NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

FULL POSE AND PARTIAL POSE CALIBRATION OF A SIX DEGREE OF FREEDOM ROBOT MANIPULATOR ARM

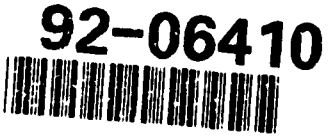
by

Scott A. Potter

September 1991

Thesis Advisor: Morris R. Driels

Approved for public release; distribution is unlimited.



REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			
4. PERFORMING ORGANIZATION REPORT NUMBER(S)		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b. OFFICE SYMBOL *(If Applicable)* 34	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
6c. ADDRESS *(city, state, and ZIP code)* Monterey, CA 93943-5000		7b. ADDRESS *(city, state, and ZIP code)* Monterey, CA 93943-5000	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	6b. OFFICE SYMBOL *(If Applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS *(city, state, and ZIP code)*		10. SOURCE OF FUNDING NUMBERS	
		PROGRAM ELEMENT NO.	PROJECT NO.
			TASK NO.
			WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*
FULL POSE AND PARTIAL POSE CALIBRATION OF A SIX DEGREE OF FREEDOM ROBOT MANIPULATOR ARM

12. PERSONAL AUTHOR(S)
Potter, Scott Alton

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(year, month, day)*	15. PAGE COUNT
Master's Thesis	FROM 1/91 TO 9/91	September 1991	131

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17. COSATI CODES			18. SUBJECT TERMS *(continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUBGROUP	Robot Manipulator Calibration

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

A six degree of freedom robot manipulator arm, a PUMA 560, is calibrated using full pose and partial pose methods in order to improve the accuracy of the manipulator arm. The theory applicable to modeling of mechanisms is introduced. A thirty parameter kinematic model is developed for use in the full pose calibration method and a twenty-six parameter kinematic model is developed for the partial pose calibration. A simulation study is performed to determine the applicability and feasibility of each model. Experimental pose measurements are performed using each method to obtain data with which to perform an actual calibration of the manipulator and compare with the predicted results. The effects of noise in each measurement system employed and in the manipulator's joint position encoders are discussed. The measurement systems employed are examined in detail and a comparison between the two is performed.

The measured kinematic parameters of each model are presented as results.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Professor Morris R. Driels	22b. TELEPHONE (Include Area Code) (408) 646-3383	22c. OFFICE SYMBOL 69Dr

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted SECURITY CLASSIFICATION OF THIS PAGE
All other editions are obsolete Unclassified

Approved for public release; distribution is unlimited.

Full Pose and Partial Pose Calibration of a Six Degree of Freedom Robot Manipulator Arm

by

Scott Alton Potter
Lieutenant, United States Navy
B.E., Vanderbilt University, 1986

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1991

Author: _____
Scott A. Potter

Approved by: _____
Morris Driels, Thesis Advisor

_____
Anthony J. Healey, Chairman
Department of Mechanical Engineering

ABSTRACT

A six degree of freedom robot manipulator arm, a PUMA 560, is calibrated using full pose and partial pose methods in order to improve the accuracy of the manipulator arm. The theory applicable to modeling of mechanisms is introduced. A thirty parameter kinematic model is developed for use in the full pose calibration method and a twenty-six parameter kinematic model is developed for the partial pose calibration. A simulation study is performed to determine the applicability and feasibility of each model. Experimental pose measurements are performed using each method to obtain data with which to perform an actual calibration of the manipulator and compare with the predicted results. The effects of noise in each measurement system employed and in the manipulator's joint position encoders are discussed. The measurement systems employed are examined in detail and a comparison between the two is performed.

The measured kinematic parameters of each model are presented as results.

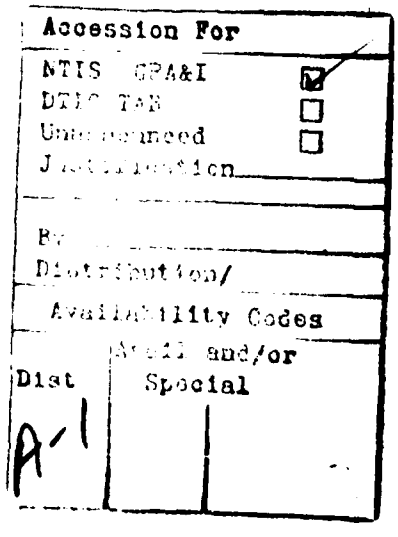


TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

In the calibration of a manipulator the desired result is to significantly improve the manipulator's accuracy. Accuracy is defined as the ability of a robot to move to a commanded pose defined in the manipulator's working volume [Ref. 1]. Stated simply, how accurately can the origin of a coordinate frame and the orientation of the coordinate axes attached to the end effector of the manipulator arm, to be termed achieving a certain pose, be positioned in relation to a commanded point and orientation in the manipulator's working volume? Present experimentation shows that manipulators have an accuracy of about 10.0 mm [Ref. 2].

Another indication of a manipulator's ability to achieve a pose is it's repeatability. Repeatability is the ability of a manipulator to return to a previously achieved pose [Ref. 3]. To *illustrate repeatability*, consider the following. A manipulator arm's end effector, termed the tool, is moved to a certain position and orientation in the working volume. After the position of each joint angle of the manipulator is recorded, the tool's position and orientation is altered. The manipulator is commanded to assume the previously recorded joint angles. Examination will reveal that the tool's position and orientation will differ from the originally learned pose. This is the error in repeatability of a manipulator arm. Present experimentation shows that manipulators have an repeatability of around 0.3 mm [Ref. 4].

To summarize, the fundamental difference between accuracy and repeatability is that repeatability is the ability of the manipulator to return to a previously achieved pose and accuracy is the ability of the manipulator to move to a pose that is specified in the working volume and that may have never been previously reached. It is apparent that a more accurate manipulator is desired over a more repeatable manipulator. The accurate manipulator's working volume is unconstrained while the repeatable manipulator does not

have a working volume, only discrete positions which must be taught to the manipulator. Correspondingly, the objective of the calibration process is to improve the accuracy of the manipulator.

In order to understand how a calibration is performed on a manipulator one must understand how a commanded pose would be achieved by any manipulator. When a specific position and orientation is commanded to the manipulator, it must be specified relative to a world coordinate frame defined by the user. To the manipulator this defined world coordinate frame is, as described, the center of the universe. By means of several mathematical operations the coordinate frame originally attached to the world coordinate frame is moved from the world coordinate frame to the manipulator' base and through each arm of the manipulator until it reaches the tool. This implies a knowledge of a mathematical model of the manipulator which includes a world coordinate frame.

As it is prohibitively expensive to manufacture and assemble each manipulator to the exact specifications required for desired accuracy, every manipulator built has a unique geometry. Since each manipulator has this unique geometry, each will have a unique mathematical model. The calibration process will identify the kinematic parameters embedded within the manipulator in order that the individual mathematical model will accurately represent the physical manipulator.

In the calibration process, several sequential steps enable the precise kinematic parameters of the manipulator to be identified, leading to improved accuracy. These steps are described as follows [Ref. 5]:

1. A kinematic model of the manipulator and calibration process is developed. The resulting model is used to define a pose error quantity based on a nominal kinematic parameter set, an unknown parameter set representing the true geometry of the manipulator and a set of joint angles. The nominal kinematics would be supplied by the manufacturer of

the manipulator, while the unknown, actual parameter set will be identified in the calibration process.

2. Experimental measurements of the robot pose, which could be either full pose or partial pose, are taken in order to obtain data which correspond to the actual kinematic parameters of the manipulator.

3. The actual kinematic parameters are identified by systematically changing the nominal parameter set so as to reduce the error quantity defined in the modeling phase. This is performed as a multidimensional optimization search in which the identified kinematic parameters are changed in order to reduce an error function to a minimum amount.

4. The final step involves incorporating the identified parameters into the actual controlling software of the manipulator.

Once the calibration process is complete the next step is to implement the identified kinematic parameters into the operation of the manipulator arm. In an operational setting the required position and orientation of the manipulator arm's end effector would be input to a controlling software program. This control program would perform an inverse kinematic solution through the mathematical model of the manipulator encoded within the control scheme, calculating the required joint angles to position and orient the end effector as dictated by the operator.

This work addresses the issue of developing the kinematic model which represents the PUMA manipulator arm and gathering the experimental data used in the calibration process. Two methods will be described in this thesis: a full pose calibration process and a partial pose calibration process.

The full pose calibration process will provide the benchmark calibration since the full pose of the end effector will be measured in each observation.

Once the benchmark kinematic parameters are identified the second method of calibration, the partial pose calibration will be performed. The resulting kinematic model produced by this calibration will be contrasted with the values identified in the benchmark.

II. THEORY

A. MODELING

The theory presented and several of the diagrams in this chapter follows closely that presented by Paul [Ref. 6] in Chapters I through III. Also, several diagrams and explanations in this chapter are based on readings from Mooring, Roth, and Driels [Ref. 7].

1. Homogeneous Transformations

The most basic premise on which the calibration process is associated is the movement of a set of coordinate axes from one position and orientation in space to another position and orientation, as illustrated in Figure 1. In Figure 1, the coordinate axes at point **A** are transformed, via translations and rotations, to point **B**.

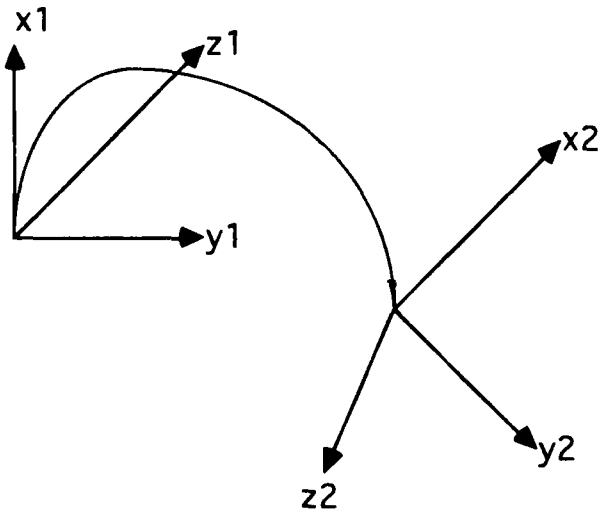


Figure 1. A Homogeneous Transformation

In addition to the translation of the origin of the coordinate axes from one position in space to another there is a change in the orientation of the coordinate axes. The process of the movement of the coordinate axes is known as a homogeneous transformation. This homogeneous transformation is accomplished by a series of translational transformations and rotational transformations.

The first type of transformation is the translational transformation. Given a point in space, **A**, illustrated in Figure 2, the point's position is fixed and known relative to a fixed, three axis coordinate frame, defined as the world coordinate frame, by means of the following vector, **u**:

$$\vec{u} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \tag{1}$$

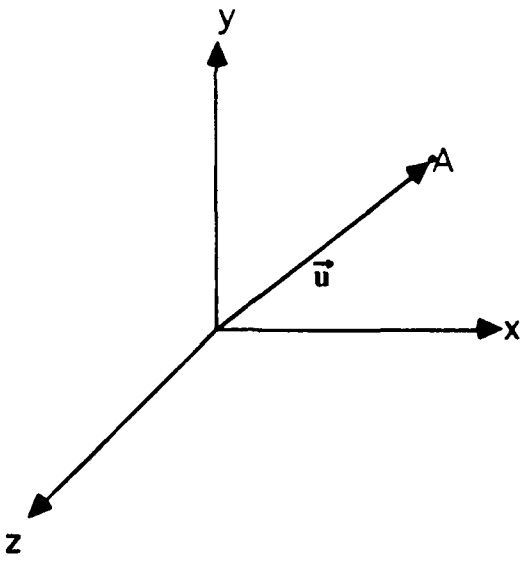


Figure 2. Position of Point A with respect to a World Coordinate Frame

The matrix element w is any non-zero scale factor. In the applications reported in this thesis w will be set equal to unity, producing the following vector:

$$\vec{u} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2}$$

The transformation from point **A** to point **B**, illustrated in Figure 3, requires the transformation matrix [**H**]:

$$\mathbf{H} = \text{Trans}(x,y,z) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

The translation from point **A** to point **B** is represented by the vector $\vec{\mathbf{v}}$:

$$\vec{\mathbf{v}} = a\vec{\mathbf{i}} + b\vec{\mathbf{j}} + c\vec{\mathbf{k}} \tag{4}$$

The transformed vector is the product of the transformation matrix, **H**, and the original position vector, **u**.

$$\vec{\mathbf{v}} = \mathbf{H}\,\vec{\mathbf{u}}$$

$$\vec{\mathbf{v}} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{5}$$

The translation is to be interpreted as the summation of the vectors **u** and **v**. This is illustrated in Figure 3 and described in equation form by equation (6).

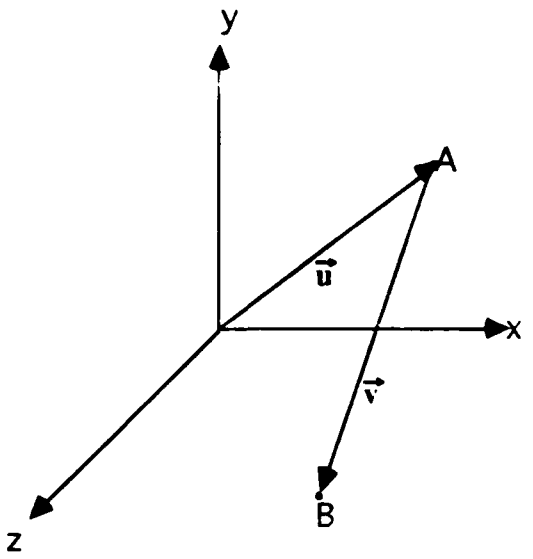


Figure 3. Translation of a Point in Space from Point A to Point B

$$\vec{u} + \vec{v} = \left(\frac{x}{w}\right)\vec{i} + \left(\frac{y}{w}\right)\vec{j} + \left(\frac{z}{w}\right)\vec{k} + a\vec{i} + b\vec{j} + c\vec{k} \tag{6}$$

The second type of transformation is the rotational transformation. Given the fixed coordinate frame of the previously discussed translational transformation the transformation corresponding to a rotation, θ, about the x-axis is:

$$\text{Rot}(x,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

The transformation corresponding to a rotation, θ, about the y-axis is:

$$\mathrm{Rot}(y,\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

The transformation corresponding to a rotation, θ, about the z-axis is:

$$\mathrm{Rot}(z,\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

As an illustration of a rotational transformation consider the following. Given another point in space, **A**, illustrated in Figure 4, if it was desired to rotate the point around the z-axis by 90 degrees the Rot(z) matrix above would be used. The value for θ, 90 degrees, would be entered into the matrix, producing:

$$\mathrm{Rot}(z,90^{\circ}) = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

This matrix would multiply the vector representing the original position of point **A**:

$$\vec{\mathbf{u}} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{11}$$

producing the new position vector, **v**, of the point **A**.

$$\vec{\mathbf{v}} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{12}$$

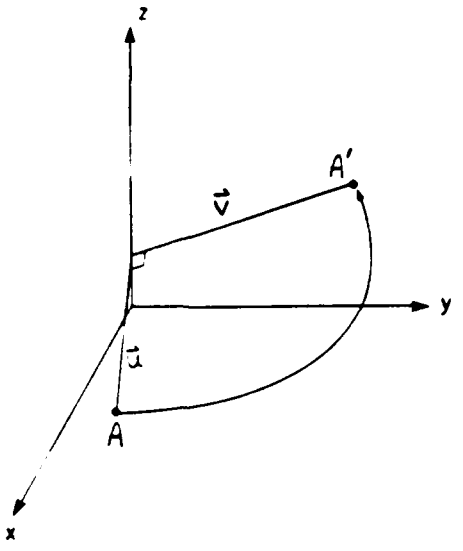


Figure 4. Rotation of Point A 90 Degrees about the Z-axis, Rot(z,90)

Translational and rotational transformations are performed right to left as they are read, for example:

$$\text{Trans}(a,b,c)\text{Rot}(y,90)\text{Rot}(z,90) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

The transformation illustrated in Figure 5 is a a rotation of 90 degrees about the z-axis, a rotation of 90 degrees about the y-axis, a translation of 'a' mm in the x direction, of 'b' mm in the y direction, and of 'c' mm in the z direction.

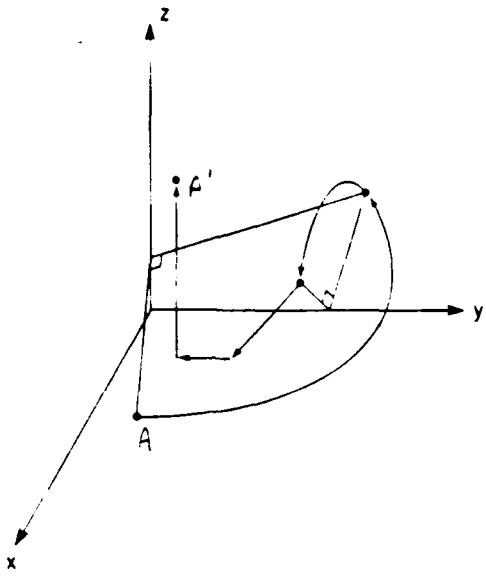


Figure 5. A Multiple Translation and Rotation Transformation of Point A

It is clear that an infinite number of translations or rotations are permitted. Each transformation is simply represented by another matrix multiplication operation.

2. Application of Basic Homogeneous Transformations to Simple Kinematic Mechanisms

Armed with the knowledge of homogeneous translations and rotations, a genera, two link kinematic mechanism will be analyzed. When homogeneous transformations are applied to the linkage shown in Figure 6, equations (3) and (9) will be utilized to move coordinate frame 1, located at point 1, to coordinate frame 2, located at point 2, by specifying the multiple transformation:

$$\mathbf{A} = \text{Trans}(d_1,0,0)\ \text{Rot}(z,\theta_1)\ \text{Trans}(d_2,0,0) \qquad (14)$$

The product of the three matrix multiplications, a frame to frame homogeneous transformation, is known as an **A** matrix. Referring to Figure 6, the coordinate frame at point 1 has been assigned such that the x-axis lies along the centerline of link 1. As illustrated in equation (14), the homogeneous transformation moves the coordinate frame along the x-axis a distance d_1, rotates the frame about the z-axis θ_1 degrees, and then translates the coordinate frame a distance d_2 along the newly rotated x-axis.

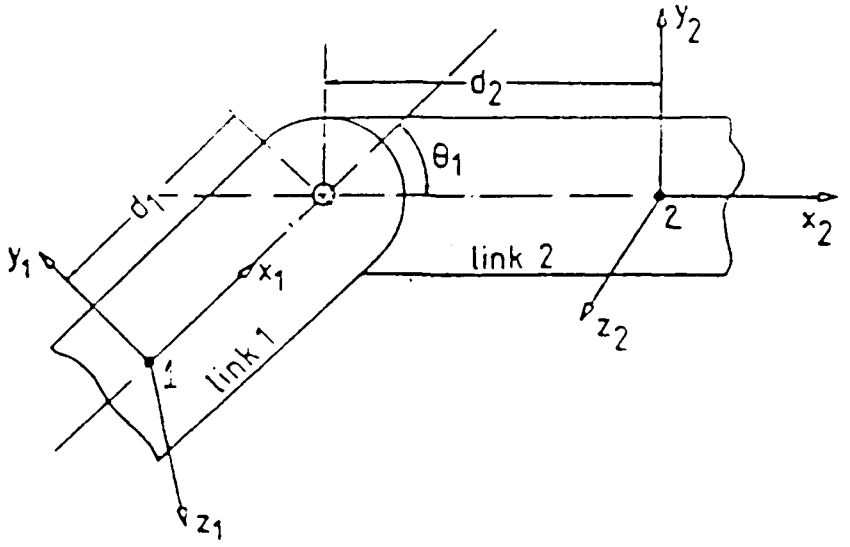


Figure 6. A Two-Link Kinematic Mechanism

For more general link structures, which most likely will be three dimensional, the coordinate frame transformations are more complex than the example shown in Figure 6. Due to this complexity, a standardization of the allocation of coordinate frames and the transformation process of the coordinate frames is required. These standardizations will be discussed in the next section.

3. Links, Joints, and Assignment of Coordinate Frames

Any manipulator can be described as a series of links connected together by joints. A coordinate frame is placed on each link in the manipulator and, as discussed earlier, homogeneous transformations are used to describe the relative position and orientation difference between two successive links. Each homogeneous transformation matrix operation rotates or translates the coordinate frame to various positions in the manipulator.

The base of a manipulator is defined to be link 0 and is not considered to be one of the defined number of links composing the manipulator. In between the base of the manipulator, link 0, and the first joint of the manipulator is link 1. Thus, link 0 is attached to link 1 by joint 1. Link 2 is attached to link 1 by joint 2, and so on.

A link is characterized by two dimensions: the common normal distance, a_n, and the angle, α_n, between the axes in a plane perpendicular to a_n. The common normal distance is known as the length of the link and the angle is known as the twist of the link. These two link dimensions are illustrated in Figure 7.

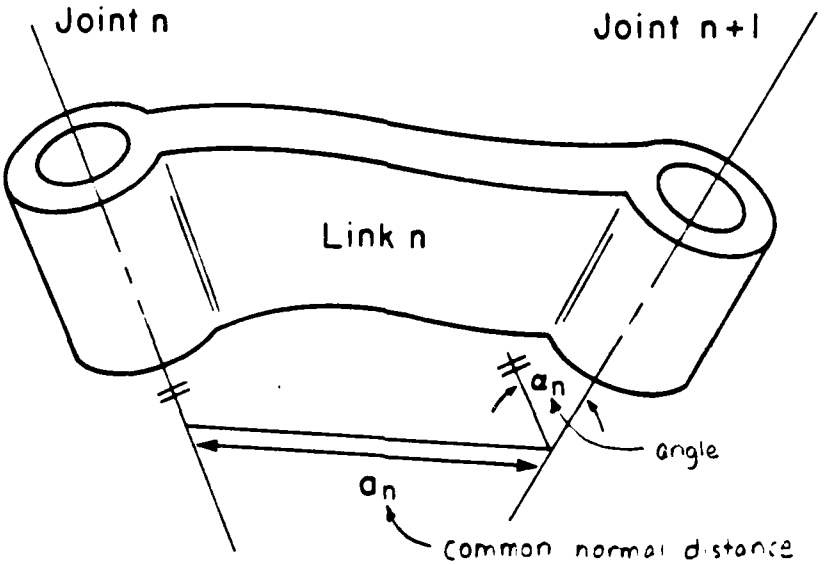


Figure 7. The Length, a, and Twist, α, of a Link

To aid in the visualization of these two parameters perform the following. Visualize a small diameter bar of reasonable length. At each end of the bar is a joint which pivots. Now the bar looks like a goal post with short uprights and a long crossbar. Take the uprights and bend them in. The bar will appear as Figure 8.

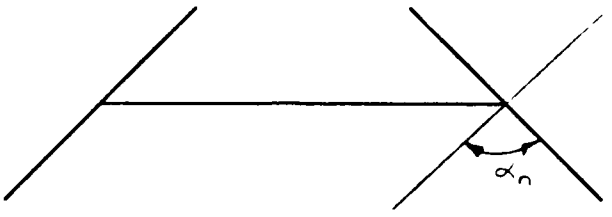


Figure 8. Illustration of Link Dimensions a and α

The next step is to grasp the uprights and twist them in opposite directions. The length of the line which connects the left upright to the right upright, while maintaining right angles with each, is a_n. To visualize α_n, draw a line which is parallel to the left upright through the point where the crossbar is attached to the right upright. The angle between the line which has been drawn and the right upright is the angle of twist, α_n.

As stated earlier, any manipulator is composed of a series of links which are attached by joints. The relationship between links is described using the distance and angle between links. The distance between the links is defined as d_n and the angle between two successive links is defined as θ_n. Refer to Figure 9 in the following discussion.

Two links will be connected at a joint. The joint will have a joint axis, around which the two links will rotate. A normal from each link intersects the joint axis. The relative position of two successive links is defined as d_n, the distance separating the points of intersection on the joint axis of the normals of the two links under consideration. Define the normal of the link to the left of the joint as N_1. Define the normal of the link to the right

of the joint as N_2. Extend N_1 through the intersection of the joint axis. Draw a line parallel and coplanar to N_2 through the point of intersection of N_1 with the joint axis. The angle between the extension of N_1 and the line parallel to N_2 is defined as θ_n.

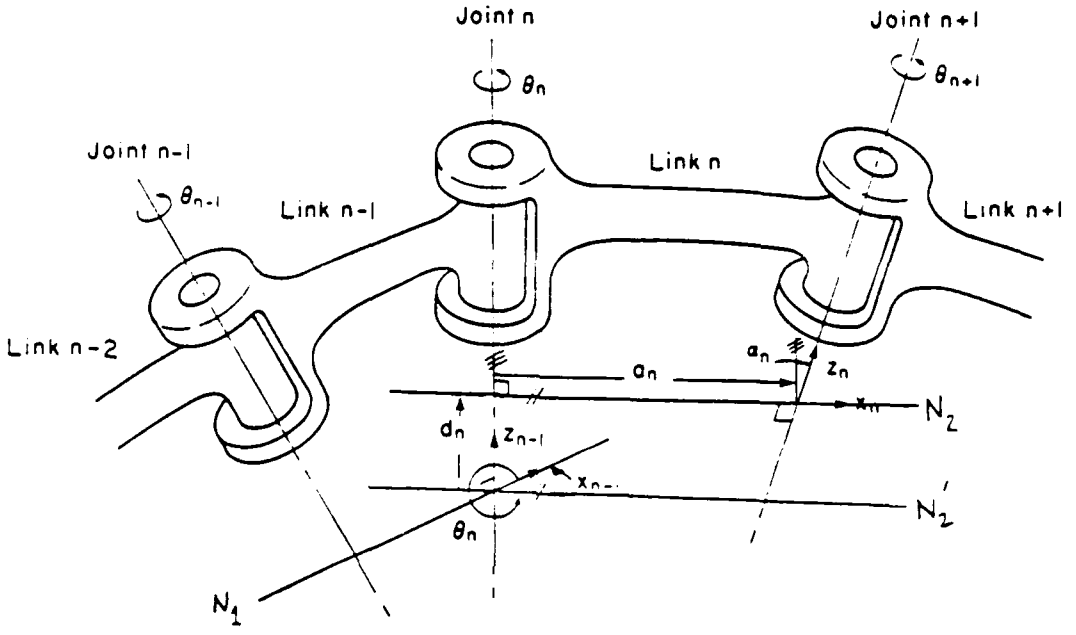


Figure 9. Link Parameters θ, d, a, and α

In order to describe the relationships between links, a coordinate frame will be assigned to each link, based on the type of joint to which the link is attached. There are two types of joints, revolute and prismatic.

Consider a revolute joint, illustrated in Figure 10. Revolute joints are characterized by θ_n, the joint variable. The origin of the coordinate frame of link n is set to be at the intersection of the common normal between the axes of joints n and n+1, which will be the same line as the one measured for the length of link n, extended if necessary,

and the axis of joint n+1. In the case of intersecting joint axes, the origin is at the point of intersection of the joint axes. If the axes are parallel the coordinate origin is not uniquely defined if the above specifications are used, so the origin is chosen to make the joint distance zero for the next link whose coordinate origin is defined. The z-axis for link n will be aligned with the axis of joint n+1. The x-axis will be aligned with any common normal which exists and is directed along the normal from joint n to joint n+1. In the case of intersecting joints, the direction of the x-axis is parallel or antiparallel to the vector cross product $z_{n-1} \times z_n$.

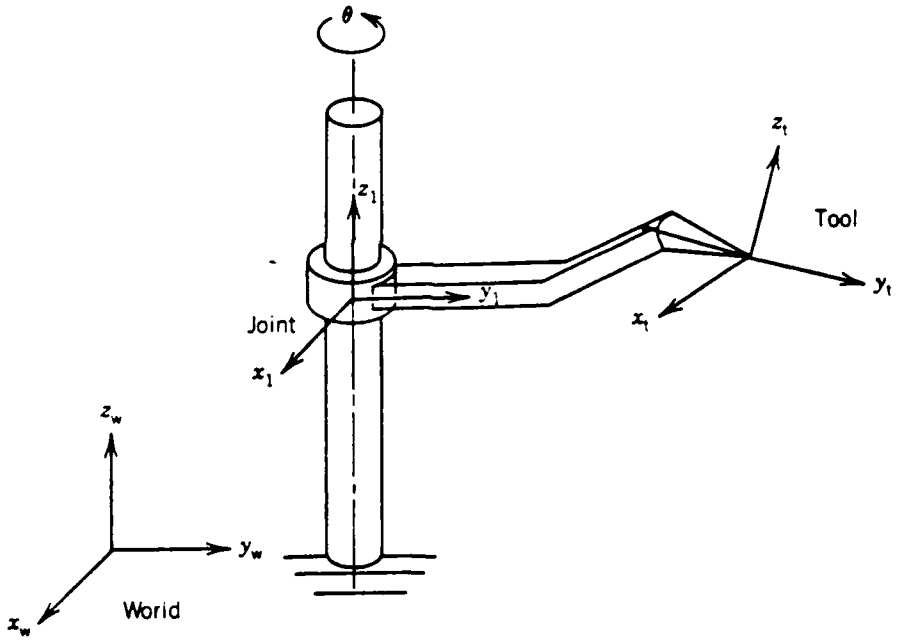


Figure 10. Revolute Joint

Now consider prismatic a joint, which is illustrated in Figure 11. A prismatic joint is characterized by the distance d_n being the joint variable. The direction of the joint

axis is the direction in which the joint moves. The direction of the axis is defined but, unlike a revolute joint, the position in space is not defined. In the case of a prismatic joint the link length, a_n, has no meaning and is set to be zero. The origin of the coordinate frame for the prismatic joint is coincident with the next defined link origin. The z-axis of the prismatic link is aligned with the axis of joint n+1. The x_n-axis is parallel or antiparallel to the vector cross product of the direction of movement of the prismatic joint and z_n.

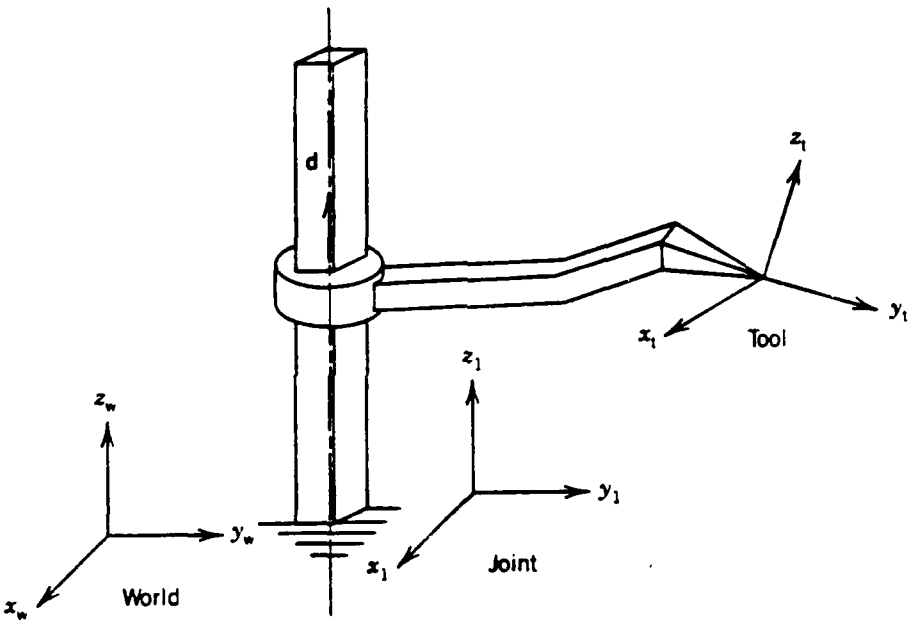


Figure 11. Prismatic Joint

4. The Denavit-Hartenburg Transformation

Now that a coordinate frame has been assigned to each joint of the manipulator, the next step is to develop the relationship between coordinate frames. The relationship between coordinate frame n and n-1 will now be examined in detail. In the calibration

processes reported in this thesis two types of relationships will be used. They are known as the modified Denavit-Hartenburg (MDH) transformation, a frame to axis transformation, and the Euler transformation, a frame to frame transformation.

The modified Denavit-Hartenburg transformation is a derivative of the standard Denavit-Hartenburg transformation. The standard Denavit-Hartenburg (DH) transformation incorporates the translations and rotations introduced earlier in the thesis, and result in an **A** matrix of the following form:

$$\mathbf{A}_n = \text{Rot}(z,\theta_n) \times \text{Trans}(z,d_n) \times \text{Trans}(x,a_n) \times \text{Rot}(x,\alpha_n) \qquad (15)$$

This is to be interpreted as:

1. A rotation of angle θ_n about the z-axis.
2. A translation of length d_n along the z-axis.
3. A translation of length a_n along the newly rotated x-axis.
4. A rotation of angle α_n about the newly rotated x-axis.

The standard Denavit-Hartenburg transformation is not implemented in the modeling of manipulator arms for use in the calibration procedures described in this thesis. As described earlier, the location of coordinate frames are functions of manipulator geometry. Any variation in the manipulator geometry will cause a cascading change in the positions of the coordinate frames. This results in three effects which are undesirable for manipulator calibration:

1. Selection of the base frame is not arbitrary
2. The zero position of the manipulator is not arbitrary.
3. If two joints having nominally parallel axes are found to have non-parallel axes, then the transformation parameters will dramatically change.

The third result is the most important. Many manipulators are designed to have consecutive parallel axes. In this case, there is no common normal between the two axes. In conforming to the rules introduced earlier the coordinate frame is chosen such that the joint distance will be zero for the next link whose coordinate link whose origin is defined. In almost every calibration procedure a misalignment of the two nominally parallel joint axes will be identified. A misaligned pair of joint axes produces a common normal between them, and the common normal fixes the position of the coordinate frame. The new, fixed position of the coordinate frame could be quite different from the user defined position of the coordinate frame based on the joint distance being zero. This could cause a cascading change in position of coordinate frames downstream of the coordinate frame in question. Such significant changes in several positions of coordinate frames causes difficulty in the kinematic parameter identification program.

The problem with the parallel axes is eliminated by using the MDH transformation, which introduces a fifth matrix multiplication in the **A** matrix, as described in the following section.

5. The Modified Denavit-Hartenburg Transformation

The MDH transformation is, as stated earlier, a frame to axis transformation and is represented by the following **A** matrix,

$$\mathbf{A}_n = \mathrm{Rot}(z,\theta_n) \times \mathrm{Trans}(z,d_n) \times \mathrm{Trans}(x,a_n) \times \mathrm{Rot}(x,\alpha_n) \times \mathrm{Rot}(y,\beta_n) \qquad (16)$$

This is to be interpreted as:

1. A rotation of angle θ_n about the z-axis.
2. A translation of length d_n along the z-axis.
3. A translation of length a_n along the newly rotated x-axis.
4. A rotation of angle α_n about the newly rotated x-axis.

5. A rotation of angle β_n about the newly rotated y-axis.

In reduced matrix form, for a rotational joint, the operation is as follows:

$$A_n = \begin{matrix} \cos(\theta) & -\sin(\theta)\cos(\alpha) & \sin(\theta)\sin(\alpha) & a\cos(\theta) \\ \sin(\theta) & \cos(\theta)\cos(\alpha) & -\cos(\theta)\sin(\alpha) & a\sin(\theta) \\ 0 & \sin(\alpha) & \cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{matrix} \qquad (17)$$

In reduced matrix form, for a prismatic joint, the operation is as follows:

$$A_n = \begin{matrix} \cos(\theta) & -\sin(\theta)\cos(\alpha) & \sin(\theta)\sin(\alpha) & 0 \\ \sin(\theta) & \cos(\theta)\cos(\alpha) & -\cos(\theta)\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{matrix} \qquad (18)$$

6. The Euler Transformation

The Euler transformation is a frame to frame transform which requires six parameters:

$$A_n = \mathrm{Rot}(z,\varphi_n) \times \mathrm{Rot}(y,\theta_n) \times \mathrm{Rot}(x,\varphi_n) \times \mathrm{Trans}(p_{x_n}, p_{y_n}, p_{z_n}) \qquad (19)$$

This will be interpreted as a rotation φ_n about the z-axis, followed by a rotation θ_n about the newly rotated y-axis, followed by a rotation φ_n about the newly rotated x-axis. At the conclusion of these rotations the axes of frame n, the coordinate frame to which the transform is occurring, will be perfectly aligned with the transforming coordinate frame. The translations p_x, p_y, and p_z will move the aligned coordinate axes from the origin of coordinate frame n-1 to coordinate frame n.

7. Application to the PUMA

It has been established that an **A** matrix is a homogeneous transformation from one coordinate frame to the next. This **A** matrix can either be, in applications described in this thesis, a modified Denavit-Hartenburg transformation matrix or an Euler transformation matrix. Matrix $\mathbf{A}_1$ will describe the position and orientation of the first coordinate frame with respect to the world coordinate frame. The coordinate frame attached to link 0 is translated and rotated such that it is aligned in the proper position and orientation, in accordance with the rules described earlier, on link 1 of the manipulator. Matrix $\mathbf{A}_2$ will describe the position and orientation of the second coordinate frame with respect to the first coordinate frame. The coordinate frame attached to link 1 is translated and rotated such that it is aligned in the proper position and orientation on link 2 of the manipulator. Thus, the position and orientation of the second coordinate frame with respect to the base coordinate frame is defined to be:

$$\mathbf{T}_2 = \mathbf{A}_1 \times \mathbf{A}_2 \tag{20}$$

Since the PUMA 560 manipulator is a six link manipulator, the $\mathbf{T}_6$ matrix, which the indicate the position and orientation of the tool frame with respect to the world coordinate frame, will be represented by:

$$\mathbf{T}_6 = \mathbf{A}_1 \times \mathbf{A}_2 \times \mathbf{A}_3 \times \mathbf{A}_4 \times \mathbf{A}_5 \times \mathbf{A}_6 \tag{21}$$

Figure 12 illustrates that any coordinate frame may be reached from any other coordinate frame by executing the matrix multiplications between the initial coordinate frame and the final coordinate frame, moving clockwise or counter-clockwise.

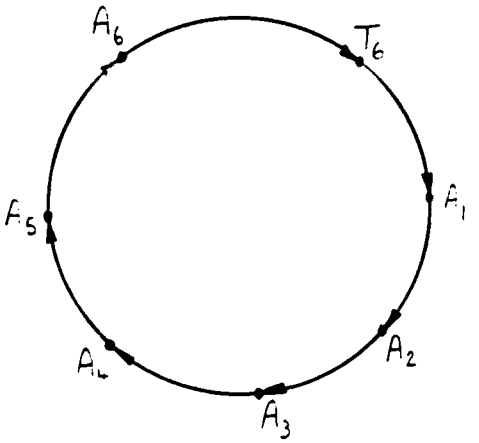


Figure 12. Homogeneous Transformation Loop

Moving clockwise would generate a forward kinematic solution, equivalent to moving through the manipulator from, for example, coordinate frame 3 to the tool frame. Moving counter-clockwise will generate an inverse kinematic solution, equivalent to moving backwards through the manipulator from coordinate frame 4 to coordinate frame 2.

With the acquired knowledge of links, joints, how to assign reference frames and transform from one reference frame to the next, the PUMA 560's coordinate frames and the nominal kinematic parameters will be introduced.

Figure 13 illustrates individual links and joints of the PUMA 560.

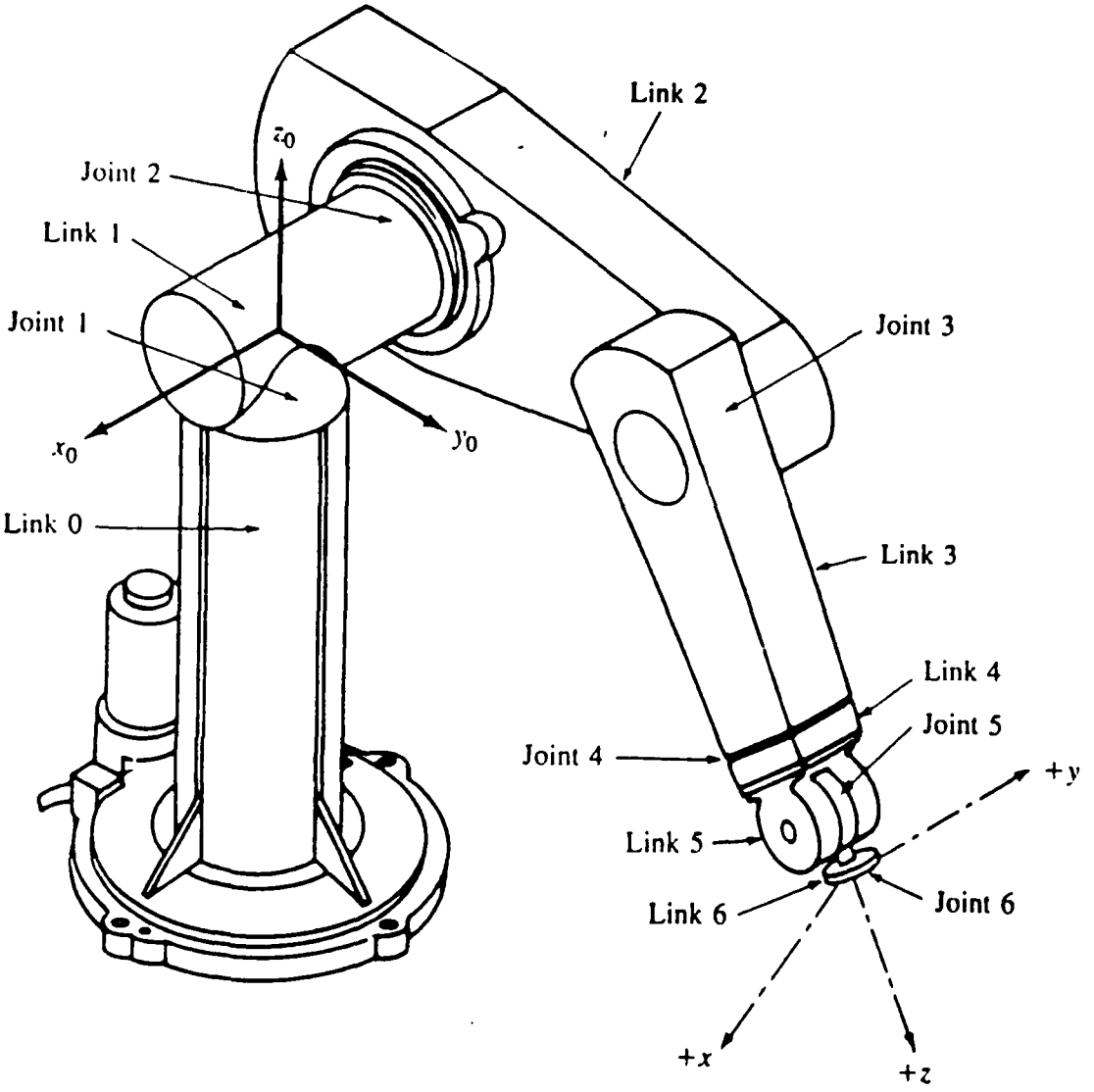


Figure 13. Links and Joints of the PUMA 560 Robot Manipulator Arm

If all of the rules governing the placement of coordinate frames are followed, the position and orientation of the coordinate frame corresponding to each joint may be displayed. Figure 14 illustrates the allocation of each coordinate frame.

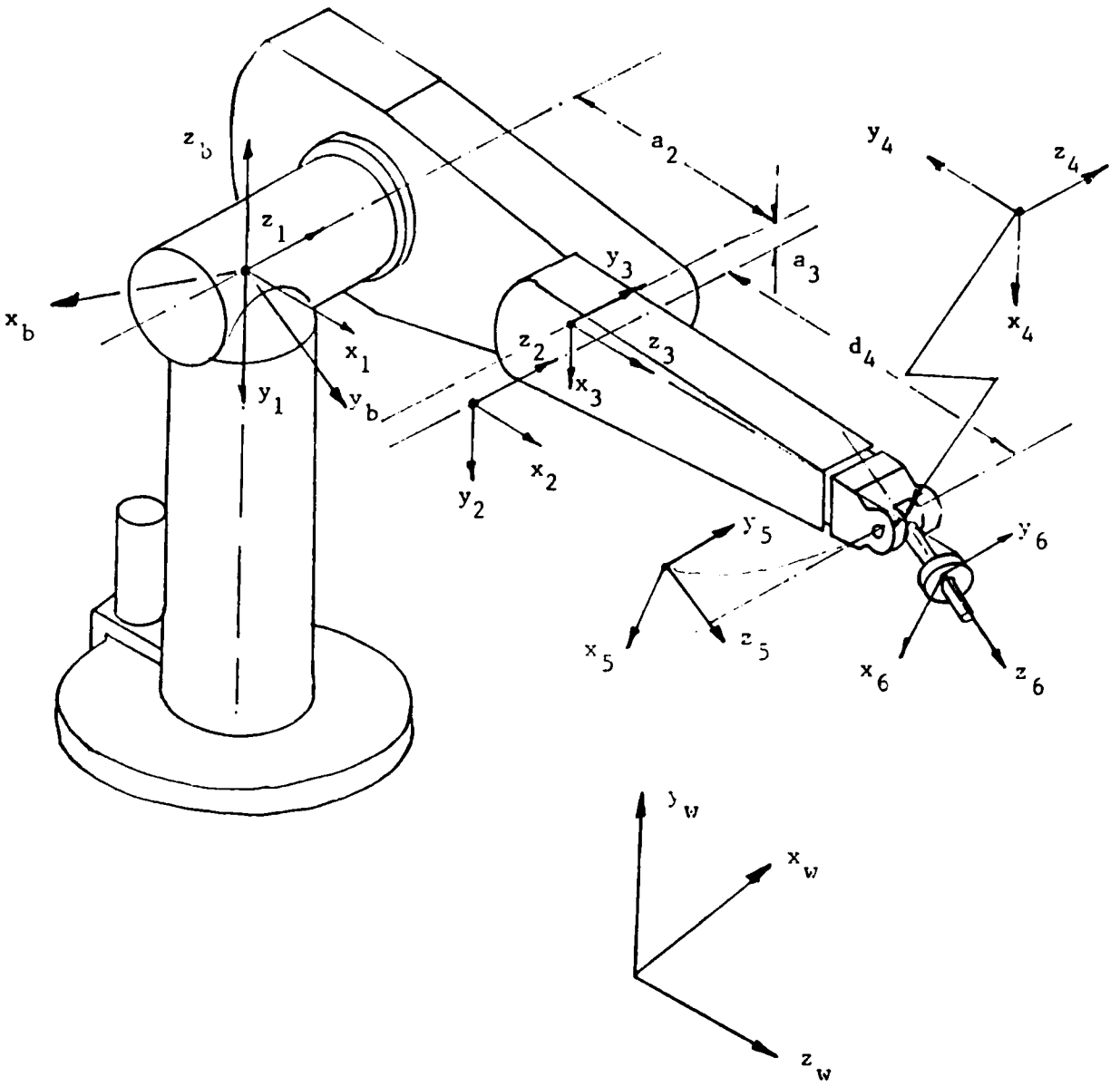


Figure 14. PUMA Frame Allocation

The internal kinematics of the PUMA will remain constant regardless of the world coordinate frame used or type of measurement system employed. Refer to Figure 14

to follow the discussion of the transformation from the base coordinate frame of the manipulator to frame 5. Note that the position of the manipulator depicted in Figure 14 does not show the manipulator in its zero position. The transforms which will be described in the next paragraph will move from joint n-1's zero position to joint n's zero position. All of the transformations within the PUMA are MDH transformations. In both of the calibration procedures reported in this thesis the transformation from the world coordinate frame to the base frame and the transformation from frame 5 to frame 6, the tool frame, is an Euler transformation. The details of those transformations will be discussed during the simulation phase of each calibration procedure.

1. Base frame, frame 0, to frame 1: No rotation about Z_0, no translation along the Z_0-axis, no translation along the X_0-axis, rotate -90° about X_0. This fixes coordinate frame 1.

2. Frame 1 to frame 2: No rotation about Z_1, no translation along the Z_1-axis, translate along the X_1-axis 431.85 mm, no rotation about X_1. This fixes coordinate frame 2.

3. Frame 2 to frame 3: No rotation about Z_2, translate along the Z_2-axis 149.09 mm, translate along the X_2-axis -20.33 mm, rotate 90° about X_2. This fixes coordinate frame 3.

4. Frame 3 to frame 4: No rotation about Z_3, translate along the Z_3-axis 433.0 mm, no translation along the X_3-axis, rotate -90° about X_3. This fixes coordinate frame 4.

5. Frame 4 to frame 5: No rotation about Z_4, no translation along the Z_4-axis, no translation along the X_4-axis, rotate 90° about X_4. This fixes coordinate frame 5.

Since the coordinate frames have been assigned to each joint and the relationship between each frame has been determined, a table of nominal kinematic parameters may be constructed. Note that the transform from the base frame to frame 1 and

the transform from frame 5 to the tool frame are Euler transformations. All other transformations are Modified Denavit-Hartenburg transformations. The bold entries are defined to be zero.

TABLE 1. NOMINAL KINEMATIC PARAMETERS FOR THE PUMA 560

φ_b	θ_b	φ_b	Pxb	Pyb	Pzb
degrees	degrees	degrees	mm	mm	mm
180.0	0.0	90.0	-394.0	-383.0	474.0

link number	$\Delta\theta_i$ degrees	d_i mm	a_i mm	α_i degrees	β_i degrees
1	**0**	**0**	0.0	-90.0	**0**
2	0.0	**0**	431.85	0.0	0.0
3	0.0	149.09	-20.33	90.0	**0**
4	0.0	433.00	0.0	-90.0	**0**
5	0.0	0.0	0.0	90.0	**0**

φ_6	θ_6	φ_6	Px6	Py6	Pz6
degrees	degrees	degrees	mm	mm	mm
90.0	0.0	0.0	0.0	0.0	134.0

At this point the concept of the forward kinematic solution and the inverse kinematic solution will be introduced. A forward kinematic solution is necessary to determine the position and orientation of the tool frame with respect to the world coordinate frame if all six joint angles are known. Given six joint angles, the pose can be determined using a forward kinematic solution. An inverse kinematic solution is necessary to determine the six required joint angles which would place the the tool frame in a specified position and orientation. Given the tool pose, the six joint angles can be determined using an inverse kinematic solution.

B. IDENTIFICATION METHODOLOGY

1. Introduction

Consider a generic function in the x-y plane, illustrated in Figure 15.

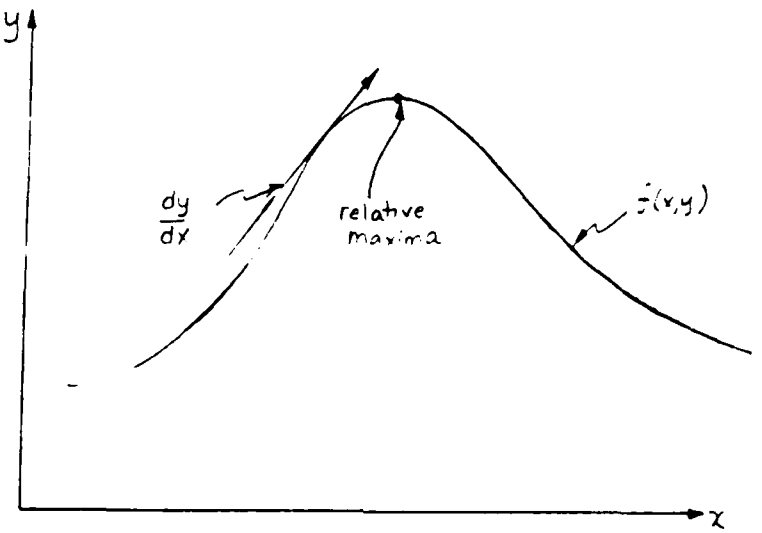


Figure 15. Generic Function in the x-y Plane

If a relative maxima of the function was to be determined within a range of x then the derivative of the function could be calculated at some x within that range. If the derivative was positive then the maxima would be to the right of the x value and if the derivative was negative then the maxima would be to the left. The x value would be incremented and the derivative calculated until the the derivative equaled zero. The function would be optimized at this point within the specified range of x.

This concept can be extended to surfaces. The problem of climbing a hill in the most efficient manner is one example. Consider a generic two variable function illustrated in Figure 16.

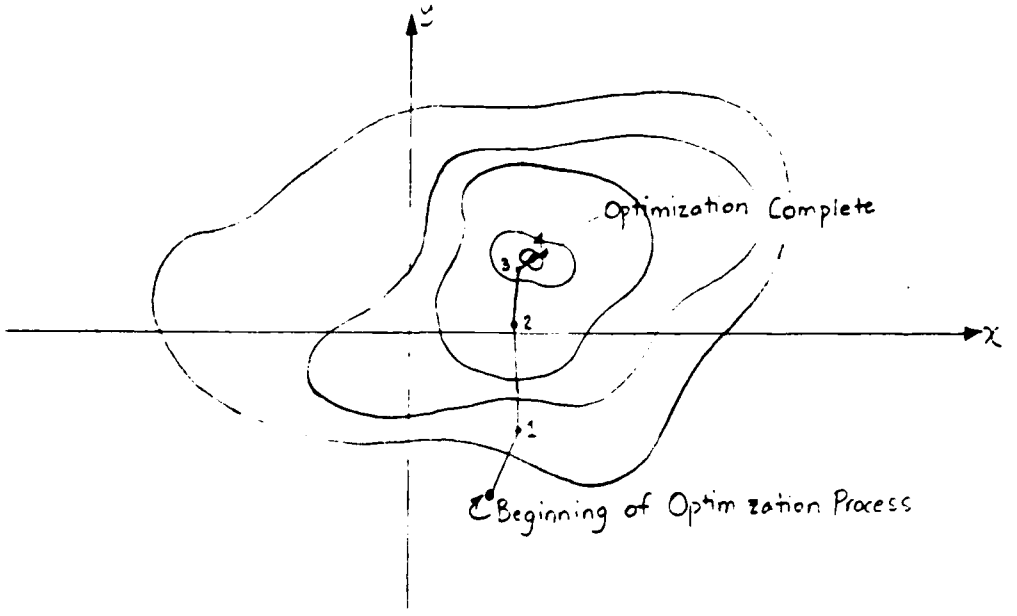


Figure 16. Example of Two-Variable Function G(x,y)

It is desired to find maximum G. To determine the direction of maximum increase of the function G at a specific point, calculate the gradient of the function and evaluate the gradient at the point in question:

$$\nabla G(x_1,y_1) = \left(\frac{\partial G}{\partial x}\right)_{x_1,y_1} \vec{i} + \left(\frac{\partial G}{\partial y}\right)_{x_1,y_1} \vec{j} \qquad (22)$$

Calculating the gradient provides the direction of maximum increase based on (x1+Δx), (x1-Δx), (y1+Δy), and (y1-Δy).

A question arises: How is the gradient calculated if G(x,y) is not specifically known? The only way to determine the gradient is through a method of orderly estimation and elimination. For a two-dimensional function this provides a challenge. If the problem

is extended to n-dimensional space the problem is extremely difficult to solve. The difficulty in n-dimensional optimization is the development of an algorithm which will change the n variables of the function in an efficiently and orderly manner in order that the gradient can be determined.

2. The IMSL routine ZXSSQ

The optimization procedure used in this thesis is an IMSL routine called ZXSSQ. ZXSSQ is a routine which will vary the n variables of a function in order that the function will be minimized. As a basic example of the operation of ZXSSQ consider the following equation:

$$y = (X_1 \times Z) + \sin(X_2 \times Z) \tag{23}$$

This equation will be the model of some physical system. The parameters X_1 and X_2 are to be determined based on the observation of y at incremental values of Z. The following observations are taken:

$$\begin{aligned} y_1 &= 1 + \sin(1) + (-0.1) = 1.74 \\ y_2 &= 2 + \sin(2) + (0.1) = 3.00 \\ y_3 &= 3 + \sin(3) + (-0.1) = 3.04 \\ y_4 &= 4 + \sin(4) + (0.1) = 3.34 \\ y_5 &= 5 + \sin(5) + (-0.1) = 3.94 \end{aligned} \tag{24}$$

which is:

$$y = X_1 \times Z + \sin(X_2 \times Z) \pm 0.1 \tag{25}$$

with:

$$\begin{aligned} X_1 &= 1 \\ X_2 &= 1 \end{aligned} \tag{26}$$

The model predicts:

$$y_1 = 1 + \sin(1) = 1.84$$
$$y_2 = 2 + \sin(2) = 2.91$$
$$y_3 = 3 + \sin(3) = 3.14$$
$$y_4 = 4 + \sin(4) = 3.24$$
$$y_5 = 5 + \sin(5) = 4.04 \qquad (27)$$

ZXSSQ will not discriminate the addition or subtraction of 0.1. This is equivalent to the presence of noise in a measurement system. So, ZXSSQ will use the flowchart in Figure 17 to select the values of X_1 and X_2 which will best fit the above data.

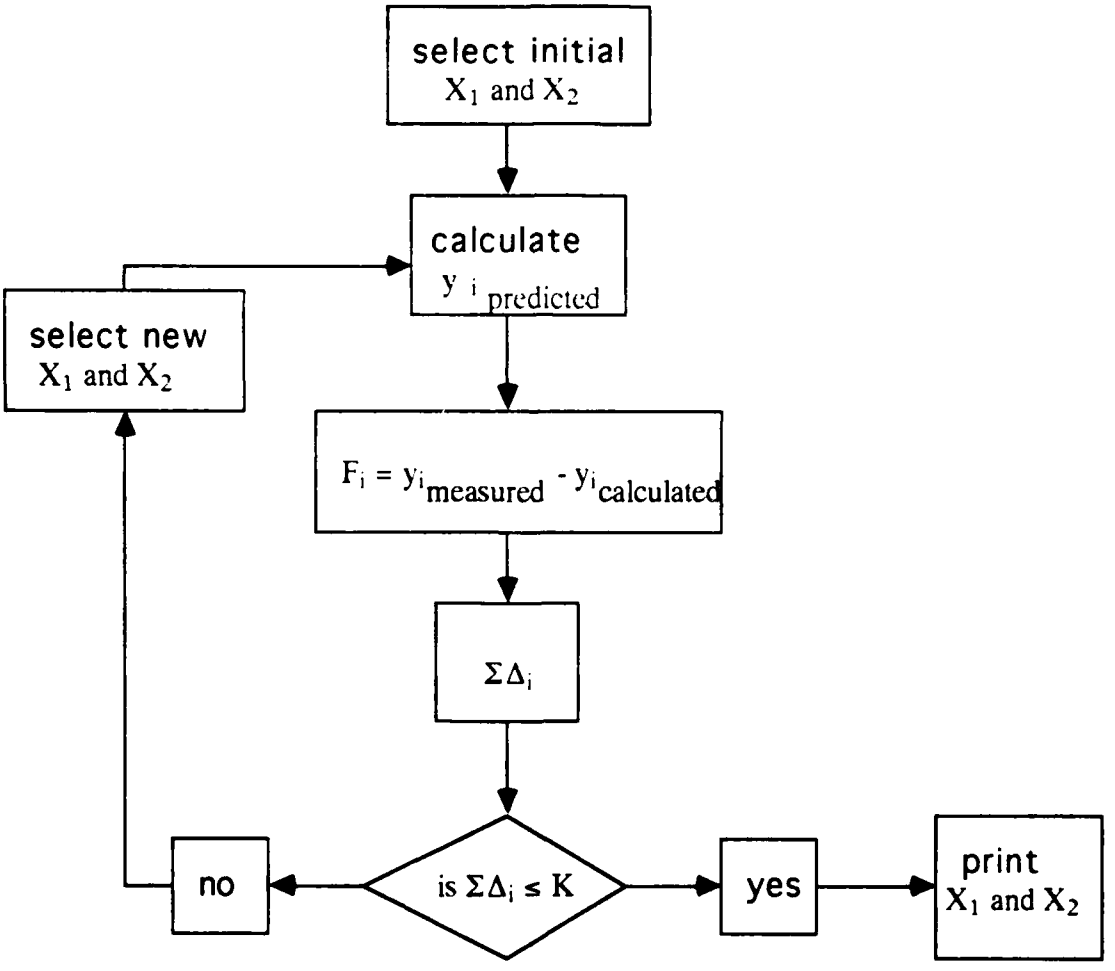


Figure 17. Flowchart of the Operation of ZXSSQ

The initial values of X_1 and X_2 are defined by the user. For each observation, producing a $y_{i_{measured}}$ and a $y_{i_{predicted}}$ will be calculated based on the current X_1 and X_2. The difference between $y_{i_{predicted}}$ and $y_{i_{measured}}$ will be calculated for each observation and the sum of these differences will be calculated. If this sum is less than the user defined convergence criteria the current values of X_1 and X_2 are acceptable. If not ZXSSQ selects new values for X_1 and X_2 and the process begins again. The process of the selection of the values of X_1 and X_2 performed by ZXSSQ is a mathematical consideration which is not important to the results reported in this thesis, but it must be understood that ZXSSQ makes the identification of the kinematic parameters of the PUMA within a reasonable period period of time possible.

As an illustration of the implementation of ZXSSQ in this thesis, consider the following. Suppose a ball is suspended in space in an unknown position relative to a defined set of coordinate axes, as illustrated in Figure 18.

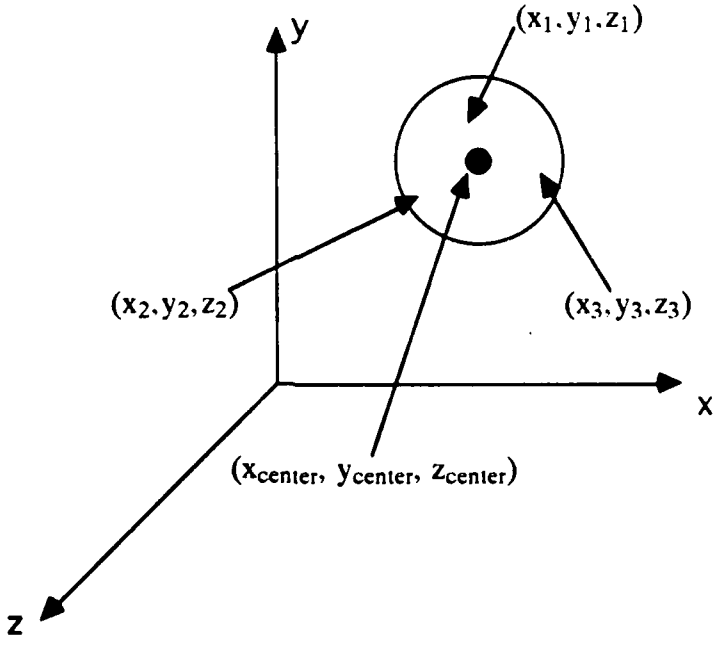


Figure 18. Determination of the Center of a Ball in Space

It is possible to use a touch probe to measure (x_i, y_i, z_i), the coordinates of an i^{th} position on the surface of the ball relative to the fixed coordinate frame. However, the exact position of the center of the ball is required in order to assist in the determination of the pose of the tool. ZXSSQ accepts the coordinates of at least three surface positions on the ball and calculates the coordinates of the center of the ball. The procedure is as follows.

The end effector used in the experimental phase of the initial calibration has a pattern of precision tooling balls which have a known radius, R_{tb}. The touch probe also has a known radius, R_{tp}. That means that the center of the ball will be $(R_{tb} + R_{tp})$ from each measured position of the touch probe. If only one touch probe measurement is made the actual position of the center of the ball could be at any position on a sphere of $(R_{tb} + R_{tp})$ radius surrounding the single measurement. As more measurements are taken the position of the center of the ball can be determined more accurately. It has been determined that a minimum of three touch probe measurements are required to accurately determine the coordinates of the center of the ball. Each measurement produces an error function:

$$F_1 = \left| (R_{tb} + R_{tp}) - \sqrt{(x_c - x_1)^2 + (y_c - y_1)^2 + (z_c - z_1)^2} \right|$$

$$F_2 = \left| (R_{tb} + R_{tp}) - \sqrt{(x_c - x_2)^2 + (y_c - y_2)^2 + (z_c - z_2)^2} \right|$$

$$F_3 = \left| (R_{tb} + R_{tp}) - \sqrt{(x_c - x_3)^2 + (y_c - y_3)^2 + (z_c - z_3)^2} \right| \qquad (28)$$

ZXSSQ will systematically vary (x_c, y_c, z_c) until each F_i is minimized. If there is no noise in the system, each F_i will be reduced to zero. If noise is present the best fit (x_c, y_c, z_c) will be calculated. Once the functions are minimized, the coordinates of the center of the ball are known.

3. Application to the Calibration Process

The kinematic parameter identification of the PUMA 560 will be performed as a multi-dimensional minimization process performed by ZXSSQ. The process will proceed in the following manner.

1. Begin with the nominal set of kinematic parameters. This set will most likely be the manufacturer's predicted parameters based on the design and construction of a generic manipulator.

2. Select a set of six joint angles, θ_1 through θ_6, for the manipulator.

3. Perform a forward kinematic solution for the PUMA, using the nominal set of kinematic parameters. This will calculate the predicted pose of the end effector of the manipulator.

4. Measure the actual pose of the end effector of the manipulator. In most cases measured pose will be different from the predicted pose.

5. Modify the kinematic parameters such that the predicted pose, determined by the forward kinematic solution using the modified kinematic parameters, matches the measured pose [Ref. 8].

This process will be applied to several sets of joint angles. The number of physical measurements, meaning the number of sets of joint angles required, must satisfy:

$$K_p \leq N \times D_f \tag{29}$$

where: K_p = number of kinematic parameters to be identified

N = number of measurements (poses) taken

D_f = number of degrees of freedom present in each measurement

Two calibration processes will be reported in this thesis. Each of the calibration procedures will have a simulation phase and an experimental phase. The details of each

kinematic identification process will be described in each section of the appropriate calibration procedure. Refer to the preceeding paragraph for a general explanation of the kinematic parameter identification procedure.

III. FULL POSE CALIBRATION

A. THEORY

1. Introduction

As stated earlier, the nominal coordinate transformations from the base frame to frame 5 will not change, regardless of the type of calibration process used or the type of measurement system used in the calibration. The transformation from the world coordinate frame to the base frame is an Euler transform, as well as the transform from frame 5 to frame 6, the tool frame.

The transformation from the world coordinate frame to frame 1 of the manipulator needs to be carefully considered, since there are potential parameter dependencies if certain transforms are chosen. Consider Figure 19 which shows the world coordinate frame x_w,y_w,z_w, frame x_0,y_0,z_0, frame x_b,y_b,z_b, and frame x_1,y_1,z_1.

Frame x_0,y_0,z_0 is defined by a DH transform from the world frame to the first joint axis of the manipulator, frame x_b,y_b,z_b is the PUMA manufacturer's base frame and frame x_1,y_1,z_1 is the second DH frame of the manipulator. What are the minimum number of parameters that are required to move from the world frame to frame x_1,y_1,z_1? There are two transformation paths which will accomplish this transform [Ref. 9].

Path 1: A DH transform from the world coordinate frame to frame x_0,y_0,z_0 involving four parameters followed by another DH from frame x_0,y_0,z_0 to frame x_b,y_b,z_b which will involve only two parameters ϕ' and d' in the transform:

$$\mathbf{T}_0^b = \mathrm{rot}(z_0,\phi')\mathrm{trans}(z_0,d') \tag{30}$$

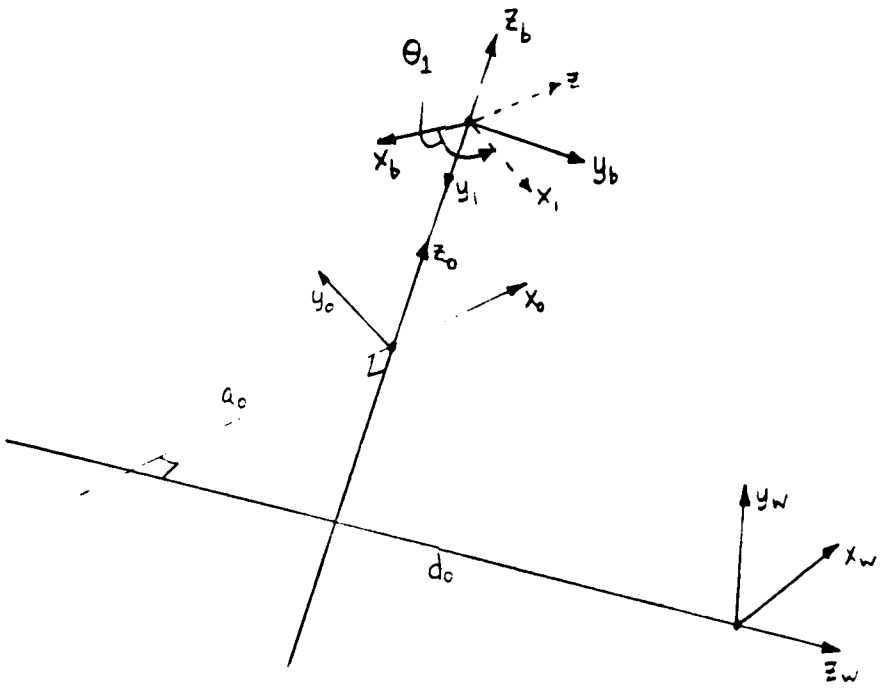


Figure 19. Base Transformations

Another DH transform from x_b,y_b,z_b to x_1,y_1,z_1 which involves four parameters. Note that $\Delta\theta_1$ and Φ' both rotate about z_0. That means that a rotation $\Delta\theta_1$ about the z_0-axis is indiscernible. Any rotation about the z_0-axis can be accomplished using an infinite number of combinations of values of $\Delta\theta_1$ and Φ', which means that $\Delta\theta_1$ and Φ' can not be identified independently. This leads to a conclusion that eight independent kinematic parameters are required to move the coordinate frame from the world frame to the first frame of the PUMA using this path.

Path 2: A transform can be defined directly from the world coordinate frame x_w,y_w,z_w, to the base frame x_b,y_b,z_b. This is a frame to frame transform so an Euler transform is required:

$$A_b = \text{rot}(z,\phi_b)\text{rot}(y,\theta_b)\text{rot}(x,\varphi_b)\text{trans}(p_{xb},p_{yb},p_{zb}) \quad (31)$$

The DH transform from x_b,y_b,z_b to x_1,y_1,z_1 which will follow would normally involve four parameters, but there is another dependency involving $\Delta\theta_1$ and Δd_1. $\Delta\theta_1$ can be resolved into $\phi_b,\theta_b,\varphi_b$ and Δd_1 can be resolved into (p_{xb},p_{yb},p_{zb}). This reduces the number of parameters required for the transform from the base frame to frame 1 from four to two. Coupled with the six parameters required to transform from the world frame to the base frame it is seen again that eight parameters are required to transform from the world frame to frame 1, but a different set of parameters than found in path 1. In this simulation the second path is chosen.

The tool transform is an Euler transform, requiring the specification of six parameters:

$$A_6 = \text{rot}(z,\phi_6)\text{rot}(y,\theta_6)\text{rot}(x,\varphi_6)\text{trans}(p_{x6},p_{y6},p_{z6}) \quad (32)$$

2. Full Model of the PUMA, World Coordinate Frame, and Coordinate Measuring Machine

Figure 20 shows the full pose calibration apparatus. The operation of the coordinate measuring machine will be explained in the experimental section of this chapter.

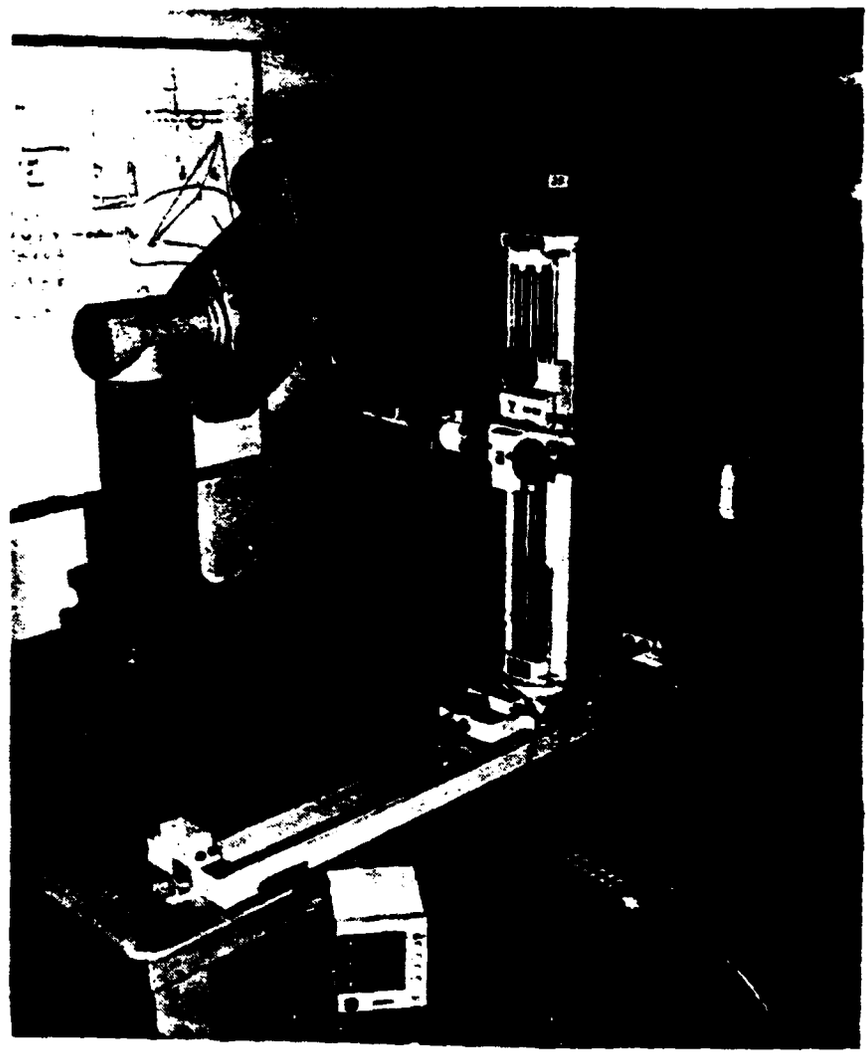


Figure 20. Full Pose Calibration Apparatus: PUMA, World Coordinate Frame and Coordinate Measuring Machine

Table 2 represents the kinematic parameters of the full pose model [Ref. 10]. In the full pose model all thirty parameters not previously defined to be zero, indicated in boldface type, are able to be determined. Note the transform from the world coordinate frame to the base frame of the PUMA.

TABLE 2. NOMINAL KINEMATIC PARAMETERS FOR THE FULL POSE CALIBRATION

φ_b	θ_b	φ_b	pxb	pyb	pzb
degrees	degrees	degrees	mm	mm	mm
180.0	0.0	90.0	-394.0	-383.0	474.0

link number	$\Delta\theta_i$	d_i	a_i	α_i	β_i
	degrees	mm	mm	degrees	degrees
1	**0**	**0**	0.0	-90.0	**0**
2	0.0	**0**	431.85	0.0	0.0
3	0.0	149.09	-20.33	90.0	**0**
4	0.0	433.00	0.0	-90.0	**0**
5	0.0	0.0	0.0	90.0	**0**

φ_6	θ_6	φ_6	px6	py6	pz6
degrees	degrees	degrees	mm	mm	mm
90.0	0.0	0.0	0.0	0.0	134.0

B. SIMULATION

1. The Suite of Programs Used and the Strategy Involved

In performing a simulation study of a proposed experimental calibration procedure the objectives are to:

1. Confirm that the numerical algorithm proposed for the identification converges to the correct values.

2. Predict the number of experimental poses required to identify the kinematic parameters to a defined degree of accuracy.

3. Estimate the resulting accuracy of the manipulator if the new kinematic model was embedded in the control software of the manipulator [Ref. 11].

Several computer programs were written to perform the simulation. These programs, and their relationships to each other are shown in Figure 21.

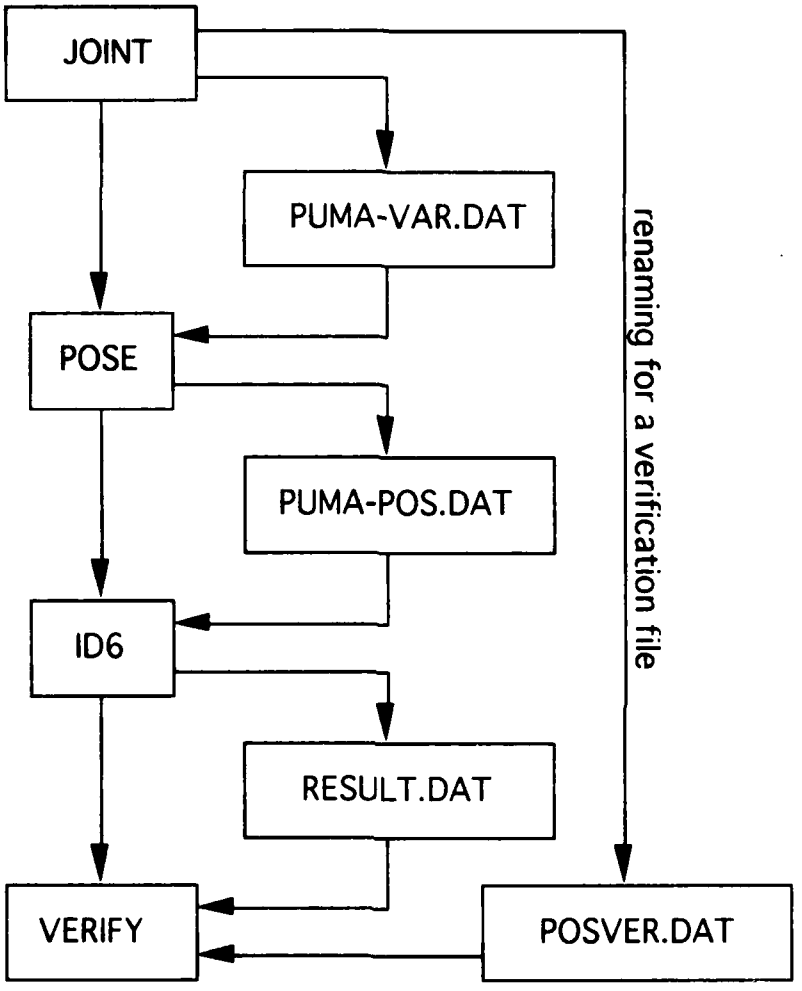


Figure 21. Flowchart for Full Pose Simulation

The complete simulation may be regarded as a tool to plan the experiment in which the independent variables are the number of observations and the range of joint angles allowed by the common work volume of the PUMA and the coordinate measuring machine, while the dependent variables are the accuracy of the parameter identification and

the resulting manipulator accuracy [Ref. 12]. A detailed explanation of each program follows.

a. The program JOINT

Program JOINT does not require an input data file.

Program JOINT generates sets of six joint angles which comprise a simulated pose of the PUMA. The program can be modified such that the simulated working volume of the PUMA is restricted from the maximum possible to either one-half of one-quarter of the maximum volume.

The joint angles are selected using a random number generator. For instance, if the working volume of the PUMA has been selected as the maximum possible, the program takes the difference between the extreme joint angles possible, +180-(-180)=360, then multiplies this value by a number between 0 and 1, which has been generated randomly, thus generating one joint angle. Since six independent joint angles are required, six different random numbers and six different joint angles are generated for each pose, or observation.

The output of program JOINT is a data file, PUMA-VAR.DAT. PUMA-VAR.DAT will have n sets of six joint angles, which comprise n simulated poses of the PUMA. In the verification phase program JOINT is run again, with the output data file being renamed POSEVER.DAT for use in program VERIFY.

b. The program POSE

Program POSE requires two input data files, PUMA-VAR.DAT, the output data file from program JOINT, and INPUT.DAT, the nominal kinematic parameters of the PUMA, listed in Table 1. POSE also inputs values "dangle" and "dlenth." The value assigned to "dangle" is added to all of the angular parameters except for the ones which are defined to be zero. The value assigned to "dlenth" is added to all of the length

parameters. Adding these values creates a manipulator which is significantly different from the manipulator reflected in the nominal kinematic parameter table, and is equivalent to supplying the kinematic identification program with pseudo-experimental data. Creating this different manipulator tests the integrity of the kinematic parameter identification program. Since the initial guess of the parameters will be the nominal parameters the identification program must rigorously solve for the actual kinematic parameters, which will be the nominal parameters plus "dangle" and "dlenth" as appropriate.

Program POSE generates a T_6 matrix, representing the simulated position and orientation of the tool frame of the PUMA, for each set of joint angles which were generated by program JOINT.

When the joint angles are read from PUMA-VAR.DAT they are added to the parameters read from the nominal kinematic table which apply to the individual joint. For example, in POSE the variable TH1 equals the sum of DT1 and THETA1. DT1 is input from the nominal kinematic table and THETA1 has been generated by JOINT. The nominal kinematic parameters are added to the simulated actual joint angles as any non-zero THETA value at any joint represents a movement from the zero position of that joint.

Once the joint angles are calculated and the nominal kinematic parameters are adjusted, a forward kinematic solution is calculated. This will produce one T_6 matrix for each set of joint angles. In order to accurately model an actual measurement in the laboratory noise is added to the position vector of the T_6 matrix and the six joint angles of each simulated pose. The position noise arises from the uncertainty of the precision with which the coordinate measuring machine and associated software is capable of identifying the position of the center of a tooling ball. The position noise is generated by multiplying a random number which is generated in the same manner as in JOINT by a number read from INPUT.DAT, "magnx." The joint angle noise, called the encoder error, is generated by

multiplying a random number by another value which is read from INPUT.DAT, "magn1." The position noise is 100 times the magnitude of the encoder noise.

The output of POSE will be the six joint angles for each simulated pose and the corresponding T_6 matrix for each set of joint angles. This data is stored in the output file PUMA-POS.DAT.

c. The program ID6

Program ID6 requires two input files, INPUT.DAT and PUMA-POS.DAT. The nominal kinematic parameters are read from INPUT.DAT in order to provide the kinematic parameter identification with a beginning point. The T_6 matrices contained in PUMA-POS.DAT which were generated by program POSE contain the simulated poses with which the actual kinematic parameters will be identified in this program.

ZXSSQ will perform the kinematic parameter identification. The function to be minimized by ZXSSQ is actually an array of six functions. Each pose will have six functions to be minimized, so if there are ten poses, or observations, ZXSSQ will have sixty functions to minimize using thirty variables.

A subroutine within ZXSSQ, PUMA_ARM, calculates the T_6 matrix corresponding to the current kinematic parameters for each set of six joint angles input from PUMA-POS.DAT. The simulated measured T_6 matrix read from PUMA-POS.DAT for each set of joint angles is known. Since the original nominal kinematic parameters were altered before the simulated measured T_6 matrix was calculated, the simulated measured T_6 matrix will be different from the T_6 matrix calculated using the generated joint angles and the nominal kinematic parameters. Also, the added measurement noise is present, providing more of a difference in the two matrices. For each observation the array of six F functions is calculated. After performing the matrix subtraction:

$$\Delta T = T_{measured} - T_{calculated} \tag{33}$$

the six F functions are determined for each observation:

$$\begin{aligned} F_1 &= \Delta T(1,4) \\ F_2 &= \Delta T(2,4) \\ F_3 &= \Delta T(3,4) \\ F_4 &= \frac{(\Delta T(3,2) - \Delta T(2,3))}{2} \times \text{scaling factor} \\ F_5 &= \frac{(\Delta T(1,3) - \Delta T(3,1))}{2} \times \text{scaling factor} \\ F_6 &= \frac{(\Delta T(2,1) - \Delta T(1,2))}{2} \times \text{scaling factor} \end{aligned} \tag{34}$$

The scaling factor which multiplies the functions in equation (34) is used to remove the emphasis from the position error. The position error in a pose will be much larger than the orientation error between two T_6 matrices. For this reason the emphasis in the optimization program will be on eliminating the position error, while largely ignoring the orientation error. This is unacceptable, since the orientation error is equally important as the position error. The scaling factor places the orientation error at the same level of magnitude as the position error.

The F functions in equation (34) represent the error matrix [Ref. 13]:

$$\Delta = \begin{bmatrix} 0 & -\delta_z & \delta_y & d_x \\ \delta_z & 0 & -\delta_x & d_y \\ -\delta_y & \delta_x & 0 & d_z \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{35}$$

It must be noted that equation (35) can only be used when the two matrices being subtracted are almost equal. In an actual experimental process the measured poses usually significantly differ from the predicted pose. This difference causes the Δ

matrix, equation 35, to have relatively large elements. ZXSSQ is often unable to converge under these conditions. The different version of ID6 will be introduced in the Experimental section of this chapter.

After all the observations have been calculated for each pose, ZXSSQ compares each F value with a user defined convergence criteria. Convergence is defined when the kinematic parameters selected from one iteration to the next agree to four significant digits. If every F for each pose is less than this convergence criteria, then the current kinematic parameters are saved as the correct parameters. If not, ZXSSQ changes the thirty kinematic parameters and performs the process again until the convergence criteria is satisfied.

The output of ID6 is RESULT.DAT. The identified kinematic parameters will be stored in the same format as INPUT.DAT. In the simulation the identified kinematic parameters will be the parameters of INPUT.DAT plus dangle for orientation parameters and dlenth for length parameters plus some small error value created by the addition of the simulated measurement noise.

d. The program VERIFY

Program VERIFY requires three input programs. The first, POSEVER.DAT, is the output from program JOINT, and includes sets of six joint angles. The other two are kinematic parameters, INPUT.DAT and RESULT.DAT. INPUT.DAT, as explained earlier, contains the nominal kinematic parameters, and RESULT.DAT contains the kinematic parameters identified in program ID6.

VERIFY measures the position and orientation difference between poses calculated using the nominal kinematic parameters and the nominal kinematic parameters. For each set of joint angles, VERIFY calculates a forward kinematic solution using both the nominal and identified kinematic parameters and takes the difference between the

corresponding T_6 matrices. The position error and orientation error for all of the sets of joint angles are added and these values are divided by the total number of sets of joint angles. This will produce the total position error and total orientation error for a given number of observations.

C. EXPERIMENTATION

1. The Tooling Ball End Effector

The tooling ball end effector is illustrated in Figure 22. Its' purpose is to provide a means for determining the position and orientation of frame 6 of the PUMA. The five balls attached to the effector are precision tooled to a radius of 6.35 mm. This known radius, along with the known radius of the touch probe of the coordinate measuring machine provides the essential information required for identification of the center of each tooling ball by the routine BALL described earlier in the thesis. Also, each of the four tooling balls contained in the plane normal to the axis of the effector are nominally 90 degrees apart, creating orthogonality between any two in series around the circumference of the tool. The fifth tooling ball's center lies on an axis which passes the two lines which attach balls on opposite sides of the tool, creating another orthogonality between the fifth ball and each of the other four. The orientation of frame six is fixed on the tool and is required information for the determination of the position and orientation of the tool frame by program CMMPOSE.

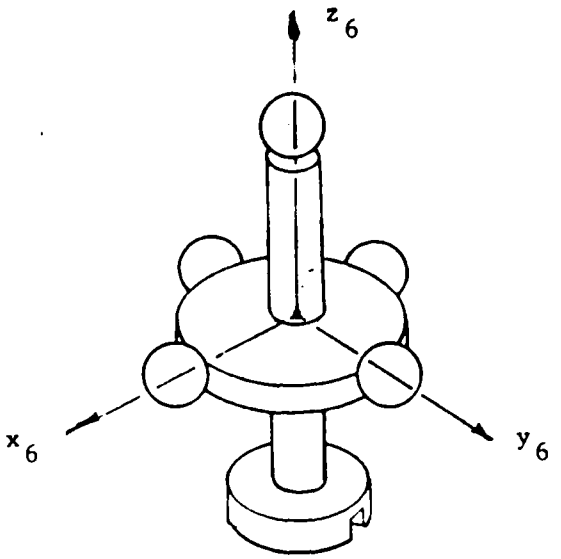


Figure 22. Tooling Ball End Effector

2. The Coordinate Measuring Machine and World Coordinate Frame

a. Construction

The coordinate measuring machine (CMM) used for data acquisition in this thesis is illustrated in Figure 23.

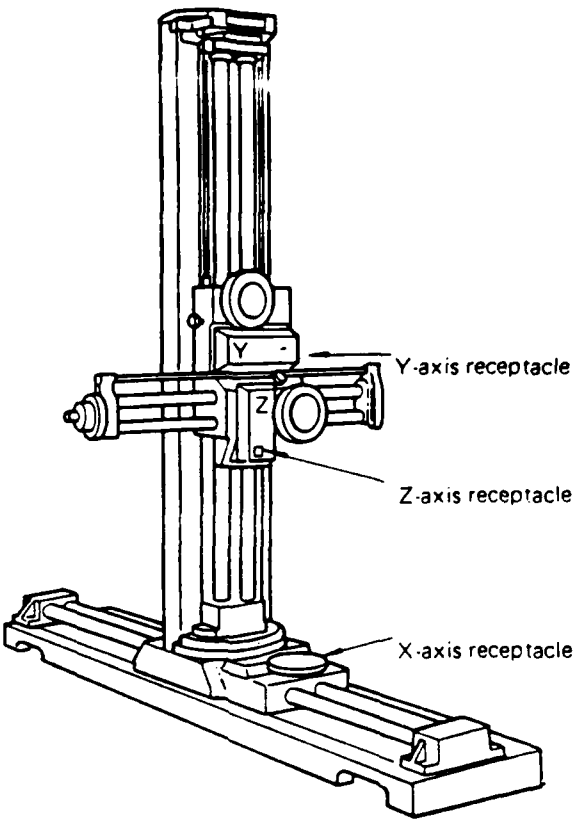


Figure 23. Coordinate Measuring Machine

The x-axis lies on the base of the CMM. Rotating the x wheel will move the x-axis receptacle to the left or right, changing the x-value displayed. The y-axis is contained within a post which is attached to the slide that changes the x position. Rotating the y wheel moves the y-axis receptacle up or down, changing the y-value displayed. The z-axis is contained is the y-axis receptacle assembly. Rotating the z wheel moves the z receptacle in or out, changing the z value displayed. Consider the point (1,1,1) relative to a known world reference frame. Rotating the x wheel will produce a change in position (Δx,1,1). Rotating the y wheel will produce a change in position (1,Δy,1). Rotating the z

wheel will produce a change in position (1,1,Δz). It is clear that any point in the working volume of the CMM may be reached by the touch probe.

A significant advantage of this CMM is its' ability to zero at any position. Each axis may be zeroed independently or concurrently. This is important in identifying the location of the world coordinate frame. A diagram of the world coordinate frame used in the full pose calibration is shown in Figure 24.

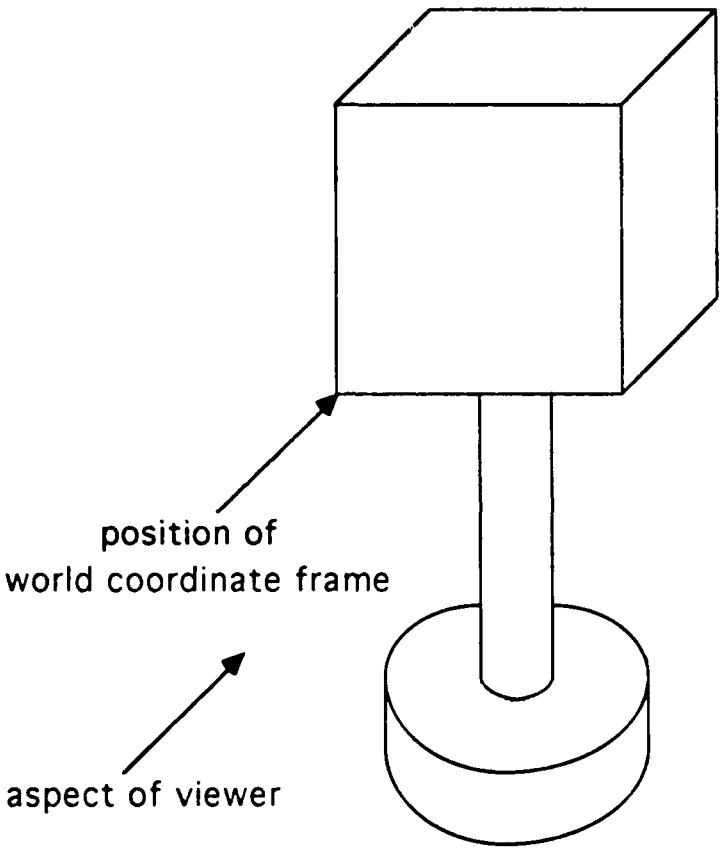


Figure 24. The Full Pose World Coordinate Frame Cube

The cube itself does not actually contain the world coordinate frame. Using the three faces of the cube which form the corner of the bottom of the cube closest to and most left of the operator, also identified in Figure 24, allow for a procedure which

identifies point (0,0,0) of the world coordinate frame. Once this point is identified, the axes of the world coordinate frame match the axes of the CMM. The procedure for identifying the world coordinate frame is as follows. Note that the order of the identification process of the world coordinate frame is arbitrary.

1. The touch probe is positioned in area near the bottom, left, forward face of the cube.

2. Activate the 'x-zero' mode of the CMM. Move, using all three axes if necessary, the touch probe such that its' position is as close to the forward corner of the bottom of the left face as possible. Slowly move the probe to the right, using the x-axis wheel only, until contact is made with the forward face of the cube. When contact is made the display unit (DU) will beep and the x position will indicate zero.

3. Activate the 'y-zero' mode of the CMM. Move, using all three axes if necessary, the touch probe such that its' position is as close to the forward, left corner of the bottom face of the cube as possible. Slowly move the probe upwards, using the y-axis wheel only, until contact is made with the bottom of the cube. When contact is made the DU will beep and the y position will indicate zero.

4. Activate the 'z-zero' mode of the CMM. Move, using all three axes if necessary, the touch probe such that its' position is as close to the bottom, left corner of the forward face of the cube as possible. Slowly move the probe towards the cube, using the z-axis wheel only, until contact is made with the forward face of the cube. When contact is made the DU will beep and the y position will indicate zero.

Once this procedure is completed a point in space located a distance equal to the diameter of the touch probe along each axis, $(|x_{dia_{tip}}|,|y_{dia_{tip}}|,|z_{dia_{tip}}|)$, will be identified as the location of the world coordinate frame. This process is easily repeatable and has a high

degree of accuracy, so for different sets of experimental measurements the position of the world coordinate frame should not vary to a significant degree.

b. Data Acquisition Using the Coordinate Measuring Machine

Data acquisition using the coordinate measuring machine consists of a repetitive measurement of the position of three different tooling balls on the end effector illustrated in Figure 22. The PUMA is operated in joint mode, which allows each of the six joints to be rotated independently. The end effector, which is attached to the tooling ball assembly, is moved to a variety of positions. When the operator is satisfied with the position of the tool, the measurement process begins.

Execution of the pose measurement program, CMMPOSE, begins by inquiring which tooling ball on the tool is being measured. This information is required in order to determine the orientation of frame 6. The touch probe of the CMM is moved slowly towards the selected tooling ball. As the probe touches the surface of the ball the display unit of the CMM beeps. The position displayed by the unit is the position of the probe at the first instant that the probe touches the surface of the ball, and is displayed in millimeters to two significant digits. If the probe slightly slides after initial contact the displayed position will not change, remaining until one of the three axes of the CMM is moved. CMMPOSE asks for three touch probe measurement sets per ball and determines the center of each tooling ball using the procedure described in section II.B.2. Once the center of each ball is calculated a residual error reflecting the uncertainty of the accuracy of the predicted position of the center of the ball is displayed. An option of storing or purging the data is given at this point. Based on the judgement of the operator, the data is sent for further manipulation by CMMPOSE, or the data is purged and another set of ball measurements is taken. An acceptable residual error has a value near 0.000001 mm. Once the center of three of any of the five tooling balls, P_1, P_2, and P_3, is determined, the

program synthesizes the position of the center of a fourth ball, P_4, by calculating the vector cross product [Ref. 14]:

$$P_4 = (P_3 - P_1) \times (P_2 - P_1) \tag{36}$$

The relationship between the coordinates of the center of each ball expressed in terms of the tool frame and the world coordinate frame is

$$[\mathbf{P}_i^*] = [\mathbf{A}][\mathbf{P}_i] \tag{37}$$

where $\mathbf{P}_i^*$ is the 4x1 column vector of the coordinates of the i^{th} ball expressed with respect to the world frame, $\mathbf{P}_i$ is the 4x1 column vector of the coordinates of the i^{th} ball with respect to the tool frame, and $\mathbf{A}$ is the 4x4 homogeneous transformation matrix from the world frame to the tool frame.

If $\mathbf{P}_i$ is known by precalibrating the tool, and $\mathbf{P}_i^*$ is measured then $\mathbf{A}$ may be calculated and used as the measured pose in the calibration process. The difficulty arises in inverting equation (37) to obtain $\mathbf{A}$. The following information is known:

$$\begin{aligned} [\mathbf{P}_1^*] &= [\mathbf{A}][\mathbf{P}_1] \\ [\mathbf{P}_2^*] &= [\mathbf{A}][\mathbf{P}_2] \\ [\mathbf{P}_3^*] &= [\mathbf{A}][\mathbf{P}_3] \\ [\mathbf{P}_4^*] &= [\mathbf{A}][\mathbf{P}_4] \end{aligned} \tag{38}$$

Algebraic manipulation produces:

$$[\mathbf{P}_1^*][\mathbf{P}_2^*][\mathbf{P}_3^*][\mathbf{P}_4^*] = [\mathbf{A}][\mathbf{P}_1][\mathbf{P}_2][\mathbf{P}_3][\mathbf{P}_4] \tag{39}$$

reducing:

$$[\mathbf{P}^*] = [\mathbf{A}][\mathbf{P}] \tag{40}$$

Since $\mathbf{P}^*$, **A**, and **P** are all 4x4 matrices, equation (40) can be inverted, producing matrix **A**, the pose of the tool:

$$[\mathbf{A}] = [\mathbf{P}^*][\mathbf{P}^{-1}] \tag{41}$$

Two operators can expect to measure one pose in ten minutes. One operator will be utilized to move the PUMA and operate the CMM and the other operator will input data as instructed by program CMMPOSE.

c. Identification of Actual Kinematic Parameters

A modified version of program ID6 is used for the identification of the actual kinematic parameters of the PUMA. The modification is in the functions which are minimized by the IMSL routine ZXSSQ. As mentioned earlier, the delta matrix of equation (35) is suitable for use only when the predicted pose and the measured pose are quite close. Experimentation will usually produce matrices which are dissimilar. For this reason, each element of the matrix resulting from the difference between the predicted and measured pose is minimized:

$$\begin{bmatrix} F_1 & F_4 & F_7 & F_{10} \\ F_2 & F_5 & F_8 & F_{11} \\ F_3 & F_6 & F_9 & F_{12} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}_{\text{predicted}} - \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}_{\text{measured}} \tag{42}$$

This produces twelve functions which are to be minimized. This method provides a more rigorous method for the identification of the kinematic parameters. Table 3 shows the kinematic parameters identified using the full pose calibration procedure [Ref. 15].

TABLE 3. FULL POSE IDENTIFIED PARAMETERS

Parameter	Nominal Value	Identified Value
φ_b	180.0	179.9579
θ_b	0.0	1.5120
φ_b	90.0	89.0219
p_{xb}	-394.0	-393.9838
p_{yb}	-383.0	-405.0608
p_{zb}	474.0	466.8381
a_1	0.0	-0.04923
α_1	-90.0	-89.9977
$\delta\theta_2$	0.0	-0.4888
a_2	431.9	432.1216
α_2	0.0	-0.0303
β_2	0.0	-0.01515
$\delta\theta_3$	0.0	-1.2069
d_3	149.1	149.1455
a_3	-20.3	-19.2270
α_3	90.0	90.0512
$\delta\theta_4$	0.0	-0.9144
d_4	433.0	432.8899
a_4	0.0	0.0040
α_4	-90.0	-89.9909
$\delta\theta_5$	0.0	2.2364
d_5	0.0	-0.6629
a_5	0.0	-0.0258
α_5	90.0	89.9345
φ_6	90.0	91.2400
θ_6	0.0	-0.0979
φ_6	0.0	-0.0575
p_{x6}	0.0	0.1863
p_{y6}	0.0	-0.2329
p_{z6}	134.0	133.1557

IV. PARTIAL POSE CALIBRATION

A. THEORY

1. Introduction

As stated earlier, the nominal coordinate transformations from the base frame of the manipulator to frame 5 will not change. These transformations are inherent to the manipulator and are independent of the pose measurement scheme being employed. The transformations from the world coordinate frame to the base frame and from frame 5 to frame 6 are the same transformations as employed in the full pose calibration.

The world coordinate frame is not in the same position, and is not defined in the same manner, as it was in the full pose calibration. The partial pose calibration measurement system uses a linear slide to which the tool frame of the manipulator is attached. The only variable which is measured is the position of the slide relative to a zero position defined at the beginning of each pose measurement session. The world frame is defined as the T_6 matrix of the tool when the slide's position on the slide equals zero. The slide acts as a prismatic joint and hence is defined only in direction. The location of the axis is undefined. The three translational components of the transform from frame five to frame six, T_5^6, are arbitrary and may be set equal to zero. This defines that the origins or frame five and six are coincident, as shown in Figure 25. This will reduce the full thirty parameter model to twenty-seven parameters.

The world coordinate frame's orientation is not known, either. The direction of one axis is able to be defined. In this experiment the axis of the linear slide has been defined as the x axis. The y and z axes of the world coordinate frame are orthogonal to the

x-axis, but their orientation relative to the x-axis is undefinable, arbitrary, and therefore set equal to zero. This further reduces the model to twenty-six parameters.

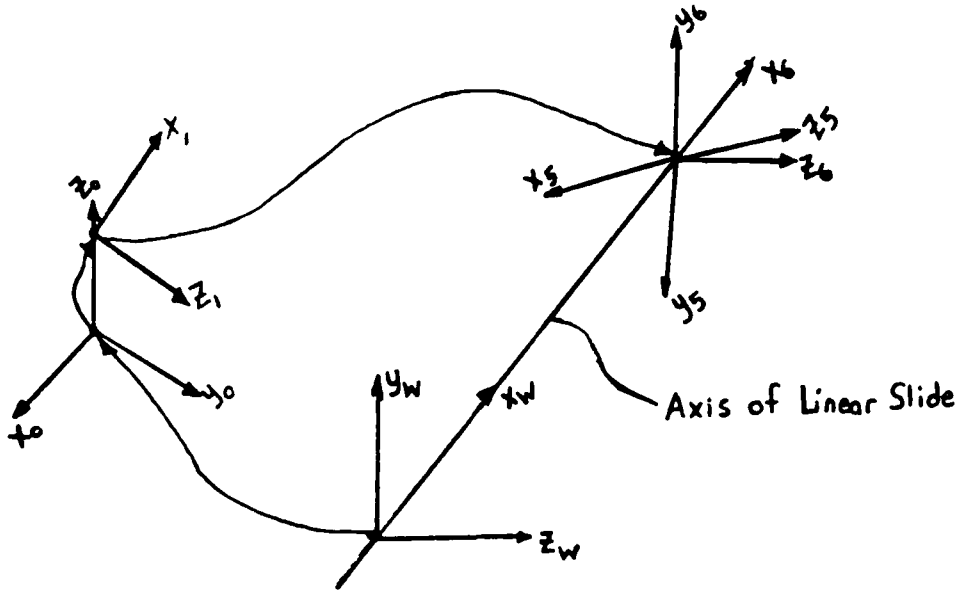


Figure 25. Linear Slide Transformations

The partial pose calibration model dictates that the tool frame pose is invariant. Each pose will have the following orientation:

$$T_6 = \begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{43}$$

This T_6 matrix defines the x-axis of the tool frame to be in the same direction as the x-axis of the world coordinate frame, the y-axis of the tool frame to be in the same direction as the z-axis of the world coordinate frame, and the z-axis of the tool frame to be in the same direction as the negative y-axis of the world coordinate frame. Also, the

position of the origin of frame six is at position (X,0,0), where X is the position of the linear slide. Note that it is still not possible to uniquely define the orientation of the world coordinate frame with respect to any frame on the PUMA. It is only known that the orientation of the world coordinate frame is consistently defined with respect to the orientation of the tool frame.

2. Full Model of the Puma and the Linear Slide

Figure 26 shows the partial pose calibration apparatus. The operation of the coordinate measuring machine as a linear slide will be explained in the experimental section of this chapter.

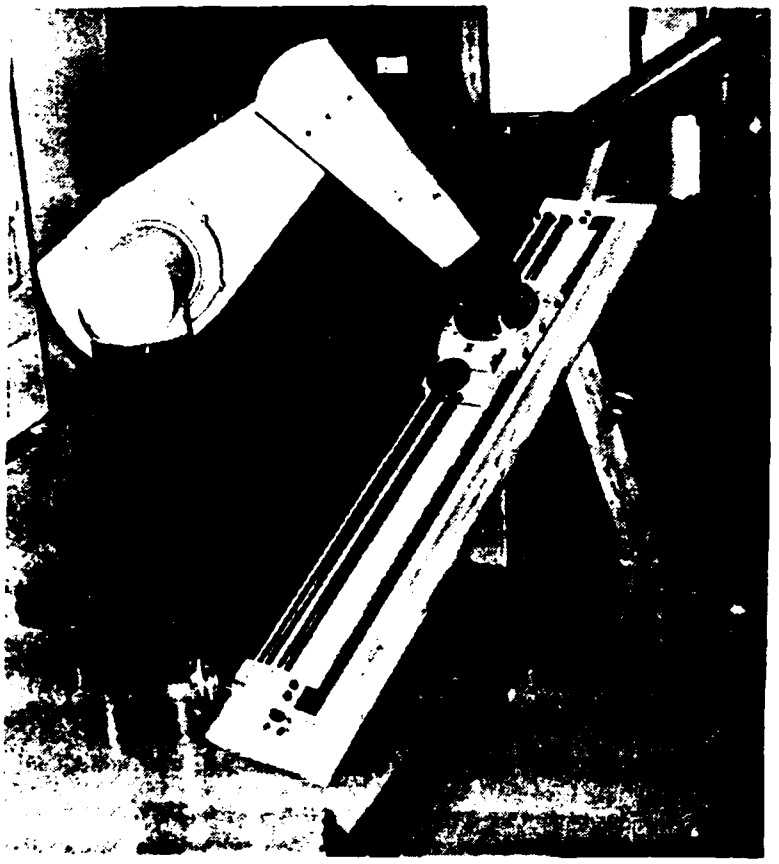


Figure 26. Partial Pose Calibration Apparatus: PUMA and Linear Slide

Table 4 displays the kinematic parameters of the partial pose model. In the partial pose model twenty-six parameters are able to be determined. The parameters in **boldface type are unable to be experimentally determined consequently their value is** defined as equaling zero, or, in the case of the four parameters in the sixth transformation, **the parameters are not independent, as explained in the previous section.** Note the transform from the world coordinate frame to the base frame of the PUMA has changed from the full pose calibration. The world coordinate frame is not at the corner of the cube, as it was in the full pose calibration. The nominal kinematic parameters within the PUMA do not change, regardless of the type model used in the calibration process.

TABLE 4. NOMINAL KINEMATIC PARAMETERS FOR THE PARTIAL POSE CALIBRATION

φ_b	θ_b	φ_b	p_{xb}	p_{yb}	p_{zb}
degrees	degrees	degrees	mm	mm	mm
151.0	-20.0	90.0	-203.0	-254.0	457.2

link number	$\Delta\theta_i$	d_i	a_i	α_i	β_i
	degrees	mm	mm	degrees	degrees
1	**0**	**0**	0.0	-90.0	**0**
2	0.0	**0**	431.85	0.0	0.0
3	0.0	149.09	-20.33	90.0	**0**
4	0.0	433.00	0.0	-90.0	**0**
5	0.0	0.0	0.0	90.0	**0**

φ_6	θ_6	φ_6	p_{x6}	p_{y6}	p_{z6}
degrees	degrees	degrees	mm	mm	mm
90.0	0.0	**0.0**	**0.0**	**0.0**	**0.0**

B. SIMULATION

1. The Suite of Programs Used and the Strategy Involved

As stated earlier, the simulation study is performed to:

1. Confirm that the numerical algorithm proposed for the identification converges to the correct values.

2. Predict the number of experimental poses required to identify the kinematic parameters to a defined degree of accuracy.

3. Estimate the resulting accuracy of the manipulator if the new kinematic model was embedded in the control software of the manipulator.

In the partial pose calibration, the forementioned "correct" values are the kinematic parameters identified using the full pose method. The computer programs written, and their relationships, are illustrated in Figure 27.

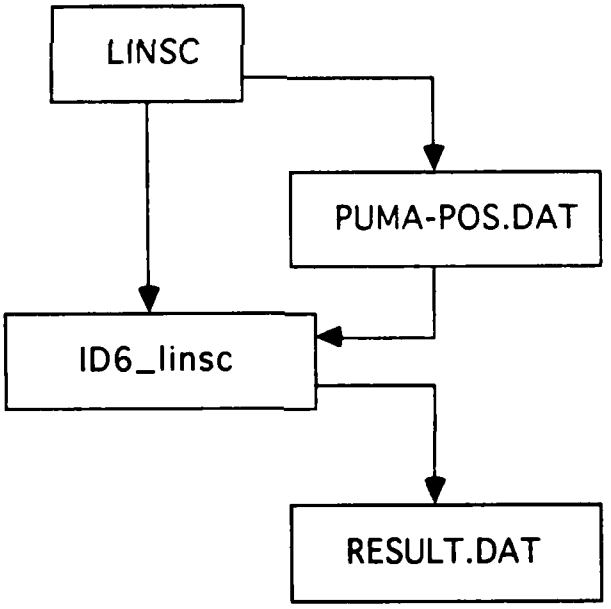


Figure 27. Flowchart for Partial Pose Simulation

The complete simulation may be regarded as a tool to plan the experiment in which the independent variables are the number of observations and the range of joint angles allowed by the common work volume of the PUMA and the linear slide, while the dependent variables are the accuracy of the parameter identification and the resulting manipulator accuracy. A detailed explanation of each program follows.

a. The program LINSC

Program LINSC requires the input file INPUT.DAT, the nominal kinematic data for the PUMA, which are listed in Table 1. LINSC also inputs values "dangle" and "dlenth," as did program POSE in the simulation phase of the full pose calibration. It will be instructive to reexamine the function of these two parameters. The value assigned to "dangle" is added to all of the angular parameters except for the ones which are defined to be zero. The value assigned to "dlenth" is added to all of the length parameters. Adding these values creates a manipulator which is significantly different from the manipulator reflected in the nominal kinematic parameter table, and is equivalent to supplying the kinematic identification program with pseudo-experimental data. Creating this different manipulator tests the integrity of the kinematic parameter identification program. Since the initial guess of the parameters will be the nominal parameters the identification program must rigorously solve for the actual kinematic parameters, which will be the nominal parameters plus "dangle" and "dlenth" as appropriate.

Program LINSC generates six joint angles which will produce a $\mathbf{T}_6$ matrix with an orientation that is predefined and fixed and a position on the slide which has been randomly generated. This is done by performing an inverse kinematic solution, since the $\mathbf{T}_6$ matrix is known. The set of joint angles, when used with the nominal kinematic parameters of the PUMA, which will produce the required $\mathbf{T}_6$ matrix is the required data. The IMSL routine ZXSSQ is again used to make this determination.

ZXSSQ uses an initial guess of all six joint angles equaling zero. A forward kinematic solution is calculated based on these joint angles and the nominal kinematic parameters plus dangle and dlenth. The difference between each element of the calculated $\mathbf{T}_6$ matrix and the required $\mathbf{T}_6$ matrix is calculated. These elements represent the functions to be minimized:

$$\begin{bmatrix} F_1 & F_4 & F_7 & F_{10} \\ F_2 & F_5 & F_8 & F_{11} \\ F_3 & F_6 & F_9 & F_{12} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}_{predicted} - \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}_{measured} \quad (44)$$

ZXSSQ modifies the joint angle set until a convergence criteria is satisfied. At convergence the required $\mathbf{T}_6$ matrix and the six joint angles are returned to the main program.

It has been stated that in order to accurately model an actual measurement in the laboratory noise must be added to the simulated $\mathbf{T}_6$ matrix and the six joint angles of each pose. The position noise and encoder noise is generated in the same manner as in program POSE in the full pose simulation. The position noise in the partial pose simulation arises from the uncertainty in the measurement of the position of the slide. The orientation error arises from inconsistencies in the bar supporting the slide. The bar almost certainly will have elements of twist and bend in it. These elements will cause differences in orientation as the slide moves along its' length.

After the noise is added to the $\mathbf{T}_6$ matrix and the joint angles, the data is stored in the output file PUMA-POS.DAT.

b. The program ID6_LINSC

Program ID6_linsc requires two input files, INPUT.DAT and PUMA-POS.DAT. The nominal kinematic parameters are read from INPUT.DAT in order to

provide the kinematic parameter identification with a beginning point. The T_6 matrices and associated joint angles contained in PUMA-POS.DAT which were generated by program LINSC contain the simulated poses with which the actual kinematic parameters will be identified in this program.

ZXSSQ will perform the kinematic parameter identification. Each element of the matrix resulting from the difference between the predicted and measured pose is minimized, as was done during the actual kinematic parameter identification of the full pose calibration method. See equation (43) for the equation of the functions to be minimized.

This produces twelve functions which are to be minimized. Subroutine PUMA_ARM calculates the T_6 matrix corresponding to the current kinematic parameters for each set of six joint angles input from PUMA-POS.DAT. The simulated measured T_6 matrix read from PUMA-POS.DAT for each set of joint angles is known. Since the original nominal kinematic parameters were altered before the simulated measured T_6 matrix was calculated, the simulated measured T_6 matrix will be different from the T_6 matrix calculated using the generated joint angles and the nominal kinematic parameters. Also, the added measurement noise is present, providing more of a difference in the two matrices.

After all the observations have been calculated for each pose, ZXSSQ compares each F value with a user defined convergence criteria. Convergence is defined when the kinematic parameters selected from one iteration to the next agree to four significant digits. If every F for each pose is less than this convergence criteria, then the current kinematic parameters are saved as the correct parameters. If not, ZXSSQ changes the twenty-six available kinematic parameters and repeats the process until the convergence criteria is satisfied.

The output of ID6_linsc is RESULT.DAT. The identified kinematic parameters will be stored in the same format as INPUT.DAT. In the simulation the identified kinematic parameters will be the parameters of INPUT.DAT plus dangle for orientation parameters and dlenth for length parameters plus some small error value created by the addition of the simulated measurement noise.

C. EXPERIMENTATION

1. The Coordinate Measuring Machine as a Linear Slide

a. Construction

The coordinate measuring machine's base, which supports the x-axis, is mounted on an incline, as illustrated in Figure 28. The incline is placed on the table on which the PUMA is mounted, in an orientation which is significantly different from the x-axis of the PUMA. This position, concurrent with the inclined mounting, provides for the maximum possible joint rotation of all six joints of the PUMA.

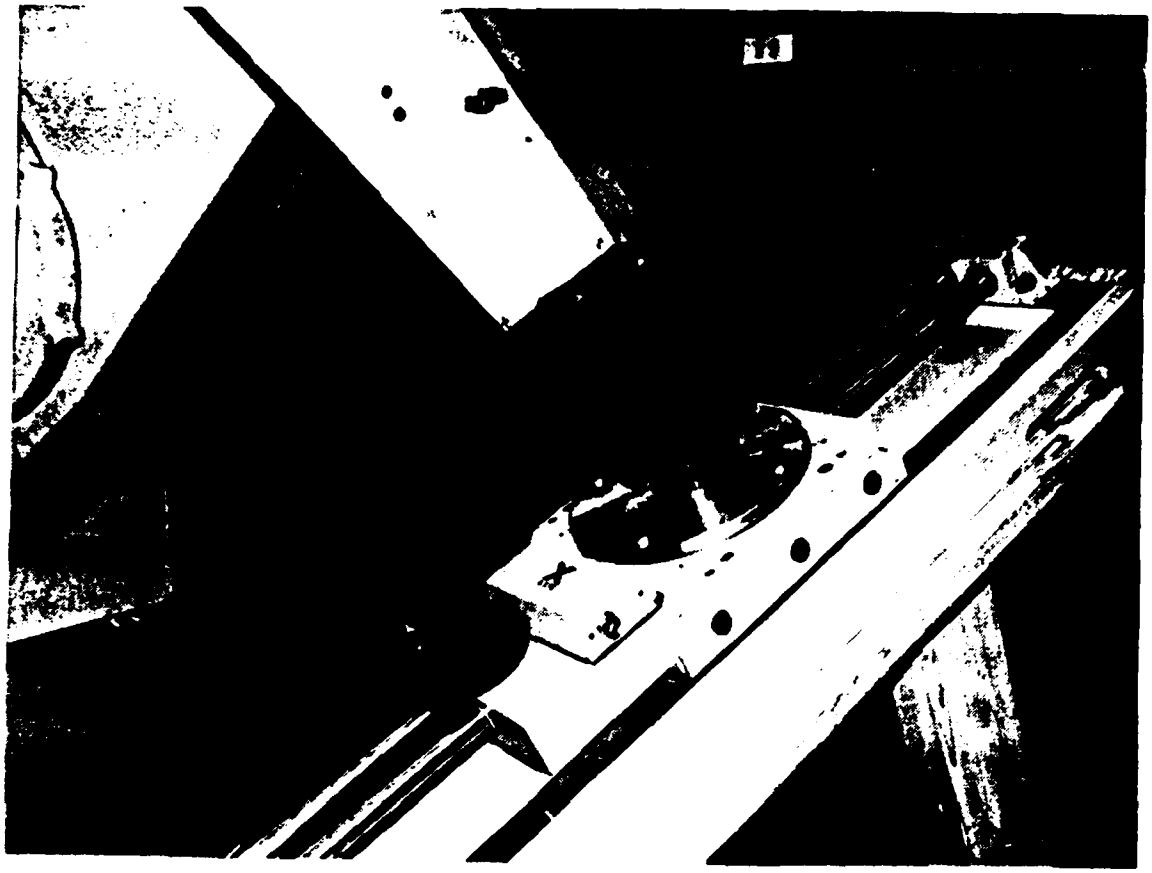


Figure 28. Coordinate Measuring Machine as Linear Slide

b. Data Acquisition Using the Linear Slide

Data acquisition using the linear slide is as follows. The PUMA is fixed to the slide using a plate fitted with holes to facilitate attachment to both joint six and the linear slide. The mounting plate has an offset which allows the kinematic parameters associated with the sixth transform to be identified. The PUMA is placed in "Free" mode, allowing unimpeded rotation of each joint, while continually being supported by one member of the measurement team. The PUMA is quite heavy and great care must be taken to prevent abrupt movement of the manipulator arm while joint six is attached to the slide.

The slide is moved to one end of the slide and the x-axis position indicator on the display unit is zeroed. The slide should be positioned at the lower end of the slide when being zeroed in order to have positive x-axis position indications on the display unit. The slide is moved up the scale in incremental distances, and after each increment the six joint angle positions of the PUMA and the x position of the slide is input using program ACDAT. The position which has been input is stored with the orientation matrix, which will not change, regardless of the position of the slide.

Two operators can expect to measure one pose in one minute. One operator is utilized to move and support the PUMA while the other will input the position of the slide and the joint angles of the PUMA, as directed by program ACDAT.

c. Identification of Actual Kinematic Parameters

The same version of program ID6_linsc which was used in the identification of the kinematic parameters in the simulation phase of the partial pose calibration is used in the identification of the actual kinematic parameters. Table 5 shows the kinematic parameters identified using the full pose calibration procedure.

TABLE 5. PARTIAL POSE IDENTIFIED PARAMETERS

Parameter	Nominal Value	Identified Value
ϕ_b	151.0	150.4016
θ_b	-20.0	-18.1370
φ_b	90.0	90.1163
p_{xb}	-203.0	-167.1160
p_{yb}	-254.0	-382.9224
p_{zb}	457.2	408.2330
a_1	0.0	0.3920
α_1	-90.0	-89.9778
$\delta\theta_2$	0.0	-0.5600
a_2	431.85	431.9559
α_2	0.0	-0.0648
β_2	0.0	0.0326
$\delta\theta_3$	0.0	-1.0704
d_3	149.1	149.4573
a_3	-20.3	-19.3452
α_3	90.0	90.1312
$\delta\theta_4$	0.0	-0.8549
d_4	433.0	433.0048
a_4	0.0	0.6809
α_4	-90.0	-89.5235
$\delta\theta_5$	0.0	2.3123
d_5	0.0	-0.3900
a_5	0.0	-0.8460
α_5	90.0	89.5360
ϕ_6	90.0	78.6654
θ_6	0.0	-0.1972

V. DISCUSSION OF RESULTS

A. GENERAL OBSERVATIONS

Both the full pose and partial pose calibration methods successfully provided kinematic parameters which could be used in a control program to use the PUMA most accurately. Each simulation predicted convergence to a unique set of kinematic parameters, thirty for the full pose method and twenty-six for the partial pose method. Poses measured in the laboratory provided the data which proved this prediction.

The results of the kinematic parameter identification using the full pose calibration method indicates that the resulting accuracy of the manipulator is 0.3 mm. The experimentally predicted accuracy was considerably larger than the predicted accuracy from the simulation, which was approximately 0.15 mm. This is most likely attributed to a higher magnitude of noise being present in the pose measurements than predicted in the simulation. It may be concluded that using this type of calibration method will improve the accuracy of the manipulator to about the same value of the manipulator repeatability [Ref. 16].

The results of the kinematic parameter identification using the partial pose calibration method indicate that the resulting accuracy of the manipulator is 0.9 mm. It may be concluded that using this type of calibration method will provide a manipulator which is three times less accurate than a manipulator calibrated using the full pose method, but still more accurate than an uncalibrated manipulator.

The method to determine accuracy of the partial pose calibration method is as follows. A total of 42 poses were measured, 21 in lefty configuration, 21 in righty. The lefty and righty configuration is a description of joint 1's position. If, while the operator is facing

the PUMA, joint 1 is rotated such that the axis of joint one points to the operator's right, the PUMA is in lefty configuration. The opposite holds for righty configuration. Figure 29 shows the PUMA in lefty, elbow down configuration. Note the tooling ball end effector attached to joint six.

Figure 29. The PUMA 560 in Lefty, Elbow Down Configuration

While all 42 poses were used in the identification of the kinematic parameters shown in Table 5, to determine the accuracy, the poses were divided. The first ten poses measured in the lefty and righty configuration were merged to form a set of twenty poses which were used in program ID6_LINSC to determine a set of identified kinematic parameters, an output file KINPARAM.DAT. The last ten poses measured in the lefty and righty configuration were merged into a data file, MEASURE.DAT, to act as the measured poses. The program F.FOR was executed, calculating the predicted poses based on the actual joint angles of the PUMA, read from MEASURE.DAT, and the identified kinematic parameters, read from KINPARAM.DAT. The result is two sets of $\mathbf{T}_6$ matrices, one the actual measured $\mathbf{T}_6$ matrix, the other the predicted $\mathbf{T}_6$ matrix. Program COMP.FOR calculates the RMS position difference and orientation difference between the two sets of $\mathbf{T}_6$ matrices.

B. COMPARATIVE OBSERVATIONS

Table 6 shows the identified kinematic parameters for the PUMA using the full pose calibration method and the partial pose calibration method. A comparison of the two methods of calibration described in this thesis, along with an analysis of the identified calibrated parameters, will follow.

TABLE 6. COMPARISON BETWEEN FULL AND PARTIAL POSE IDENTIFIED KINEMATIC PARAMETERS

Parameter	Nominal Value	Identified Value FULL POSE	Identified Value PARTIAL POSE
a_1	0.0	-0.04923	0.3920
α_1	-90.0	-89.9977	-89.9778
$\delta\theta_2$	0.0	-0.4888	-0.5600
a_2	431.9	432.1216	431.9559
α_2	0.0	-0.0303	-0.0648
β_2	0.0	-0.01515	0.0326
$\delta\theta_3$	0.0	-1.2069	-1.0704
d_3	149.1	149.1455	149.4573
a_3	-20.3	-19.2270	-19.3452
α_3	90.0	90.0512	90.1312
$\delta\theta_4$	0.0	-0.9144	-0.8549
d_4	433.0	432.8899	433.0048
a_4	0.0	0.0040	0.6809
α_4	-90.0	-89.9909	-89.5235
$\delta\theta_5$	0.0	2.2364	2.3123
d_5	0.0	-0.6629	-0.3900
a_5	0.0	-0.0258	-0.8460
α_5	90.0	89.9345	89.5360

Most robot manipulator arms will be used in an industrial environment. While it is important that the manipulator be properly calibrated, the amount of time the manipulator is unavailable for operation due to the calibration process being performed is significant. The experiments performed clearly have shown the full pose method takes considerably more time to accomplish than the partial pose method. The full pose method takes approximately

ten minutes to make one pose measurement, while the partial pose measurement takes only one minute to measure one pose. The amount of time required could be cut dramatically if an interface between the PUMA control computer and the data acquisition computer were built. This interface would eliminate the need for manual entry of the six joint angles for each pose. The relative difference in the amount of time required to measure one pose would still be present.

The full pose calibration method provides for a more accurate calibration of the manipulator. This fact arises from two considerations. First, more of the kinematic parameters of the manipulator can be identified using the full pose method, thirty, than the partial pose method, twenty-six. A thirty parameter model is logically a more detailed model of the manipulator than a twenty-six parameter model. Second, the full pose calibration uses a working volume of the manipulator which is significantly larger than the working volume used in the partial pose method. In actuality, the area used in the full pose calibration is indeed a volume. The area used in the partial pose model is comprised of only a line which is slightly longer than 900 mm. This thesis did not examine the validity of the calibration of the PUMA once the PUMA operates outside the calibration volume. It seems logical that if the tool of the PUMA was placed a considerable distance from the region in which it was calibrated, the calibration becomes invalid.

Another difference between the full and partial pose methods is the association between the number of poses required for a calibration. Each pose measured using the full pose calibration provides six units of information: the position, (x_6, y_6, z_6), of the origin of frame 6, and the orientation of frame 6, the direction cosines of each coordinate axis. The full pose calibration determined thirty kinematic parameters. In the complete absence of noise five poses would be required to identify the kinematic parameters, since thirty unknowns require thirty units of information for a unique solution. Each pose measured

using the partial pose calibration provides two units of information, since the frame which the pose is defined with respect to, is not known. Four units of information are required to define the frame, the sum of which is six units of information, as is available in the full pose calibration. In the complete absence of noise fourteen poses would be required to identify the kinematic parameters, since twenty-six unknowns require twenty-six units of information for a unique solution.

The full pose calibration showed that it is very important to exercise all six of the joints as much as possible during the pose measurement phase. This requirement arises due to the presence of measurement noise [Ref. 17]. The simulation's level of measurement noise was not as large as the noise determined to be present in the actual measurement phase. This allowed for a unique solution in the simulation without large joint rotations. The measured pose calibration process showed the increased amount of noise being present, and it was determined that maximizing the joint rotation alleviated the presence of the measurement noise in the coordinate measuring machine and in the encoders within the joints of the PUMA. The PUMA was exercised by taking a series of poses which would transcribe a cube in front of the PUMA, in elbow up and elbow down (joint 3 is the elbow), and in lefty and righty configurations.

The problem of exercising the joints resurfaced in the partial pose calibration. The slide's length and position relative to the PUMA prevented adequate exercising of the joints. If poses which were taken in only one configuration, i.e. lefty or righty, were used in a calibration attempt using ID6_LINSC, the identified kinematic parameters were found to be significantly different from those identified using the full pose method, and were disregarded as inaccurate. The presence of noise arising from the slide's flexure on its' mount dominated in this case. The first slide used in the partial pose calibration process had dramatic flexure, and was discarded in preference of the more stiff coordinate

measurement machine x-axis slide. Even with the reasonably stiff coordinate measuring machine it was shown that regardless of how many poses were measured in one configuration, an accurate calibration was not possible. To conduct an accurate calibration, a series of poses were measured along the length of the slide and then the PUMA was disconnected from the slide. Joint 1 was rotated such that the configuration was reversed from the lefty configuration to the righty configuration, the tool frame was reattached to the slide, and another series of poses was measured along the length of the slide. It was not possible to obtain poses in both the elbow up an elbow down configuration due to the proximity of the PUMA to the slide. All of the poses were taken in the elbow up configuration. Measuring poses in both the lefty and righty configuration countered the presence of the measurement noise, but the resultant accuracy of the calibration still did not match that of the full pose calibration.

The orientation of the slide presented a problem as well. Two calibration attempts were attempted with the slide laying flat on the table on which the PUMA is mounted. It was found that the orientation of the slide prevented the joint rotation necessary to overcome the measurement noise present. The next attempt placed the slide on an incline of about thirty degrees. The slide/incline apparatus was first oriented nominally in line with the x-axis of the PUMA. Again, adequate joint rotation was not possible. The difficult joints to move were the wrist joints, joints 4 through 6. Also, there was a problem with the tool frame entering areas of singularity. The area is said to be singular if the tool frame can be restrained in one position while a joint is moved freely. Simply stated, this means a singular pose can be defined with several different sets of joint angles. To remove the problem of singularity and to achieve the required joint rotation, the slide/incline apparatus was placed at an angle from the x-axis of the PUMA. Using this orientation and the pose

measurement procedure described in the previous paragraph, the calibration was completed.

VI. CONCLUSIONS

The results of the research reported in this thesis provide the following conclusions:

1. It is possible to improve the accuracy of a PUMA 560 robot manipulator to significantly less than the current possible accuracy of 10.0 mm, and equal to the repeatability, 0.3 mm.

2. Full pose calibration of the PUMA 560 is possible and will identify thirty kinematic parameters, producing an accuracy of 0.3 mm.

3. Partial pose calibration of the PUMA 560 is possible and will identify twenty-six kinematic parameters, producing an accuracy of 0.9 mm.

4. Obtaining the maximum possible joint rotation of all six joints is required for robust convergence to the identified kinematic parameters of the manipulator.

5. Measuring poses in lefty, righty, elbow up, and elbow down configuration facilitates maximum joint rotation and corresponding robust convergence.

6. The partial pose measurement phase showed that the orientation of the slide relative to the PUMA is of paramount importance in the identification of the kinematic parameters.

7. More information per pose is determined using the full pose calibration, but the excessive time required, ten minutes per pose, detracts from the efficiency of the method.

8. To use the partial pose method of calibration requires that the manipulator have a "Free" mode, in which each joint can be physically rotated.

9. The "Delta" matrix, showed in equation 35 should not be used as the functions to be minimized in the calibration procedure. Each element of the $\mathbf{T}_6$ matrix should be used as functions to be minimized.

APPENDIX A. PROGRAM BALL1

```
      Program BALL

c     This program takes four sets of Cartesian coordinates
c     assumed to be from a CMM touching the surface of a precision tooling
c     ball, and finds the center of the ball.

      integer n,m,nsig,maxfn,iopt,i,infer,ier,ixjac
      real*8  parm(4),x(3),f(4),xjac(4,3),work(29),eps,delta,
     &        xjtj(6),y(4,3)
      external resid
      common  y

      m=4
      n=3
      ixjac=4
      nsig=3
      eps=0.0
      delta=0.0
      maxfn=500
      iopt=1

c get touch data

 1000 do i=1,4
      write(6,*)'Move the CMM and type in data for point #',i
      read (5,*) y(i,1),y(i,2),y(i,3)
      enddo

c set up start point for identification

      x(1)=y(1,1)
      x(2)=y(1,2)
      x(3)=y(1,3)

      call zxssq(resid,m,n,nsig,eps,delta,maxfn,iopt,parm,x,ssq,f,
     &           xjac,ixjac,xjtj,work,infer,ier)

      write(6,800) 'center (x,y,z) is:',x(1),x(2),x(3)
      write(6,*) 'residual error is:', ssq
  800 format(1x,a,3f9.3)

      end

c-----------------------------------------------------------------------

      subroutine resid(x,m,n,f)

c     This subroutine calculates the residual functions used by the
c     IMSL routine ZXSSQ, called from subroutine BALL.

      integer m,n
      real*8 x(n),f(m),y(4,3),res,re,r0
      common y
```

```
r0=7.85
do i=1,4
        res=(x(1)-y(i,1))**2+(x(2)-y(i,2))**2+(x(3)-y(i,3))**2
        re=dsqrt(res)
        f(i)=abs(re-r0)
enddo

sum=0.0
do i=1,4
        sum=sum+f(i)
enddo
write(6,*) sum
return
end
```

APPENDIX B. PROGRAM CMMPOSE

```
c-------------------------------------------------------------------------------

      Program CMMPOSE

c     This program accepts coordinate data from the CMM, together
c     with the tooling ball i.d. and generates a 4x4 pose matrix
c     corresponding to the pose. Four measurements on each of three
c     balls is required.

      parameter (lda=4,ldainv=4,n=4,ld3=3)
      real*8 x(3,3),z(3),p(5,3),pa(3),pb(3),pc(3),pd(3),p4(4,4),
     &       pd4(4,4),pinv(4,4),t(4,4),sum,val,x1,y1,z1,
     &       x2,y2,z2
      integer id(3),flag
      character reply
      external wrrrn

c Precalibration data for the tool (in column order)

      data p/0.0,50.740,0.0,-50.913,0.0,
     &       0.0,0.0,50.703,0.0,-50.988,
     &       51.111,0.0,0.0,0.0,0.0/

c open data file

      open(9,name='posex.dat',status='new')

c gather data from three balls

 1000 do j=1,3
      write(6,*) 'Input ball reference number'
      read(5,*) id(j)
      call ball(z)
           do k=1,3
           x(j,k)=z(k)
           enddo
      enddo

c synthesize the fourth ball - measured

      do j=1,3
      pa(j)=x(1,j)
      pb(j)=x(2,j)
      pc(j)=x(3,j)
      enddo

      x1=(pb(1)-pa(1))/70.71
      y1=(pb(2)-pa(2))/70.71
      z1=(pb(3)-pa(3))/70.71

      x2=(pc(1)-pa(1))/70.71
      y2=(pc(2)-pa(2))/70.71
      z2=(pc(3)-pa(3))/70.71
```

```
      pd(1)=(y1*z2-y2*z1)*70.71+pa(1)
      pd(2)=(x2*z1-x1*z2)*70.71+pa(2)
      pd(3)=(x1*y2-x2*y1)*70.71+pa(3)

c compose the PD4 matrix

      do i=1,3
      pd4(i,1)=pa(i)
      pd4(i,2)=pb(i)
      pd4(i,3)=pc(i)
      pd4(i,4)=pd(i)
      enddo

      do i=1,4
      pd4(4,i)=1.0
      enddo

c synthesize the fourth ball - tool

      do j=1,3
      pa(j)=p(id(1),j)
      pb(j)=p(id(2),j)
      pc(j)=p(id(3),j)
      enddo

      x1=(pb(1)-pa(1))/70.71
      y1=(pb(2)-pa(2))/70.71
      z1=(pb(3)-pa(3))/70.71

      x2=(pc(1)-pa(1))/70.71
      y2=(pc(2)-pa(2))/70.71
      z2=(pc(3)-pa(3))/70.71

      pd(1)=(y1*z2-y2*z1)*70.71+pa(1)
      pd(2)=(x2*z1-x1*z2)*70.71+pa(2)
      pd(3)=(x1*y2-x2*y1)*70.71+pa(3)

c compose the P4 matrix

      do i=1,3
      p4(i,1)=pa(i)
      p4(i,2)=pb(i)
      p4(i,3)=pc(i)
      p4(i,4)=pd(i)
      enddo

      do i=1,4
      p4(4,i)=1.0
      enddo

c calculate the T matrix

      call dlinrg(n,p4,lda,pinv,ldainv)
      call matmulc(t,pd4,pinv)
      call dwrrrn('T',n,n,t,ldainv,0)

c orthogonality and size check

      flag=0
      sum=0.0
            do j=1,3
```

```
                sum=sum+t(j,1)*t(j,2)
                enddo
                val=dabs(sum)
                write(6,*)'val=',val
                if(val.gt.0.01) flag=1

                do j=1,3
                sum=sum+t(j,2)*t(j,3)
                enddo
                val=dabs(sum)
                write(6,*)'val=',val
                if(val.gt.0.01) flag=1

                do j=1,3
                sum=sum+t(j,1)*t(j,3)
                enddo
                val=dabs(sum)
                write(6,*)'val=',val
                if(val.gt.0.01) flag=1

        do i=1,3
                sum=0.0
                do j=1,3
                sum=sum+t(j,i)*t(j,i)
                enddo
                val=dabs(sum-1.00000)
                write(6,*)'val=',val
                if(val.gt.0.01) flag=1
        enddo

        if(flag.eq.1) then
                write(6,*) 'orthogonality test: FAIL'
        else
                write(6,*) 'orthogonality test: PASS'
        endif

c save data to file

        do i=1,4
        write(9,700) t(i,1),t(i,2),t(i,3),t(i,4)
        enddo
  700   format(4e17.8)
        write(9,*)

c reminder to get joint angles

        write(6,*)'Type WHERE on the PUMA console'

c need more data ?

 1020   write(6,*)'More data...(y=yes, n=no)'
        read (5,'(a)') reply
        if (reply.eq.'y') goto 1000
        if (reply.eq.'n') goto 1010
        goto 1020


 1010   close (9)
        close(10)
        end
```

```
c ------------------------------------------------------------------------

      subroutine BALL(x)

c     This program takes three or four sets of Cartesian coordinates
c     assumed to be from a CMM touching the surface of a precision tooling
c     ball, and finds the center of the ball.

      external resid
      integer n,m,nsig,maxfn,iopt,i,infer,ier,ixjac
      real*8  parm(4),x(3),f(4),xjac(4,3),work(29),eps,delta,
     &        xjtj(6),y(4,3)
      character reply
      common  y

      open(10,name='test.dat',status='old')

      m=4
      n=3
      ixjac=4
      nsig=3
      eps=0.0
      delta=0.0
      maxfn=500
      iopt=1

c get touch data

 1020 do i=1,4
      write(6,*)'Move the CMM and type in data for point #',i
c     read (10,*)y(i,1),y(i,2),y(i,3)
      read (5,*) y(i,1),y(i,2),y(i,3)
      write(10,*)y(i,1),y(i,2),y(i,3)
      enddo

c set up start point for identification

      x(1)=y(1,1)
      x(2)=y(1,2)
      x(3)=y(1,3)

      call zxssq(resid,m,n,nsig,eps,delta,maxfn,iopt,parm,x,ssq,f,
     &           xjac,ixjac,xjtj,work,infer,ier)

      write(6,800) 'center (x,y,z) is:',x(1),x(2),x(3)
      write(6,*) 'residual error is:', ssq
  800 format(1x,a,3f9.3)

 1000 write(6,*) 'Keep this data (y=yes, n=no)'
      read (5,'(a)') reply
      if (reply.eq.'y') goto 1010
      if (reply.eq.'n') goto 1020
      goto 1000

 1010 return
      end

c-------------------------------------------------------------------------

      subroutine resid(x,m,n,f)
```

```
c       This subroutine calculates the residual functions used by the
c       IMSL routine ZXSSQ, called from subroutine BALL.

        integer m,n
        real*8 x(n),f(m),y(4,3),res,re,r0
        common y

        r0=7.85
        do i=1,4
                res=(x(1)-y(i,1))**2+(x(2)-y(i,2))**2+(x(3)-y(i,3))**2
                re=dsqrt(res)
                f(i)=abs(re-r0)
        enddo

        sum=0.0
        do i=1,4
                sum=sum+f(i)
        enddo

        return
        end
```

APPENDIX C PROGRAM JOINT

```
      PROGRAM JOINT

c     This program generates the six joint angles for the
c     PUMA manipulator arm for the simulation of the PUMA's
c     calibration using the coordinate measurement machine.
c     The six joint angles are generated using a random number
c     generator.

      INTEGER I, J, K, NOBS, MAXNOBS
      PARAMETER (MAXNOBS=360)
      REAL Q(MAXNOBS,6), QMIN(6), QMAX(6)

      COMMON /C1/ Q, QMAX, QMIN

c     In this section the working volume of the PUMA can be selected.
c     The working volume can be full, half, or one quarter, based on
c     extent of the allowed joint movement.

      DATA QMIN/ -180.0, -180.0, -180.0, -180.0, -180.0, -180.0 /
      DATA QMAX/  180.0, 180.0, 180.0, 180.0, 180.0, 180.0 /
       WRITE (6,*) 'Volume is FULL'

c     DATA QMIN/ -90.0, -90.0, -90.0, -90.0, -90.0, -90.0 /
c     DATA QMAX/ 90.0, 90.0, 90.0, 90.0, 90.0, 90.0 /
c      WRITE (6,*) 'Volume is HALF'

c     DATA QMIN/ -45.0, -45.0, -45.0, -45.0, -45.0, -45.0 /
c     DATA QMAX/ 45.0, 45.0, 45.0, 45.0, 45.0, 45.0 /
c      WRITE (6,*) 'Volume is QUARTER'

c     Open the output data file, PUMA-VAR.DAT.

      OPEN (8, NAME='PUMA-VAR.DAT', STATUS='NEW')

c     Input the number of observations from the nominal kinematic
c     data file, INPUT.DAT.

      OPEN (9, NAME='INPUT.DAT', STATUS='OLD')

      DO I=1,10
       READ (9,*)
      ENDDO

      READ (9,*) NOBS

      CLOSE (9)

c     Call the random number generation routine to obtain a set
c     of six random joint angles.

      CALL MSPREAD (NOBS)

c     Save the joint variable data in the output file, PUMA-VAR.DAT.
```

```
      DO II = 1, NOBS
       WRITE (8,*) Q(II,1),Q(II,2),Q(II,3),Q(II,4),Q(II,5),Q(II,6)
      ENDDO

      CLOSE (8)

      STOP

      END

c  **SUBROUTINES**

c  **SUBROUTINE 1**

      SUBROUTINE MSPREAD (NOBS)

c     This subroutine generates the six required joint angles
c     by the Monte Carlo method. The six joint angles are
c     generated by using six independently random numbers.

      INTEGER I, J, K, NOBS, MAXNOBS
      PARAMETER (MAXNOBS=360)
      REAL Q(MAXNOBS,6), QMIN(6), QMAX(6), MAGQ(6), NUM
      INTEGER*4 ISEED
      COMMON /C1/ Q, QMAX, QMIN

c     Obtaining the random number seed.

      WRITE (6,*) 'Type in a 6-digit random number seed'
      READ  (5,*) ISEED

c     Calculating the scaling factor for each joint angle.

      DO I = 1, 6
       MAGQ(I) = QMAX(I)-QMIN(I)
      ENDDO

c     Generating the six joint angles.

      DO J = 1, NOBS
       DO I = 1, 6
        CALL RANDOM (ISEED,NUM)
        Q(J,I) = QMIN(I) + MAGQ(I) * NUM
       ENDDO
      ENDDO

      RETURN

      END

c  **SUBROUTINE 2**

      SUBROUTINE RANDOM (X,Z)

      REAL FM, FX, Z
      INTEGER A, X, I, M
      DATA I/1/
```

```
      IF ( I .EQ. 0 ) GO TO 1000
       I=0
       M= 2 ** 20
       FM= M
       A= 2**10 + 3

1000  X= MOD( A*X ,M)
      FX= X
      Z= FX/ FM

      RETURN

      END
```

APPENDIX D PROGRAM POSE

```
      PROGRAM POSE

c     This program generates a T6 matrix for each set of joint
c     angles which were generated using the program JOINT.  The T6
c     matrix is calculated by performing a forward kinematic solution
c     using the six joint angles which are read from PUMA-VAR.DAT
c     and the nominal kinematic data which are read from INPUT.DAT.

      INTEGER*4 ISEED
      INTEGER I, J, K, NOBS, MAXNOBS, N

      REAL*8 RNX, RNY, RNZ, MAGNX, MAGN1, DANGLE, DLENTH
      REAL*8 RN1, RN2, RN3, RN4, RN5, RN6, PI

      PARAMETER (PI=3.141592653589793)
      PARAMETER (MAXNOBS=360)

      REAL*8 FI0, SI0, TH0, PX0, PY0, PZ0
      REAL*8 DT1, DT2, DT3, DT4, DT5
      REAL*8 DD1, DD2, DD3, DD4, DD5
      REAL*8 AA1, AA2, AA3, AA4, AA5
      REAL*8 AL1, AL2, AL3, AL4, AL5
      REAL*8 BL1, BL2, BL3, BL4, BL5
      REAL*8 FI6, TH6, SI6, PX6, PY6, PZ6, DF6
      REAL*8 THETA1, THETA2, THETA3, THETA4, THETA5, THETA6
      REAL*8 TH1, TH2, TH3, TH4, TH5
      REAL*8 T0(4,4), T1(4,4), T2(4,4), T3(4,4)
      REAL*8 T4(4,4), T5(4,4), T6(4,4), TRPY(4,4), TXYZ(4,4)
      REAL*8 TIMAT(4,4), T(4,4)

c     Initializing the TIMAT matrix to an identity matrix.

      DATA TIMAT/1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1/

c     Obtain a random number seed.  This random number will be used
c     later in the program to generate random noise which will model
c     an actual measurement using the coordinate measurement machine.

      WRITE (6,*) 'Type in a 6-digit random number seed.'
      READ  (5,*) ISEED

c     Open the two input files, PUMA-VAR.DAT and INPUT.DAT, and
c     the output data file, PUMA-POS.DAT, which will contain the
c     six previously determined joint angles and the corresponding
c     T6 matrix calculated in this program.

      OPEN (8, NAME='PUMA-VAR.DAT', STATUS='OLD')
      OPEN (9, NAME='PUMA-POS.DAT', STATUS='NEW')
      OPEN (10,NAME='INPUT.DAT', STATUS='OLD')

c     Reading nominal kinematic parameters from INPUT.DAT.

      read (10,*)
```

```
      read (10,*) fi0,th0,si0,px0,py0,pz0
      read (10,*) dt1,dd1,aa1,al1,bl1
      read (10,*) dt2,dd2,aa2,al2,bl2
      read (10,*) dt3,dd3,aa3,al3,bl3
      read (10,*) dt4,dd4,aa4,al4,bl4
      read (10,*) dt5,dd5,aa5,al5,bl5
      read (10,*)
      read (10,*) df6,th6,si6,px6,py6,pz6
      read (10,*)
      read (10,*) nobs,n,dangle,dlenth,magnx,magnl

c     Adding encoder offsets and setting link parameters, creating
c     a PUMA manipulator different from the manipulator represented
c     by the nominal kinematic parameters.

      dt1 = dt1
      dt2 = dt2 + dangle
      dt3 = dt3 + dangle
      dt4 = dt4 + dangle
      dt5 = dt5 + dangle

      fi0 = fi0 + dangle
      th0 = th0 + dangle
      si0 = si0 + dangle
      px0 = px0 + dlenth
      py0 = py0 + dlenth
      pz0 = pz0 + dlenth

      al1 = al1 + dangle
      al2 = al2 + dangle
      al3 = al3 + dangle
      al4 = al4 + dangle
      al5 = al5 + dangle

      aa1 = aa1 + dlenth
      aa2 = aa2 + dlenth
      aa3 = aa3 + dlenth
      aa4 = aa4 + dlenth
      aa5 = aa5 + dlenth

      dd1 = dd1
      dd2 =    0.0            ! defined
      dd3 = dd3 + dlenth
      dd4 = dd4 + dlenth
      dd5 = dd5 + dlenth

      bl1 = bl1               ! defined
      bl2 = bl2 + dangle
      bl3 = bl3               ! defined
      bl4 = bl4               ! defined
      bl5 = bl5               ! defined

      df6 = df6 + dangle
      th6 = th6 + dangle
      si6 = si6 + dangle
      px6 = px6 + dlenth
      py6 = py6 + dlenth
      pz6 = pz6 + dlenth

c     Loop through the program, generating a T6 matrix for each
c     set of 6 joint angles.  The number of loops will be reflected
```

```
c   by the entry "nobs" in the nominal kinematic parameter file,
c   INPUT.DAT.

        DO I = 1, NOBS

c   Initializing the T matrix to an identity matrix.

         DO J=1,4
          DO K=1,4
           T(J,K) = TIMAT(J,K)
          ENDDO
         ENDDO

c   Read the sets of six joint angles from the data file PUMA-VAR.DAT.

        READ (8,*) THETA1, THETA2, THETA3, THETA4, THETA5, THETA6

        TH1 = DT1 + THETA1
        TH2 = DT2 + THETA2
        TH3 = DT3 + THETA3
        TH4 = DT4 + THETA4
        TH5 = DT5 + THETA5
        FI6 = DF6 + THETA6

c   Computing the six individual T matrices, T1 thru T6.

        CALL T3RPY ( FI0, TH0, SI0, TRPY )
        CALL T3XYZ ( PX0, PY0, PZ0, TXYZ )
        CALL MATMULC ( T0, TRPY, TXYZ )

        CALL TRANSFORM ( AL1, AA1, DD1, TH1, BL1, T1 )
        CALL TRANSFORM ( AL2, AA2, DD2, TH2, BL2, T2 )
        CALL TRANSFORM ( AL3, AA3, DD3, TH3, BL3, T3 )
        CALL TRANSFORM ( AL4, AA4, DD4, TH4, BL4, T4 )
        CALL TRANSFORM ( AL5, AA5, DD5, TH5, BL5, T5 )

        CALL T3RPY ( FI6, TH6, SI6, TRPY )
        CALL T3XYZ ( PX6, PY6, PZ6, TXYZ )
        CALL MATMULC ( T6, TRPY, TXYZ )

c   Computing the overall transformation, T.

        CALL MATMULA ( T, T0 )
        CALL MATMULA ( T, T1 )
        CALL MATMULA ( T, T2 )
        CALL MATMULA ( T, T3 )
        CALL MATMULA ( T, T4 )
        CALL MATMULA ( T, T5 )
        CALL MATMULA ( T, T6 )

c   Generating the random noise.  This noise will be added to
c   the theoretical measurement data and the encoder readings
c   to accurately simulate the actual measurements which will
c   be made in the lab.

        CALL RANDOM(ISEED,RNX)
        CALL RANDOM(ISEED,RNY)
        CALL RANDOM(ISEED,RNZ)

        CALL RANDOM(ISEED,RN1)
        CALL RANDOM(ISEED,RN2)
```

```
      CALL RANDOM(ISEED,RN3)
      CALL RANDOM(ISEED,RN4)
      CALL RANDOM(ISEED,RN5)
      CALL RANDOM(ISEED,RN6)

      RNX = MAGNX*( 2.0*RNX - 1.0 )
      RNY = MAGNX*( 2.0*RNY - 1.0 )
      RNZ = MAGNX*( 2.0*RNZ - 1.0 )

      RN1 = MAGN1*( 2.0*RN1 - 1.0 )
      RN2 = MAGN1*( 2.0*RN2 - 1.0 )
      RN3 = MAGN1*( 2.0*RN3 - 1.0 )
      RN4 = MAGN1*( 2.0*RN4 - 1.0 )
      RN5 = MAGN1*( 2.0*RN5 - 1.0 )
      RN6 = MAGN1*( 2.0*RN6 - 1.0 )

c     Adding the noise to measurement and encoder readings.

      T(1,4) = T(1,4) + RNX
      T(2,4) = T(2,4) + RNY
      T(3,4) = T(3,4) + RNZ

      THETA1 = THETA1 +RN1
      THETA2 = THETA2 +RN2
      THETA3 = THETA3 +RN3
      THETA4 = THETA4 +RN4
      THETA5 = THETA5 +RN5
      THETA6 = THETA6 +RN6

c     Storing the manipulator joint angles, calculated in the
c     program JOINT and the theoretical measured tool pose,
c     calculated in this program, in the data file PUMA-POS.DAT.

      WRITE (9,991) THETA1, THETA2, THETA3, THETA4, THETA5, THETA6
      WRITE (9,992) T(1,1), T(1,2), T(1,3), T(1,4)
      WRITE (9,992) T(2,1), T(2,2), T(2,3), T(2,4)
      WRITE (9,992) T(3,1), T(3,2), T(3,3), T(3,4)
      WRITE (9,*)

c     Format below decides the digits of accuracy of the
c     simulation data.

991   FORMAT ( 6F12.6 )                    !Joint vector data
992   FORMAT ( 3F16.10, F12.5 )            !Measurement data

      ENDDO

      WRITE (6,*) 'Data stored in F12.5, F12.4 format'

      CLOSE (8)
      CLOSE (9)

      STOP

      END


c **SUBROUTINES**

c **SUBROUTINE 1**
```

```
      SUBROUTINE RANDOM (X,Z)

      REAL FM, FX, Z
      INTEGER A, X, I, M
      DATA I/1/

      IF ( I .EQ. 0 ) GO TO 1000
       I=0
       M= 2 ** 20
       FM= M
       A= 2**10 + 3

1000  X= MOD( A*X ,M)
      FX= X
      Z= FX/ FM

      RETURN

      END
```

APPENDIX E PROGRAM ID6

```
      PROGRAM ID6

c This is the program which identifies the kinematic parameters
c of the simulated PUMA 560 manipulator using the Non-linear
c Least Squares method in IMSL routine ZXSSQ using function
c PUMA_ARM. The simulated poses are read from the data file
c PUMA-POS.DAT

      INTEGER LDFJAC, MM, M, NN, N, NSIG, MAXFN, IOPT, IXJAC, INFER, IER
      INTEGER I, J, K, NOBS, MAXNOBS

      REAL*8 FJAC(LDFJAC,NN), XJTJ((NN+1)*NN/2)
      REAL*8 PARM(4), F(LDFJAC), WORK((5*NN)+(2*MM)+((NN+1)*NN/2))
      REAL*8 DANGLE, DLENTH, TQ, DQ, EPS, DELTA, SSQ, SQERR1, SQERR2
      REAL*8 FI0, TH0, SI0, PX0, PY0, PZ0
      REAL*8 DT1, DT2, DT3, DT4, DT5
      REAL*8 DD1, DD2, DD3, DD4, DD5
      REAL*8 AA1, AA2, AA3, AA4, AA5
      REAL*8 AL1, AL2, AL3, AL4, AL5
      REAL*8 BL1, BL2, BL3, BL4, BL5
      REAL*8 DF6, TH6, SI6, PX6, PY6, PZ6, FI6
      REAL*8 TET1(MAXNOBS), TET2(MAXNOBS), TET3(MAXNOBS)
      REAL*8 TET4(MAXNOBS), TET5(MAXNOBS), TET6(MAXNOBS)
      REAL*8 TM(3,4,MAXNOBS), SCALE, X(NN)

      PARAMETER (LDFJAC=6*360, MM=LDFJAC, NN=30)
      PARAMETER (MAXNOBS=360)

      EXTERNAL PUMA_ARM

      COMMON /PDATA/ NOBS, TM, SCALE,
     &          TET1, TET2, TET3, TET4, TET5, TET6

c Open data file for storage of output. This will be the identified
c kinematic parameters.

      OPEN (8, NAME='RESULT.DAT', STATUS='NEW')

c Open data file for the simulated poses. These were generated in
c program POSE.

      OPEN (9, NAME='PUMA-POS.DAT', STATUS='OLD')

c Open data file of nominal kinematic parameters.

      OPEN (10,NAME='INPUT.DAT', STATUS='OLD')

c Read nominal kinematic parameters from INPUT.DAT.

      read (10,*)
      read (10,*) fi0,th0,si0,px0,py0,pz0
      read (10,*) dt1,dd1,aa1,al1,bl1
      read (10,*) dt2,dd2,aa2,al2,bl2
```

```
      read (10,*) dt3,dd3,aa3,al3,bl3
      read (10,*) dt4,dd4,aa4,al4,bl4
      read (10,*) dt5,dd5,aa5,al5,bl5
      read (10,*)
      read (10,*) df6,th6,si6,px6,py6,pz6
      read (10,*)
      read (10,*) nobs,n,dangle,dlenth,magnx,magnl

      CLOSE (10)

c Initialize the kinematic parameter data variables in the X array.

      X(1)=FI0
      X(2)=TH0
      X(3)=SI0
      X(4)=PX0
      x(5)=PY0
      x(6)=PZ0

      X(7)=AA1
      X(8)=AL1

      X(9)=DT2
      X(10)=AA2
      X(11)=AL2
      X(12)=BL2

      X(13)=DT3
      X(14)=DD3
      X(15)=AA3
      X(16)=AL3

      X(17)=DT4
      X(18)=DD4
      X(19)=AA4
      X(20)=AL4

      X(21)=DT5
      X(22)=DD5
      X(23)=AA5
      X(24)=AL5

      X(25)=DF6
      X(26)=TH6
      X(27)=SI6
      X(28)=PX6
      X(29)=PY6
      X(30)=PZ6

c Read simulated joint angles and tool pose from PUMA-POS.DAT.

      DO J = 1, NOBS
        READ (9,*) TET1(J), TET2(J), TET3(J), TET4(J), TET5(J), TET6(J)
        READ (9,*) TM(1,1,J), TM(1,2,J), TM(1,3,J), TM(1,4,J)
        READ (9,*) TM(2,1,J), TM(2,2,J), TM(2,3,J), TM(2,4,J)
        READ (9,*) TM(3,1,J), TM(3,2,J), TM(3,3,J), TM(3,4,J)
        READ (9,*)
      ENDDO
      CLOSE (9)

c Set the scale factor for the direction cosines within the T6 matrix.
```

```
      SCALE=100.0

c The following lines set the parameters necessary for the operation
c of the IMSL routine, ZXSSQ, for identification of the kinematic
c parameters.

c The subroutine which ZXSSQ is calling is PUMA_ARM.

      NSIG=4
      EPS=0.0
      DELTA=0.0
      MAXFN=1500
      IOPT=1
      IXJAC=LDFJAC
      M=6*NOBS

      CALL ZXSSQ(PUMA_ARM,M,N,NSIG,EPS,DELTA,MAXFN,IOPT,
     &           PARM,X,SSQ,F,FJAC,IXJAC,XJTJ,WORK,INFER,IER)

c Save identified kinematic parameters to data file RESULT.DAT.

      WRITE (8,*)
      WRITE (8,*) 'FI0, TH0, SI0, PX0, PY0, PZ0'
      WRITE (8,888) X(1), X(2), X(3), X(4), x(5), x(6)
      WRITE (8,*)
      WRITE (8,*) 'DT1, DD1, AA1, AL1, BL1'
      WRITE (8,888) 0.0, 0.0, X(7), X(8), 0.0
      WRITE (8,*)
      WRITE (8,*) 'DT2, DD2, AA2, AL2, BL2'
      WRITE (8,888) X(9), 0.0, X(10), X(11), X(12)
      WRITE (8,*)
      WRITE (8,*) 'DT3, DD3, AA3, AL3, BL3'
      WRITE (8,888) X(13), X(14), X(15), X(16), 0.0
      WRITE (8,*)
      WRITE (8,*) 'DT4, DD4, AA4, AL4, BL4'
      WRITE (8,888) X(17), X(18), X(19), X(20), 0.0
      WRITE (8,*)
      WRITE (8,*) 'DT5, DD5, AA5, AL5, BL5'
      WRITE (8,888) X(21), X(22), X(23), X(24), 0.0
      WRITE (8,*)
      WRITE (8,*) 'DF6, TH6, SI6, PX6, PY6, PZ6'
      WRITE (8,889) X(25), X(26), X(27), X(28), X(29), X(30)

888   FORMAT ( 5F12.5 )
889   FORMAT ( 6F12.5 )

c Calculate the root mean square error in identification of the
c direction cosine and position parameters.

      TQ = DANGLE
      DQ = DLENTH

c This calculates the error in identification of the direction
c cosine parameters.

      SQERR1 =
     & (FI0+TQ-X(1))**2+(TH0+TQ-X(2))**2+(SI0+TQ-X(3))**2
     & +(AL1+TQ-X(8))**2+(DT2+TQ-X(9))**2+(AL2+TQ-X(11))**2
     & +(BL2+TQ-X(12))**2+(DT3+TQ-X(13))**2+(AL3+TQ-X(16))**2
     & +(DT4+TQ-X(17))**2+(AL4+TQ-X(20))**2
```

```
     &  +(DT5+TQ-X(21))**2+(AL5+TQ-X(24))**2
     &  +(DF6+TQ-X(25))**2+(TH6+TQ-X(26))**2+(SI6+TQ-X(27))**2

        SQERR1 = DSQRT( SQERR1/16 )

c This calculates the error in identification of the position
c parameters.

        SQERR2 =
     &  (PX0+DQ-X(4))**2+(PY0+DQ-X(5))**2+(PZ0+DQ-X(6))**2
     &  +(AA1+DQ-X(7))**2+(AA2+DQ-X(10))**2+(DD3+DQ-X(14))**2
     &  +(AA3+DQ-X(15))**2+(DD4+DQ-X(18))**2+(AA4+DQ-X(19))**2
     &  +(DD5+DQ-X(22))**2+(AA5+DQ-X(23))**2
     &  +(PX6+DQ-X(28))**2+(PY6+DQ-X(29))**2+(PZ6+DQ-X(30))**2
        SQERR2 = DSQRT( SQERR2/14 )

c Write the position and orientation error to the data file
c RESULT.DAT.

        WRITE (8,*)
        WRITE (8,*) 'RMS PARMS (LENGTH), RMS PARMS (ANGLE)'
        WRITE (8,*) SQERR2, SQERR1

c Write the position and orientation error to the screen.

        WRITE (6,*) 'RMS PARMS (LENGTH), RMS PARMS (ANGLE)'
        WRITE (6,*) SQERR2,SQERR1

c Write the ZXSSQ convergence criteria to the data file
c RESULT.DAT.

        WRITE (8,*)
        WRITE (8,*) 'INFER, IER, NOBS, NSIG'
        WRITE (8,*)  INFER, IER, NOBS, NSIG
        WRITE (8,*)

c Write the ZXSSQ convergence criteria to the screen.

        WRITE (6,*) 'INFER, IER, NOBS, NSIG'
        WRITE (6,*)  INFER, IER, NOBS, NSIG

        CLOSE (8)

        END

c **SUBROUTINES**

c **SUBROUTINE 1**

        SUBROUTINE PUMA_ARM (X, M, N, F)

c This subroutine calculates the non-linear function for the use of
c the IMSL routine ZXSSQ. It is the forward kinematic solution for
c the PUMA manipulator.

        INTEGER M, N, II, JJ
        INTEGER I, J, K, NOBS, MAXNOBS

        REAL*8 X(N), F(M)
        REAL*8 FI0, TH0, SI0, PX0, PY0, PZ0
        REAL*8 DT1, DT2, DT3, DT4, DT5
```

```
      REAL*8 DD1, DD2, DD3, DD4, DD5
      REAL*8 AA1, AA2, AA3, AA4, AA5
      REAL*8 AL1, AL2, AL3, AL4, AL5
      REAL*8 BL1, BL2, BL3, BL4, BL5
      REAL*8 FI6, TH6, SI6, PX6, PY6, PZ6, DF6
      REAL*8 TH1, TH2, TH3, TH4, TH5
      REAL*8 T0(4,4), T1(4,4), T2(4,4), T3(4,4), T4(4,4)
      REAL*8 T5(4,4), T6(4,4), TRPY(4,4), TXYZ(4,4)
      REAL*8 TIMAT(4,4), T(4,4)
      REAL*8 TINV(4,4), TMJ(4,4), TDELTA(4,4)
      REAL*8 TET1(MAXNOBS), TET2(MAXNOBS), TET3(MAXNOBS)
      REAL*8 TET4(MAXNOBS), TET5(MAXNOBS), TET6(MAXNOBS)
      REAL*8 TM(3,4,MAXNOBS), SCALE

      PARAMETER (MAXNOBS=360)

      COMMON /PDATA/ NOBS, TM, SCALE,
     &          TET1, TET2, TET3, TET4, TET5, TET6

c Initializing the TIMAT matrix to an identity matrix.

      DATA TIMAT/1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1/

c Assign kinematic parameters to the variable X array.

      FI0 = X(1)
      TH0 = X(2)
      SI0 = X(3)
      PX0 = X(4)
      PY0 = x(5)
      PZ0 = x(6)

      DT1 = 0.0
      DD1 = 0.0
      AA1 = X(7)
      AL1 = X(8)
      BL1 = 0.0

      DT2 = X(9)
      DD2 = 0.0
      AA2 = X(10)
      AL2 = X(11)
      BL2 = X(12)

      DT3 = X(13)
      DD3 = X(14)
      AA3 = X(15)
      AL3 = X(16)
      BL3 = 0.0

      DT4 = X(17)
      DD4 = X(18)
      AA4 = X(19)
      AL4 = X(20)
      BL4 = 0.0

      DT5 = X(21)
      DD5 = X(22)
      AA5 = X(23)
      AL5 = X(24)
      BL5 = 0.0
```

```
      DF6 = X(25)
      TH6 = X(26)
      SI6 = X(27)
      PX6 = X(28)
      PY6 = X(29)
      PZ6 = X(30)

c Loop through NOBS times.

      K = 0
      DO J = 1, NOBS

c Initializing the T matrix to an identity matrix.

      DO II = 1,4
       DO JJ = 1,4
        T(II,JJ) = TIMAT(II,JJ)
       ENDDO
      ENDDO

c Calculate the actual manipulator joint angles.

      TH1 = DT1 + TET1(J)
      TH2 = DT2 + TET2(J)
      TH3 = DT3 + TET3(J)
      TH4 = DT4 + TET4(J)
      TH5 = DT5 + TET5(J)
      FI6 = DF6 + TET6(J)

c Compute the T matrices, T1 thru T6.

      CALL T3RPY( FI0, TH0, SI0, TRPY )
      CALL T3XYZ( PX0, PY0, PZ0, TXYZ )
      CALL MATMULC( T0, TRPY, TXYZ )

      CALL TRANSFORM ( AL1, AA1, DD1, TH1, BL1, T1 )
      CALL TRANSFORM ( AL2, AA2, DD2, TH2, BL2, T2 )
      CALL TRANSFORM ( AL3, AA3, DD3, TH3, BL3, T3 )
      CALL TRANSFORM ( AL4, AA4, DD4, TH4, BL4, T4 )
      CALL TRANSFORM ( AL5, AA5, DD5, TH5, BL5, T5 )

      CALL T3RPY( FI6, TH6, SI6, TRPY )
      CALL T3XYZ( PX6, PY6, PZ6, TXYZ )
      CALL MATMULC(T6, TRPY, TXYZ )

c Compute the complete transformation from the base frame to
c the tool frame.

      CALL MATMULA ( T, T0 )
      CALL MATMULA ( T, T1 )
      CALL MATMULA ( T, T2 )
      CALL MATMULA ( T, T3 )
      CALL MATMULA ( T, T4 )
      CALL MATMULA ( T, T5 )
      CALL MATMULA ( T, T6 )

c Read the simulated measured T matrix for this observation:

      DO II = 1,3
       DO JJ = 1,4
```

```
          TMJ(II,JJ) = TM(II,JJ,J)
         ENDDO
        ENDDO

c Remember that row 4 is 0, 0, 0, 1.

        TMJ(4,1) = 0.0
        TMJ(4,2) = 0.0
        TMJ(4,3) = 0.0
        TMJ(4,4) = 1.0

c Compute the difference between the measured T matrix and the
c T matrix generated using the latest table of kinematic data.

        CALL MATSUB ( TDELTA, TMJ, T )

c Calculate the function F (six rows at a time).

c First, position.

        K = K + 1
        F(K) = TDELTA(1,4)
        K = K + 1
        F(K) = TDELTA(2,4)
        K = K + 1
        F(K) = TDELTA(3,4)

c Now orientation.

        K = K + 1
        F(K) = ((TDELTA(3,2)-TDELTA(2,3))/2.0) * SCALE
        K = K + 1
        F(K) = ((TDELTA(1,3)-TDELTA(3,1))/2.0) * SCALE
        K = K + 1
        F(K) = ((TDELTA(2,1)-TDELTA(1,2))/2.0) * SCALE

c Ending the DO loop for J counter.

        ENDDO

c Write RMS error in function F.

        XSSQ=0.0

        DO II=1,6*NOBS
          XSSQ=XSSQ+F(II)*F(II)
        ENDDO

        XER=SQRT(XSSQ)

c Write RMS error to the screen. This allows the operator
c to track how the identification process is proceeding.

        WRITE(6,*) XER

        RETURN

        END
```

APPENDIX F PROGRAM VERIFY

```
      PROGRAM VERIFY

C This program generates the six-dof pose  error for the PUMA manipulator.
C It contains the identified calibration parameters and the exact parameter.
C It uses a data file of verification joint angle sets POSEVER.DAT, and the
C file RESULT.DAT from the program ID6.

      INTEGER I, J, K, NPOSE, N, nob
      REAL*8 DANGLE, DLENTH

      REAL*8 DT(5),dd(5),aa(5),al(5),bl(5)
      REAL*8 eDT(5),edd(5),eaa(5),eal(5),ebl(5)
      REAL*8 edf6, EFI6, ETH6, ESI6, EPX6, EPY6, EPZ6
      REAL*8 EDF0, EFI0, ETH0, ESI0, EPX0, EP60, EPZ0
      REAL*8 THETA(1000,6), TDELTA(4,4)
      REAL*8 T0(4,4), T1(4,4), T2(4,4), T3(4,4)
      REAL*8 T4(4,4), T5(4,4), T6(4,4), TRPY(4,4), TXYZ(4,4)
      REAL*8 TIMAT(4,4), T(4,4), et(4,4)

      REAL*8 DF0, DT0, DS0, PX0, PY0, PZ0
      REAL*8 DT1, DT2, DT3, DT4, DT5
      REAL*8 DD1, DD2, DD3, DD4, DD5
      REAL*8 AA1, AA2, AA3, AA4, AA5
      REAL*8 AL1, AL2, AL3, AL4, AL5
      REAL*8 BL1, BL2, BL3, BL4, BL5
      REAL*8 DF6, DT6, DS6, PX6, PY6, PZ6

      COMMON TIMAT,THETA

      EXTERNAL FKS

C Initialize the TIMAT matrix to an I matrix:

      DATA TIMAT/1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1/

C Open data file

      OPEN (9,  NAME='posever.DAT',STATUS='OLD')
      OPEN (10, NAME='input.DAT', STATUS='OLD')
      OPEN (11, NAME='result.DAT', STATUS='OLD')
      open (12, name='error.dat',access='append',status='unknown')

c Read input parameters

      read (10,*)
      read (10,*) df0,dt0,ds0,px0,py0,pz0
      read (10,*) dt1,dd1,aa1,al1,bl1
      read (10,*) dt2,dd2,aa2,al2,bl2
      read (10,*) dt3,dd3,aa3,al3,bl3
      read (10,*) dt4,dd4,aa4,al4,bl4
      read (10,*) dt5,dd5,aa5,al5,bl5
      read (10,*)
      read (10,*) df6,dt6,ds6,px6,py6,pz6
      read (10,*)
```

```
      read (10,*) nobs,n,dangle,dlenth,magnx,magnl

      CLOSE (10)


c Read in joint angle sets for verification poses


      do i=1,nobs
       read(9,*)theta(i,1),theta(i,2),theta(i,3),theta(i,4),
     &          theta(i,5),theta(i,6)
      enddo
      close (9)

C Set exact link parameters for the manipulator:

       DT(1)=0.0
       do i=2,5
          dt(i)=dangle
       enddo

      DF0 = DF0 + Dangle
      DT0 = DT0 + DANGLE
      DS0 = DS0 + DANGLE
      PX0 = PX0 + DLENTH
      PY0 = PY0 + DLENTH
      PZ0 = PZ0 + DLENTH

      al(1) = al1 + DANGLE
      al(2) = al2 + DANGLE
      al(3) = al3 + DANGLE
      al(4) = al4 + DANGLE
      al(5) = al5 + DANGLE

      AA(1) = aa1 + DLENTH
      AA(2) = aa2 + DLENTH
      AA(3) = aa3 + DLENTH
      AA(4) = aa4 + DLENTH
      AA(5) = aa5 + DLENTH

      DD(1) = dd1      ! defined
      DD(2) = dd2      ! defined
      DD(3) = dd3 + DLENTH
      DD(4) = dd4 + DLENTH
      DD(5) = dd5 + DLENTH

      BL(1) = bl1              ! defined
      BL(2) = bl2 + DANGLE
      BL(3) = bl3              ! defined
      BL(4) = bl4              ! defined
      BL(5) = bl5              ! defined

      DF6 = DF6 + Dangle
      DT6 = DT6 + DANGLE
      DS6 = DS6 + DANGLE
      PX6 = PX6 + DLENTH
      PY6 = PY6 + DLENTH
      PZ6 = PZ6 + DLENTH

c Read in and set up estimated parameter table
```

```
      READ (11,*)
      READ (11,*)
      READ (11,*) EDF0,ETH0,ESI0,EPX0,EPY0,EPZ0
      do i=1,5
       read (11,*)
       read (11,*)
       read (11,*) edt(i),edd(i),eaa(i),eal(i),ebl(i)
      enddo

      read(11,*)
      read(11,*)
      read(11,*) edf6,eth6,esi6,epx6,epy6,epz6

c Main loop through nobs joint angle sets

      do k=1,nobs

      call fks (k,DF0,DT0,DS0,PX0,PY0,PZ0,dt,al,aa,dd,bl,df6,
     &          DT6,DS6,PX6,PY6,PZ6,t)
      call fks (k,EDF0,ETH0,ESI0,EPX0,EPY0,EPZ0,edt,eal,eaa,edd,
     &          ebl,edf6,eth6,esi6,epx6,epy6,epz6,et)

c Compute the differential tool matrix

      call matsub (tdelta,t,et)

c Compute the pose errors

      poserr=sqrt(tdelta(1,4)**2+tdelta(2,4)**2+tdelta(3,4)**2)
      orerr1=(tdelta(3,2)-tdelta(2,3))/2
      orerr2=(tdelta(1,3)-tdelta(3,1))/2
      orerr3=(tdelta(2,1)-tdelta(1,2))/2
      orerr=sqrt(orerr1**2+orerr2**2+orerr3**2)

c Update total error counts

      posterr=(poserr+(k-1)*posterr)/k
      orterr =(orerr +(k-1)*orterr)/k

c End of main loop

      enddo

       write (6,*) 'Position error, orientation error'
       write (6,*) posterr,orterr

c     write (6,*) 'how many nobs in this run?'
c     read  (6,*) nob
      nob = 12
      write (12,*) 'for nobs of:',nob
      write (12,*) 'position error, orientation error'
      write (12,*) posterr,orterr
      write (12,*)

      close (12)

      end

c ********************************************************************

      subroutine fks (n,DF0,DT0,DS0,PX0,PY0,PZ0,dt,al,aa,dd,bl,
```

```
     &                    df6,th6,si6,px6,py6,pz6,t)

          REAL*8 T0(4,4), T1(4,4), T2(4,4), T3(4,4)
          REAL*8 T4(4,4), T5(4,4), T6(4,4), TRPY(4,4), TXYZ(4,4)
          REAL*8 TIMAT(4,4), T(4,4), dt(5),al(5),aa(5),dd(5),bl(5)
          REAL*8 DF0,DT0,DS0,PX0,PY0,PZ0,DF6,TH6,SI6,PX6,PY6,PZ6,FI6
          real*8 theta(1000,6), ang(6)
          common timat,theta

C Initialize the T matrix to an I matrix:

          DO J=1,4
          DO K=1,4
                T(J,K) = TIMAT(J,K)
          ENDDO
          ENDDO

C Set up joint angles


          do i=1,5
          ang(i)=theta(n,i)+dt(i)
          enddo

          fi6=df6+theta(n,6)

C Compute the T matrices, T1 thru T6:

          CALL T3RPY (DF0,DT0,DS0,TRPY )
          CALL T3XYZ (px0,py0,pz0,TXYZ )
          CALL MATMULC ( T0, TRPY, TXYZ )

          CALL TRANSFORM (al(1),aa(1),dd(1),ang(1),bl(1),T1)
          CALL TRANSFORM (al(2),aa(2),dd(2),ang(2),bl(2),T2)
          CALL TRANSFORM (al(3),aa(3),dd(3),ang(3),bl(3),T3)
          CALL TRANSFORM (al(4),aa(4),dd(4),ang(4),bl(4),T4)
          CALL TRANSFORM (al(5),aa(5),dd(5),ang(5),bl(5),T5)

          CALL T3RPY (fi6,th6,si6,TRPY )
          CALL T3XYZ (px6,py6,pz6,TXYZ )
          CALL MATMULC ( T6, TRPY, TXYZ )

C Compute the overall transformation, T:

          CALL MATMULA ( T, T0 )
          CALL MATMULA ( T, T1 )
          CALL MATMULA ( T, T2 )
          CALL MATMULA ( T, T3 )
          CALL MATMULA ( T, T4 )
          CALL MATMULA ( T, T5 )
          CALL MATMULA ( T, T6 )

          return
          end
```

APPENDIX G PROGRAM LINSC

```
      PROGRAM linsc

C  This program generates joint angles for the Puma manipulator
C  arm.  It presumes that the tool frame of the manipulator is
C  constrained to move in the positive x direction only.  The
C  tool is constrained by a sliding linear scale.  The values along
C  the x direction are determined by a random number generator.  The
C  orientation of the tool is constrained as well.  The x direction of
C  tool is in the x direction of the WCF, the y direction of the tool
C  is in the z direction of the WCF and the z direction of the tool is
C  in the negative y direction of the WCF.


      INTEGER LDFJAC, M, N, obs, nobs
      PARAMETER (LDFJAC=12, M=LDFJAC, N=6)

      real*8 fi0, th0, si0, px0, py0, pz0
      REAL*8 DT1, DT2, DT3, DT4, DT5
      REAL*8 DD1, DD2, DD3, DD4, DD5
      REAL*8 AA1, AA2, AA3, AA4, AA5
      REAL*8 AL1, AL2, AL3, AL4, AL5
      REAL*8 BL1, BL2, BL3, BL4, BL5
      REAL*8 DF6, FI6, TH6, SI6, PX6, PY6, PZ6

      REAL*8 RN1,RN2,RN3,RN4,RN5,RN6
      REAL*8 RN7,RN8,RN9,RN10,RN11,RN12
      REAL*8 RN13,RN14,RN15,RN16,RN17,RN18

      INTEGER infer,ier,iopt,nsig,maxfn

      REAL*8 FJAC(LDFJAC,N), xjtj((n+1)*n/2), xjac(ldfjac,n)
      REAL*8 parm(4), f(ldfjac), work((5*n)+(2*m)+((n+1)*n/2))
      REAL*8 X(N)
      REAL*8 magnx,magn1

      EXTERNAL PUMA_ARM

      INTEGER I, J, K, nou
      REAL*8 TDES(4,4), T(4,4), SCALE, DANGLE, DLENTH, NUM

      COMMON /PDATA/ TDES, DANGLE, DLENTH, T
      COMMON /KIN/ fi0, th0, si0, px0, py0, pz0,
     &             DT1,DT2,DT3,DT4,DT5,
     &             AL1,AL2,AL3,AL4,AL5,
     &             AA1,AA2,AA3,AA4,AA5,
     &             DD1,DD2,DD3,DD4,DD5,
     &             BL1,BL2,BL3,BL4,BL5,
     &             DF6,TH6,SI6,PX6,PY6,PZ6

C Initialize data variables
```

```
      obs=0

C Open data files for input

      OPEN (10, NAME='puma-pos.dat', STATUS='NEW')
      OPEN (9, NAME='input.dat', STATUS='OLD')

C Read input kinematic data

      read (9,*)
      read (9,*) fi0,th0,si0,px0,py0,pz0
      read (9,*) dt1,dd1,aa1,al1,bl1
      read (9,*) dt2,dd2,aa2,al2,bl2
      read (9,*) dt3,dd3,aa3,al3,bl3
      read (9,*) dt4,dd4,aa4,al4,bl4
      read (9,*) dt5,dd5,aa5,al5,bl5
      read (9,*)
      read (9,*) df6,th6,si6,px6,py6,pz6
      read (9,*)
      read (9,*) nobs,nou,dangle,dlenth,magnx,magnl


      close (9)

C Adjust nominal values

      fi0=fi0+dangle
      th0=th0+dangle
      si0=si0+dangle
      px0=px0+dlenth
      py0=py0+dlenth
      pz0=pz0+dlenth

      dt1=0.0
      dt2=dt2+dangle
      dt3=dt3+dangle
      dt4=dt4+dangle
      dt5=dt5+dangle

      al1=al1+dangle
      al2=al2+dangle
      al3=al3+dangle
      al4=al4+dangle
      al5=al5+dangle

      aa1=aa1+dlenth
      aa2=aa2+dlenth
      aa3=aa3+dlenth
      aa4=aa4+dlenth
      aa5=aa5+dlenth

      dd1=0.0
      dd2=0.0
      dd3=dd3+dlenth
      dd4=dd4+dlenth
      dd5=dd5+dlenth

      bl1=bl1
      bl2=bl2+dangle
      bl3=bl3
      bl4=bl4
```

```
      bl5=bl5

      df6=df6+dangle
      th6=th6+dangle
c     si6=si6+dangle
c     px6=px6+dlenth
c     py6=py6+dlenth
c     pz6=pz6+dlenth

C Get random number seed

      write (6,*) 'Type in a 6-digit random number seed'
      read (5,*) iseed

C Start of main loop

 1010 obs=obs+1

C Set joint angles to zero

      do i=1,n
      x(i)=0.0
      enddo

C Get random bar angles

 1000 call random (iseed,num)
      num=num*900.0

C Establish desired tool pose

      do ii=1,4
      do jj=1,4
            TDES(ii,jj)=0.0
      enddo
      enddo

      TDES(1,4)= num
      TDES(4,4)=  1.0
      TDES(1,1)=  1.0
      TDES(3,2)=  1.0
      TDES(2,3)= -1.0

C Call IMSL ZXSSQ for inverse kinematic solution

      nsig=4
      eps=0.0
      delta=0.0
      maxfn=500
      iopt=1
      ixjac=ldfjac

      CALL ZXSSQ(puma_arm,m,n,nsig,eps,delta,maxfn,iopt,parm,x,
     &                  ssq,f,xjac,ixjac,xjtj,work,infer,ier)

C Check for singularities

      if (ssq .gt. 0.00001) goto 1000

C Print results to 2 decimal places
```

```
      write(6,*) obs,ssq

C Generate the random noise

      CALL RANDOM (ISEED,RN1)
      CALL RANDOM (ISEED,RN2)
      CALL RANDOM (ISEED,RN3)
      CALL RANDOM (ISEED,RN4)
      CALL RANDOM (ISEED,RN5)
      CALL RANDOM (ISEED,RN6)
      CALL RANDOM (ISEED,RN7)
      CALL RANDOM (ISEED,RN8)
      CALL RANDOM (ISEED,RN9)
      CALL RANDOM (ISEED,RN10)
      CALL RANDOM (ISEED,RN11)
      CALL RANDOM (ISEED,RN12)
      CALL RANDOM (ISEED,RN13)
      CALL RANDOM (ISEED,RN14)
      CALL RANDOM (ISEED,RN15)
      CALL RANDOM (ISEED,RN16)
      CALL RANDOM (ISEED,RN17)
      CALL RANDOM (ISEED,RN18)

      RN1 =  MAGNX * (2.0 * RN1 - 1.0)
      RN2 =  MAGNX * (2.0 * RN2 - 1.0)
      RN3 =  MAGNX * (2.0 * RN3 - 1.0)

      RN4 =  MAGN1 * (2.0 * RN4 - 1.0)
      RN5 =  MAGN1 * (2.0 * RN5 - 1.0)
      RN6 =  MAGN1 * (2.0 * RN6 - 1.0)
      RN7 =  MAGN1 * (2.0 * RN7 - 1.0)
      RN8 =  MAGN1 * (2.0 * RN8 - 1.0)
      RN9 =  MAGN1 * (2.0 * RN9 - 1.0)
      RN10 = MAGN1 * (2.0 * RN10 - 1.0)
      RN11 = MAGN1 * (2.0 * RN11 - 1.0)
      RN12 = MAGN1 * (2.0 * RN12 - 1.0)
      RN13 = MAGN1 * (2.0 * RN13 - 1.0)
      RN14 = MAGN1 * (2.0 * RN14 - 1.0)
      RN15 = MAGN1 * (2.0 * RN15 - 1.0)
      RN16 = MAGN1 * (2.0 * RN16 - 1.0)
      RN17 = MAGN1 * (2.0 * RN17 - 1.0)
      RN18 = MAGN1 * (2.0 * RN18 - 1.0)

     T(1,4) = T(1,4) + RN1
     T(2,4) = T(2,4) + RN2
     T(3,4) = T(3,4) + RN3

     T(1,1) = T(1,1) + RN4
     T(1,2) = T(1,2) + RN5
     T(1,3) = T(1,3) + RN6

     T(2,1) = T(2,1) + RN7
     T(2,2) = T(2,2) + RN8
     T(2,3) = T(2,3) + RN9

     T(3,1) = T(3,1) + RN10
     T(3,2) = T(3,2) + RN11
     T(3,3) = T(3,3) + RN12

      X(1) = X(1) + RN13
      X(2) = X(2) + RN14
```

```
         X(3) = X(3) + RN15
         X(4) = X(4) + RN16
         X(5) = X(5) + RN17
         X(6) = X(6) + RN18

        write (10,*) X(1),X(2),X(3),X(4),X(5),X(6)
        write (10,*)
        write (10,*) T(1,1),T(1,2),T(1,3),T(1,4)
        write (10,*) T(2,1),T(2,2),T(2,3),T(2,4)
        write (10,*) T(3,1),T(3,2),T(3,3),T(3,4)
        write (10,*)

 991    FORMAT ( 6F12.6 )
 992    FORMAT ( 3F16.10, F12.5 )

C Continue for other bar angles

        if (obs .lt. nobs) go to 1010

        CLOSE (10)

        END

C ***************************************************************

        SUBROUTINE PUMA_ARM (X,M,N,F)

C This subroutine calculates the non-linear function for the use of
C the IMSL routine ZXSSQ. It is the forward kinematic solution for
C the PUMA manipulator.

        INTEGER M, N
        REAL*8 X(N), F(M)

        INTEGER II, JJ
        real*8 fi0, th0, si0, px0, py0, pz0
        REAL*8 DT1, DT2, DT3, DT4, DT5
        REAL*8 DD1, DD2, DD3, DD4, DD5
        REAL*8 AA1, AA2, AA3, AA4, AA5
        REAL*8 AL1, AL2, AL3, AL4, AL5
        REAL*8 BL1, BL2, BL3, BL4, BL5
        REAL*8 DF6, FI6, TH6, SI6, PX6, PY6, PZ6

        REAL*8 TH1, TH2, TH3, TH4, TH5
        REAL*8 T0(4,4), T1(4,4), T2(4,4), T3(4,4), T4(4,4)
        REAL*8 T5(4,4), T6(4,4), trpy(4,4), txyz(4,4)
        REAL*8 TIMAT(4,4), T(4,4), td(4,4)

        INTEGER I, J, K
        REAL*8 TDES(4,4), DANGLE, DLENTH,scale

        COMMON /PDATA/ TDES, DANGLE, DLENTH, T
        COMMON /KIN/ fi0,th0,si0,px0,py0,pz0,
     &               DT1,DT2,DT3,DT4,DT5,
     &               AL1,AL2,AL3,AL4,AL5,
     &               AA1,AA2,AA3,AA4,AA5,
     &               DD1,DD2,DD3,DD4,DD5,
     &               BL1,BL2,BL3,BL4,BL5,
     &               DF6,TH6,SI6,PX6,PY6,PZ6

C Initialize the TIMAT matrix to an I matrix:
```

```
      DATA TIMAT/1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1/

      scale=100.0

C Initialize the T matrix to an I matrix

      DO II = 1,4
      DO JJ = 1,4
            T(II,JJ) = TIMAT(II,JJ)
      ENDDO
      ENDDO

C Manipulator joint angles

      TH1 = DT1 + X(1)
      TH2 = DT2 + X(2)
      TH3 = DT3 + X(3)
      TH4 = DT4 + X(4)
      TH5 = DT5 + X(5)
      FI6 = DF6 + X(6)

C Compute the T matrices, T1 thru T6:

      call t3rpy (fi0,th0,si0, trpy)
      call t3xyz (px0,py0,pz0, txyz)
      call matmulc (t0,trpy,txyz)

      CALL TRANSFORM ( AL1, AA1, DD1, TH1, BL1, T1 )
      CALL TRANSFORM ( AL2, AA2, DD2, TH2, BL2, T2 )
      CALL TRANSFORM ( AL3, AA3, DD3, TH3, BL3, T3 )
      CALL TRANSFORM ( AL4, AA4, DD4, TH4, BL4, T4 )
      CALL TRANSFORM ( AL5, AA5, DD5, TH5, BL5, T5 )

      CALL t3rpy ( fi6, th6, si6, trpy )
      CALL T3XYZ ( PX6, PY6, PZ6, txyz )
      CALL matmulc ( t6, trpy, txyz )

C Compute the overall transformation, T:

      CALL MATMULA ( T, T0 )
      CALL MATMULA ( T, T1 )
      CALL MATMULA ( T, T2 )
      CALL MATMULA ( T, T3 )
      CALL MATMULA ( T, T4 )
      CALL MATMULA ( T, T5 )
      CALL MATMULA ( T, T6 )

      call matsub(td,tdes,t)

C Calculate the function F

      f(1)=td(1,4)
      f(2)=td(2,4)
      f(3)=td(3,4)
      f(4)=td(1,1)*scale
      f(5)=td(1,2)*scale
      f(6)=td(1,3)*scale
      f(7)=td(2,1)*scale
      f(8)=td(2,2)*scale
      f(9)=td(2,3)*scale
```

```
        f(10)=td(3,1)*scale
        f(11)=td(3,2)*scale
        f(12)=td(3,3)*scale

       sum = 0.0
       do i=1,12
        xssq=sum+f(i)
       enddo
C      write (6,*) xssq

        RETURN
        END

C ***************************************************************

        SUBROUTINE RANDOM (x,z)

C This subroutine generates random numbers in the range 0-1
C using a supplied seed x, the returned random number being z.

        REAL FM, FX, Z
        INTEGER A, X, I, M
        DATA I/1/

        IF ( I .EQ. 0 ) GO TO 1000
        I=0
        M= 2 ** 20
        FM= M
        A= 2**10 + 3

 1000   X= MOD( A*X ,M)
        FX= X
        Z= FX/ FM

        RETURN
        END
```

APPENDIX H PROGRAM ACDAT

```
  program acdat

C  This program is the data acquisition program for the linear
C  slide version of the puma manipulator calibration experiments.
C  It requires that the value for the distance down the linear
C  slide and the six joint angles of the puma when it is in that
C  configuration.  The pose data is saved in the file "slide-pos.dat."
C  That file is set up in the append mode.

        real*8 x, th1, th2, th3, th4, th5, th6
        real*8 tm(4,4)

        open (10, name='slide-pos.dat',access='append',status='unknown')

C  Inputting the distance down the linear slide.

100     write (6,*) 'input the value for x'
        read  (6,*) x

C  Inputting the six joint angles.

        write (6,*) 'input the 6 joint angles'
        read  (6,*) th1, th2, th3, th4, th5, th6

C  This will set the T6 matrix for the value of x measured.

        do ii=1,4
          do jj=1,4
            tm(ii,jj)=0.0
          enddo
        enddo

        tm(1,4)=    x
        tm(4,4)=   1.0
        tm(1,1)=   1.0
        tm(3,2)=   1.0
        tm(2,3)=  -1.0

C  Writing the pose data to "slide-pos.dat"

        write (10,991) th1, th2, th3, th4, th5, th6
        write (10,*)
        write (10,992) tm(1,1),tm(1,2),tm(1,3),tm(1,4)
        write (10,992) tm(2,1),tm(2,2),tm(2,3),tm(2,4)
        write (10,992) tm(3,1),tm(3,2),tm(3,3),tm(3,4)
        write (10,992) tm(4,1),tm(4,2),tm(4,3),tm(4,4)
        write (10,*)

        write (6,*)
        write (6,*) 'data saved'
        write (6,*)
```

```
      write (6,*) 'do you wish to make another observation'
      write (6,*) 'enter a 1 for yes or a 0 for no'
      write (6,*)
      read  (6,*) ans

      if (ans .eq. 1) go to 100

991   FORMAT ( 6F12.6 )
992   FORMAT ( 3F16.10, F12.5 )

      close(10)
      stop

      end
```

APPENDIX I PROGRAM ID6_LINSC

```
      PROGRAM ID6_LINSC

C Robot Identification using the Non-linear Least Squares method.
C Simulation data is read for the PUMA manipulator from
C the data file PUMA-POS.DAT

C Change parameter LDFJAC to change the number of observations,
C set LDFJAC = 6 * Number of observations

      INTEGER LDFJAC, MM, M, NN, N, NSIG, MAXFN, IOPT, IXJAC, INFER, IER
      PARAMETER (LDFJAC=12*42, MM=LDFJAC, NN=26)

      REAL*8 FJAC(LDFJAC,NN), XJTJ((NN+1)*NN/2)
      REAL*8 PARM(4), F(LDFJAC), WORK((5*NN)+(2*MM)+((NN+1)*NN/2))
      REAL*8 X(NN)
      EXTERNAL PUMA_ARM

      REAL*8 DANGLE, DLENTH, TQ, DQ, EPS, DELTA, SSQ
      REAL*8 SQERR1, SQERR2

      real*8 fi0,th0,si0,px0,py0,pz0
      REAL*8 DT1, DT2, DT3, DT4, DT5
      REAL*8 DD1, DD2, DD3, DD4, DD5
      REAL*8 AA1, AA2, AA3, AA4, AA5
      REAL*8 AL1, AL2, AL3, AL4, AL5
      REAL*8 BL1, BL2, BL3, BL4, BL5
      REAL*8 DF6, TH6, SI6, PX6, PY6, PZ6, FI6

      INTEGER I, J, K, NOBS, MAXNOBS
      PARAMETER (MAXNOBS=360)
      REAL*8 TET1(MAXNOBS), TET2(MAXNOBS), TET3(MAXNOBS)
      REAL*8 TET4(MAXNOBS), TET5(MAXNOBS), TET6(MAXNOBS)
      REAL*8 TM(3,4,MAXNOBS), SCALE
      COMMON /PDATA/ NOBS, TM, SCALE,
     &          TET1, TET2, TET3, TET4, TET5, TET6

C Open data files for inputs and results

      OPEN (8, NAME='RESULT.DAT', STATUS='NEW')
      OPEN (9, NAME='PUMA-POS.DAT', STATUS='OLD')
      OPEN (10,NAME='INPUT.DAT', STATUS='OLD')

c Read input parameters

      read (10,*)
      read (10,*) fi0,th0,si0,px0,py0,pz0
      read (10,*) dt1,dd1,aa1,al1,bl1
      read (10,*) dt2,dd2,aa2,al2,bl2
      read (10,*) dt3,dd3,aa3,al3,bl3
      read (10,*) dt4,dd4,aa4,al4,bl4
      read (10,*) dt5,dd5,aa5,al5,bl5
      read (10,*)
      read (10,*) df6,th6,si6,px6,py6,pz6
```

```
      read (10,*)
      read (10,*) nobs,n,dangle,dlenth,magnx,magnl

      CLOSE (10)

      write (6,*) 'enter nobs'
      read (6,*) nobs

C Initialize data variables

      X(1)=fi0
      X(2)=th0
      X(3)=si0
      X(4)=px0
      x(5)=py0
      x(6)=pz0

      X(7)=AA1
      X(8)=AL1

      X(9)=DT2
      X(10)=AA2
      X(11)=AL2
      X(12)=BL2

      X(13)=DT3
      X(14)=DD3
      X(15)=AA3
      X(16)=AL3

      X(17)=DT4
      X(18)=DD4
      X(19)=AA4
      X(20)=AL4

      X(21)=DT5
      X(22)=DD5
      X(23)=AA5
      X(24)=AL5

      x(25)=df6
      x(26)=th6
c     x(27)=si6
c     x(28)=px6
c     x(29)=py6
c     x(30)=pz6

C Read simulated joint data and tool pose

      DO J = 1, NOBS
        READ (9,*) TET1(J), TET2(J), TET3(J), TET4(J), TET5(J), TET6(J)
        read (9,*)
        READ (9,*) TM(1,1,J), TM(1,2,J), TM(1,3,J), TM(1,4,J)
        READ (9,*) TM(2,1,J), TM(2,2,J), TM(2,3,J), TM(2,4,J)
        READ (9,*) TM(3,1,J), TM(3,2,J), TM(3,3,J), TM(3,4,J)
        READ (9,*)
c       READ (9,*)
      ENDDO
      CLOSE (9)

C Initialize scale for the angular rows of the Jacobian
```

```
      SCALE=100.0

C Call IMSL routine for non-linear identification

      NSIG=3
      EPS=0.0
      DELTA=0.0
      MAXFN=1500
      IOPT=1
      IXJAC=LDFJAC
      M=12*NOBS

      CALL ZXSSQ(PUMA_ARM,M,N,NSIG,EPS,DELTA,MAXFN,IOPT,
     &           PARM,X,SSQ,F,FJAC,IXJAC,XJTJ,WORK,INFER,IER)

C Save results to data file

      WRITE (8,*)
      WRITE (8,*) 'fi0, th0, si0, px0, py0, pz0'
      WRITE (8,888) X(1), X(2), X(3), X(4), x(5), x(6)
      WRITE (8,*)
      WRITE (8,*) 'DT1, DD1, AA1, AL1, BL1'
      WRITE (8,888) 0.0, 0.0, X(7), X(8), 0.0
      WRITE (8,*)
      WRITE (8,*) 'DT2, DD2, AA2, AL2, BL2'
      WRITE (8,888) X(9), 0.0, X(10), X(11), X(12)
      WRITE (8,*)
      WRITE (8,*) 'DT3, DD3, AA3, AL3, BL3'
      WRITE (8,888) X(13), X(14), X(15), X(16), 0.0
      WRITE (8,*)
      WRITE (8,*) 'DT4, DD4, AA4, AL4, BL4'
      WRITE (8,888) X(17), X(18), X(19), X(20), 0.0
      WRITE (8,*)
      WRITE (8,*) 'DT5, DD5, AA5, AL5, BL5'
      WRITE (8,888) X(21), X(22), X(23), X(24), 0.0
      WRITE (8,*)
      WRITE (8,*) 'DF6, TH6, SI6, PX6, PY6, PZ6'
      WRITE (8,889) x(25),x(26), si6, px6, py6, pz6

888   FORMAT ( 5F12.5 )
889   FORMAT ( 6F12.5 )

C Calculate root mean square error in identification

      TQ = DANGLE
      DQ = DLENTH

C Error in identification (angular parameters)

      SQERR1 =
     & (FI0+TQ-X(1))**2 +(TH0+TQ-X(2))**2 +(SI0+TQ-X(3))**2
     & +(DT3+TQ-X(13))**2 +(DT4+TQ-X(17))**2 +(DT5+TQ-X(21))**2
     & +(AL1+TQ-X(8))**2 +(AL2+TQ-X(11))**2
     & +(AL3+TQ-X(16))**2 +(AL4+TQ-X(20))**2 +(AL5+TQ-X(24))**2
     & +(BL2+TQ-X(12))**2 +(DT2+TQ-X(9))**2
     & +(df6+tq-x(25))**2 +(th6+tq-x(26))**2
      SQERR1 = DSQRT( SQERR1/15 )

C Error in identification (length parameters)
```

```
      SQERR2 =
     &  (PX0+DQ-X(4))**2 +(AA1+0.0+DQ-X(7))**2 +(AA2+DQ-X(10))**2
     &  +(AA3+DQ-X(15))**2 +(AA4+DQ-X(19))**2 +(AA5+DQ-X(23))**2
     &  +(PY0+DQ-X(5))**2 +(PZ0+DQ-X(6))**2
     &  +(DD3+DQ-X(14))**2 +(DD4+DQ-X(18))**2 +(DD5+DQ-X(22))**2
      SQERR2 = DSQRT( SQERR2/11 )

      WRITE (8,*)
      WRITE (8,*) 'RMS PARMS (LENGTH), RMS PARMS (ANGLE)'
      WRITE (8,*) SQERR2, SQERR1
      WRITE (6,*) 'RMS PARMS (LENGTH), RMS PARMS (ANGLE)'
      WRITE (6,*) SQERR2,SQERR1


      WRITE (8,*)
      WRITE (8,*) 'INFER, IER,NOBS,NSIG'
      WRITE (8,*)  INFER, IER,NOBS,NSIG
      WRITE (6,*) 'INFER, IER,NOBS,NSIG'
      WRITE (6,*)  INFER, IER,NOBS,NSIG

      WRITE (8,*)

      CLOSE (8)

      END

C *****************************************************************

      SUBROUTINE PUMA_ARM (X, M, N, F)

C This subroutine calculates the non-linear function for the use of
C the IMSL routine DUNLSF. It is the forward kinematic solution for
C the PUMA manipulator.

      INTEGER M, N
      REAL*8 X(N), F(M)

      INTEGER II, JJ

      real*8 fi0,th0,si0,px0,py0,pz0
      REAL*8 DT1, DT2, DT3, DT4, DT5
      REAL*8 DD1, DD2, DD3, DD4, DD5
      REAL*8 AA1, AA2, AA3, AA4, AA5
      REAL*8 AL1, AL2, AL3, AL4, AL5
      REAL*8 BL1, BL2, BL3, BL4, BL5
      REAL*8 FI6, TH6, SI6, PX6, PY6, PZ6, DF6

      REAL*8 TH1, TH2, TH3, TH4, TH5
      REAL*8 T0(4,4), T1(4,4), T2(4,4), T3(4,4), T4(4,4)
      REAL*8 T5(4,4), T6(4,4), TRPY(4,4), TXYZ(4,4)
      REAL*8 TIMAT(4,4), T(4,4)
      REAL*8 TINV(4,4), TMJ(4,4), TDELTA(4,4)

      INTEGER I, J, K, NOBS, MAXNOBS
      PARAMETER (MAXNOBS=360)
      REAL*8 TET1(MAXNOBS), TET2(MAXNOBS), TET3(MAXNOBS)
      REAL*8 TET4(MAXNOBS), TET5(MAXNOBS), TET6(MAXNOBS)
      REAL*8 TM(3,4,MAXNOBS), SCALE
      COMMON /PDATA/ NOBS, TM, SCALE,
     &         TET1, TET2, TET3, TET4, TET5, TET6
```

```
C Initialize the TIMAT matrix to an I matrix:

      DATA TIMAT/1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1/

C Set parameters for the manipulator:

      fi0 = X(1)
      th0 = X(2)
      si0 = X(3)
      px0 = X(4)
      py0 = x(5)
      pz0 = x(6)

      DT1 = 0.0
      DD1 = 0.0
      AA1 = X(7)
      AL1 = X(8)
      BL1 = 0.0

      DT2 = X(9)
      DD2 = 0.0
      AA2 = X(10)
      AL2 = X(11)
      BL2 = X(12)

      DT3 = X(13)
      DD3 = X(14)
      AA3 = X(15)
      AL3 = X(16)
      BL3 = 0.0

      DT4 = X(17)
      DD4 = X(18)
      AA4 = X(19)
      AL4 = X(20)
      BL4 = 0.0

      DT5 = X(21)
      DD5 = X(22)
      AA5 = X(23)
      AL5 = X(24)
      BL5 = 0.0

      DF6 = x(25)
      TH6 = x(26)
      SI6 = 0.0  !x(27)
      px6 = 0.0  !x(28)
      py6 = 0.0  !x(29)
      pz6 = 55.9 !x(30)

C Loop NOBS times

      K = 0
      DO J = 1, NOBS

C Initialize the T matrix to an I matrix

      DO II = 1,3
      DO JJ = 1,4
            T(II,JJ) = TIMAT(II,JJ)
      ENDDO
```

```
      ENDDO

C Manipulator joint angles

      TH1 = DT1 + TET1(J)
      TH2 = DT2 + TET2(J)
      TH3 = DT3 + TET3(J)
      TH4 = DT4 + TET4(J)
      TH5 = DT5 + TET5(J)
      FI6 = DF6 + TET6(J)

C Compute the T matrices, T1 thru T6:

      call t3rpy(fi0,th0,si0, trpy)
      call t3xyz(px0,py0,pz0, txyz)
      call matmulc(t0,trpy,txyz)

      CALL TRANSFORM ( AL1, AA1, DD1, TH1, BL1, T1 )
      CALL TRANSFORM ( AL2, AA2, DD2, TH2, BL2, T2 )
      CALL TRANSFORM ( AL3, AA3, DD3, TH3, BL3, T3 )
      CALL TRANSFORM ( AL4, AA4, DD4, TH4, BL4, T4 )
      CALL TRANSFORM ( AL5, AA5, DD5, TH5, BL5, T5 )

      CALL T3RPY ( FI6, TH6, SI6, TRPY )
      CALL T3XYZ ( PX6, PY6, PZ6, TXYZ )
      CALL MATMULC (T6, TRPY, TXYZ )

C Compute the overall transformation, T:

      CALL MATMULA ( T, T0 )
      CALL MATMULA ( T, T1 )
      CALL MATMULA ( T, T2 )
      CALL MATMULA ( T, T3 )
      CALL MATMULA ( T, T4 )
      CALL MATMULA ( T, T5 )
      CALL MATMULA ( T, T6 )

C Get the "T-measured" matrix for this observation:

      DO II = 1,4
       DO JJ = 1,4
             TMJ(II,JJ) = tm(ii,jj,j)
       ENDDO
      ENDDO

c     tmj(3,2)=-1.0
c     tmj(2,3)=1.0

      TMJ(4,1) = 0.0
      TMJ(4,2) = 0.0
      TMJ(4,3) = 0.0
      TMJ(4,4) = 1.0

C Compute the elements of the "delta-Tn":

      CALL MATSUB ( TDELTA, TMJ, T )

C Calculate the function F (six rows at a time)

      f(k+1)=tdelta(1,4)
      f(k+2)=tdelta(2,4)
```

```
        f(k+3)=tdelta(3,4)

        f(k+4)=tdelta(1,1)*scale
        f(k+5)=tdelta(1,2)*scale
        f(k+6)=tdelta(1,3)*scale
        f(k+7)=tdelta(2,1)*scale
        f(k+8)=tdelta(2,2)*scale
        f(k+9)=tdelta(2,3)*scale
        f(k+10)=tdelta(3,1)*scale
        f(k+11)=tdelta(3,2)*scale
        f(k+12)=tdelta(3,3)*scale

        k=k+12

C End the do-loop for counter J

        ENDDO

C Write RMS error in F

        XSSQ=0.0
        DO II=1,12*NOBS
          XSSQ=XSSQ+F(II)*F(II)
        ENDDO

        XER=SQRT(XSSQ)
          WRITE(6,*) XER

        RETURN
        END
```

LIST OF REFERENCES

1. Mooring, Roth, and Driels, *Fundamentals of Manipulator Calibration*, p. 5, John Wiley and Sons, Inc., 1991.

2. Driels, M. R., Swayze, W., and Potter, S. A., "Full Pose Calibration of a Robot Manipulator Using a Coordinate Measuring Machine," Submitted for Publication, p. 1, 1991.

3. Mooring, Roth, and Driels, *Fundamentals of Manipulator Calibration*, John Wiley and Sons, Inc., 1991.

4. Driels, M. R., Swayze, W., and Potter, S. A., "Full Pose Calibration of a Robot Manipulator Using a Coordinate Measuring Machine," Submitted for Publication. p. 1, 1991.

5. Driels, M. R., "Using Passive End-Point Motion Constraints to Calibrate Kinematic Mechanisms," Submitted for Publication, 1991.

6. Paul, Robot Manipulators: Mathematics, Programming and Control, The MIT Press, 1982.

7. Mooring, Roth, and Driels, *Fundamentals of Manipulator Calibration*, John Wiley and Sons, Inc., 1991.

8. Driels, M. R., Swayze, W., and Potter, S. A., "Full Pose Calibration of a Robot Manipulator Using a Coordinate Measuring Machine," Submitted for Publication, p. 5, 1991.

9. Driels, M. R., Swayze, W., and Potter, S. A., "Full Pose Calibration of a Robot Manipulator Using a Coordinate Measuring Machine," Submitted for Publication, p. 4, 1991.

10. Driels, M. R., Swayze, W., and Potter, S. A., "Full Pose Calibration of a Robot Manipulator Using a Coordinate Measuring Machine," Submitted for Publication, p. 12, 1991.

11. Driels, M. R., "Using Passive End-Point Motion Constraints to Calibrate Kinematic Mechanisms," Submitted for Publication, p. 7, 1991.

12. Driels, M. R., "Using Passive End-Point Motion Constraints to Calibrate Kinematic Mechanisms," Submitted for Publication, p. 9, 1991.

13. Paul, Robot Manipulators: Mathematics, Programming and Control, The MIT Press, 1982.

14. Driels, M. R., Swayze, W., and Potter, S. A., "Full Pose Calibration of a Robot Manipulator Using a Coordinate Measuring Machine," Submitted for Publication, p. 5-6, 1991.

15. Driels, M. R., Swayze, W., and Potter, S. A., "Full Pose Calibration of a Robot Manipulator Using a Coordinate Measuring Machine," Submitted for Publication, p. 13, 1991.

16. Driels, M. R., Swayze, W., and Potter, S. A., "Full Pose Calibration of a Robot Manipulator Using a Coordinate Measuring Machine," Submitted for Publication, p. 9, 1991.

17. Mooring, Roth, and Driels, *Fundamentals of Manipulator Calibration*, John Wiley and Sons, Inc., 1991.