CLOSED-FORM AND GENERALIZED INVERSE KINEMATIC SOLUTIONS FOR ANIMATING THE HUMAN ARTICULATED STRUCTURE

Kwan W. CHIN

938354B

Project Supervisors: Dr. Brian von Konsky *and*
Andrew Marriott

PRESENTED AS PART OF THE REQUIREMENTS FOR
THE AWARD OF THE DEGREE OF
BACHELOR OF SCIENCE (COMPUTER SCIENCE) (HONOURS)
OF THE
CURTIN UNIVERSITY OF TECHNOLOGY (1996)

Contents

List of Figures

List of Tables

Abstract

This thesis presents inverse kinematics (IK) solutions specifically for the human articulated structure. The main area of research is in robotics where the algebraic and iterative methods for solving IK are investigated in detail. Extensive closed-form solutions (algebraic) for the human arm are derived which can be directly used for solving the IK of the human arm. For the iterative method, various minimization and root finding algorithms have been investigated for solving IK. Results and analysis showed that the Broyden-Fletcher-Goldfarb-Shanno (BFGS) minimization algorithm is ideal in that it has a reasonable overall rate of convergence and function evaluations. Therefore it is used in the resulting generalized IK engine which can be used to solve the IK of an articulated structure having N degrees of freedom (DOF). Further, this thesis provides implementation details (high-level interface) which demonstrates the application of the IK solutions presented for animating human motion. This is demonstrated in the *Human Arm Animator* applet which demonstrates the use of the work herein for animating the human arm. The significance of this work is that this thesis provides a generalized IK solution for articulated structure having N DOF. Thus other forms of figures (e.g. animals) can be animated successfully. Furthermore this thesis has extended the work of (Goldenberg and Lawrence, 1985; Goldenberg *et al.*, 1985; Sasaki, 1994) where other minimization algorithms (such as modified Powell's, modified Fletcher's and Brent's method) have been thoroughly investigated for solving IK.

Preface

This thesis would not have been completed without the help of these people who have played a significant role directly or indirectly. I would like to thank my supervisors (Brian and Andrew) for suggesting this project which has tested my capabilities in the academic field and which has brought out the insanity in me to work all nights and days (and giving me a few more white hairs!). Also I'd like to thank them for the continous support, suggestions and understanding of a student's life. Friends I am indebted to, Rita Chan who has given me the support and helped me work out some of the hairy maths involved in this thesis, James Mei who has helped me with the maths and has thrown meaningful ideas for this thesis, Allan Loh whom has made the nights at Curtin lively, being a great friend and for discussing the problems encountered in this thesis, Boon Yih Kuan for being a great friend and letting me use his car every night. Finally I would like to thank my best friend Yen Khee for being there and for reminding me that there are more important things in life apart from studying (for example, health). In my opinion, friends are very important and I've been lucky in my academic years in Australia to have met some great friends (Allan Loh, Yih kuan, James Mei, Kok yong and Rita Chan) whose main goal is to achieve and by knowing them, the urge to compare myself to them plays a significant role, for without them I would not have reached this far. Thanks very much guys.

Chapter 1

INTRODUCTION

Successful human animation incorporates research in modeling and animation of the human body, facial animation, hand animation and mechanical aspects of articulated body segments (Magnenat-Thalmann and Thalmann, 1988). Early research involved using a script (labanotation) which described the position and orientation of every joint. This script was fed into a program which choreographed a synthetic actor through time. Another method uses rotoscopy (image-based keyframe animation) (Magnenat-Thalmann and Thalmann, 1988). Zeltzer (1982) used high-level control where users only have to specify commands such as "walk to the door". As motions produced by humans are extremely subtle, the task of animating humans is made much more difficult. For example, when humans walk, the shoulders and hips do not move dramatically but if the movement is not there, the motion is not right (Coco, 1995).

The possibility of animating humans has been made possible by more powerful computers and the advent of better algorithms. This can be seen in the movie "Toy Story" (Robertson, 1995). Also, the simulation of humans in a dynamic environment has been realized; e.g. the simulation of humans in a car crash. Other areas include biomedical visualization (modeling of internal organs, skeleton and muscles which include their deformations), medical simulations (training medical staff in emergency situations), and virtual reality (virtual humans interacting with the simulated environment). Interested readers are referred to http://www.cis.upenn.edu/~hms for papers concerning human animation and the *Jack* system

In the field of robotics, the basis for controlling highly articulated structures (more than 6 DOF) has been heavily researched for more than two decades. Most industrial robots have a simple design so that the mathematical formulation needed to control their trajectory is simple. There are various problems associated with manipulating articulated structures but most of these problems are concerned with formulating and solving equations for controlling manipulators. Manipulators having DOF higher than seven or eight are difficult to formulate. Even if the formulation is possible, it would be susceptible to errors and inefficient such that real-time control cannot be achieved.

The two main methods which are used extensively in robotics for controlling articulated structure are kinematics and dynamics. *Kinematics* is the science of motion without regard to forces affecting it. Its only concern is the position, velocity and acceleration relative to time (Craig, 1989). Kinematics is separated into forward and inverse kinematics (IK). In forward kinematics, the angles for each joint are given and the position and orientation of the end-effector (hand) is calculated. In inverse kinematics, the position and orientation of the end-effector is given and the angle of each joint is determined. On the other hand, *dynamics* deals with the forces, moments of inertia and mass of each limb segment. It can also be separated into forward dynamics and inverse dynamics. The definition is similar to kinematics but the variables deal with forces, masses and moment of inertia for each joint.

This thesis provides the core foundation for a human animation package through inverse kinematics and to consider other minimization algorithms which have not been considered. Algebraic and iterative methods for solving IK are investigated in detail. This research work presents extensive formulation and solutions for both of these methods specifically for animating the human articulated structure.

Chapter 2

PROBLEM DESCRIPTION

The following sections formally outline the problems investigated in this research. The human arm will be used as the kinematic model for discussion throughout because it can be modelled with a minimum of six degrees of freedom (DOF) and is closely related to a generic robotic arm.

2.1 Statement of the problem

The aim of this research is to review and incorporate methods for solving inverse kinematics (IK) from robotics for human animation. The focus is two fold, investigate the closed-form solution and the generalized IK solutions. These methods for solving IK are investigated in detail. Extensive closed-form solutions are derived for the human arm and various minimization algorithms have been investigated for solving IK. The second aspect of this research is to demonstrate the application of forward and inverse kinematics for animating the human articulated structure. A detailed review of various methods for solving IK in robotics and the incorporation of these methods for animating human motion are presented.

2.2 Subproblems

1. *Closed-form Solution*

   The feasibility of a closed-form IK solution was investigated. This involved analysis of the algebraic method for deriving a closed-form solution for manipulators having six degrees of freedom (DOF) or more.

2. *Iterative Solutions*

   Generalized IK solutions which utilized iterative methods including minimization and root finding algorithms were investigated. The main focus was to implement and incorporate the most robust algorithms which are able to handle singular and non-square Jacobian. Moreover real-time solutions were investigated for iterative manipulation of articulated structures. The main area of research was on minimization algorithms where issues such as global and local convergence were considered in detail.

3. *Dealing With Multiple Solutions*

   For a particular end-effector position there are multiple configurations which can be used. The problem was to determine which solutions are feasible subject to a set of constraints. Analysis of algebraic and iterative methods for deriving different set of solutions are presented.

4. *IK Interface*

   A high-level interface was implemented to demonstrate the capabilities of the IK engine. The main objectives of the interface were interactive manipulation, ease of use and to demonstrate the implementation details for the theoretical concepts presented in this thesis.

2.3 Significance of Study

There has been significant work in robotics for solving IK which can be used in computer graphics, specifically human animation. This includes generalized (Sasaki, 1994; Manocha and Zhu, 1994; Goldenberg *et al.*, 1985) and real-time (Zomaya, 1992) solutions in robotics which are applicable to animating humans. The human joints constitute an articulated structure with redundant DOF, therefore the aim of this thesis was to investigate algebraic and iterative methods for solving IK and hence for animating the human articulated structure.

The possibility of developing a generalized IK engine was made possible by generalized IK solutions in robotics. Therefore this study looked at the feasibility of these methods and provides the basis for the implementation of a generalized IK engine which can be extended to handle other parts of the human skeletal structure (i.e legs). Further, this thesis extends the work by Goldenberg *et al.* (1985); Goldenberg and Lawrence (1985) and Sasaki (1994) where other minimization algorithms have been considered for solving IK iteratively. Thus the work described herein presents a contribution to the robotic and computer graphics fields.

Finally this study provides the foundation needed for applications which involve human animation in the School of Computing. Another example would be to incorporate the resulting work into a teaching tool for the School of Physiotherapy.

Chapter 3

PREVIOUS RESEARCH

The following sections review literature related to solving IK in robotics and the use of IK in human animation. The extent of work done in both fields provides the reader with significant background necessary to understanding this thesis. Readers who are interested in a general explanation of IK are referred to Korein and Badler (1982).

3.1 Kinematics Representation

An articulated figure is a structure that consists of a series of rigid links connected at joints (Watt and Watt, 1992). To define the representation of these joints relative to one another, a kinematic notation is needed. Such a notation was developed by Denavit and Hartenberg which specifies a coordinate frame for each joint where each coordinate frame relates joint i with joint $i+1$ (Paul *et al.*, 1981).

DH-Notation is a set of rules which specify the assignment of coordinate frames (Lee, 1983). The coordinate frame for each joint is specified using the following systematic steps:

1. Identify the joint axis. The joint axis for joint i, is the axis the joint rotates about.

2. Assign Z_i axis pointing along the i*th* joint axis.

3. Assign X_i axis perpendicular to the Z_i and Z_{i+1} axis.

4. Assign Y_i to complete the coordinate frame.

After the frames are assigned, the following four parameters can be obtained (Lee, 1983, 1982):

- θ_i: The joint angle from the x_{i-1} axis to the x_i axis about the z_{i-1} axis.
- d_i: The distance from the origin of the (i-1)th coordinate frame to the intersection of the z_{i-1} axis with the x_i axis along the z_{i-1} axis.
- a_i: The offset distance from the intersection of the z_{i-1} axis with the x_i axis (shortest distance between z_{i-1} and z_i.
- α_i: The offset angle from the z_{i-1} axis to the z_i axis about the x_i axis.

These four parameters describe the position and orientation of joint i relative to joint $i+1$. Figure 3.1 gives an illustration of these parameters for describing a two DOF kinematic chain. The corresponding values for the four parameters

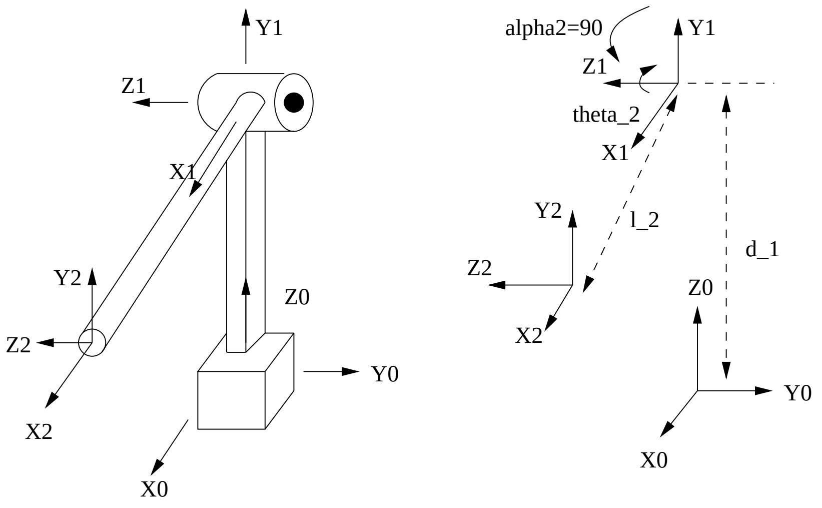


Figure 3.1: Coordinate Frames for a 2 DOF Kinematic Chain (McKerrow, 1991)

outlined above (θ_i to α_i) are shown in Table 3.1. The use of DH-Notation for the specification of the human arm (6 DOF) can be found in Appendix B.

The parameters defined above can then be used to construct a general transformation matrix which describes the position and orientation of joint i and joint $i+1$. This transformation matrix transforms joint $i+1$ to joint i with the following transformation (Lee, 1983),

$$A^i_{i-1} = Translate(Z,d) \cdot Rotate(z,\theta) \cdot Translate(X,a) \cdot Rotate(X,\alpha)$$

Joint	theta (θ_i)	alpha (α_i)	a_i	d_i
1	θ_1	90°	0	d_1
2	θ_2	0°	l_2	0

Table 3.1: DH-Notation Parameters

Thus,

$$A_{i-1}^{i} = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) & 0 & a_{i-1} \\ sin(\theta_i)cos(\alpha_{i-1}) & cos(\theta_i)cos(\alpha_{i-1}) & -sin(\alpha_{i-1}) & -sin(\alpha_{i-1}d_i \\ sin(\theta_i)sin(\alpha_{i-1}) & cos(\theta_i)sin(\alpha_{i-1}) & cos(\alpha_{i-1}) & cos(\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.1)$$

Once A_{i-1}^{i} is defined for each joint, the position and orientation of the end-effector can be found (relative to the base frame). This is defined by Craig (1989); Lee (1983) as,

$$T_n = \prod_{i=1}^{n} A_{i-1}^{i}(\theta_i) \quad (3.2)$$

where T_n is the end-effector and n is the DOF of the kinematic chain. Equation (3.2) can be used to describe the pose of the end-effector in cartesian space given the angles for each joint.

DH-Notation is a system where each joint coordinate is related to its previous, but it is not applicable to branching joints and links. Another intuitive system is called the *Axis-Position* method which stores the position of the joint, its orientation of the joint axis and pointers to the link(s) each joint is attached to (Watt and Watt, 1992).

3.2 Solving For Inverse Kinematics

Solving for IK is one of the major research areas in robotics. This is due to the need for more complex manipulators with N degrees of freedom (DOF). There are three main methods for solving IK: *algebraic, geometric* and *iterative* (Korein and Badler, 1982). Each method has its advantages and limitations. Algebraic and geometric methods provide closed-form solutions for a given kinematic chain but obtaining a solution space is difficult. Iterative methods provide generalized solutions to IK but converge to one solution space where a kinematic chain has multiple solutions (Korein and Badler, 1982). Also, algebraic and geometric methods provide real-time computation of IK and find all solutions.

3.2.1 Algebraic

Details of an algebraic solution to the PUMA 500 manipulator can be found in Snyder (1985), Paul *et al.* (1981) and Craig (1989). To solve IK algebraically, it is necessary to solve equations for $\theta_1 \ldots \theta_N$ (joint angles) for N DOF.

Given the required end-effector frame, the problem can be formulated as,

$$\prod_{i=1}^{n} A_{i-1}^{i}(\theta_i) = \begin{bmatrix} Nx & Ox & Ax & Px \\ Ny & Oy & Ay & Py \\ Nz & Oz & Az & Pz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

where the right hand side describes the required position and orientation of the end-effector. The problem comes down to solving n equations for n unknowns (Paul *et al.*, 1981).

For example, to solve for a six DOF PUMA 500 (robot arm) given its coordinate frames A_1 to A_6, equate (Paul *et al.*, 1981),

$$\prod_{i=2}^{6} A_{i-1}^{i}(\theta_i) = A_1^{-1}(\theta_1) \cdot^0 T_6$$

where ${}^0T_6$ is the right hand side of equation (3.3). Then solve for θ_i by equating the left and right hand side. If no more θ_i can be solved, A_2 is inverted and brought over to the right hand side and solved further for joint angles. This continues until all angles have been determined.

This method does not guarantee a closed-form solution for a manipulator. Thus, mechanical engineers usually design simple manipulators where closed-form solutions exist (Craig, 1989). Manocha and Canny (1994) and Manocha and Zhu (1994) proposed a generalized closed-form solution which can be derived for 6 DOF (or less) kinematic chain. Manocha and Canny (1994) outlined a method for solving IK algebraically using symbolic manipulation to derive univariate polynomial and matrix computations. This method reduces the complexity of finding the closed-form solution of a manipulator using a matrix polynomial to reduce the problem to an eigenvalue problem.

3.2.2 Geometric

As opposed to the algebraic method, a closed-form solution is derived using the geometry of the manipulator. Lee (1982) used theorems in coordinate geometry which can be found in Anon (1992) to derive the closed-form solution for a six DOF manipulator. This involves projecting link coordinate frames onto the x_{i-1} and $-y_{i-1}$ plane (Lee, 1983). This method can be applied to any manipulator with known geometry. One limitation of this method is that the solution to the first three joints must be obtained before the rest of the solutions can be found. Another limitation is that the closed-form solution only applies to a specific geometry (Craig, 1989). Algebraic and geometric methods can be used together to obtain the closed-form solution for a given manipulator. Sasaki (1995) used both methods to derived the solution for a 7 DOF manipulator.

3.2.3 Iterative

The iterative method solves IK by iteratively solving for the joint angles. Generally this method is slower and converges to only one solution. The following subsections serve to provide the background needed for solving IK iteratively. The derivation of the Jacobian is presented which is needed in some minimization algorithms, the pseudoinverse for inverting a non-square and singular Jacobian and minimization algorithms are reviewed in detail.

The Jacobian

The Jacobian is used extensively in the field of robotics to relate the differential changes of the end-effector and the target object. Apart from that, it can be used to map the end-effector rate to joint rates or vice-versa (Paul, 1981).

Watt and Watt (1992) and Snyder (1985) represented the Jacobian as:

$$\dot{X} = J(\theta)\dot{\theta} \tag{3.4}$$

where θ representes the (6×1) position vector of joint angles and $J(\theta)$ is a $(6 \times N)$ matrix whose J_{ij} elements represent the partial derivative of the ith cartesian coordinate of the end-effector with respect to the jth joint angle (Leahy Jr *et al.*, 1987). Each element of the Jacobian matrix is a non-linear function of the joint variables (Featherstone, 1983). The Jacobian's inverse is defined as (Watt and Watt, 1992; Snyder, 1985):

$$\dot{\theta} = J(\theta)^{-1}\dot{X} \tag{3.5}$$

Paul *et al.* (1981) defined the Jacobian as a $6 \times N$ matrix for N DOF. Each column of the Jacobian consists of the differential translation and rotation vector corresponding to the differential changes of each of the joint coordinates (Paul, 1981). $\dot{X}$ (6×1 vector) represents the linear velocity (*dx,dy,dz*) and rotational velocity $(\delta_x, \delta_y, \delta_z)$ of the end-effector. $\dot{\theta}$ ($N \times 1$ vector) is the time derivative of the state vector (rotational velocity for each joint)(Watt and Watt, 1992).

There are various methods for deriving the Jacobian. One method is to take the partial derivative of the closed-form equations (Orin and Schrader, 1984; Angeles, 1985). According to Leahy Jr *et al.* (1987), partial differentiating is straightfoward but it is inefficient for computer implementation.

Paul (1981) represented the Jacobian in the following form (6 DOF):

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{y} \\ \dot{\theta}_x \\ \dot{\theta}_y \\ \dot{\theta}_z \end{bmatrix} = \begin{bmatrix} d_{1_x} d_{2_x} d_{3_x} d_{4_x} d_{5_x} d_{6_x} \\ d_{1_y} d_{2_y} d_{3_y} d_{4_y} d_{5_y} d_{6_y} \\ d_{1_z} d_{2_z} d_{3_z} d_{4_z} d_{5_z} d_{6_z} \\ \delta_{1_x} \delta_{2_x} \delta_{3_x} \delta_{4_x} \delta_{5_x} \delta_{6_x} \\ \delta_{1_y} \delta_{2_y} \delta_{3_y} \delta_{4_y} \delta_{5_y} \delta_{6_y} \\ \delta_{1_z} \delta_{2_z} \delta_{3_z} \delta_{4_z} \delta_{5_z} \delta_{6_z} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \\ \dot{\theta}_5 \\ \dot{\theta}_6 \end{bmatrix} \tag{3.6}$$

which directly corresponds to the form shown in equation (3.4).

Each element of the Jacobian is then defined as (Paul, 1979, 1981),

$$\begin{aligned} {}^{T6}\mathbf{d}_{i_x} &= \mathbf{n} \bullet (\delta \times \mathbf{p}) \\ {}^{T6}\mathbf{d}_{i_y} &= \mathbf{o} \bullet (\delta \times \mathbf{p}) \\ {}^{T6}\mathbf{d}_{i_z} &= \mathbf{a} \bullet (\delta \times \mathbf{p}) \end{aligned} \tag{3.7}$$

$$\begin{aligned} {}^{T6}\delta_{i_x} &= \mathbf{n} \bullet \delta \\ {}^{T6}\delta_{i_y} &= \mathbf{o} \bullet \delta \\ {}^{T6}\delta_{i_z} &= \mathbf{a} \bullet \delta \end{aligned} \tag{3.8}$$

where δ is the differential rotation vector and $i = 1 \ldots n$ DOF. **n,o,a** and **p** are vectors which correspond to columns of coordinate frames which defined the frame's x,y,z axis and its position relative to the reference frame. (*See* equation (3.1)).

Then compute the columns of the Jacobian shown in (3.6) using (3.7) and (3.8) . To use (3.7) and (3.8), the differential coordinate transformations corresponding to $\dot{\theta}_1$ to $\dot{\theta}_6$ which are the coordinate frames A_1 to A_6 respectively as defined in Section 3.1 (Paul, 1981) are needed. This method is generalized in that it can be implemented easily and applied to various kinematic chains.

Craig (1989) derived the Jacobian in terms of the screw axis (*see* McKerrow (1991) for explaination of screw axis) variables ω (angular velocity) and V (linear velocity). Craig (1989) defined the angular velocity (ω) at frame {i+1} as,

$${}^{i+1}\omega_{i+1} = {}^{i+1}_{i} R^{i}\omega_i + \dot{\theta_{i+1}}\,{}^{i+1}\hat{Z}_{i+1} \tag{3.9}$$

where ${}^{i+1}_{i}R$ is the transform frame from i to $i+1$ and $\hat{Z}$ is the joint axis of joint $i+1$ (usually the Z-axis) and linear velocity (at frame $i+1$) is defined as:

$${}^{i+1}V_{i+1} = {}^{i+1}_{i} R \left({}^{i}V_i + {}^{i}\omega_i \times {}^{i}P_{i+1}\right) \tag{3.10}$$

where ${}^{i}P_{i+1}$ is the position vector from joint i to $i+1$.

Using equations (3.9) and (3.10) and rearranging the final equations and factoring out $\dot{\theta}_1$ from equation (3.9) we can derive the Jacobian which relates the end-effector rate to joint rates. This method is used in Maciejewski and Klein (1985) for defining figures in their animation software. According to Maciejewski and Klein (1985) this method formulates the Jacobian in a minimum number of computation steps because the majority of work has been done in generating the homogeneous transformations.

Performance measures for various methods in deriving the Jacobian can be found in Orin and Schrader (1984). Whitney (1972) derived the Jacobian using vector cross products where each element of the Jacobian is defined using V_j (linear velocity vector), ω_j (rotational velocity vector) and b_iN (a vector from joint j to N) for the jth joint in a N DOf manipulator. This method was later improved by Paul (1981).

According to Zomaya (1992), the difference between various methods for computing the Jacobian are:

- the components of the Jacobian may be defined in any coordinate frame;
- the reference point of the end-effector for which the translational velocity is computed may be chosen arbitrarily.

Inverting the Jacobian

The inverse of the Jacobian is needed to solve equation (3.5). Given the linear and angular velocity of the hand (end-effector) the joint rates can be computed. There are a number of numerical methods which can be used for calculating joint rates. For example, Gaussian Elimination can be used to invert the Jacobian if it is invertible (Snyder, 1985). Press *et al.* (1992) used **LU** decomposition to solve equation (3.5). Maciejewski (1990) outlined the following problems when mapping effector-rate to joint rates:

1. **Singularity**
   The Jacobian is singular if it is not of full rank or if its determinant is zero. Nakamura and Hanafusa (1986) solved this problem using the **singularity robust inverse** (SR-Inverse) as an alternative to the pseudoinverse in the area of singularity. Alternatively, Whitney (1972) suggested the use of redundant DOF to avoid singularities.

2. **Non-Square**
   The Jacobian is a $(6 \times N)$ matrix for N DOF. Therefore for DOF less than

or more than six, the Jacobian is non-square. A general solution would be needed to handle $N < 6$ and $N > 6$ (Fletcher, 1968). When $N = 6$, any inverse methods (e.g. Gaussian Elimination) can be used to obtain J^{-1} if and only if the Jacobian is non-singular. Otherwise, the pseudoinverse or the SR-Inverse is used.

3. **Degeneracies**
   This is due to the fact that there is an infinite number of solutions for the same end-effector configuration. When degeneracies occur, certain joint velocities exist but do not contribute to the overall end-effector velocity (Zomaya, 1992).

The Pseudoinverse

As mentioned, the pseudoinverse is used to invert the Jacobian (**J**) when it is singular or non-square. Thus, it provides a best approximation which gives a least squares solution of minimum norm. Golub and Kahan (1965) defined the pseudoinverse using Singular Value Decompostion (SVD). Other methods of defining the pseudoinverse can be found in the comprehensive book by Rao and Mitra (1971).

The pseudoinverse of a rectangular matrix (J^+) satisfies,

$$\begin{aligned} JJ^+J &= J \\ J^+JJ^+ &= J^+ \\ (J^+J)^T &= J^+J \\ (JJ^+)^T &= JJ^+ \end{aligned}$$

where T denotes the complex conjugate transpose (Klein and Huang, 1983; Golub and Kahan, 1965). Proof of the above properties and that the pseudoinverse satisfies least squares can be found in Noble and Daniel (1989). Comprehensive examples can found in Noble and Daniel (1989). Golub and Kahan (1965) presented proof and basis for deriving SVD.

Noble and Daniel (1989) defined the pseudoinverse using SVD as,

$$J^+ = V\Sigma^+U^H \tag{3.11}$$

where V and U are unitary matrices of dimension $p \times p$ and $q \times q$ respectively. Σ^+ contain the reciprocals of the eigenvalues of J. Albert (1972) computed the pseudoinverse as:

$$A^+ = \lim_{x \to \infty}((A^T A + x^2 I)^{-1} A^T) \tag{3.12}$$

where T denotes the transpose and I is the identity matrix.

Klein and Huang (1983) reviewed the pseudoinverse and outlined problems in solving IK when the end-effector follows a rectilinear space. Klein and Kee (1989) provided proof and experimental results to support Klein and Kee (1989). They identified various configurations of a simple manipulator where this situation arises. Maciejewski and Klein (1985) presented various methods for defining the pseudoinverse used in their animation system. This included determining the pseudoinverse if the rank of J is unknown. Furthermore, Maciejewski and Klein (1985) pointed out that the pseudoinverse is ideal when redundant DOF are present.

Maciejewski (1990) used SVD to identify ill-conditioning. This is done using SVD to determine the rank of the Jacobian. If the rank is zero, the Jacobian is singular, thus the pseudoinverse is needed to provide a least-square and minimum norm solution. Maciejewski (1990) suggested that the use of the pseudoinverse is spurious at the transition from an approximate solution to an actual solution. He overcame this limitation by using damped least squares or regularization.

Numerical Methods

This section reviews papers on numerical analysis which are used for solving IK iteratively. The minimization methods provided here give a brief introduction to existing methods. Comprehensive details are covered in Ortega and Rheinboldt (1970), Massara (1991), Rabinowitz (1970), Press *et al.* (1992), Adby and Dempster (1974) and CSEP (1995). This treatment is significant because it provides a mathematical foundation for understanding methods for solving IK iteratively. The basic structure of iterative methods can be summarized as (CSEP, 1995),

- Supply an initial guess X_o
- For $k = 0, \ldots$ until convergence

1. Test X_k for convergence
2. Calculate a search direction P_k
3. Determine an appropriate step length λ_k (or modified step S_k)
4. Set $X_{k+1} = X_k + \lambda_k P_k$ (or $X_k + S_k$)

In minimization, it is required to find the local minimum of f (Brent, 1973) of the function $f : R^n \to R$. In relation to IK, we are solving for the joint angles (n dimension) which minimizes the error between the end-effector and the required (destination) coordinate frame. A detailed explanation can be found in Chapter 5.

Powell's Method

Powell (1964) outlined a method for the minimization of a function with several variables which did not require the derivative of the given function. One limitation of Powell's method is that the function variables cannot have constraints (unconstrained optimization). Interested readers are directed to Powell (1964) for proof that Powell's method is quadratically convergent.

The main idea of Powell's method for achieving convergence is by searching down a given direction vector $U = [u_1 \dots u_n]$ with each element in the direction vector updated iteratively. The direction vector produces N mutually conjugate directions which means that once all elements of the direction vector are linearly independent (mutually conjugate directions), the next N line minimizations (one dimensional minimization method) will put the function at the minimum (Press *et al.*, 1992).

Massara (1991) pointed out that Powell's method need $O(n^2)$ line searches and fails to converge when the set of search directions become linearly dependent.

PRAXIS method

PRAXIS is another method which also does not require derivatives of the objective function. This method was originally presented in Brent (1973). According to Brent (1973), this method performs better than Powell's method presented above. The Praxis method is a modification of Powell's algorithm such that it ensures quadratic convergence, and avoids difficulties associated with accepting new search directions which are inherent in Powell's method. This is achieved by reseting the search direction $U = [u_1 \ldots u_n]$ after every n or $n + 1$ iterations. Brent (1973) reset $U = [u_1 \ldots u_n]$ to an orthogonal matrix and ensured that the new directions are conjugate. This effectively removes the linear dependence of the search directions with the Powell's method.

Fletcher-Reeves-Polak-Ribiere (FRPR)

Fletcher-Reeves-Polak-Ribiere (FRPR) is a method which has been modified by Polar-Ribiere (Press *et al.*, 1992). This method makes use of the conjugate gradient vectors (generated from gradient evaluations) which replaces the Hessian matrix. In this method, a direction vector is calculated such that it produces a basis set of mutually conjugate directions such that each successive step continually refines the direction toward the minimum. In this method the new direction vector (h_{i+1}) leading from point $i + 1$ is computed by the gradient at point $i + 1$ denoted by g_{i+1} to the previous h_i scaled by a constant γ_i. Mathematicaly this is defined as:

$$h_{i+1} = g_{i+1} + \gamma_i h_i \tag{3.13}$$

In the FRPR method, γ_i is defined as:

$$\gamma_i = \frac{g_{i+1} \cdot g_{i+1}}{g_i \cdot g_i} \tag{3.14}$$

Another variation called Fletcher-Reeves defined γ_i as:

$$\gamma_i = \frac{(g_{i+1} - g_i) g_{i+1}}{g_i \cdot g_i} \tag{3.15}$$

Newton's Method for Non-Linear System

Generally, in non-linear systems, the aim is to minimize the set of equations (functions with N variables):

$$F_i(x_1, x_2, \ldots, x_N) = 0 \qquad \text{i=1} \ldots \text{N} \tag{3.16}$$

According to Press *et al.* (1992), equation (3.16) can be expanded in Taylor Series and by neglecting second order terms or higher, obtain the following equation:

$$J(x_\circ) \cdot \delta x = -F(x_\circ) \tag{3.17}$$

where J is the Jacobian matrix, $(x_\circ)$ is the initial guess and F denotes the entire vector of functions F_i defined in equation (3.16). Solving for δx using LU decomposition we obtain the correction vector for the next iteration:

$$x_{new} = x_\circ + \delta x \tag{3.18}$$

Equation (3.17) can be solved as,

$$\delta x = J(x_\circ)^{-1} \cdot -F(x_\circ) \tag{3.19}$$

This requires the Jacobian to be inverted. Section 3.2.3 gives a detailed discussion on inverting the Jacobian for solving equation (3.19).

Iteration stops when the sum of the magnitudes of the functions F_i is less than ψ (tolerance value) or the sum of $|\delta_i x| < \varepsilon$, where ε is the given accuracy is true. The mathematical basis can be found in Ortega and Rheinboldt (1970) and implementation details and source code can be found in Mathews (1992).

Broyden-Fletcher-Goldfarb-Shanno (BFGS)

The BFGS method belongs to the class of methods called Quasi-Newton or variable metrics. Quasi-Newton is a term used to define a class of methods which replaces the Jacobian in the ordinary Newton's method with an approximation matrix that is updated at each iteration. The advantage of this method is that the Jacobian does not need to be recalculated and inverted at each iteration. On the other hand, the Quasi-Newton method in general does not have quadratic convergence and is not self correcting (results does not correct for round-off error) (Burden and Faires, 1985).

In Quasi-Newton methods the inverse of the Hessian (H) matrix is approximated iteratively (Press *et al.*, 1992). The Hessian (H) matrix for a given objective function $F(x)$ (n variables) is defined as $G_{i,j} = \delta^2 F(x)/\delta x_i \delta x_j (i, j = 1, \ldots, n)$; the second partial derivatives of the objective function. However in Quasi-Newton methods, H^{-1} is approximated by H^r. H^r is positive definite and the sequence of H^r for $r = 0 \ldots \infty$ tends to the value of H^{-1} (Massara, 1991).

The iterative equation is defined as (Massara, 1991),

$$x^{i+1} = x^i + \alpha^i d^i \tag{3.20}$$

where the direction vector d^i is defined as,

$$d^i = -H^i g^i \tag{3.21}$$

Equation (3.20) is guaranteed to converge if H^r is positive definite and the scalar α is chosen such that it reduces the objective function (Massara, 1991).

The difference between the variations of the Quasi-Newton method is the computation of the Hessian matrix (Press *et al.*, 1992). In BFGS, the Hessian matrix is approximated iteratively using the following formula,

$$H_{i+1} = H_i + Term1 - Term2 + Term3 \tag{3.22}$$

where

$$Term1 = \frac{(x_{i+1} - x_i) \otimes (x_{i+1} - x_i)}{(x_{i+1} - x_i) \cdot (\bigtriangledown f_{i+1} - \bigtriangledown f_i)}$$

$$Term2 = \frac{[H_i \cdot (\bigtriangledown f_{i+1} - \bigtriangledown f_i)] \otimes [H_i \cdot (\bigtriangledown f_{i+1} - \bigtriangledown f_i)]}{(\bigtriangledown f_{i+1} - \bigtriangledown f_i) \cdot H_i \cdot (\bigtriangledown f_{i+1} - \bigtriangledown f_i)}$$

$$Term3 = [(\bigtriangledown f_{i+1} - \bigtriangledown f_i) \cdot H_i \cdot (\bigtriangledown f_{i+1} - \bigtriangledown f_i)] u \otimes u$$

$$u \equiv \frac{(x_{i+1} - x_i)}{(x_{i+1} - x_i) \cdot (\bigtriangledown f_{i+1} - \bigtriangledown f_i)} - \frac{H_i \cdot (\bigtriangledown f_{i+1} - \bigtriangledown f_i)}{(\bigtriangledown f_{i+1} - \bigtriangledown f_i) \cdot H_i \cdot (\bigtriangledown f_{i+1} - \bigtriangledown f_i)}$$

$\bigtriangledown f_i$ is the gradient of the objective function at i^{th} iteration. In the *Davidon-Fletcher-Powell* 's variation, the expression **Term3** is omitted. Sasaki (1994) provided a flow diagram and explanation to the Davidon-Fletcher-Powell method which was used to solve IK.

Iterative Solutions for IK

Goldenberg and Lawrence (1985) gave an overview of a generalized method for solving IK using an iterative method. The method formulates the problem by obtaining the residual vector which is defined as the difference between the hand frame and the target frame. The joint constraints are not considered in their solution. Goldenberg *et al.* (1985) extended the generalized method presented in Goldenberg and Lawrence (1985). Steps and equations to their algorithm are outlined and joint constraints are considered. This is done by reducing the step size in their computation.

Wang and Chen (1991) used two minimization algorithms, namely *Cyclic Coordinate Descent* (CCD) and *Broyden-Fletcher-Shanno* (BFS). The CCD algorithm is used to find the starting value for BFS which ensure that for any given initial value, BFS will sucessfully converge. Furthermore, the joint limits are considered.

Tsai and Orin (1987) developed a generalized method for solving IK. Their method solves *N* DOF based on integration of joint rates and includes a position feedback term for convergent checking. Moreover, the developed method converged near singular points. The developed method does not solve IK iteratively but uses a feedback mechanism which enables the determination of deviation of joint rates from actual rate. Sasaki (1994) provided an overview of numerical methods used to solve IK and also provided test results and evaluation for each numerical method used. Refer to Press *et al.* (1992) for general overview and source code for the *Broyden-Fletcher -Goldfarb-Shanno (BFGS)* minimization algorithm which is a variation of the BFS method. Source code for other minimization algorithms such as steepest descent and conjugate gradient can also be found in Press *et al.* (1992). The mathematical background on Quasi-Newton can be found in Mathews (1992); Dennis and et al (1974). Other methods for solving non-linear systems can be found in Byrne and Hall (1973).

3.3 Kinematics in Computer Graphics

IK is used to animate all aspects of human motion. This includes locomotion and manipulating the skeletal system which includes feet, pelvis, spine and torso and center of mass (Cary B and Badler, 1991). IK or goal directed motion used in human animation seeks to aid in positioning and orienting a body model which consists of joints and segments (Badler, 1987). The PODA system (Girard, 1987) linearizes kinematics at a point and solves for the joint angles iteratively. This is done by computing the Jacobian and iteratively solving for IK. The pseudoinverse is used when the Jacobian is not square and singular. Otherwise Gaussian elimination with pivoting is used to remove the complexity of computation (Girard and Maciejewski, 1985). One downside of using the pseudoinverse is its expensive computational cost (Sasaki, 1994). In order to provide ease of manipulation, the PODA system incorporates both inverse and forward kinematics. Figures in the

PODA system are animated using a parameterized model where a set of end-effector positions are specified and IK is invoked to move the figure through the predetermined path. The limitation in the PODA system is that joint limitations are not considered. The *Jack* system employs an iterative method to solve IK as well. The *Jack* system uses a combination of the BFGS method and Rosen (1960)'s method to solve IK. This method is used to solve multiple constraints which are specified by the user. Moreover it supports joint constraints.

Badler *et al.* (1987) outlined an innovative application of IK by having multiple constraints on a human model. IK is then used to solve for each constraint (through minimization). This allows the desired end-effector configuration to be reached in fewer iterations. For example, to position a figure for sitting, the destination position and orientation of the shoulders, hip, arms and knees are specified. Then IK is used to solve for each constraint iteratively. Also, the centre of mass of the figure positioned at the hip is taken into account. The downside of this is that the figure is seen hanging about its centre of mass. Also, each constraint can be solved in parallel. Phillips *et al.* (1990) provided the mathematical formulation of the method presented in Badler *et al.* (1987). With multiple constraints, Phillips *et al.* (1990) and Zhao and Badler (1994) determined the computation is of order $O(n) \times O(n)$ where n is the number of degrees of freedom (DOF).

Rijpkema and Girard (1991) used IK to choreograph the fingers in their knowledge-based approach to human grasping. The difficulty in this problem is that the movement of each finger may influence one another. Also, the thumb is very dextrous. To represent the fingers, DH-Notation is used to describe the kinematics of each finger. IK is then invoked to position and orient each finger when grasping is done. The thumb is solved using an iterative method since a closed-form solution cannot be found.

The animation of human locomotion relies on IK to position and orient joints. A path describing the end-effector's trajectory is specified and IK is used to solve for the joint angles along this path. In human locomotion, a parameterized model is used to simulate the limb motion through time. The *Jack* system is able to specify motion using scripts which consists of constraints, velocity, and start and ending time (Badler *et al.*, 1993). Keyframing techniques for articulated structures are presented in Korein and Badler (1982) where a kinematic chain of N DOF is parameterized as the vector $q = (q_1, q_2, \ldots, q_n)$. Then keyframing can be used to animate each element to the desired goal. A parameterized model for gaits is also presented in van de Panne (1996), along with various parameterized models for animating creatures.

Apart from using IK to interactively manipulate human figures and controlling joints for a specified path as in Girard and Maciejewski (1985) and Girard (1987), there are other applications of IK such as *Reachable Workspace* and *Collision Avoidance* (Zhao and Badler, 1994). The use of IK in systems incorporating dynamics provides an intuitive mechanism in human animation. Ko and Badler (1996) combines both kinematics and dynamics for better control in the *speedy* system. They used IK as a locomotion generator based on collected biomechanical data of human locomotion. At each frame, dynamics is used to give a realistic characteristic of human locomotion. This includes balance control and force control.

3.4 Summary

As can be seen various methods are available for solving IK and little work has been presented which incorporates the advancement in solving the IK of manipulators in robotic for animating the human figure. Example of this is the algebraic and geometric methods which offer fast and efficient computation cost. Futhermore generalized solutions are available which can be used to animate articulated structure having N DOF. This thesis contributes by incorporating various methods for solving IK in robotics for animating the human figure. Furthermore as can be seen in the robotics field, minimization algorithms such as modified POWELL (Press *et al.*, 1992), PRAXIS (Brent, 1973)) and Modified Fletcher (Lau, 1995) methods have not been investigated for solving IK. Thus this thesis contributes to research in animating human motion and generalized IK solutions in robotics.

Chapter 4

RESEARCH METHODOLOGY

This chapter outlines the hypotheses, deliminitions and the research methodology undertaken in this work used to prove or disprove the hypotheses outlined.

4.1 Hypotheses

The following outlines hypotheses which were of concerned to this study. These hypotheses are evaluated in Chapter 6.

1. Iterative methods are better than the closed-form method presented at handling articulated structures having N (e.g $N < 6$ or $N > 6$) DOF. This is because the formulation of algebraic equations is time consuming and error prone. Furthermore, articulated structures having more than six DOF might not have a closed-form solution.

2. Solutions from iterative methods do not necessarily give the required joint configurations. Since iterative methods only converge to one solution, it is time consuming to obtain other solutions.

3. Generalized solutions presented in robotics can be incorporated to manipulate different joints in humans. This is done by applying IK to a segment of joints.

4. In iterative methods, if the required target position and orientation is near the current configuration of the articulated structure, there will not be

sudden changes to each DOF.

4.2 Types of Investigation

The following outlined methodologies were used to test each hypothesis and solve the stated problems in Section 2.

- **Comparative**
  The performance and feasibility of closed-form and iterative solutions in providing real-time manipulation of the human arm was compared and contrasted. This is needed to compare the limitations and advantages of each method. This is vital because the objective was to provide a real-time engine and the problems associated with each method needed to be thoroughly investigated.

- **Evaluation**
  The effectiveness of closed-form solutions and iterative methods in deriving specified end-effector configuration was investigated. This methodology was incorporated to determine the effectiveness of the closed-form and iterative solutions in providing the required results. The algebraic solutions yield multiple solutions and iterative solutions converge to one solution.

- **Design and Demonstration**
  A simple graphical interface was developed to demonstrate the use of the concepts and solutions presented in this thesis. This shows the application of inverse kinematics (IK) solution in animating the human articulated structure. The human arm is used to demonstrate this aspect of the thesis.

4.3 Deliminations and Assumptions

The main aims of this thesis were to provide ways of manipulating articulated structures through IK and to show the use of IK for human animation. Therefore the following do not apply:

1. *Dynamics*

   The dynamics of an articulated structure is not considered. The forces and mass acting on the articulated structure are omitted.

2. *Skin Deformation*

   Skin deformation which occurs when the arm is manipulated is not considered. The model used is a skeletal structure, therefore the joint constraints are of importance. The arm will be modelled using the joint angles constraints as measured by Norkin and White (1988)

3. *Collision Detection*

   Collision detection is not considered. Thus, interaction with the environment is omitted.

4. *Target Tracking*

   The target position and orientation which the IK solution will solve is assumed to be stationary.

5. *Human Motion Package*

   A full human animation package is not implemented. This thesis mainly covers methods for solving IK. The simple graphics system developed serves to demonstrate the application of this thesis for animating human motion. Therefore this thesis provide the foundation for a full package where developer can concentrate fully on the high level interface.

6. *Constrained Optimization*

   Constraint minimization is not within the scope of this thesis. This thesis is concerned mainly with investigating un-constrained minimization and root finding algorithms for solving IK.

Chapter 5

IMPLEMENTATION

This chapter outlines algebraic and iterative methods for solving inverse kinematics (IK). Further, the application of kinematics for human animation is outlined. The first section deals with the derivation of closed-form solution for the human arm modelled with 6 degrees of freedom (DOF). The closed-form solution derived can be used directly for solving IK of a human arm. The second section presents iterative methods for solving IK. Various minimization and root finding algorithms have been investigated and are outlined herein. Finally the implementation details of a simple human arm animator is outlined which incorporates forward and inverse kinematics.

5.1 Solving IK Algebraically

5.1.1 Problem Formulation

In the algebraic method, finding a closed-form solution for each angle of the articulated structure is of interest. Solving IK algebraically starts with the following problem formulation,

$$T_6 = A_1(\theta_1) * A_2(\theta_2) * A_3(\theta_3) * A_4(\theta_4) * A_5(\theta_5) * A_6(\theta_6) \tag{5.1}$$

where A_1 to A_6 are transformation matrices. In equation (5.1), we have six unknowns (θ_1 to θ_6). The problem is formulated as:

Given the required end-effector position and orientation, find θ_1 to θ_6

(T_6 is a 4x4 matrix which describes the orientation and position of the required end-effector).

To solve for the angles, the following steps are taken; the following matrix multiplication are derived with the help of the symbolic mathematical package MAPLE.

1. Invert $A_1(\theta_1)$ and bring it over to the left hand side.
2. Equate both sides and find the equation where there is only one unknown. Sometimes, this involves squaring two equations together, adding (or subtracting) them together and simplifying.
3. Once such an equation is found, algebraically solve for the unknown.
4. Find equations which can be solved by substituting the angle calculated in step 3.
5. If no more equations can be found, invert another matrix (e.g. $A_2(\theta_2)$) and solve further. This continues until all angles are solved.

In summary, this method systematically solves for the joint angles by inverting the **A** matrix and equating row and column from the right and left hand side. A more comprehensive example can be found in Snyder (1985).

According to Paul *et al.* (1981), the algebraic method is difficult because both sine and cosine are needed to determine a given joint angle correctly. Moreover, the manipulator has more than one solution and there are twelve equations in six unknowns.

5.1.2 Solving IK for the Human Arm

The first step in solving IK, is to assign coordinate frames to the human arm. This is needed to relate joint i with joint $i-1$ which we can then determine the end-effector's position and orientation relative to the base frame. A variation of the DH-Notation was used to describe a human arm modeled as 6 DOF. The coordinate frames and the **A** matrices derived for an arm of 6 DOF is shown below: (C_i and S_i denotes the *Cosine* and *Sine* respectively)

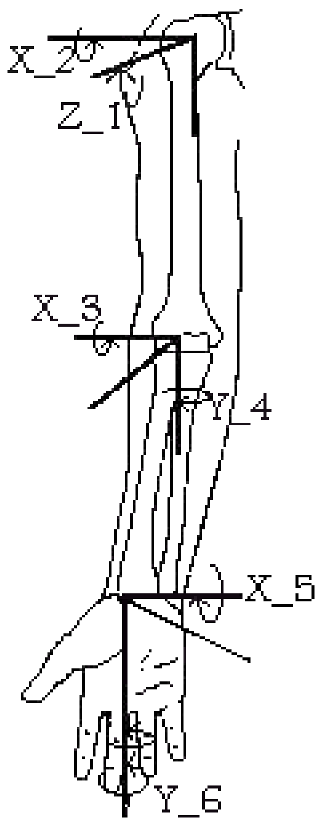


Figure 5.1: Axis of Rotation for Human Arm

$$A_1 = \begin{bmatrix} C_1 & -S_1 & 0 & 0 \\ S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_2 & -S_2 & 0 \\ 0 & S_2 & C_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_3 & -S_3 & -d3 \\ 0 & S_3 & C_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_4 = \begin{bmatrix} C_4 & 0 & S_4 & 0 \\ 0 & 1 & 0 & 0 \\ -S_4 & 0 & C_4 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_5 & -S_5 & -d4 \\ 0 & S_5 & C_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_6 = \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To solve for θ_1 formulate the following,

$$A_1^{-1} * T_6 = A_2 * A_3 * A_4 * A_5 * A_6 \tag{5.2}$$

where A_i describes the position and orientation of joint i to $i-1$.

By multiplying and simplifying both sides (see Appendix A.1 for calculated expressions), equate the equations in R_Left[1,4] and R_Right[1,4] (**See** Appendix A.1 for R_Left[] and R_Right[]) to get,
(Note. C_i,S_i and S_{ij} denotes $cos(\theta_i)$, $sin(\theta_i)$ and $sin(\theta_i + \theta_j)$. P_x to P_z, O_x to O_z, A_x to A_z and N_x to N_z are columns of the target frame).

$$C_1P_x + S_1P_z = 0 \tag{5.3}$$

Equation (5.3) has the following form Craig (1989):

$$a\ cos(\theta) + b\ sin(\theta) = 0 \tag{5.4}$$

where θ is solved as $\theta = ATAN2(a, -b)$ or $\theta = ATAN2(-a, b)$. Therefore θ_1 is solved as:

$$\begin{aligned} \theta_1 &= ATAN2(Px, -Py) \\ \theta_1 &= ATAN2(-Px, Py) \end{aligned} \tag{5.5}$$

Once having solved for θ_1, other equations which can be solved by substituting θ_1 or equations containing only one unknown are solved for. By inspection, θ_2 can be solved by equating equations in R_Left[2,4] with R_Right[2,4] and equate R_Left[3,4] with R_Right[3,4]; see Appendix A.1 for R_Left[] and R_Right[]. The following sets of equation are derived:

$$-d_4C_{23} - C_2d_3 = -S_1P_x + C_1P_y \tag{5.6}$$

$$-d_4S_{23} - S_2d_3 = P_z \tag{5.7}$$

Rewrite equations (5.6) and (5.7) so that C_{23} and S_{23} can be eliminated,

$$-d_4C_{23} = -S_1P_x + C_1P_y + C_2d_3 \tag{5.8}$$

$$-d_4S_{23} = P_z + S_2d_3 \tag{5.9}$$

Then square equation (5.8) and (5.9), add them together, simplify and factor out C_2 and S_2 to obtain the form shown in equation (5.11),

$$d_4^2 - P_x^2 + P_x^2C_1^2 + 2S_1P_xC_1P_y - C_1^2P_y^2 - P_z^2 - d_3^2 = C_2[2d_3(-S_1P_x + C_1P_y)] + S_2(2P_zd_3) \tag{5.10}$$

$$a * cos(\theta) + b * sin(\theta) = d \tag{5.11}$$

As shown in (Craig, 1989), θ in Equation (5.11) can be solved as,

$$\theta = ATAN2(a, b) \pm ATAN2(\sqrt{a^2 + b^2 - c^2}, c)$$

Then solve for θ_2 using equation (5.11):

$a = 2d_3(-S_1P_x + C_1P_y)$

$b = S_2(2P_zd_3)$

$c = d_4^2 - P_x^2 + P_x^2C_1^2 + 2S_1P_xC_1P_y - C_1^2P_y^2 - P_z^2 - d_3^2$

$$\theta_2 = ATAN2(b, a) \pm ATAN2(\sqrt{a^2 + b^2 - c^2}, c) \tag{5.12}$$

There are two solutions for θ_2. Each of these solutions can be used in the following expression to derive different configurations of the arm for a required end-effector's position and orientation.

By inspection, there are no other equations which can be used to solve for other angles. Therefore bring over A_2 to the left hand side, and formulate the following,

$$A_2^{-1} * A_1^{-1} * T_6 = A_3 * A_4 * A_5 * A_6 \tag{5.13}$$

From the above results, by equating S_Lft[2,4] with S_Rgt[2,4] and S_Lft[3,4] with S_Rgt[3,4] θ_3 can be solved. See Appendix A.2 for S_Lft[] and S_Rgt[]. Equating both of these equations and factoring out C_3 and S_3,

$$C_3 = \frac{C_2S_1P_x - C_2C_1P_y - S_2P_z - d_3}{d_4}$$

$$S_3 = \frac{-S_2S_1P_x + S_2C_1P_y - C_2P_z}{d_4}$$

Therefore θ_3 can be solved as,

$$\theta_3 = ATAN2(S_3, C_3) \tag{5.14}$$

To solve for θ_4, bring over A_3 and A_4 to the right hand side to formulate the following,

$$A_4^{-1}A_3^{-1} * A_2^{-1} * A_1^{-1} * T_6 = A_5 * A_6 \tag{5.15}$$

The full expression for the above is shown in Appendix A.3.

θ_4 can be solved by equating Q_Lft[1,4] and Q_Rgt[1,4]. Refer to Appendix A.3 for full expression of Q_Lft[1,4].

$$C_4(A_xC_1 + A_yS_1) + S_4(A_yC_1S_{23} - A_xS_1S_{23} - A_zC_{23}) = 0 \tag{5.16}$$

Equation (5.16) has the form shown in Equation (5.4).

Therefore equate,
$a = A_xC_1 + A_yS_1$
$b = A_yC_1S_{23} - A_xS_1S_{23} - A_zC_{23}$
Then the closed-form equation for θ_4 is:

$$\begin{aligned} \theta_4 &= ATAN2(a, -b) \quad or \\ \theta_4 &= ATAN2(-a, b) \end{aligned} \tag{5.17}$$

θ_6 can be solved by equating Q_Lft[1,1] with Q_Rgt[1,1] and Q_Lft[1,2] with Q_Rgt[1,2]:

$$C_6 = C_4(N_yS_1 - N_xC_1) + S_4(N_yC_1S_{23} - N_xS_1S_{23} - N_zC_{23}) \tag{5.18}$$

$$S_6 = C_4(-O_xC_1 - O_yS_1) + S_4(O - zC_{23} + O_yC_1S_{23} + O_xS_1S_{23}) \tag{5.19}$$

Therefore θ_6 is,

$$\theta_6 = ATAN2(S_6, C_6) \tag{5.20}$$

Finally, once θ_6 is solved, θ_5 is solved using equating in R_Lft[3,1] and R_Rgt[3,1]. After factoring out C_5 and S_5,

$$C_5(S_65S_{23}) + S_5(S_6C_4C_{23}) = N_z - S_4C_6C_{23}$$

The closed-form solution for θ_5 has the same form as Equation (5.11):
Therefore
$a = S_6S_{23}$
$b = S_6C_4C_{23}$
$c = N_z - S_4C_6C_{23}$

$$\theta_5 = atan2(a, b) \pm atan2(\sqrt{a^2 + b^2 - c^2}, c) \tag{5.21}$$

Once each closed-form equation for θ_1 to θ_6 is derived, these equations can then be implemented (hard-coded) directly. The required end-effector's position and orientation is described by a 4x4 homogenous coordinates. The use of the above closed-form solutions in the *Human Arm Animator Applet* will be explained in Section 5.3.2.

The steps described above can be applied to any revolute articulated structure of 6 DOF. Articulated structures having more than 6 DOF (redundant) may be solved using both algebraic and geometric method as in Sasaki (1995).

5.2 Solving IK Iteratively

5.2.1 Problem Formulation

The second method of solving IK uses iterative methods such as minimization, root finding and least-square algorithms. There has been little reported work done which uses iterative methods because iterative methods are generally slow. On the other hand, iterative methods provide a generalised solution for solving IK. Compared to the algebraic method where fixed solutions for each joint angle are analytically calculated, the iterative method calculates the joint angles dynamically based on the transformation matrices. Goldenberg and Lawrence (1985) use a modified Newton's method to solve for IK. Moreover, constraints on each joint are taken into consideration. Sasaki (1994) provided comparisons of various gradient minimization methods where derivative of the objective function is required.

Iterative methods need an objective function to minimize. One method is to represent this function as a vector of 6 elements $R = [\delta_x \; \delta_y \; \delta_y \; \omega_x \; \omega_y \; \omega_z]$ which represents the residual position and orientaion between the end-effector (EE) and the target frame. Figure 5.2 gives a graphical view of this process.

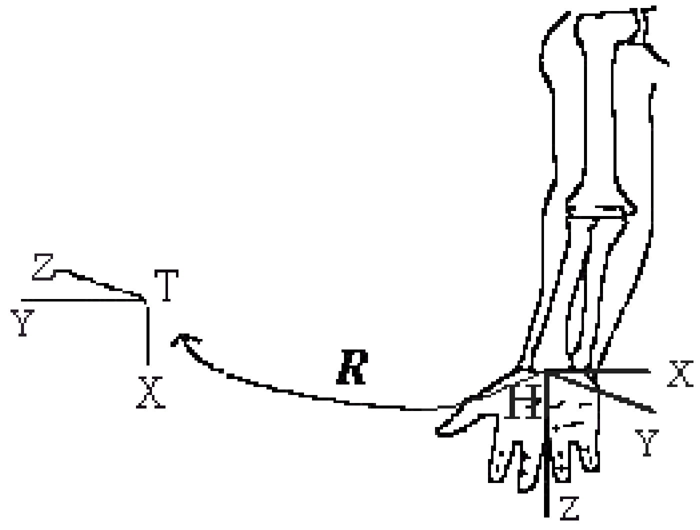


Figure 5.2: Mapping hand frame to target frame

Wu and Paul (1982) and Goldenberg *et al.* (1985) formulated R as follows:

$$\begin{aligned}
r_x &= n^H \cdot (p^T - p^H) \\
r_y &= o^H \cdot (p^T - p^H) \\
r_z &= a^H \cdot (p^T - p^H) \\
r_\Phi &= \tfrac{1}{2}(a_H \cdot o_T - a_T \cdot o_H) \\
r_\theta &= \tfrac{1}{2}(n_H \cdot a_T - n_T \cdot a_H) \\
r_\psi &= \tfrac{1}{2}(o_H \cdot n_T - o_T \cdot n_H)
\end{aligned} \tag{5.22}$$

where n^H, o^H and a^H are columns of the hand frame and n^T, o^T and a^T are columns of the Target frame. Collectively, equation (5.22) transform the hand frame to the target frame. Derivation of (5.22) can be found in Paul (1981).

When both frames coincide, the joint angles $q^*_{1\ldots6}$ satisfy (T^B_T is a constant),

$$R(q^*) = 0 \tag{5.23}$$

So the problem becomes finding q^* such that (5.23) is true.

Sasaki (1994) provides a simple formulation for solving IK. Sasaki (1994) formulated the objective function as follows:

1. Compute the current state of the end-effector using forward kinematics.

The current end-effector frame is described as:

$$\begin{bmatrix} NX_c & OX_c & AX_c & PX_c \\ NY_c & OY_c & AY_c & PY_c \\ NZ_c & OZ_c & AZ_c & PZ_c \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.24}$$

which is equivalent to T_6 in equation (5.1). Recalling that each of (5.24) depends on $\theta_{1 \ldots n}$ for n DOF.

2. Given the required position and orientation of the end-effector which is given as,

$$\begin{bmatrix} NX_d & OX_d & AX_d & PX_d \\ NY_d & OY_d & AY_d & PY_d \\ NZ_d & OZ_d & AZ_d & PZ_d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.25}$$

where each element of the matrix is constant. To formulate the problem, equate matrix (5.24) and (5.25) and obtain the following 12 sets of non-linear equations,

$$\begin{aligned} f_1(\theta) &= NX_c(\theta) - NX_d \\ f_2(\theta) &= NY_c(\theta) - NY_d \\ f_3(\theta) &= NZ_c(\theta) - NZ_d \\ f_4(\theta) &= OX_c(\theta) - OX_d \\ f_5(\theta) &= OY_c(\theta) - OY_d \\ f_6(\theta) &= OZ_c(\theta) - OZ_d \\ f_7(\theta) &= AX_c(\theta) - AX_d \\ f_8(\theta) &= AY_c(\theta) - AY_d \\ f_9(\theta) &= AZ_c(\theta) - AZ_d \\ f_{10}(\theta) &= PX_c(\theta) - PX_d \\ f_{11}(\theta) &= PY_c(\theta) - PY_d \\ f_{12}(\theta) &= PZ_c(\theta) - PZ_d \end{aligned} \tag{5.26}$$

The IK problem than becomes finding $\theta_{1 \ldots n}$ which minimizes the set of non-linear equations ($IK(\theta) = f_i(\theta) = 0$). In the least-square form, we are solving for the least value of

$$IK(\theta) = \sum_{i=1}^{12} [f_i(\theta)]^2 \tag{5.27}$$

As can be seen in Equation 5.27, only 12 non-linear equations are derived. If the number of DOF is more than 12, then it is necessary to either add constraint conditions or to optimize by the number of unspecified dimensions (Sasaki, 1994).

Two of these problem formulations define the objective function required in the minimization algorithms. A simpler method for calculating the residual vector (R) can be found in section 5.2.4.

5.2.2 Computing the Jacobian

In order to present iterative solutions, the Jacobian needs to be calculated. The Jacobian plays an important role in iterative methods where it is used to calculate the joints direction $(\dot{\theta}_1 \ldots \dot{\theta}_n)$. The Jacobian is calculated at the frame where motion is applied. In this work, the Jacobian is calculated for the End-Effector's frame since manipulation is done at the end-effector's frame.

As outlined in Chapter 2, the Jacobian (J) relates the differential changes in joint space to cartesian space:

$$\dot{X} = J(\theta)\dot{\theta} \tag{5.28}$$

where $\dot{X}$ is a 6 element vector $[d_x, d_y, d_z, \delta_x, \delta_y, \delta_z]^T$, and $\dot{\theta}$ is the differential joint vector $\dot{\theta}_1 \ldots \dot{\theta}_N$ $(N = DOF)$. The application of the Jacobian can be found in section 5.2.4 where Newton's method for a multi-dimension function is used to solve IK.

Although there are numerous methods for computing the Jacobian as presented in Chapter 2 the method used in this thesis is based on the formulation presented in (Zomaya, 1992). This method was chosen because of its simple and algorithmic nature.

The method presented by (Zomaya, 1992) defines the linear and rotational velocity $([dx_i, dy_i, dz_i, \psi_{x_i}, \psi_{y_i}, \psi_{z_i}]^T)$ components as follows:

- *Linear Velocity* $[dx_i, dy_i, dz_i]^T$

$$[dx_i, dy_i, dz_i]^T = Z_{i-1} \times (P_N - P_{i-1}) \tag{5.29}$$

where $\times$ is the cross-product operator, N is the DOF and P_i denotes the position vector from the i^{th} joint to the base frame (Frame 0). Z_i is just the 3^{rd} column of the coordinate frame for joint i where i ranges from 1 to the number of joints.

- *Angular Velocity* $[\psi_{x_i}, \psi_{y_i}, \psi_{z_i}]^T$

$$[\psi_{x_i}, \psi_{y_i}, \psi_{z_i}]^T = Z_{i-1} \tag{5.30}$$

  Equation (5.30) assumes the axis of rotation is about Z_{i-1}.

The following outlines the implementation steps for computing the Jacobian based on the above formulation.

1. Compute $T_6 = \prod_{i=1}^{N} A_{i-1}^{i}(\theta_i)$
2. Store the 4th column of T_6 as P_N.
3. FOR i= 1 to N DO STEP 4 to 7
4. Compute $G = GA_{i-1}$. At $i = 1$, G and A_0 is initialized to the identity matrix.
5. Store the 4th column of G as P_{i-1} (3 element vector).
6. Store the 3rd column of G as Z_{i-1}.
7. Compute column i of the Jacobian using equation (5.29) and (5.30).

Once the Jacobian is computed, it is used in iterative methods to calculate the end-effector's differential change. Moreover for calculating the gradient and Hessian matrix of the *Objective* function being minimized.

Computing the Gradient

Computing the gradient is often required in minimization algorithms such as Broyden, BFGS, FRPRM and Fletcher. The gradient can be calculated easily from the Jacobian as (Rabinowitz, 1970; Sasaki, 1994; Goldenberg *et al.*, 1985):

$$g_j^{(k)} = 2\sum_{i=1}^{6} f_i(x^{(k)})J_{ij}^{(k)},\ j = 1, 2, \ldots, n \tag{5.31}$$

where $J_{ij}^{(k)}$ denotes the Jacobian calculated at iteration k, f_i is the residual error as defined in equation (5.26) or (5.22), $x^{(k)}$ is the joint angles at iteration k and n is the degree of freedom (DOF) being modelled.

5.2.3 The Objective Function

An objective function is needed by all minimization algorithms. The principle role of this function is to calculate the relative error between the required and current effector's position. The skeleton structure of the objective function used in this research is defined as follows:

```
Pseudocode for Objective Function

Input: Joint Angles, $q_{1...n}$
Exit : Error, $\epsilon$

1. Calculate current end-effector (EE)'s position, $C_{EE} = FK(q_{1...n})$
   where FK is the forward kinematic function.
2. Calculate the residual vector $r$ using the formulation shown in
   equation (5.26) or (5.22).
3. Calculate error, $\epsilon = \sum_{i=1}^{G} r_i^2$
   where $G = 6$ and $G = 12$ for equation (5.22) and (5.26)
   respectively. This step can vary where some minimization
   methods require the residual vector $r$.
```

5.2.4 Algorithms for Solving IK Iteratively

The following are the minimization and root finding algorithms investigated for solving IK. The following is by no means a comprehensive explanation of each algorithm. Comprehensive details can be found in Press *et al.* (1992); Lau (1995) and Chong and Zak (1996).

Newton's Method

Newton's method is the simplest root finding method and converges efficiently if the initial guess is close (Press *et al.*, 1992). Netwon's method solves $F(q_k) = 0$ iteratively by computing $q_k = q_{k+1} + \dot{q_k}$ until convergence, where F is the objective function, q_k is the joint vector (θ_1 to θ_3) and $\dot{q_k}$ is the direction vector (*see* below). The iterative process for solving IK is shown in the following steps (From $k = 0$ to convergence) (Burden and Faires, 1985):

```
Newton's Method
Input: Initial guess for joint angles, $q^k$
Exit : solution or FAIL
1. Compute the Jacobian for the current joint configuration $q_k$,
        using the formulation shown in section 5.2.2.
2. Compute $\delta_{p_{k+1}} = Objective(q_k)$, where $Objective$ is the
        objective function defined in section 5.2.3.
3. Invert the Jacobian (J) using pseudoinverse and solve for the
    joint rates:
            $\dot{q_k} = J^{-1}\delta_{p_k}$
        $\delta_{p_k}$ is the EE differential change as shown in
        equation (7).
4. Compute the new joint angles by $q_{k+1} = q_k + \dot{q_k}$.
5. Set $k = k + 1$.
6. Exit this iteration when one of the following is true:
        A. $\sum_1^3 \dot{q}_{k2} <= TOLX$ (Tolerance factor)
        B. $\sum_1^6 \delta_{p_{k+1}} <= TOLX$ (Tolerance factor)
        C. $k > MAX$ (Max iteration)
```

Results showed that by applying the above algorithm on a 3 DOF kinematic chain requires at least 200 iterations before convergence is achieved. Therefore the improved method (NEWT) presented in Press *et al.* (1992) was used. This method uses LU decompsition to solve the equation in step 4 above and uses a *LineSearch* routine for faster convergence. The role of the line search routine is to determine a step size α_i which decreases $Objective(q^{(k+1)} < Objective(q^{(k)})$. One advantage of this method is that it solves step 4 with less computation but fails when the Jacobian is singular. On the other hand, the method presented above is robust in that the inverse of Jacobian can be obtained using pseudoinverse when the Jacobian is singular.

Powell's Method

This method solves IK without calculating the Jacobian. This method is implemented in Press *et al.* (1992). The pseudcode for the modified Powell's method is shown below (Brent, 1973):

```
Modified Powell's Method (POWELL)

Input: Initial guess for joint angles, $x^0 = (x_1, \ldots, x_n)^t$
            $u_1, \ldots, u_n$ be the colums of the identity matrix
Exit :

1. For $i = 1, \ldots, n$, compute $\beta_i$ to minimize $Objective(x_{i-1} + \beta_i u_i)$
   and define $x_i = x_{i-1} + \beta_i u_i$. See Press et al. (1992) (pg 419) for details of
   for details of linmin. This function moves to a direction where
   Objective takes on a minimum.
2. For $i = 1, \ldots, n-1$, replace $u_i$ by $u_{i+1}$.
3. Define $f_O \equiv f(x_0)$, $f_n \equiv f(x_n)$, $f_E \equiv f(2x_n - x_0)$
      IF $(f_E \geq f_O)$ OR $2(f_O - 2f_n + f_E)[(f_O - f_n) - \triangle f]^2 \geq (f_O - f_E)^2 \triangle f$
      THEN keep the direction computed in step 1, ELSE
      Compute new direction, $x_n - x_0$.
4. Compute $\beta$ to minimize $Objective(x_0 + \beta u_n)$ and replace $x_0$ by $x_0 + \beta u_n$
```

Brent's Method (PRAXIS)

Brent's is an improvement of Powell's method. The implementation details can be found in Lau (1995). This method improved Powell's method by reseting the search direction $U = [u_1, \ldots, u_n]$ every n or $n + 1$ iterations. U is reset to an orthogonal matrix $Q = [q_1, \ldots, q_n]$ which is computed using singular value decomposition (SVD) (Brent, 1973). See Golub and Kahan (1965) for a detail explaination of SVD. This modification produces a search direction which is orthogonal. Therefore the search can never become restricted to a subspace and the new direction is conjugate with respect to a matrix which is close to the Hessian matrix of the *Objective* function, thus achieving a faster convergence rate (Brent, 1973).

Fletcher-Reeves-Polak-Ribiere (FRPRM)'s Method

FRPRM solves quadratics of n variables in n steps and requires no Hessian evaluations and matrix inversion. The difference between this method and Powell's method (or Brent's) is that it does not use prespecified conjugate directions but computes the directions dynamically (Chong and Zak, 1996). The following outlines the pseudcode for the FRPRM's method (Chong and Zak, 1996; Press *et al.*, 1992):

```
FRPRM's Method
Input: Initial guess for joint angles, $x^0 = (x_1, \ldots, x_n)^t$
    TOLX (tolerance) and N (maximum iterations allowed).
Exit : Estimated minimum or FAIL

1. Set $k = 0$ and select $x^{(0)}$
2. Compute,
        $g^{(0)} = \bigtriangledown Objective(x^{(0)})$ (Compute the Gradient at $x^{(0)}$)
      IF $g^{(0)} < TOLX$ return $x^(k)$ (solved)
      ELSE Set $d^{(0)} = -g^{(0)}$
3. Find $\alpha_k$ (step length) which minimizes Objective.
        See Press et al. (1992); Chong and Zak (1996).
4. $x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$
5. $g^{(k+1)} = \bigtriangledown Objective(x^{(k+1)})$. (Gradient at $x^{(k+1)}$
      IF $g^{(k+1)} < TOLX$ return $x^(k)$ (solved)
6. Compute,
        $\beta_k = \frac{g^{(k+1)} - g^{(k)} \cdot g^{(k+1)}}{g^{(k)} \cdot g^{(k)}}$
7. $d^{(k+1)} = -g^{(k+1)} + \beta_k d^{(k)}$
8. Set $k = k + 1$
9. IF $k > N$ return ITERATION MAX
    ELSE goto Step 3.
```

Broyden's Method

Broyden's method provides cheap computational approximations to the Jacobian for zero finding. This method is also called *secant* method because it reduces to the secant method in one dimension (Press *et al.*, 1992). The pseudocode for this method is shown below (Burden and Faires, 1985):

```
Broyden's Method

Input: Initial guess for joint angles, $x^0 = (x_1, \ldots, x_n)^t$
    TOLX (tolerance) and N (maximum iterations allowed).
Exit : Solution or MAX iteration exceeded

1. Set $A_0 = J(x^0)$, where $J$ is the Jacobian.
   $v = Objective(x^0)$
2. Set $A = A_0^{-1}$
3. Set $k = 1$;
   $s = -Av$
   $x^1 = x^0 + s$
4. WHILE $(k \leq N)$ DO Steps 5-13
5. Set $w = v$
    $v = Objective(x^k)$
    $y_k = v - w$
6. Set $z = -Ay_k$
7. Set $p = -s^t z$
8. Set $C = pI + (s + z)s^t$
9. Set $A = (1/p)CA$
10. Set $s = -Av$
11. Set $x^{k+1} = x^k + s$
12. If $||s|| < TOLX$ then return $x$
13. Set $k = k + 1$
14. Return ITERATION N EXCEEDED
```

Broyden-Fletcher-Goldfarb-Shanno (BFGS)'s Method

This main feature of this method is that it provides cheap approximations to the Hessian matrix (Press *et al.*, 1992). The iterative formula used can be seen in Step 5 in the pseudocode shown below. The pseudcode for this method is as follows (Chong and Zak, 1996):

```
BFGS's Method
Input: Initial guess for joint angles, $x^0 = (x_1, \ldots, x_n)^t$
        and a positive definite $H_0$.
Exit : Estimated solution or FAIL

1. Set $k = 0$
2. Compute the gradient $g^{(k)}$.
     if $g^{(k)} == 0$ return $x^{(k)}$ else
         $d^{(k)} = -H_k g^{(k)}$
3. Compute,
         $\alpha_k = arg\ min\ Objective(x^{(k)} + \alpha d^{(k)})$
         $arg\ min\ Objective(x^{(k)} + \alpha d^{(k)})$ basically searches for
         $\alpha$ (step length) which reduces $Objective$.
         See Press et al. (1992) (pg 385) for details.
4. $x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$
5. Compute,
         $\triangle x^{(k)} = \alpha_k d^{(k)}$
         $\triangle g^{(k)} = g^{(k+1)} - g^{(k)}$
         $H_{k+1} = H_k + \left(1 + \frac{\triangle g^{(k)T} H_k g^{(k)}}{\triangle g^{(k)T} \triangle x^{(k)}}\right) \frac{\triangle x^{(k)} \triangle x^{(k)T}}{\triangle x^{(k)T} \triangle g^{(k)}}$
               $- \frac{H_k \triangle g^{(k)} \triangle x^{(k)T} + (H_k \triangle g^{(k)} \triangle x^{(k)T})^T}{\triangle g^{(k)T} \triangle x^{(k)}}$
6. Set $k = k + 1$.
7. IF $k > MAX\ ITERATION$ return FAIL
           ELSE goto step 2.
```

Fletcher's Method

This method is suitable for problems where the number of unknowns is large. The difference between this method and BFGS is the rank-two updating formulae for updating the Hessian matrix (Lau, 1995).

```
Modified Fletcher's Method
Input: Initial guess for joint angles, $x^0 = (x_1, \dots, x_n)^t$
    and a positive definite $H_0$.
Exit : Estimated solution or FAIL
1. Set $k = 0$
2. $g(x^{(k)}$ denotes a function which compute the gradient at $x^{(k)}$
        Set $d^{(k)} = -H^{(k)} g(x^{(k)})$
3.
4. Compute $\gamma^{(k)} = g(x^{(k+1)} - g(x^{(k)})$
    IF $(\delta^{(k)})^T \gamma^{(k)} < 0$ THEN
        $H^{(k+1)} = H^{(k)}$
    ELSE IF $((\delta^{(k)})^T \gamma^{(k)} > 0)$ AND $((\delta^{(k)})^T \gamma^{(k)} \geq (\gamma^{(k)})^T H^{(k)} \gamma^{(k)}$ THEN
      $H^{(k+1)} = H^{(k)} - c_1^{(k)} v^{(k)} (w^{(k)})^T - c_2^{(k)} w^{(k)} (v^{(k)})^T + (1 + (c_1^{(k)} / c_2^{(k)}) c_1^{(k)} v^{(k)} (v^{(k)})^T$
    where $c_1^{(k)} = \frac{1}{(\delta^{(k)})^T \gamma^{(k)}}, c_2^{(k)} = \frac{-1}{(\gamma^{(k)})^T H^{(k)} \gamma^{(k)}}, v_{(k)} = \delta^{(k)}, w^{(k)} = H^{(k)} \gamma^{(k)}$
    ELSE
      $H^{(k+1)} = H^{(k)} + c_1^{(k)} v^{(k)} (v^{(k)})^T + c_2^{(k)} w^{(k)} w^{(k)}$
    where $c_1^{(k)}, \dots, w^{(k)}$ as defined above.
5. IF $\|H^{(k+1)} g^{(k+1)}\| \leq \epsilon_r \|x^{(k+1)}\| + \epsilon_a$ AND $\|g(x^{(k+1)}\| \leq \epsilon_g$ AND
      $g(x^{(k+1)})^T H^{(k+1)} g(x^{(k+1)}) \geq 0$ AND $k \geq n$ ($n$=Angles (unknowns)) THEN
      $x^{(k+1)}$ is accepted as an approximation to the minimum.
```

5.3 Application of Kinematics in Human Animation

This section describes the procedures for representing the human figure. This includes assignment of coordinate frames and performing the transformations required for IK manipulation. This is needed to show the application of forward and inverse kinematics presented in the previous sections for animating human motion. An initial prototype was developed in OpenGL which provided a means for showing the *correctness* (the solved angles yield the correct position and orientation of the end-effector) of the closed-form solution derived. The final graphic system is the *Human Arm Animator* applet which uses forward and inverse kinematics to animate the human arm.

5.3.1 Human Figure Representation

Badler *et al.* (1993) classified human figure modeling into two categories, boundary and volumetric schemes. In boundary schemes, the human figure can be modeled using points and lines, polygons or surface patches. On the other hand, volumetric schemes combine the concepts of constructive solid geometry and voxels to produce the human figure.

The approach taken in this work is the boundary scheme utilizing a polygonal representation. The use of this scheme provides reasonable speed for interactive manipulation. Furthermore, if a more realistic human figure is to be produced, higher polygon counts and smooth shading can be utilized to generate a "realistic" human model (Badler *et al.*, 1993). An example of a human animation system which uses polygonal representation is the *Jack* system (Badler *et al.*, 1993).

Figure 5.3 shows the polygonal mesh and the name of each segment of the human figure in use in the Human Animator applet. The polygonal mesh for each of these segments is available in different files and is described relative to the world coordinate system. The human figure is translated to the origin of the world coordinate system with the center of the pelvis at the origin of the world frame (our view vector looks down the Z axis of the world frame). The following illustrates the steps taken to achieve this:

1. Obtain the bounding volume of the pelvis segment. This process will locate the center of the pelvis segment at the origin of the world coordinate frame. This is done by locating the minimum and maximum x (**MinX** and **MaxX**), y (**MinY** and **MaxY**) and z (**MinZ** and **MaxZ**) vertices.

2. For each segment, translate along x, y and z axis by $(MinX + MaxX)/2)$, $(MinY + MaxY)/2)$, $(MinZ + MaxZ)/2)$ respectively.

Figure 5.4 shows how body parts are linked together. This hierarchy is highly dependent on the complexity of the model (number of segments used). For example, the *Jack* system uses a model which consist of 69 segments and 68 joints

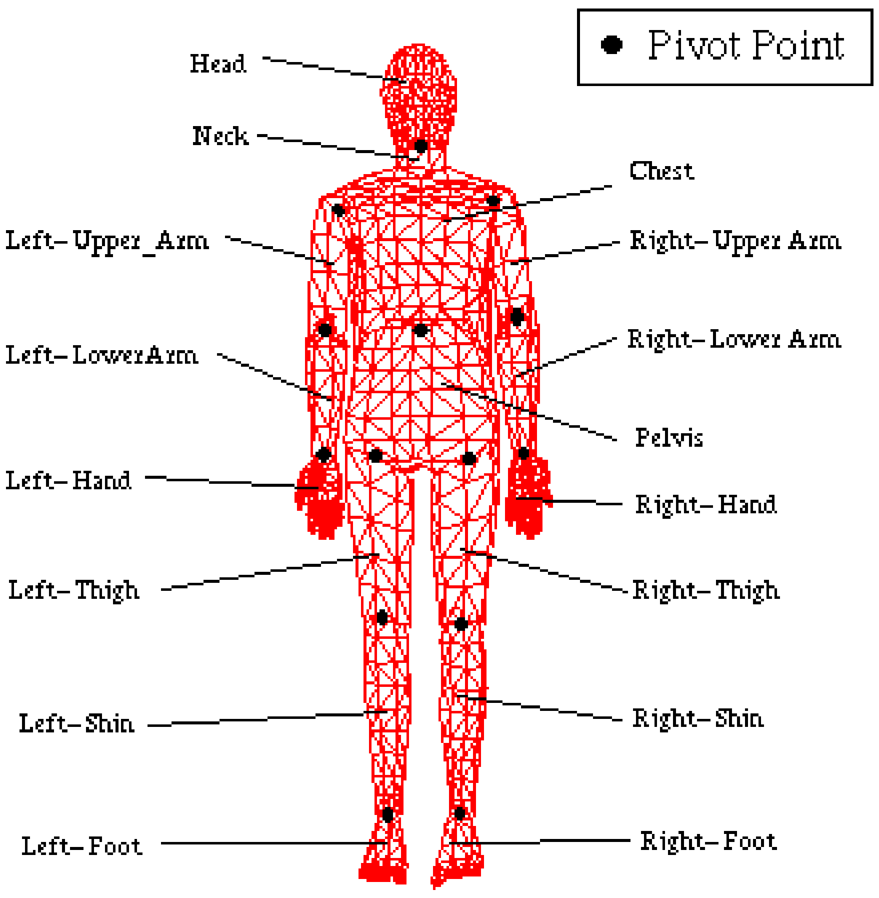


Figure 5.3: Polygonal representation of human figure and pivot points location

(Badler *et al.*, 1993). Therefore, the *Jack* system has more nodes than the model used here.

Each segment of the model is linked to another to form a hierarchy. When the human figure is manipulated (e.g. change of angle at a given node), the effect must be propagated down to its child nodes. For example, if the **Left-Upper Arm** node is manipulated, the changes also effect the **Left-Lower Arm** and **Left-Hand**. This is certainly true since if the upper-arm (humerus) is moved, it is expected that the lower arm (ulna and radius) and the hand move with the upper-arm. A change in the pelvis node would amount to expensive computation since all nodes are descedent from the pelvis node where a number of joint updates are neccessarily.

The degree of freedom (DOF) for each joint is summarized in Table 5.1. Table 5.2 shows the range for the human arm. Only the measurement for the humam arm is shown because to present the work in this research, the human arm is sufficient.

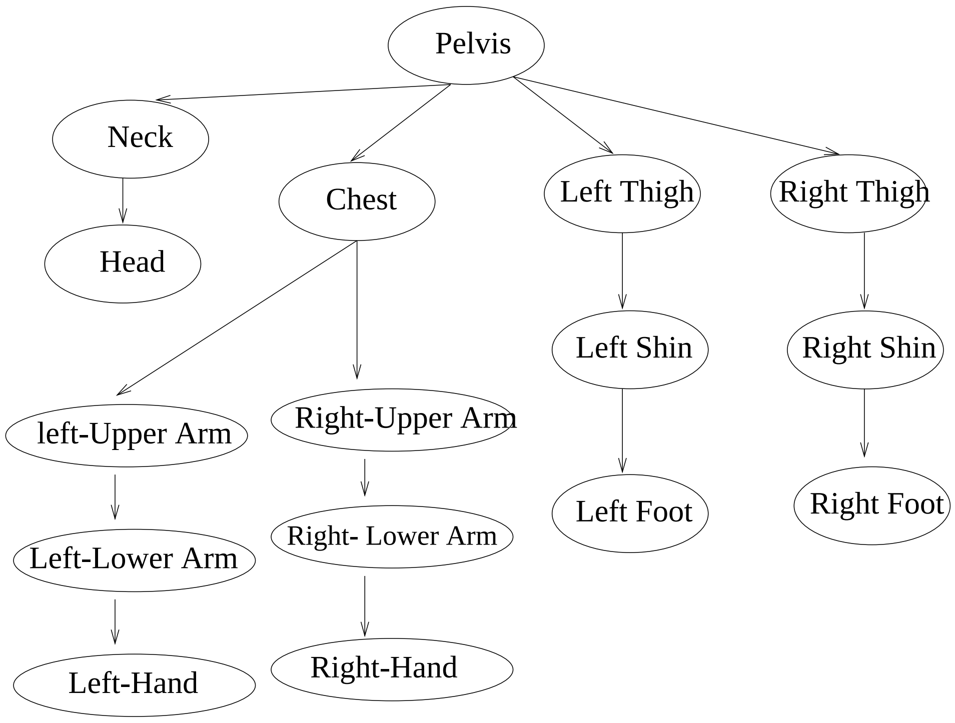


Figure 5.4: Tree Hierarchy for Human Figure

5.3.2 Human Figure Manipulation

The coordinate assignment of a given segment is very important. This is because incorrect frame assignment results in erroneous IK formulation. This is because the end-effector is describe numerically as a 4×4 matrix and is difficult to visualize without graphical means. Each joint or body segment rotates about its own local axis. This local axis is related to the world coordinate frame by a translation along the X, Y and Z axis relative to the world coordinate frame.

The procedure for assigning a local coordinate frame for each joint starts by assigning a pivot point to origin of local coordinate frame or the origin of the local axis relative to the world coordinate frame to each joint. This is done with the help of 3D modeling software such as 3D-Studio 4.0. The pivot point assignment for the human figure can be seen in Figure 5.3.

Once all pivot points are assigned, the local coordinate frame can be assigned to the given joint. The local axis is orthogonal to the world coordinate frame so no

Body Segments	Degree of Freedom (DOF)	Axis of Rotation
Neck	3	XYZ
Pelvis	3	XYZ
Left Upper Arm	3	XYZ
Left Lower Arm	2	XY
Left Hand	3	XYZ
Right Upper Arm	3	XYZ
Right Lower Arm	2	XY
Right Hand	3	XYZ
Left Thigh	3	XYZ
Left Shin	1	X
Left Foot	3	XYZ
Right Thigh	3	XYZ
Right Shin	1	X
Right Foot	3	XYZ

Table 5.1: DOF Assignment for Human Figure

Body Segments	X-axis	Y-axis	Z-axis
Shoulder	$-60 \leq \theta \leq 180$	$-70 \leq \theta \leq 90$	$0 \leq \theta \leq 180$
Elbow	$0 \leq \theta \leq 150$	$-80 \leq \theta \leq 80$	0
Wrist	$-70 \leq \theta \leq 80$	0	$-20 \leq \theta \leq 30$

Table 5.2: Angles' Range for the Human Arm (Norkin and White, 1988)

rotation is needed to align the assigned local axis.

Changes of angle in a given node propagate down to its child nodes. This is where the joint angles for the given articulated structure being modelled are specified or calculated from forward or inverse kinematics. An example of articulated structure manipulation is presented as follows:

Taking the left arm of Figure 5.3 as an example and we rotate the **Left-Upper Arm** about its own local axis, the following shows the manipulation at the shoulder (**Upper-Arm**) and the change propagation to the lower arm and hand. The steps below essentially outlines the implementation of forward kinematic for the human arm.

1. Translate the **Left-Upper Arm** to the World Coordinate Frame by (P_x,P_y,P_z), where (P_x, P_y,P_z) is a vector which describes the **Left-Upper Arm**'s pivot relative to the world coordinate frame.

2. Rotate the **Left-Upper Arm** by (θ_x, θ_y, θ_z).

3. Translate the left shoulder back to where it was by ($-P_x$,$-P_y$,$-P_z$).

4. Translate the **Left-Lower Arm** by (P_x-Q_x, P_y-Q_y, P_z-Q_z) where (Q_x, Q_y, Q_z) is the pivot point for the **Left-Lower Arm**. Then translate it again by (P_x,P_y,P_z) to move the Left-Lower Arm to the world coordinate frame.

5. Rotate the **Left-Lower Arm** by (θ_x, θ_y, θ_z).

6. Translate the **Left-Lower Arm** back to its original position by (-(P_x-Q_x), -(P_y-Q_y), -(P_z-Q_z)) and ($-Q_x$, $-Q_y$, $-Q_z$)

7. The **Left-Hand** is translated to the World origin by translating the **Left-Hand** to the **Left-Lower Arm**'s pivot point, then translate to the Left-Uper Arm's pivot point and finally onto the world origin as in step 4.

8. Rotate the **Left-Hand** by (θ_x, θ_y, θ_z).

9. Finally, translate the **Left-Hand** back to it's original position. The reverse of step 7.

This applies to each manipulated node except for the hand and foot node. This is because no other nodes are relative to these nodes. The above operations can be concatenate into a 4x4 matrix which reduces computation significantly.

Using IK for Human Figure Manipulation

The previous section outlined the implementation details of forward kinematics. The use of IK for manipulating the human articulated structure follows the following basic steps:

1. Get current joint angles for articulated structure.
2. Get the required end-effector position and orientation. This is represented as a 4×4 homogeneous matrix. If the end-effector is to be translated along a trajectory starting from the current end-effector position then, first use the joint angles from Step 1 to compute the current end-effector frame which is then translated and rotated (relative to the end-effector frame) as required by the trajectory.
3. Then use IK (iterative or algebraic) to solve for the joint angles for the required frame calculated in Step 2.
4. Finally use the algorithm presented in Section 5.3.2 to reflect the changes in joint angles.

5.4 Summary

This chapter has discussed the procedures for deriving a closed-form solution for an arm modeled as 6 DOF. The closed-form solution derived can be hard-coded directly. Further factoring was used to avoid repeated terms to reduce computation of transcendental functions. As can be seen from the algebraic method various solutions can be derived. For example the closed-form solutions for θ_2 yield two angles which can be used to derive different configuration for a given end-effector position and orientation. Solving IK iteratively was presented and the pseudocode for each minimization and root finding algorithm has been outlined. Finally the implementation details of forward and inverse kinematics for animating a human figure has been outlined. The representation of the human figure and the process of transforming each joint relative to its local frame has been discussed in detail.

Chapter 6

RESULTS AND ANALYSIS

This chapter presents the results generated from this research work, analysis of results and analysis of the algebraic and iterative methods for solving IK. Hypotheses which were addressed in Chapter 4 are evaluated.

6.1 Analysis of Algebraic Solution

The algebraic method is susceptible to problems with transcendental functions (e.g. *sin* function) (McKerrow, 1991). The problems which are inherent in closed-form solutions are:

1. The division by $\cos(-90°)$, $\cos(90°)$, $\sin(0°)$, $sin(180°)$ or $tan(0°)$ results in inaccuracy (when near these angles) and indeterminacy. This was avoided by bringing over (inverting) the next A_i transformation matrix and deriving the closed-form equation for the required angle using the new set of equations.

2. The closed-form solutions presented in section 5.1 degenerate when the square root portion of the closed-form equation for θ_2 or θ_5 becomes negative $((a^2 + b^2 - c^2) < 0)$. This occurs when the arm is out-stretched. This is because the axes of joint 1 and 2 are aligned (McKerrow, 1991). When this occurs, θ_2 and θ_5 were set to an arbitrary angle (e.g 0). This is because this articulated structure is in a degenerated state where there are infinite number of solutions for the required end-effector position and orientation.

Algebraic solutions are significantly faster than iterative solutions. Computation expense can be decreased by using lookup tables for transcendental (i.e *sin, cos, atan*) functions, factoring out repeated terms and storing computed expression. Further all solutions for the given end-effector position can be derived. This can be seen in the closed-form equation for θ_2 where there are two unique solutions. These two solutions can then be substituted into the closed-form solutions for (θ_3 to θ_6). Choosing the appropriate solutions involve eliminating solutions which violate joint constraints. If there are multiple remaining solutions given that the articulated structure is not in a degenerated state, then the closest (the difference between the current and solved joint angles is at a minimum) solution to the current arm solution is chosen (Craig, 1989). The problem with the algebraic method is that not all kinematic chains have a closed-form solution especially kinematic chains which have more than six DOF (McKerrow, 1991).

6.2 Analysis and Results of Iterative Solutions of a 6 DOF Kinematic Chain

The following presents IK results solved iteratively for a human arm modeled at 6 DOF. The corresponding transformation frames used to describe the human arm (using DH-Notation) can be found in Appendix B. The following data were generated by translating the end-effector (hand) along a line. At each unit step, a new pose for the hand was generated and IK was used to solve for the joint angles.

Example 1

Given that the current frame has the following position and orientation:

$$C_{EE} = \begin{bmatrix} -0.999960 & 0.008954 & -0.000079 & 0.959823 \\ -0.000155 & -0.026099 & -0.999659 & -109.984770 \\ -0.008953 & -0.999619 & 0.026099 & 1.483471 \\ 0.000000 & 0.000000 & 0.000000 & 1.000000 \end{bmatrix}$$

and the joint angles are at [0.5 0.5 0.5 0.5 0.5 0.5].

Now translate the hand (C_{EE}) by one along x and y,

$$D_{EE}=\text{Translate}(C_{EE},\ 1,\ 1,\ 0)$$

where *Translate* is a function which translates a given frame by x,y and z relative to the world coordinate frame. Orientation of the end-effector can be done by the rotation transformation if needed. The required frame is then calculated as (returned from *Translate*):

$$D_{EE}=\begin{bmatrix} -0.999960 & 0.008954 & -0.000079 & 1.959823 \\ -0.000155 & -0.026099 & -0.999659 & -108.98477 \\ -0.008953 & -0.999619 & 0.026099 & 1.483471 \\ 0.000000 & 0.000000 & 0.000000 & 1.000000 \end{bmatrix}$$

IK is then invoked (as presented in the previous chapter) to solve for the joint angles needed to achieve the orientation and position of the hand defined by D_{EE}. The following matrix shows the result after minimization (using NEWTON's method),

$$New_{EE}=\begin{bmatrix} -0.999960 & 0.008954 & -0.000079 & 1.959822 \\ -0.000155 & -0.026099 & -0.999659 & -108.984747 \\ -0.008953 & -0.999619 & 0.026100 & 1.483472 \\ 0.000000 & 0.000000 & 0.000000 & 1.000000 \end{bmatrix}$$

The corresponding angles (new arm configuration) which yield the above pose were [1.03021 -7.64128 15.43075 0.48902 -6.28465 1.08795]. The residual error (difference between D_{EE} and New_{EE}) was 0.000024.

6.2.1 Convergence Rate

Table 6.1 shows the average number of iterations before convergence for each of the minimization and root finding methods (NEWTON and BROYDN) used to solve the IK for a 6 degree of freedom (DOF) arm. The residual error ($\sum_{i=1}^{6} r_i^2$) for each trajectory step was 4.000. The data in Table 6.1 show the average number of iteration as the end-effector was translated along a trajectory of length 10. As can be seen from Table 6.1 (and Appendix C.1) NEWTON's method has the fastest convergence rate. This is because NEWTON's method has fast convergence when

Iterative Methods	Average Convergence Rate
NEWTON	3.2
BFGS	9.5
POWELL	11.5
FRPRM	14.1
PRAXIS	14.9
FLETCHER	15.9
BROYDN	18.5

Table 6.1: Average Convergence Rate for Iterative Methods

the initial estimate is close (Burden and Faires, 1985). The disadvantage of NEWTON's method is that the routine fails when the Jacobian becomes singular and non-square. The use of the pseudoinverse provides a solution to this where solutions can be obtained whether the system of equations is underspecified, overspecified or singular. On the other hand, the use the pseudoinverse causes discontinuity between the transition of singular and nonsingular configurations (Maciejewski, 1990).

6.2.2 Objective Function Evaluations

Table 6.2 shows the average number of function calls needed for each method before convergence was achieved. As can be seen from Table 6.2 (and Appendix C.2), the POWELL and PRAXIS minimization algorithms required many function calls since these methods are based purely on the Objective function only (i.e without calculating derivatives). The disadvantage of this is that with high DOF articulated structures, the objective function is expensive to compute (multiplication of N matrices, evaluation of sin,cos and atan functions). The computation expense is proportional to N. This affects the time needed to traverse through the trajectory.

Iterative Methods	Average Function Evaluations
POWELL	1428.5
FRPRM	279.1
PRAXIS	264.1
FLETCHER	100
BROYDN	66.3
BFGS	63.9
NEWTON	24.1

Table 6.2: Number of function evaluations (6 DOF Arm)

6.2.3 Minimization Algorithms with Large Residual Error

The aim of this is to investigate the ability of various iterative methods in handling the constraints where the residual error between the current and required end-effector frame is large. This is useful where the end-effector can be specified anywhere within the reachable workspace as opposed to translating it along fixed size steps along the trajectory to the required position and orientation. Table 6.3 and 6.4 were generated where the residual error is at 200.

Iterative Methods	Average Convergence Rate
BFGS	8.2
POWELL	11.5
NEWTON	12.1
FLETCHER	14.9
PRAXIS	15.7
FRPRM	See Appendix G
BRODYN	FAIL

Table 6.3: Convergence Rate with Large Residual Error

Iterative Methods	Average Function Evaluations
BFGS	76.6
FLETCHER	100
NEWTON	112.5
PRAXIS	274.6
POWELL	1392.5
FRPRM	See Appendix G
BRODYN	FAIL

Table 6.4: Function Evaluations with Large Residual Error

6.3 Iterative IK Solutions of 8 and 3 DOF Articulated Structure

The iterative methods presented were also used to solve the IK of 8 and 3 DOF articulated structures. This is to demonstrate its generality in solving articulated structures having N DOF. The articulated structures modeled are shown in Appendix D and Appendix E. The best three minimization algorithms were used to solve the IK of an N DOF articulated structure base on its fast convergence rate, least function evaluations and handling singularities. Another problem which arose was that the BRODYN and NEWTON routines provided by Press *et al.* (1992) assumed that the Jacobian is square and has the same dimension as the number of DOF. The POWELL or PRAXIS methods were not considered in the 8 DOF case due to the high computation cost of evaluating the *Objective* function. The convergence rate and number of function evaluations needed is shown in Table 6.5 and Table 6.6.

Iterative Methods	Average Convergence Rate
BFGS	3.7
FLETCHER	18.0
FRPRM	44.7

Table 6.5: Average Convergence Rate for Minimization Algorithms (8 DOF)

Iterative Methods	Average Function Evaluations
BFGS	43.4
FRPRM	44.5
FLETCHER	99.7

Table 6.6: Average Number of Function evaluations (8 DOF)

Iterative Methods	Average Convergence Rate
BFGS	2.6
FRPRM	13.8
FLETCHER	18.0

Table 6.7: Average Convergence Rate for Minimization Algorithms (3 DOF)

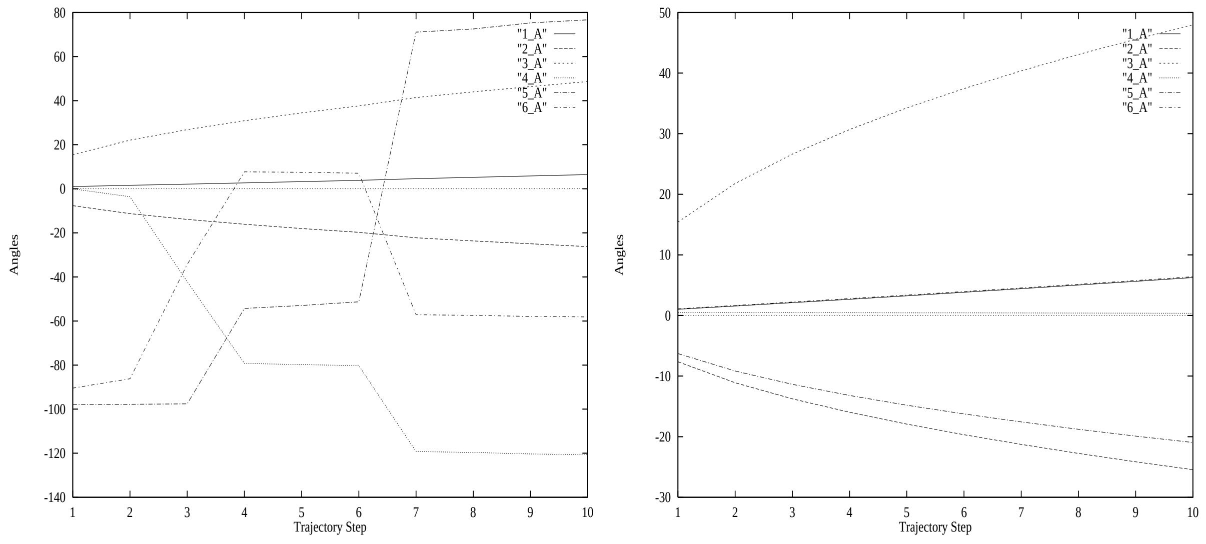
Iterative Methods	Average Function Evaluations
BFGS	23.3
FLETCHER	58.8
FRPRM	100.0

Table 6.8: Average Number of Function evaluations (3 DOF)

This shows that using iterative methods for solving IK, the IK of a given articulated structure can be generated dynamically where only the coordinate frames need to be specified. As mentioned in Section 5.3.1, the human figure is represented as a tree hierarchy where the coordinate frame between each joint is known. Therefore these coordinate frames can be used directly by the iterative IK method.

6.4 Analysis of Joint Angles Computed Iteratively

Figure 6.1(a) and 6.1(b) shows two different changes in joint angles as computed by BFGS and NEWTON's method respectively (residual error is at 4.0). The joint angles computed by other numerical methods can be found in Appendix F. (**1_A** to **6_A** correspond to θ_1 to θ_6 respectively) Figure 6.1(a) and and 6.1(b)



(a) (b)

Figure 6.1: Changes in Joint angles using (a) BFGS's (b) NEWTON's Method

and figures in Appendix F show different joint angles configurations as the 6 DOF kinematic chain was translated along a line. Different numerical methods produced different joint angles at each trajectory step. By comparison, NEWTON, BROYDEN and FLETCHER's method produced similar joint configurations. This can be seen in Figure 6.1(a), F.1 and F.4 respectively (*See* Appendix F for Figures F.1 to F.4). The joint angles generated from FLETCHER's method shown in Figure F.4 were inconsistent from step 7 to step 10. This was due to the convergence error which can be seen in Figure G.2 (Appendix G).

Hypothesis 4 states that if the residual error between the target and current end-effector (EE) is small then moving from the current frame to the target frame does not cause huge changes to each joint angle. This is useful because when an animator moves an end-effector (e.g. hand) to a required position, a sudden change in joint angles might cause the articulated structure to collide with the environment or make the motion of the articulated structure seem *unnatural*. Analysis of the joint configuration for each numerical method shows that this is only true for certain algorithms. For example, in BFGS and PRAXIS's method, huge change in joint angles at each iteration is needed. This shows the variety of joint configurations which can be used to derive the required end-effector position. In contrast with algebraic solutions , where all solutions for the required end-effector position are available, this provides the animator with better choices given the option to choose the appropriate joint configurations.

6.5 The Human Arm Animator Applet

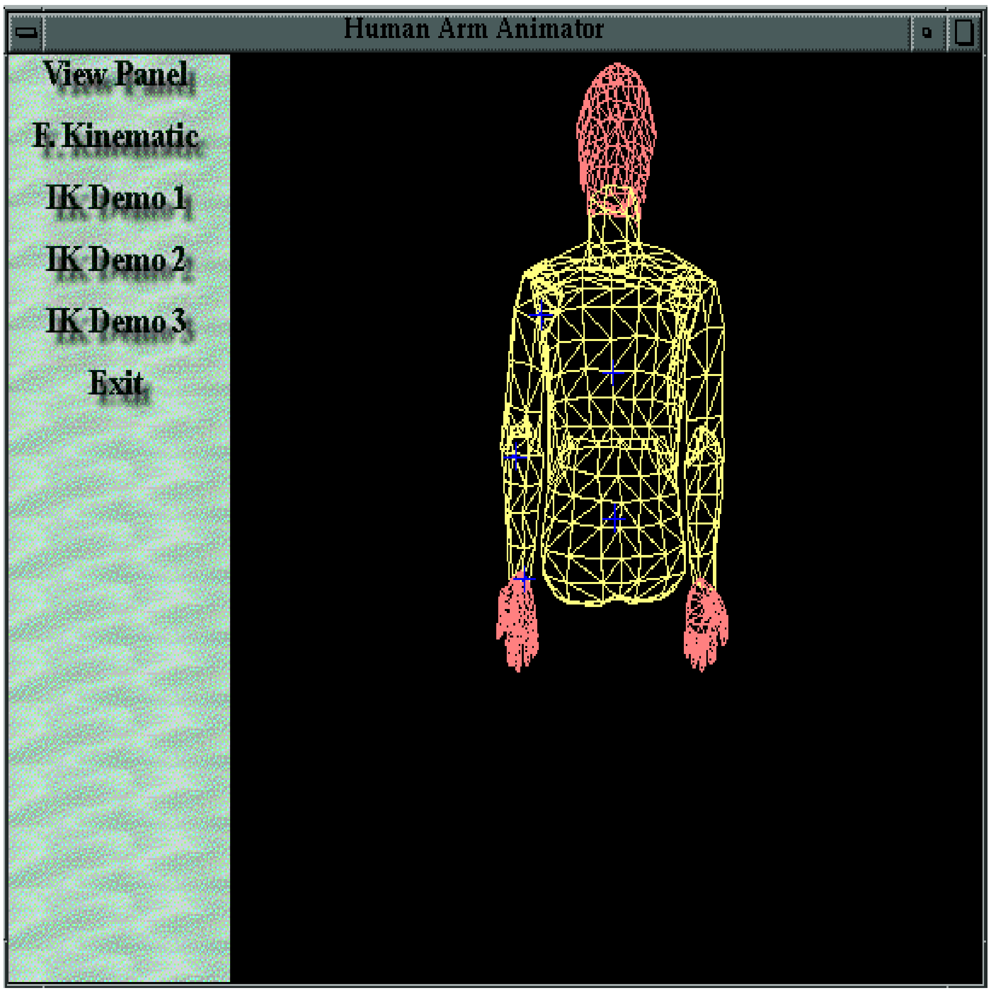


Figure 6.2: Snapshot of the Human Arm Animator Applet

Figure 6.2 shows the high-level interface where the closed-form solutions derived were used to animate the human arm. This high-level interface was implemented in the Java language which takes advantage of its portability, object-oriented and multi-threading features. The applet shown in Figure 6.2 allows the user to position and orient the left arm using forward and inverse kinematics.

6.5.1 Forward Kinematics

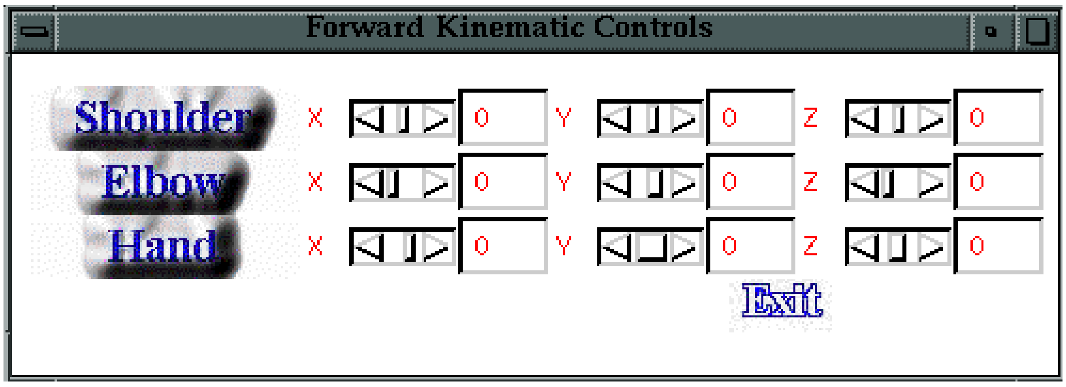


Figure 6.3: Forward Kinematic Panel

Forward kinematics of the human arm is controlled using scroll bars to specify the joint angles. This is shown in Figure 6.3. Each angle change is then applied to the given DOF using the algorithm presented in Section 5.3.2.

6.5.2 Inverse Kinematics

This applet uses closed-form solutions presented in Section 5.1 for animating flexion and abduction at the elbow and shoulder. The closed-form solutions were used because there are less computational expensive than the iterative methods and all solutions for the required end-effector frame can be obtained. Flexion of the forearm at the elbow is achieved by continuosly translating the hand as follows:

$$Hand = Translate(0,\ 0.25,\ 0.25);$$

Any amount of translation can be specified. The number given above affects the speed of flexion. Smaller numbers would mean smoother animation of the arm. Of importance to note is that the arm is flexed by translating the hand up along $\mathbf{y}$ and along $\mathbf{z}$ and this action requires movement from at least two DOF (elbow and shoulder). Then IK is used to solve for the joint angles needed to obtain the required position and orientation of the hand. Initialy the arm is at at outstretched position; if a -y translation is performed, the closed-form solution would be degenerated because the required position is unreachable.

Abduction is achieved by translating the hand as follows:

$$\textit{Hand} = \textit{Translate(}0.25\textit{, }0.25\textit{, }0\textit{);}$$

This effectively moves the hand to the side and up. IK is used to solve for the joint angles as in the flexion case. The amount to be translated (x and y) determines the speed of abduction.

Chapter 7

CONCLUSION

This thesis has investigated the algebraic and iterative methods for solving IK. Extensive formulation, investigation and implementation of both methods have been discussed. The original aims which were presented in Chapter 2 were achieved. Each of the subproblems were solved and detailed analysis and results were presented. Closed-form solutions for the human arm were derived and have been incorporated into the *Human Arm Animator* Applet. As can be seen from the process of deriving the closed-form solution, it is a lengthy process, specific for a given kinematic chain and error prone (too much human intervention). On the other hand, it provides all possible solutions for the required end-effector frame as shown. Further it is less computationally expensive (compared to the iterative methods) and is ideal for environment such as Java.

Apart from that, generalized methods for solving IK were presented. This enables any articulated structure (human, animal, robot manipulator) to be manipulated where the frames which describes the articulated structure can be dynamically fed into the generalized IK engine and the IK for the given articulated structure is solved iteratively. Comparative study has shown that this method is computationally expensive compared to the algebraic method. The BFGS's method was found to be the most robust, has fast and consistent convergence rate and overall least objective function evaluations among the iterative methods compared. The BFGS's method was used for solving the IK of a three and eight DOF articulated structure to demonstrate its flexibility in handling N DOF articulated structure. Moreover, the case where the initial estimate and the required position is not close was considered. This was to investigate the global convergence of each iterative method. This feature enables the animator to specify the position and orientation of the end-effector arbitrary instead of transversing a fixed step size trajectory. It was found that BFGS, POWELL and NEWTON's perform well in this case. The modified NEWTON's method only performs well when the Jacobian is square and non-singular. The incorporation of the pseudoinverse to invert the Jacobian when it is non-square and singular was not considered because the use of pseudoinvese and minimum norm method results in computation which is generally formidable (Sasaki, 1994) and did not address joint limits (Zhao and Badler, 1994). Other methods such as POWELL and PRAXIS methods were high in objective function evaluation. This is due to the nature of the algorithm where the derivative of the objective function is not needed. Moreover these methods have a good convergence rate.

The following summarizes the discussion in relation to the hypotheses presented in Section 4.1.

1. This thesis has shown that the iterative method provides a dynamic and flexible method for solving IK. Only coordinate frames which describe the articulated structure to be manipulated are needed. This implies that the generality of the iterative method can be used for animating other forms of articulated figures; for example animals.

2. Iterative results have shown that it only converges to one solution. Furthermore there is no way in which a different starting point can be given which guarantees a different set of solutions. Since the constraint used in this thesis is that the initial estimates and the correct solution is close, it provides efficient computation. Constraint optimization can be used to alleviate this problem but constraint optimization methods are inevitably expensive in terms of computer resources and with the added complexity of the problem which may occur after many restarts or yield and un-satisfactory solution (Massara, 1991).

3. The use of IK which is used inherently in the robotics area has been applied successfully for animating the human figure. This was demonstrated by the *Human Arm Animator* applet. This applet provides a visual feedback on

the *correctness* of the IK solutions.

4. The joint angles computed when the current and target frame is close depends on the iterative method used. Analysis of the joints angle computed as it transverses through a trajectory have shown that different minimization algorithms produce varying sets of solutions. This reflects the fact that there is more than one solution to a required end-effect frame.

7.0.3 Future Directions

The next stage for this research is to implement a full IK human motion animation package. Another application would be as a teaching tool for the School of Physiotherapy.

Apart from that, research on constrained optimization has been the current focus of optimization algorithms. In constraint optimization a set of constraints can be enforced on the variables. The optimization routine than take these constraints into account to search for a feasible solution (Massara, 1991). Constraint optimization can be incorporated easily with the existing work to provide a more refined IK engine where the upper and lower bound of each joint angle is addressed.

Bibliography

Adby, P. R. and Dempster, M. A. H. (1974). *Introduction to Optimization Methods*. Chapman and Hall.

Albert, A. E. (1972). *Regression and the Moore-Penrose Pseudoinverse*. Academic Press.

Angeles, J. (1985). On the numerical solution of the inverse kinematic problem. *The International Journal of Robotic Research*, **4**(2), 21–37.

Anon (1992). *SEA Mathematical Formulae and Statistical Tables Book*. Secondary Education Authority.

Badler, N. I. (1987). Articulated figure animation. *IEEE Computer Graphics and Applications*, **7**(6), 10–11.

Badler, N. I., Manoochehri, K. H., and Walters, G. (1987). Articulated figure positioning by multiple constraints. *IEEE Computer Graphics and Applications*, **7**(6), 28–38.

Badler, N. I., Phillips, C. B., and Webber, B. L. (1993). *Simulating Humans: Computer Graphics Animation and Control*. Oxford University Press.

Brent, R. P. (1973). *Algorithms for Minimization without Derivatives*. Prentice-Hall.

Burden, R. L. and Faires, J. D. (1985). *Numerical Analysis*. PWS Publishers, 3rd edition.

Byrne, G. D. and Hall, C. A., editors (1973). *Numerical Solutions of Systems of Nonlinear Algebraic Equations*. Academic Press.

Cary B, . P. and Badler, N. I. (1991). Interactive behaviours for bipedal articulated figures. *Computer Graphics (SIGGRAPH '91 Proceedings)*, **24**(4), 359–362.

Chong, E. K. P. and Zak, S. H. (1996). *An Introduction to Optimization.* John Wiley and Sons, INC.

Coco, D. (1995). Breathing life into 3d humans. *Computer Graphics World*, **18**(8), 41–48.

Craig, J. J. (1989). *Introduction to Robotics: Mechanics and Control.* Addison-Wesley.

CSEP (1995). Mathematical optimization, [WWW]. *Technical Report. http://csep1.phy.ornl.gov/CSEP/MO/MO.html*, Accessed: 23 Sept. 1996.

Dennis, J. E. and et al (1974). Quasi-newton methods: Motivation and theory. Technical report, Cornell University.

Featherstone, R. (1983). Position and velocity transformation between robot end-effector coordinate and joint angle. *The International Journal of Robotics Research*, **2**(2), 35–45.

Fletcher, R. (1968). Generalized inverse methods for the least square solution of systems of non-linear equations. *Computer Journal*, **10**, 392–399.

Girard, M. (1987). Interactive design of 3d computer animated legged animal motion. *IEEE Computer Graphics and Applications*, **7**(6), 39–51.

Girard, M. and Maciejewski, A. A. (1985). Computational modeling for the computer animation of legged figures. *Computer Graphics (SIGGRAPH '85 Proceedings)*, **19**(3), 263–270.

Goldenberg, A. A. and Lawrence, D. L. (1985). A generalized solution to the inverse kinematics of robotic manipulator. *Journal of Dynamic Systems, Measurement, and Control*, **107**, 103–107.

Goldenberg, A. A., Benhabib, B., and Fenton, R. G. (1985). A complete generalized solution to the inverse kinematics of robots. *IEEE Journal of Robotics and Automation*, **RA-1**(1), 14–20.

Golub, G. and Kahan, W. (1965). Calculating the singular values and pseudo-inverse of a matrix. *Journal SIAM Numerical Analysis*, **2**(2), 205–223.

Klein, C. A. and Huang, C.-H. (1983). Review of pseudoinverse control for use with kinematically redundant manipulators. *IEEE Transaction on Systems, Man, and, Cybernetics*, **SMC-13**(3), 245–250.

Klein, C. A. and Kee, K.-B. (1989). The nature of drift in pseudoinverse control of kinematically redundant manipulators. *IEEE Transactions on Robotics and Automation*, **5**(2), 231–234.

Ko, H. and Badler, N. I. (1996). Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and Applications*, **16**(2), 50–59.

Korein, J. U. and Badler, N. I. (1982). Techniques for generating the goal-directed motion of articulated structures. *IEEE Computer Graphics and Applications*, **2**(9), 71–81. Good.

Lau, H. T. (1995). *A Numerical Library in C for Scientists and Engineers*. CRC Press.

Leahy Jr, M. B., Nugent, L. M., Saridis, G. N., and Valavanis, K. P. (1987). Efficient puma manipulator jacobian calculation and inversion. *Journal of Robotic Systems*, **4**(4), 185–197.

Lee, G. C. S. (1982). Robot arm kinematics, dynamics and control. *Computer*, **15**(12), 62–79.

Lee, G. C. S. (1983). *Tutorial on Robotics*, chapter 2, pages 47–65. IEEE Computer Society.

Maciejewski, A. A. (1990). Dealing with the ill-conditioned equations of motion for articulated figures. *IEEE Computer Graphics and Applications*, **10**(3), 63–71.

Maciejewski, A. A. and Klein, C. A. (1985). Sam-animation software for simulating articulated motion. *Computer and Graphics*, **9**(4), 383–391.

Magnenat-Thalmann, N. and Thalmann, D. (1988). Synthetic actors as an interdisciplinary concept for human animation. In *ACM SIGGRAPH '88 (Course Notes)*, chapter 1, page 3. ACM Press.

Manocha, D. and Canny, J. F. (1994). Efficient inverse kinematics for general 6r manipulators. *IEEE Transactions on Robotics and Automation*, **10**(5), 648–657.

Manocha, D. and Zhu, Y. (1994). A fast algorithm and system for the inverse kinematics of general serial manipulators. *IEEE Conference on Robotics and Autmomation 94*, pages 3348–3354.

Massara, R. E. (1991). *Optimization Method in Electronic Circuit Design*. Longman Scientific and Technical.

Mathews, J. H. (1992). *Numerical Methods: For Mathematics, Science and Engineering*. Prentice-Hall.

McKerrow, P. J. (1991). *Introduction to Robotics*. Addison-Wesley.

Nakamura, Y. and Hanafusa, H. (1986). Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement and Control*, **108**(3), 163–171.

Noble, B. and Daniel, J. W. (1989). *Applied Linear Algebra*. Prentice-Hall.

Norkin, C. C. and White, D. J. (1988). *Measurement of Joint Motion: A Guide to Goniometry*. F. A. Davis Company.

Orin, D. E. and Schrader, W. W. (1984). Efficient computation of the jacobian for robot manipulators. *The International Journal of Robotic Research*, **3**(4), 66–75.

Ortega, J. M. and Rheinboldt, W. C. (1970). *Iterative Solution of NonLinear Equations In Several Variables*. Academic Press.

Paul, R. (1979). Manipulator cartesian path control. *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-9**(11), 702–711.

Paul, R. P. (1981). *Robot Manipulators: Mathematics, Programming and Control*. The MIT Press.

Paul, R. P., Shimano, B., and Mayer, G. E. (1981). Kinematic control equations for simple manipulators. *IEEE Transactions On Systems, Man, And Cybernetics*, **SMC-11**(6), 66–72.

Phillips, C. B., Zhao, J., and Badler, N. I. (1990). Interative real-time articulated figure manipulation using multiple kinematic constraints. *Computer Graphics (SIGGRAPH Proceedings '90)*, **24**(2), 245–251.

Powell, M. J. D. (1964). An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Computer Journal*, **7**, 155–162.

Press, W. H., Flannery, B. P., and Teukolsky, S. A. (1992). *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge Univeristy Press.

Rabinowitz, P. (1970). *Numerical Methods for Nonlinear Algebraic Equations.* Gordon and Breach Science Publishers.

Rao, C. R. and Mitra, S. K. (1971). *Generalized Inverse of Matrices and its Applications.* John Wiley and Sons, Inc.

Rijpkema, H. and Girard, M. (1991). Computer animation of knowledge-based human grasping. *Computer Graphics (SIGGRAPH '91 Proceedings)*, **25**(4), 339–348.

Robertson, B. (1995). Toy story: A triumph of animation. *Computer Graphics World*, **18**(8), 28–38.

Rosen, J. B. (1960). The gradient projection method for non-linear programming. *Journal of Applied Mathematics*, **8**(1), 181–217.

Sasaki, S. (1994). On numerical techniques for kinematics problems of general serial-link robot manipulators. *Robotica*, **12**, 309–322.

Sasaki, S. (1995). Feasibility studies of kinematics problems in the case of a class of redundant manipulators. *Robotica*, **13**, 233–241.

Snyder, W. E. (1985). *Industrial Robots: Computer Interfacing and Control.* Prentice-Hall.

Tsai, Y. T. and Orin, D. E. (1987). A strictly convergent real-time solution for inverse kinematics of robot manipulators. *Journal of Robotic Systems*, **4**(4), 477–501.

van de Panne, M. (1996). Parameterized gait synthesis. *IEEE Computer Graphics and Applications*, **16**(2), 40–49.

Wang, L.-C. T. and Chen, C. C. (1991). A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. *IEEE Transactions On Robotics and Automation*, **7**(4), 489–499.

Watt, A. and Watt, M. (1992). *Advanced Animation and Rendering Techniques*. Addison-Wesley.

Whitney, D. E. (1972). The mathematics of coordinated control of prosthetic arm and manipulators. *Journal of Dynamic Systems, Measurement and Control*, **122**, 303–309.

Wu, C.-H. and Paul, R. P. (1982). Resolved motion force control of robot manipulator. *IEEE Transactions on Systems, Man and Cybernetics*, **SMC-12**(3), 266–275.

Zeltzer, D. (1982). Motor control techniques for figure animation. *IEEE Computer Graphics and Application*, **2**(11), 53–59.

Zhao, J. and Badler, N. I. (1994). Inverse kinematics positioning using nonlinear programming for highly articulated structure. *ACM Transactions on Graphics*, **13**(4), 313–336.

Zomaya, A. Y. (1992). *Modelling and Simulation of Robot Manipulators: A Parallel Approach*. World Scientific.

Appendix A

Workings for Algebraic Solution

The following shows the workings for the derivation of the algebraic solution presented in this thesis. The equations above are derived with the help of MAPLE (Symbolic mathematical package).

A.1 $A_2A_3A_4A_5A_6 = A_1^{-1}T_6$

R_Left and R_Right denotes the left and right hand side of the above expression. R_Left andR_Right presents a 4×4 matrix after multiplication of the matrices in the above expression. Therefore, row 1 and column 1 can be accessed as R_Left[1,1]. C_i, S_i, C_{ij} and S_{ij} denotes $cos(\theta_i)$, $sin(\theta_i)$, $cos(\theta_i{+}\theta_j)$ and $sin(\theta_i{+}\theta_j)$ respectively.

R_Left[1,1] $= C_4C_6 + S_4S_5S_6$	R_Right[1,1]$=C_1N_x + S_1N_y$
R_Left[1,2] $= -C_4S_6 + S_4S_5C_6$	R_Right[1,1]$=C_1O_x + S_1O_y$
R_Left[1,3] $= S_4C_5$	R_Right[1,3]$=C_1A_x + S_1A_y$
R_Left[1,4] $= 0$	R_Right[1,3]$=C_1P_x + S_1P_y$
R_Left[2,1] $= S_4C_6S_{23} + S_6C_5C_{23} - S_6C_4S_5S_{23}$	R_Right[2,1]$=-S_1Nx + C_1N_y$
R_Left[2,2] $= -S_4S_6S_{23} + C_6C_5C_{23} - C_6C_4S_5S_{23}$	R_Right[2,2]$=-S_1O_x + C_1O_y$
R_Left[2,3] $= -S_5C_2C_3 + S_5S_2S_3 - C_4C_5S_{23}$	R_Right[2,3]$=-S_1A_x + C_1A_y$
R_Left[2,4] $= -d_4C_{23} - C_2d_3$	R_Right[2,4]$=-S_1P_x + C_1P_y$
R_Left[3,1] $= S_4C_6C_{23} + S_6C_5S_{23} + S_6C_4S_5C_{23}$	R_Right[3,1]$=N_z$

R_Left[3,2] $= S_4S_6C_{23} + C_6C_5S_{23} + C_6C_4S_5C_{23}$	R_Right[3,2]=O_z
R_Left[3,3] $= -S_5S_6C_{23} + C_6C_5S_{23} + C_6C_4S_5C_{23}$	R_Right[3,3]=A_z
R_Left[3,4] $= -d_4S_{23} - S_2d_3$	R_Right[3,4]=P_z
R_Left[4,1] =0	R_Right[4,1]=0
R_Left[4,2] =0	R_Right[4,2]=0
R_Left[4,3] =0	R_Right[4,3]=0
R_Left[4,4] =1	R_Right[4,4]=1

A.2 $A_3A_4A_5A_6 = A_2^{-1}A_1^{-1}T_6$

S_Lft[1,1] $= C_4C_6 + S_4S_5S_6$	S_Rgt[1,1] $= C_1N_x + S_1N_y$
S_Lft[1,2] $= -C_4S_6 + S_4S_5C_6$	S_Rgt[1,2] $= C_1O_x + S_1O_y$
S_Lft[1,3] $= S_4C_5$	S_Rgt[1,3] $= C_1A_x + S_1A_y$
S_Lft[1,4] $= 0$	S_Rgt[1,4] $= C_1P_x + S_1P_y$
S_Lft[2,1] $= S_3S_4C_6 + S_6C_{23}$	S_Rgt[2,1] $= -C_2S_1N_x + C_2C_1N_y + S_2N_z$
S_Lft[2,2] $= -S_3S_4S_6 + C_6C_3C_5 - C_6S_3C_4S_5$	S_Rgt[2,2] $= -C_2S_1O_x + C_2C_1O_y + S_2O_z$
S_Lft[2,3] $= -C_3S_5 - S_3C_4C_5$	S_Rgt[2,3] $= -C_2S_1A_x + C_2C_1A_y + S_2A_z$
S_Lft[2,4] $= -C_3d_4 - d_3$	S_Rgt[2,4] $= -C_2S_1P_x + C_2C_1P_y + S_2P_z$
S_Lft[3,1] $= -C_3S_4C_6 + S_6S_3C_5 + S_6C_3C_4S_5$	S_Rgt[3,1] $= S_2S_1N_x - S_2C_1N_y + C_2N_z$
S_Lft[3,2] $= C_3S_4S_6 + C_6S_3C_5 + C_6C_3C_4S_5$	S_Rgt[3,2] $= S_2S_1O_x - S_2C_1O_y + C_2O_z$
S_Lft[3,3] $= -S_3S_5 + C_3C_4C_5$	S_Rgt[3,3] $= S_2S_1A_x - S_2C_1A_y + C_2A_z$
S_Lft[3,4] $= -S_3d_4$	S_Rgt[3,4] $= S_2S_1P_x - S_2C_1P_y + C_2P_z$
S_Lft[4,1] =0	S_Rgt[4,1]=0
S_Lft[4,2] =0	S_Rgt[4,2]=0
S_Lft[4,3] =0	S_Rgt[4,3]=0
S_Lft[4,4] =0	S_Rgt[4,4]=1

A.3 $A_4^{-1}A_3^{-1}A_2^{-1}A_1^{-1}T_6 = A_5A_6$

Q_Lft[1,1] $= N_yS_4C_1S_{23} - N_xS_4S_1S_{23} - S_4N_zC_{23} + N_xC_4C_1 + N_yC_4S_1$

Q_Lft[1,2] $= -S_4O_zC_{23} + O_xC_4C_1 + O_yS_4S_1 + O_yS_4C_1S_{23} - O_xS_4S_1S_{23}$

Q_Lft[1,3] $= A_yC_4S_1 + A_xC_4C_1 - S_4A_zC_{23} - A_xS_4S_1S_{23} + A_yS_4C_1S_{23}$

Q_Lft[1,4] $= P_xC_4C_1 - P_xS_4S_1S_{23} + P_yS_4C_1S_{23} + S_4d_3S_3 + P_yC_4S_1 - S_4P_zC_{23}$

Q_Lft[2,1] $= -S_1N_xC_{23} + C_1N_yC_{23} + N_zS_{23}$

Q_Lft[2,2] $= -S_1O_xC_{23} + C_1O_yC_{23} + O_zS_{23}$

Q_Lft[2,3] $= -S_1A_xC_{23} + C_1A_yC_{23} + A_zS_{23}$

Q_Lft[2,4] $= -S_1P_xC_{23} + C_1P_yC_{23} + P_zS_{23} + d_3C_3$

Q_Lft[3,1] $= N_xS_4C_1 + N_yS_4S_1 + N_xC_4S_1S_{23} - N_yC_4C_1S_{23} + C_4N_zC_{23}$

Q_Lft[3,2] $= O_xS_4C_1 + O_xC_4S_1S_{23} - O_yC_4C_1S_{23} + O_yS_4S_1 + C_4O_zC_{23}$

Q_Lft[3,3] $= C_4A_zC_{23} + A_xS_4C_1 - A_yC_4C_1S_{23} + A_xC_4S_1S_{23} + A_yS_4S_1$

Q_Lft[3,4] $= C_4(P_xS_1C_2S_2 - P_yS_{23} + P_xS_1S_3C_2) + S_4(P_xC_1 + P_yS_1) + C_4P_zC_{23} - C_4d_3S_3$

Q_Lft[4,1] $= 0$

Q_Lft[4,2] $= 0$

Q_Lft[4,3] $= 0$

Q_Lft[4,4] $= 1$

A_5A_6 is computed as (Q_Rgt[] matrix):

$$\begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ C_5S_6 & C_5C_6 & -S_5 & -d_4 \\ S_5S_6 & S_5C_6 & C_6 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Appendix B

DH-Notation for the Human Arm (6 DOF Kinematic Chain)

The following represents the human arm at 6 DOF using DH-Notation. The length used (a_2 and d_4) are arbitrary.

$Joint_i$	θ_i	a_i	α_i	d_i
1	$\theta_1 - 90°$	0	$90°$	0
2	θ_2	50	$0°$	0
3	$\theta_3 + 90°$	0	$90°$	0
4	θ_4	0	$-90°$	60
5	$\theta_5 - 90°$	0	$90°$	0
6	$\theta_6 + 90°$	0	$90°$	0

Table B.1: DH-Notation for Human Arm (6 DOF)

The corresponding matrices constructed from the parameters shown in Table B follows:

$$A_1 = \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} C_2 & -S_2 & 0 & 50C_2 \\ S_2 & C_2 & 0 & 50S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} C_3 & 0 & S_3 & 0 \\ S_3 & 0 & -C_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_4 = \begin{bmatrix} C_4 & 0 & -S_3 & 0 \\ S_4 & 0 & C_3 & 0 \\ 0 & -1 & 0 & 60 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_6 = \begin{bmatrix} C_6 & 0 & S_6 & 0 \\ S_6 & 0 & -C_6 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The axis of rotation represented by each matrix is shown below:

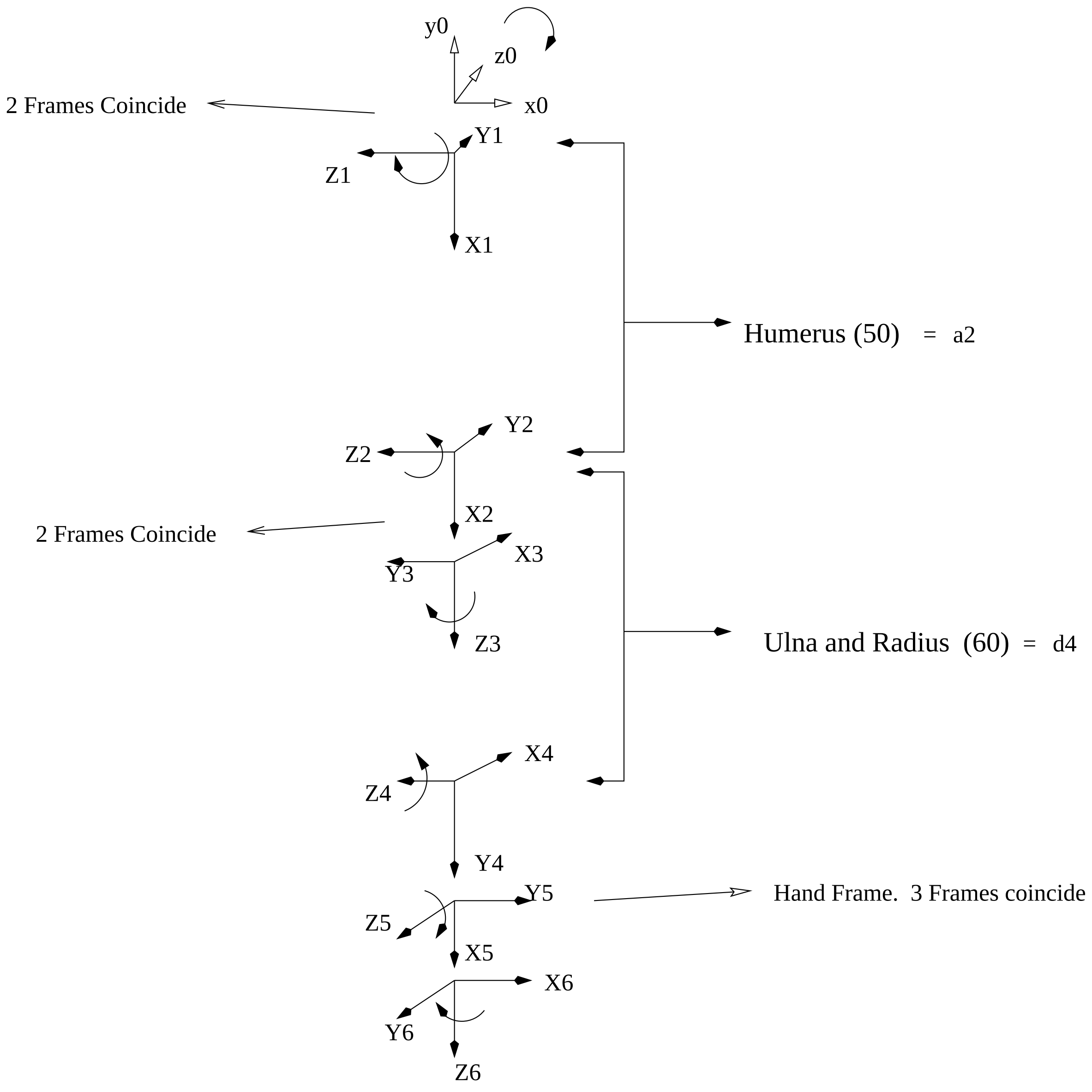


Figure B.1: Coordinate Frames assignment using DH-Notation

Appendix C

Data for Iterative Results

This appendix outlines the numerical values for the graph shown in the results chapter for a 6 DOF kinematic chain.

C.1 Convergence Rate

Trajectory	BFGS	BROYDN	FLETCHER	FRPRM	NEWTON	POWELL
1	8	14	25	10	4	23
2	6	18	19	41	3	8
3	10	13	17	13	3	10
4	9	17	15	13	4	16
5	13	16	13	12	3	8
6	9	16	14	11	3	9
7	9	20	15	11	3	11
8	9	18	14	10	3	12
9	10	19	13	10	3	9
10	12	16	14	10	3	9

Table C.1: Convergence Rate of Minimization Algorithms.

As can be seen from the above NEWTON's method has the overall fastest convergence rate. The average convergence rate between BFGS and POWELL are 9.5 and 11.5 respectively. The POWELL's method provide has a fast convergence rate but has huge number of objective function evaluation. This method is ideal for problems where the derivative of the objective function is difficult or costly to derive and the objective function is cheap to compute.

C.2 Objective Function Evaluations (6 DOF Kinematic Chain)

The following shows the number of function (*Objective*) evaluations before convergence is achieved.

Trajectory	BFGS	BROYDN	FLETCHER	FRPRM
1	73	55	100	267
2	48	64	100	856
3	61	41	100	231
4	47	65	100	247
5	104	67	100	222
6	56	67	100	202
7	54	83	100	210
8	54	74	100	185
9	58	81	100	193
10	84	66	100	178

As can be seen, methods which use only the Objective function have the most number of Objective function evaluations.

Trajectory	NEWTON	POWELL	PRAXIS
1	34	2825	265
2	22	1010	279
3	22	1257	271
4	31	2118	264
5	22	912	262
6	22	1018	267
7	22	1374	262
8	22	1528	256
9	22	1119	263
10	22	1124	252

Table C.2: Number of Function Evaluations Needed

Appendix D

DH-Notation for the Human Arm with the Pelvis (8 DOF)

The following models an articulated structure with 8 DOF. This extends the model presented in Appendix B to include an additional 2 DOF at the pelvis. The lengths used (a_2, a_3, d_2 and d_6) are arbitrary.

$Joint_i$	θ_i	a_i	α_i	d_i
1	$\theta_1 + 90°$	0	$90°$	0
2	$\theta_2 + 180°$	80	$90°$	50
3	θ_3°	50	$-90°$	0
4	θ_4	0	$0°$	0
5	$\theta_5 - 90°$	0	$90°$	0
6	θ_6	0	$-90°$	60
7	$\theta_7 + 90°$	0	$90°$	0
8	$\theta_8 + 90°$	0	$90°$	0

Table D.1: DH-Notation Parameters (8 DOF Kinematic Chain)

The corresponding matrices constructed from the parameters in the above table are shown below (C_i and S_i denotes $cos(\theta_i)$ and $sin(\theta_i)$ respectively):

$$A_1 = \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} C_2 & 0 & S_2 & 80C_2 \\ S_2 & 0 & -C_2 & 80S_2 \\ 0 & 0 & 1 & 50 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} C_3 & 0 & -S_3 & 0 \\ S_3 & 0 & C_3 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_4 = \begin{bmatrix} C_4 & -S_4 & 0 & 50C_4 \\ S_4 & C_4 & 0 & 50S_4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_6 = \begin{bmatrix} C_6 & 0 & -S_6 & 0 \\ S_6 & 0 & C_6 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_7 = \begin{bmatrix} C_7 & 0 & S_7 & 0 \\ S_7 & 0 & C_7 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_8 = \begin{bmatrix} C_8 & 0 & S_8 & 0 \\ S_8 & 0 & -C_8 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The axis of rotation represented by each matrix is shown below:

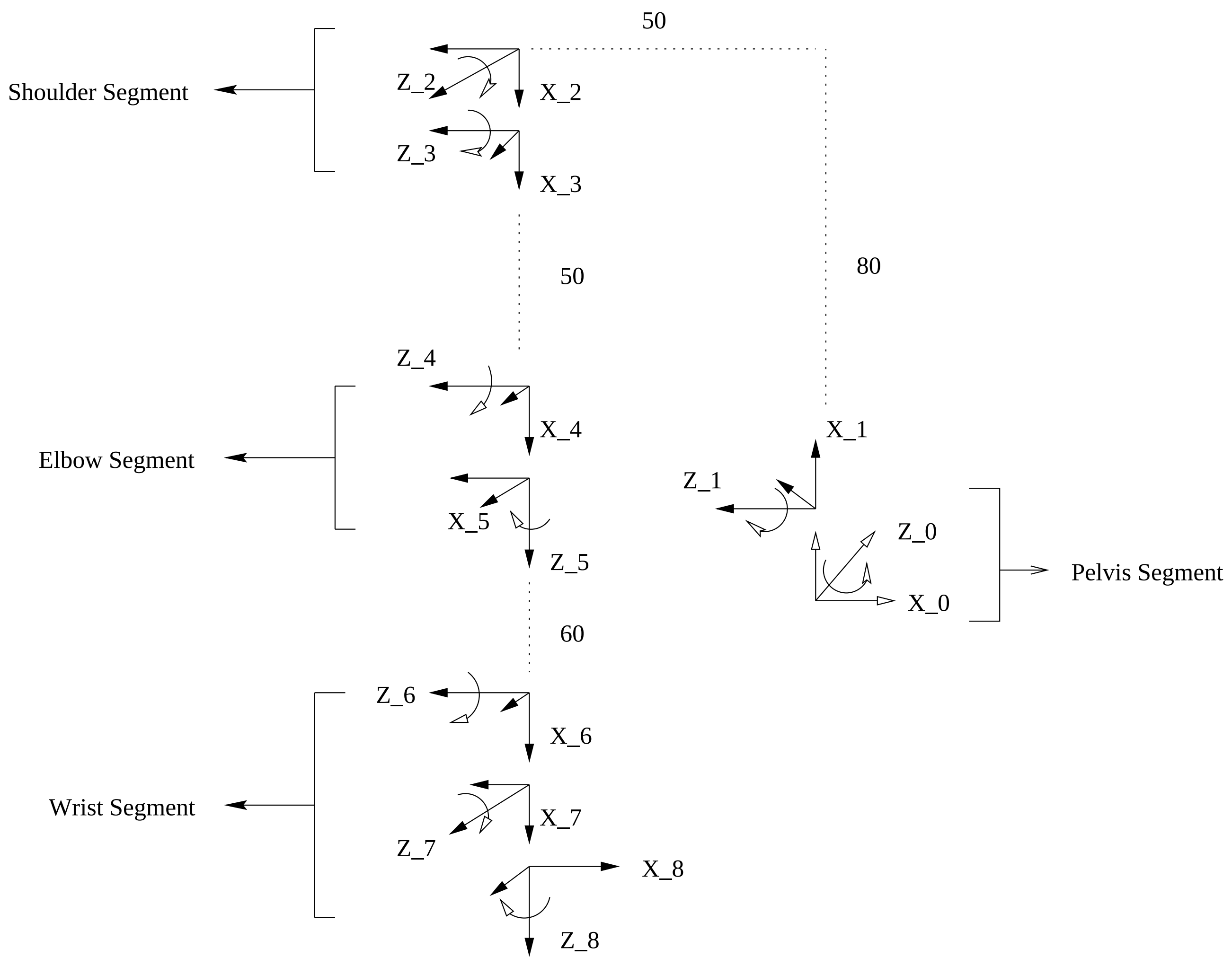


Figure D.1: Coordinate Frames assignment using DH-Notation

Appendix E

DH-Notation for a 3 DOF Kinematic Chain

$Joint_i$	θ_i	a_i	α_i	d_i
1	$\theta_1 + 90°$	10	$0°$	0
2	θ_2	10	$0°$	0
3	θ_3	10	$0°$	0

Table E.1: DH-Notation for 3 DOF Kinematic Chain

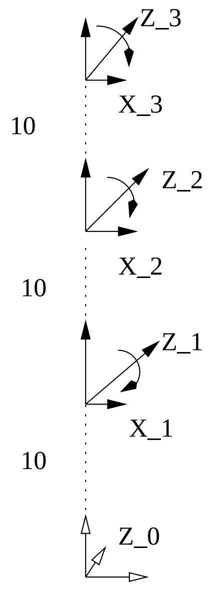


Figure E.1: Coordinate Frames for 3 DOF Kinematic Chain

The corresponding transformation matrices is as follows (C_i and S_i denotes $sin(\theta_i)$ and $cos(\theta_i)$ respectively):

$$A_1 = \begin{bmatrix} C_1 & -S_1 & 0 & 10C_1 \\ S_1 & C_1 & 0 & 10S_1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} C_2 & -S_2 & 0 & 10C_2 \\ S_2 & C_2 & 0 & 10S_2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} C_3 & -S_3 & 0 & 10C_3 \\ S_3 & C_3 & 0 & 10S_3 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Appendix F

Change in Joint Angles (6 DOF Kinematic Chain)

The graphs below show the changes in joint angles generated by the given minimization or root finding algorithms. This graphs show the different set of solutions generated by different minimization algorithms. 1_A to 6_A correspond to θ_1 to θ_6 respectively.

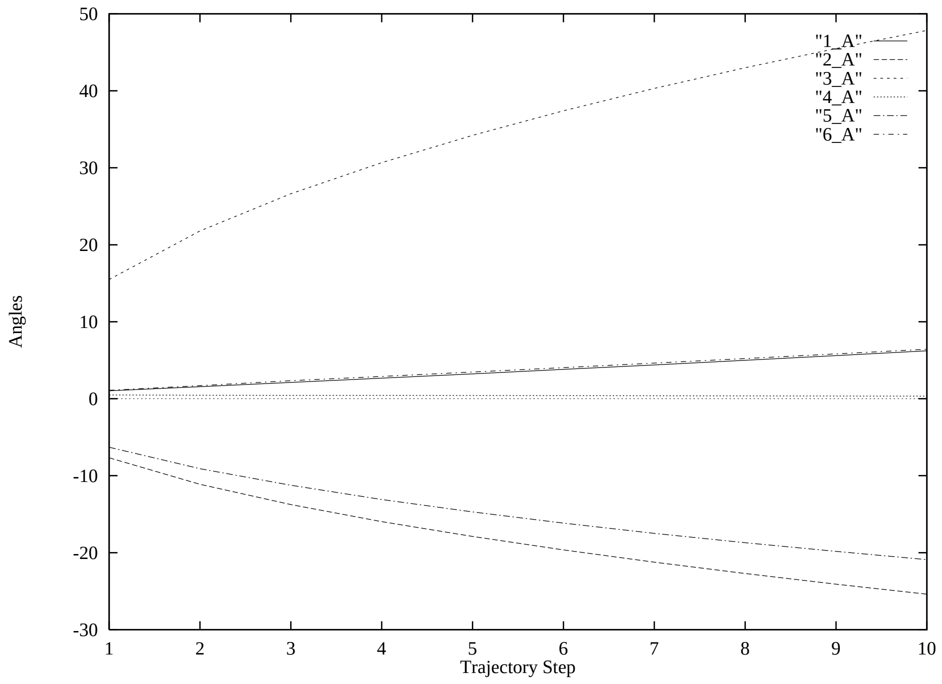


Figure F.1: Changes in Joint angles using Broyden's Method

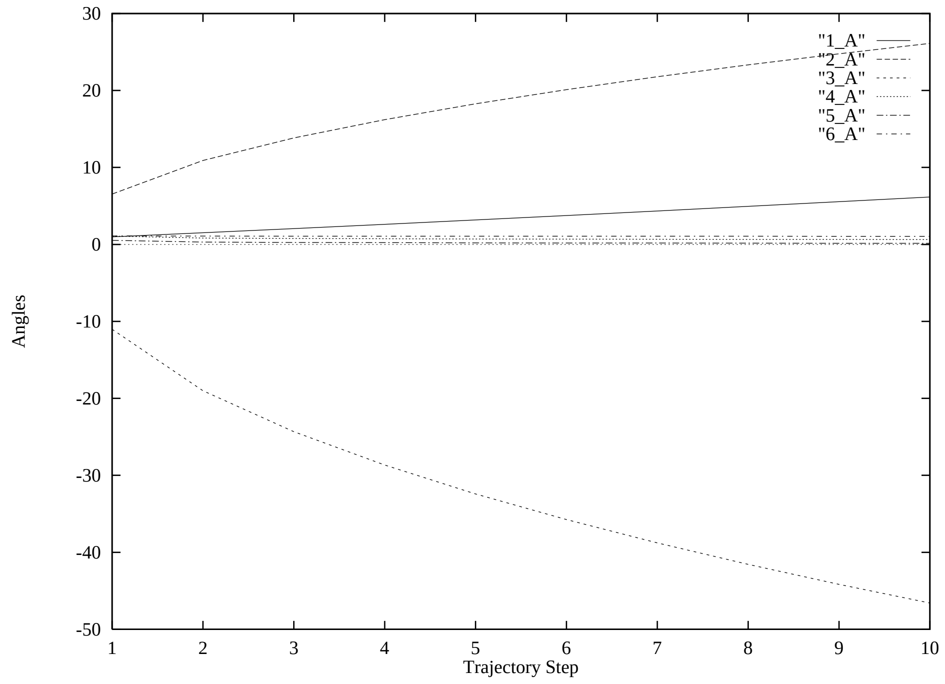


Figure F.2: Changes in Joint angles using FRPRM's Method

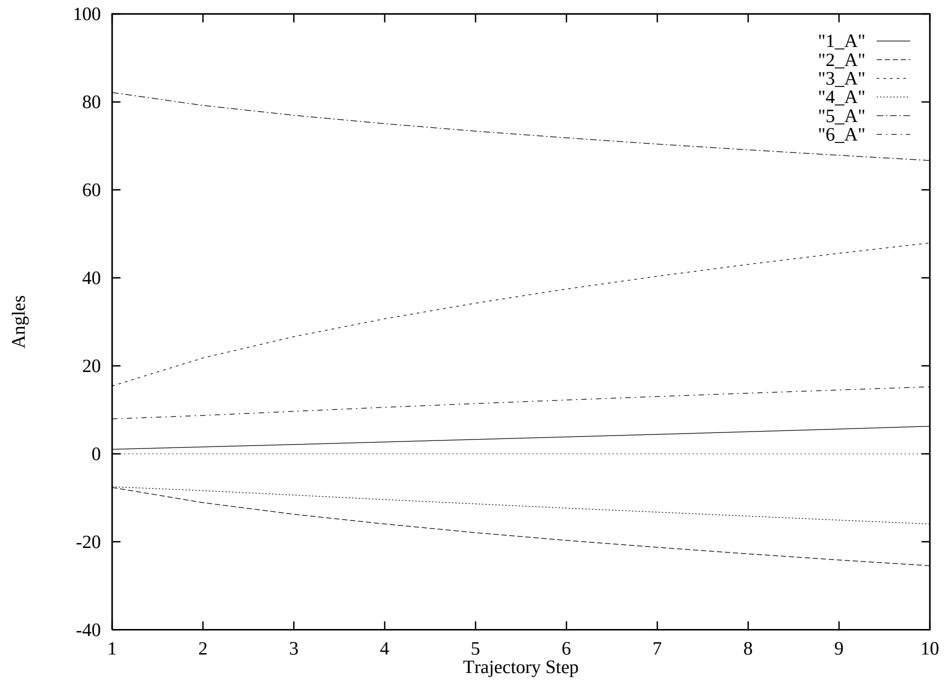


Figure F.3: Changes in Joint angles using POWELL's Method

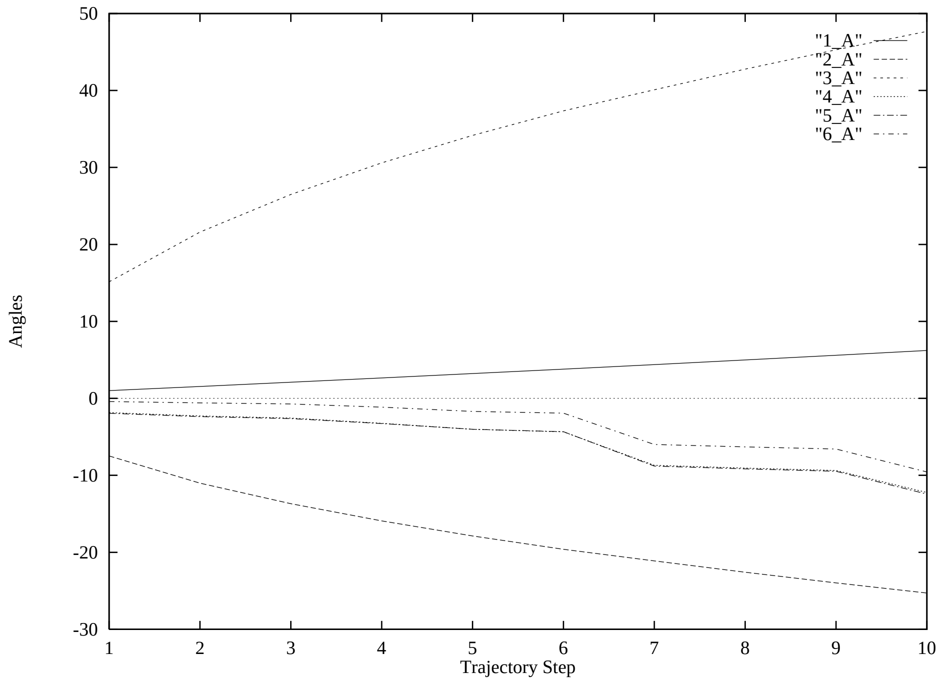


Figure F.4: Changes in Joint angles using FLETCHER's Method

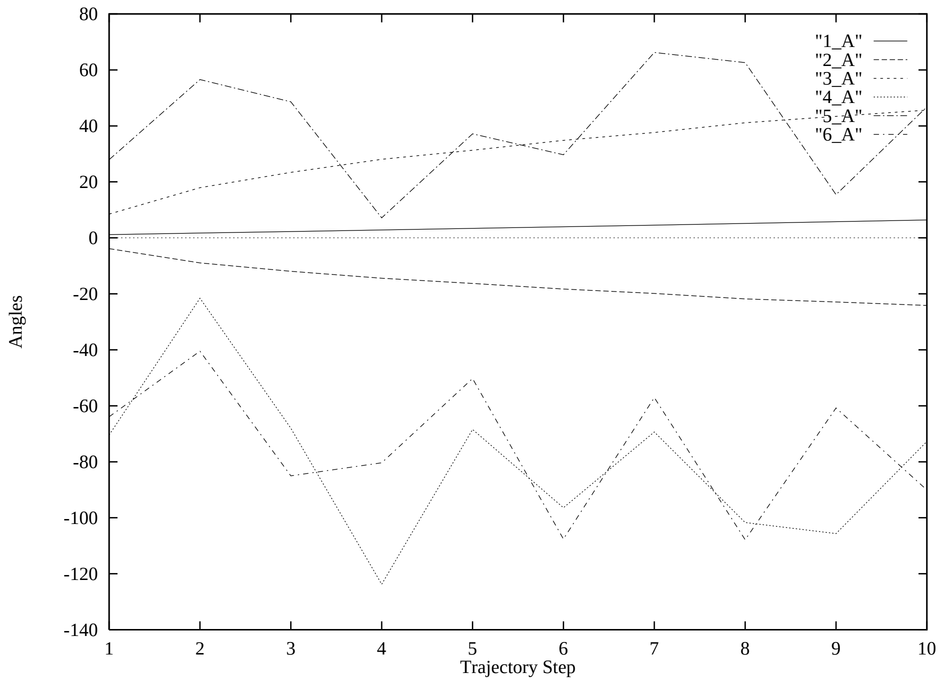


Figure F.5: Changes in Joint angles using PRAXIS's Method

Appendix G

Trajectory Graphs

G.1 FRPRM with Large Residual Error

The following figure shows the required and calculated trajectory using FRPRM's method when the residual error between current and required end-effector frame is at 200. D_{EE} denotes the required trajectory and S_{EE} denotes the calculated trajectory. This figure shows that the FRPRM method does not produce exact convergence when the residual error is large.

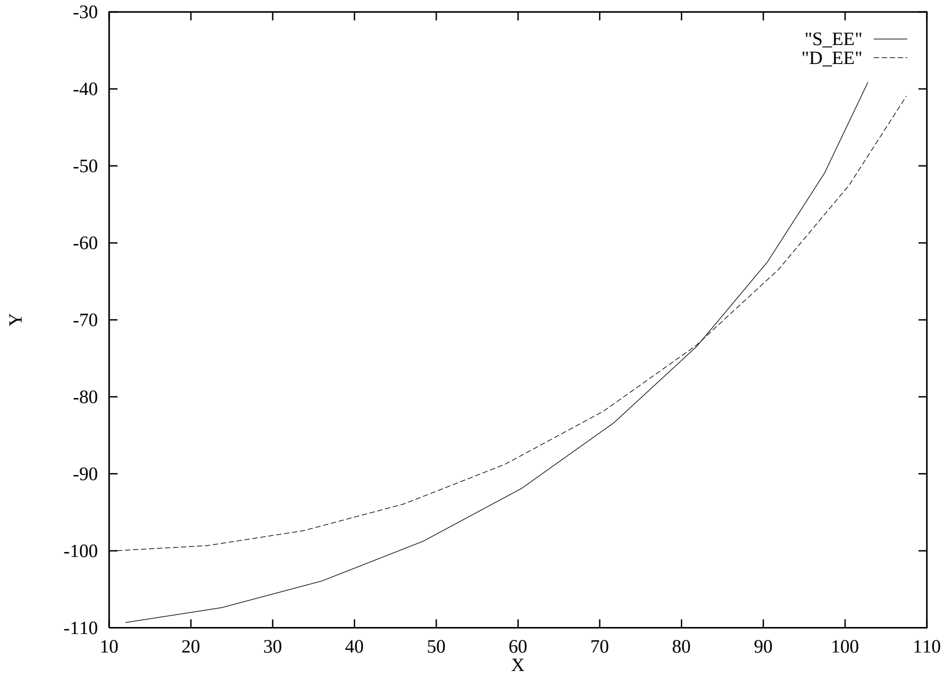


Figure G.1: Error between desired and calculated path

G.2 Trajectory Path Using FLETCHER's Method (6 DOF Kinematic Chain)

The following shows the Trajectory path taken as solved by FRPRM's method. Note the slight error between the step 7 to 9. Recalling that S_{EE} and D_{EE} denotes the calculated and the required end-effector frame respectively.

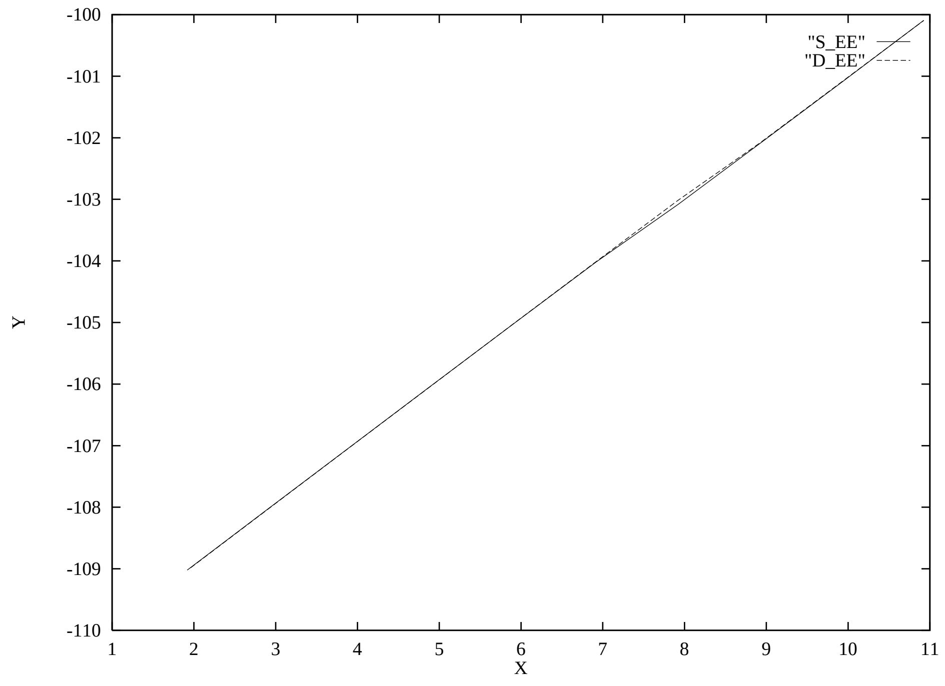


Figure G.2: Error between desired and calculated path