MANIPULATOR INVERSE KINEMATICS BASED ON RECURSIVE LEAST SQUARES ESTIMATION

Derek Charles Glenn Hutchinson

B. A. Sc. University of British Columbia

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

DEPARTMENT OF ELECTRICAL ENGINEERING

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

December 1988

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of ELECTRICAL ENGINEERING.

The University of British Columbia
Vancouver, Canada

Date 1989 01 13

Abstract

The inverse kinematics problem for six degree of freedom robots having a separable structure with the wrist equivalent to a spherical joint is considered and an iterative solution based on estimating the inverse Jacobian by recursive least squares estimation is proposed. This solution is found to have properties similar to Wampler's Damped Least Squares method and provides a stable result when the manipulator is in singular regions. Furthermore, the solution is more computationally efficient than Wampler's method; however, its best performance is obtained when the distances between the current end effector pose and the target pose are small. No knowledge of the manipulator's geometry is required provided that the end effector and joint position data are obtained from sensor information. This permits the algorithm to be readily transferable among manipulators and circumvents detailed analysis of the manipulator's structure.

Table of Contents

Abstract ii

List of Tables vi

List of Figures viii

Acknowledgement ix

1 Introduction 1

1.1 Applications of Inverse Kinematics 2

1.2 Proposed Application For This Research 2

1.3 Overview of Thesis 3

2 Kinematics 4

2.1 Manipulator Spaces 5

2.1.1 Joint Space 5

2.1.2 Work Space 5

2.1.3 Mappings Between the Joint and Work Spaces 8

2.2 The Effects of Spatial Properties on the Inverse Kinematics Solution . . . 8

2.2.1 High Manipulability Region 9

2.2.2 Singular Region 11

2.2.3 Out of Reach Region 13

2.3 Calculation Methods 13

2.3.1 Direct Kinematics 13

2.3.2 Inverse Kinematics 14
2.4 Desired Properties of an Inverse Kinematics Solution 20

3 Inverse Jacobian Estimation **21**
3.1 Inverse Jacobian Model Selection 21
3.2 Estimation Method Selected 23
3.3 Estimated Inverse Jacobian Properties 26
3.3.1 Comparision of Least Squares Methods 26
3.3.2 Estimation Using Kinematic Position Data - An Approximate Model 27
3.3.3 Numerical Measures of Estimator Properties 32

4 Proposed Inverse Kinematics Algorithm **41**
4.1 Algorithm Description 41
4.2 Algorithm Characteristics 43
4.2.1 Overview 43
4.2.2 Investigation of Algorithm Parameters 44
4.2.3 Investigation of Pose Dependence 49
4.2.4 Investigation of Manipulator Dependence 55
4.3 Comparision To Other Researcher's Work 58
4.3.1 Computations per Iteration 59
4.3.2 Functional Joint Control 60
4.3.3 Damped Least Squares with Variable Damping 61

5 Conclusions **70**
5.1 Summary 70
5.2 Conclusions 71
5.3 Recommendations 72

References 74

A Denavit-Hartenberg Parameters 81

A.1 Two Link Manipulator 81

A.2 Puma Manipulator 82

A.3 Kodiak Manipulator 83

B Direction Cosine - Euler Parameter Conversions 84

B.1 `noa_EulerParam()` 85

B.2 `EulerParam_noa()` 88

List of Tables

3.1 Comparision of Recursive Least Squares Computations 25

3.2 Comparision of Least Squares Methods 27

4.3 Solution time as a function of the forgetting factor 47

4.4 Algorithm Average Solution Times in High Manipulability Region 50

4.5 Algorithm Average Solution Times in Singular Region 52

4.6 Results from Puma Degradation Experiments 57

4.7 Comparison of Algorithm Performance Between the Puma and Kodiak . 58

4.8 Comparison of Calculations per Iteration Between the Proposed Algorithm and Wampler's 59

4.9 Comparision of RLSE with FJC - Singular Region 60

4.10 Comparision of RLSE with FJC - High Manipulability Region 60

4.11 Comparision of RLSE with Damped Least Squares with Variable Damping 61

A.2 Two Link Denavit-Hartenberg Parameters 81

A.3 Puma Denavit-Hartenberg Parameters 82

A.4 Kodiak Denavit-Hartenberg Parameters 83

List of Figures

2.1 Orientation of End Effector with $\vec{n}$, $\vec{o}$, $\vec{a}$ vectors 7

3.2 One Link Manipulator 28

3.3 Condition Number over range of θ_2 for J^{-1}, DLS, and RLSE 34

3.4 Tuneable damping shown by condition number for range of θ_2 at 3 variances of $\Delta\theta_2$ 36

3.5 Tuneable Damping effect on J_{11}^{+} 37

3.6 Tuneable Damping effect on J_{12}^{+} 38

3.7 Tuneable Damping effect on J_{21}^{+} 39

3.8 Tuneable Damping effect on J_{22}^{+} 40

4.9 Schematic of the Proposed Algorithm 42

4.10 Joint Change Limiting Algorithm 48

4.11 Puma in High Manipulability Pose 62

4.12 Error reduction - Puma in High Manipulability Pose 63

4.13 Joint Angles - Puma in High Manipulability Pose 64

4.14 Puma in Singular Pose 65

4.15 Error Reduction - Puma in Singular Pose 66

4.16 Joint Angles - Puma in Singular Pose 67

4.17 Position Error For an Out of Reach Target 68

4.18 Comparision of Kodiak and Puma Poses Used 69

A.2 Coordinate Frames of a Two Link Manipulator 81

A.3 Coordinate Frames of a Puma Manipulator 82

A.4 Coordinate Frames of a Kodiak Manipulator 83

Acknowledgement

I would like to thank my supervisor Dr. Peter D. Lawrence for his support and input during this project. Also, the efforts of Dr. David L. Pulfrey were appreciated when the project initially involved APRjr. I am indebted to Joe Poon whose graphics software, useful utilities, and programming knowledge helped me considerably throughout my work. Other people that have given me support, suggestions, and advice include my colleagues David Gagne, Rod Barman, Mike Bolotski, Steve Chan, Grant Finnighan, Jim Reimer, and Richard Smith. The departmental techinicans, Tony Leugner, and Derek Boshier, should also be recoginized for their attempts to educate me in the more practical matters of electrical engineering. Thanks to the Friends of Mickey Mouse: Willie Fajber, James Fajber, Chris Backhouse, Fraser Duncan, Dave Macquistan, Marcel Lefrançois and others for alot of fun and interesting discussions. Special thanks to Abigail Turner for her encouragement and good humor. Finally, this thesis would not be possible without the generous suppport of my parents.

This research was funded by Natural Sciences and Engineering Research Council of Canada grant A1674.

Chapter 1

Introduction

In the past robot control has been relatively simplistic. Robots were directed through a desired set of motions which were recorded or pre-programmed using a path planning approach and then played back on demand. Recently, robotics research has emphasized robots that can sense their environment and adapt to it. Such robots must be able to control their end effector in terms of environmental constraints such as picking parts off a moving conveyor belt, or accurately welding along a straight line. Most tasks are easily described in terms of Cartesian coordinates. However, the robot's motion is most naturally accomplished in terms of servoing its joints. To obtain accurate control, knowledge of the robot's kinematics and dynamics must be utilized to plan the robot's motion in its environment so that various goals can be achieved via joint motions. Although the knowledge of robot dynamics is necessary, the knowledge of kinematics is fundamentally more important. Kinematics is the study of motion without concern for the forces producing the motion and, with respect to robots, is concerned with two problems, direct and inverse kinematics. The direct kinematics problem, how motion of the robot's joints will cause motion of the robot's end effector, is readily described from the robot's geometry and joint motions. The inverse kinematics problem, which is of greater interest and accordingly greater complexity, is to determine what joint positions are required to obtain a desired end effector position. If a robot is to be controlled in terms of information gathered from its environment, the inverse kinematics problem must be solved.

1.1 Applications of Inverse Kinematics

The inverse kinematics problem is not just limited to robots. It can also be applied to the field of teleoperation. Currently, a typical teleoperator consists of a pair of manipulators; a slave manipulator which performs the manipulation, and a master manipulator which allows the operator to control the slave. The master, which is a scale model of the slave, effects control by simply reflecting its position to the slave manipulator. This system has a few drawbacks as the master is specific to a slave manipulator and needs a fairly large volume to work in. Furthermore, the master is limited to strict position control and can be awkward to use because its links interfere with the operator. The need for the master to be a replica of the slave it controls would be eliminated if the slave's inverse kinematics were known. Then, a position could be specified by an arbitrary master and the slave's position could be determined using its inverse kinematics. A standard master for an variety of manipulators could be created that could have several control modes, such as position control, rate control or even joint control. These control modes could be selected and used as the operator saw fit. This approach to teleoperation has been pursued by several researchers, most notably A. Bejczy at Jet Propulsion Labs for use in space-teleoperators [Bejczy83]. Inverse kinematics is also important in modeling human motion for ergonomic reasons [Korein85].

1.2 Proposed Application For This Research

The University of British Columbia's Robotics and Teleoperation Laboratory is currently interested in applying robotics technology to heavy equipment used in the British Columbia forest industry. The heavy equipment in use today consists of fairly simple manipulators which have a maximum of five degrees of freedom. The inverse kinematics

for this equipment can be solved in closed form. It is anticipated however, that the success of having more sophisticated control methods for heavy equipment will lead to the design and use of more complex manipulators. Once manipulators reach the 6 degree of freedom or greater stage, closed form solutions are, in general, no longer obtainable. The inverse kinematics problem is further complicated by the possibility of the manipulator's structure being altered in the field, for example, by tool changes. This thesis presents an iterative inverse kinematics solution which requires no geometric information about the manipulator it is solving provided that position data on the manipulator's joints and end effector is available.

1.3 Overview of Thesis

This thesis has been arranged in the following manner. Chapter One provides the reader with an introduction to the inverse kinematics problem and its importance. A more detailed look at robot kinematics is presented in Chapter Two providing the reader with some background information and the approaches pursued by other researchers. Chapter Three addresses the problems relevant to using estimation with inverse kinematics and demonstrates that the estimation approach is equivalent to an established inverse kinematics solution. The proposed solution to the inverse kinematics problem, which incorporates estimation, will be presented and evaluated in Chapter Four. Chapter Five contains the conclusions and recommendations from this work.

Chapter 2

Kinematics

This thesis focusses on manipulators which form an open serial chain and consist of 6 rigid links connected by revolute or prismatic joints. Of all the possible manipulator structures that could be examined, the primary interest will be focussed on manipulators that can be partitioned into major and minor linkages as suggested by Milenkovic [Milenk83]. For such manipulators the major linkage, the first 3 links, positions the end effector whereas the minor linkage orients it. The minor linkage is assumed to be formed from some combination of three revolute joints whose axes of rotation intersect at a point. Kinematics of open serial chain manipulators with less than 6 links can be considered as a subproblem of the 6 degree of freedom case so they will not be considered here. Also, the kinematics of redundant manipulators, manipulators having more than 6 links or joints that are redundantly placed, such as a planar manipulator with more than two links, will not be examined.

In this chapter the notion that the manipulator has two spaces for representing end effector position information will be presented. Both of these spaces will be discussed, their importance elucidated, and the mappings between them defined. Furthermore, the properties of the two spaces and their effect on the inverse kinematics solutions will be shown followed by a description of the calculation methods used for transforming from one space to the other. Finally, the goals that an inverse kinematics solution should satisfy are discussed.

2.1 Manipulator Spaces

The location in space of a manipulator's end effector can be expressed in the coordinates of two vector spaces: the joint space and the work space. These two vector spaces will be described and the relationship between them established.

2.1.1 Joint Space

The joint space is defined by considering each manipulator joint as contributing one axis to a vector space with dimension equal to the total number of joints. Joint constraints specify the range on each axis and the origin of the space is arbitrarily set at some convenient manipulator configuration. The units describing a coordinate's value are either in radians or metres depending on whether the joint is revolute or prismatic. Each coordinate value is derived from joint sensor information and manipulator positioning commands are expressed in the joint space. In conclusion, the joint space is the gateway to controlling the manipulator, all commands to the actuators must be made through it and all limits on the manipulator's motion are made in it.

2.1.2 Work Space

Locating objects and specifing paths within a manipulator's environment is most easily accomplished in terms of work space coordinates. The end effector's work space position can be determined by a vision system [Ishii87] or from the the joint position data given the joint space to work space mapping. However, the boundaries of a manipulator's work space are not easily defined since they are complex functions of the joint space constraints and the manipulator's structure [Gupta86b, Vijayk86]. Unlike joint space coordinates, work space coordinates are expressed in global terms relative to a fixed reference, usually the base of the manipulator. A work space vector consists of two

components, the position component, and the orientation component both of which can be expressed in a variety of ways. For example, position can be expressed in one of three coordinate systems: Cartesian (x, y, z), Cylindrical (r, θ, z) or Spherical (r, θ, ϕ). Conversions between these position coordinate systems are straightforward. There are many ways to describe a bodies' orientation in space. A few of the more commonly used methods will be described here. The direction cosine matrix is by far the most common orientation method and is used extensively in robotics [Paul81] and computer graphics [Foley82]. The method of direction cosines uses two orthogonal right handed coordinate systems and defines the projection of one coordinate system onto the other by forming the dot product between the two coordinate systems. Suppose ($\vec{a_1}$, $\vec{a_2}$, $\vec{a_3}$) and ($\vec{b_1}$, $\vec{b_2}$, $\vec{b_3}$) are orthonormal basis vectors describing coordinate system A and B respectively. Then the 3 $\times$ 3 direction cosine matrix C_a^b defining B with respect to A is defined by Equation 2.1.

$$C_{ij} = a_i \cdot b_j \quad (i,j = 1,2,3) \tag{2.1}$$

In robotics, the orthonormal basis vectors describing the end effector's coordinate frame are called normal $\vec{n}$, offset $\vec{o}$, and approach $\vec{a}$ and are oriented on the hand as shown in figure 2.1.

This approach is not minimal as only three independent parameters are needed to represent orientation whereas 9 are used. Using three variables a total of 24 different representations of orientation can be defined although each of these orientation methods suffers from a singularity, a point where the orientation representation is no longer defined [Kane83]. Two common representations in this category are Euler angles defined by the sequence of rotations Rot(z, ψ)Rot(y,ϕ)Rot(z,θ) and Roll-Pitch-Yaw angles are defined by Rot(x, ψ)Rot(y, ϕ) Rot(z, θ) [Paul81]. Euler angles have a singular point at

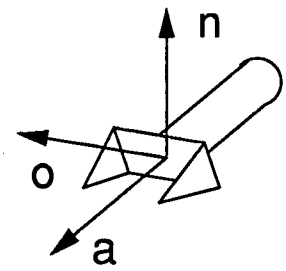

Figure 2.1: Orientation of End Effector with $\vec{n}$, $\vec{o}$, $\vec{a}$ vectors

$\theta = 0$ and Roll-Pitch-Yaw angles have a singular point at $\theta = \frac{\pi}{2}$. An orientation method which is both non-singular and easily manipulated is Euler parameters [Spring86]. Euler parameters consist of a 4-dimensional vector $\vec{q}$ defined in Equation 2.2 where θ is a rotation about an axis which is fixed in both coordinate frames and defined by the unit vector $\vec{n}$.

$$q = \{\cos\frac{\theta}{2},\ \sin\frac{\theta}{2}\vec{n}\} \tag{2.2}$$

The definition of Euler parameters is a consequence of Euler's Theorem of rigid-body dynamics [Euler] : "the attitude of a body after having undergone any sequence of rotations is equivalent to a single rotation of that body through an angle θ about an axis $\vec{n}$" [Spring86]. Euler parameters are not unique as $+q$ and $-q$ represent the same rotation. Also, Euler parameters can be considered as a unit quaternion and can be manipulated using quaternion algebra. Quaternion algebra is a generalization of vector algebra and allows rotations to be easily manipulated. In this thesis, Euler parameters will be obtained by extraction from the direction cosine matrix although direct use of quaternion

algebra may be more computationally efficient. Appendix B shows the transformations required to obtain Euler parameters from a direction cosine matrix and *vice versa.*

2.1.3 Mappings Between the Joint and Work Spaces

Since end effector path descriptions are best described in work space coordinates but control of the manipulator must be performed in terms of joint space coordinates, mappings between the two spaces must be established. The joint space to work space mapping, or direct kinematics is unique, is easily specified in closed form and is readily determined by using any of several calculation methods. Conversely, the mapping from the work space to the joint space, the inverse kinematics problem, is multivalued and can not usually be expressed in a set of closed form equations.

2.2 The Effects of Spatial Properties on the Inverse Kinematics Solution

A variety of factors complicate finding a manipulator's inverse kinematics solution. Manipulators typically have joint limits which result from physical constraints such as restricted actuator range, or links interfering with one another. Joint limits often make the manipulator's work space discontinuous. Although targets are expressed in work space coordinates, they are generally not specified with the manipulator's work space constraints in mind. Sometimes the target's coordinates will be beyond the reach of the manipulator or at some position that is difficult for the manipulator to attain. These problematic regions occur not only at the edge of the manipulator's reach but also at other locations, singular points, which are well inside the manipulator's work space. Fortunately these regions comprise only a small part of the manipulator's total work space volume; but, an inverse kinematics algorithm must be able to function adequately in these regions.

A number of researchers [Vijayk86, Gupta86b, Kumar86] have investigated manipulator work space properties. Their work has resulted in methods for describing a manipulator's work space, defining manipulator structures optimized for dexterity and identifying singular positions.

2.2.1 High Manipulability Region

The manipulator's work space can be divided into a high manipulability region and a singular region. The high manipulability region makes up the largest portion of the work space, but the boundaries of this region are arbitrary and more readily defined in terms of where the singular region is not. Distinguishing between the two regions is accomplished by determining the manipulator's dexterity or manipulability. The basis for these measures is the manipulator's Jacobian, which is a linear mapping between the joint space and the work space as described in Equation 2.3. The Jacobian is a matrix with row dimension equal to the work space dimension ($\dim m$) and the column dimension equal to the joint space dimension ($\dim n$). It is obtained by differentiating the joint space variables with respect to the work space variables as shown in Equation 2.4.

$$\dot{\mathbf{x}} = J(q)\dot{\mathbf{q}} \tag{2.3}$$

$$J_{ij} = \frac{\partial x_i}{\partial q_j} \quad \begin{matrix} i = 1\ ,\ m \\ j = 1\ ,\ n \end{matrix} \tag{2.4}$$

where $\mathbf{x}$ = work space vector

for example, [x, y, z, ϕ, θ, ψ]^T

$\mathbf{q}$ = joint space vector

for example, $[\theta_1, \theta_2, \ldots, \theta_6]^T$

The manipulability measure's function is to identify when the Jacobian matrix is near singular, that is, when two or more columns or rows are nearly linearly dependent.

The measures reported in the literature can be divided into one of two categories, those based on the Jacobian's determinant [Paul83, Yoshik85] and those based on the Jacobian's condition number [Wample85, Klein87]. A determinant method suggested by Yoshikawa [Yoshik85] is described in Equation 2.5.

$$\sqrt{\det J(q)J^T(q)} \tag{2.5}$$

With this measure the manipulability of redundant manipulators can be obtained since the matrix $J(q)J^T(q)$ is by definition positive definite and therefore has a positive determinant. The premise behind this measure is that since the Jacobian's determinant is zero at a singularity, it will be small near a singularity. Paul [Paul83] has suggested that a singular region exists when the Jacobian's determinant is less than 0.5. In contrast to the determinant method, the condition number is obtained by taking the singular value decomposition [Klema80] of the Jacobian. As shown in Equation 2.6, the Jacobian is decomposed into 3 matrices, U, Σ and V where $U \in \Re^{m \times m}$ and $V \in \Re^{n \times n}$ are orthogonal matrices and $\Sigma \in \Re^{m \times n}$ is diagonal.

$$J = U\Sigma V^T \tag{2.6}$$

The diagonal matrix Σ contains the singular values. These values, σ_i's, are in decreasing order with the largest value at element Σ_{11} and smallest at Σ_{mm}, for example $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_m \geq 0$. As the Jacobian becomes increasingly singular, the smallest singular value σ_m tends to zero. The condition number which is the ratio of the largest singular value

over the smallest, $\frac{\sigma_1}{\sigma_m}$, can also be thought of as $\|J\|\|J^{-1}\|$. From the latter result, it is seen that the condition number's smallest value is 1 and the upper limit is ∞ since σ_m can equal zero. Thus a condition number near one indicates that the manipulator is in a high manipulability region. The singular value decomposition is fairly time consuming to compute and is not practical for on-line use. Of the two measures, it should be noted that a small determinant does not necessarily mean that the Jacobian is near singular [Golub83].

Volumetric and Dextrus analysis is concerned with determining a manipulator's work space volume and how easily the manipulator can move in its work space. When analysing a manipulator's work space in order to evaluate its structure and propose improvements, the manipulability measure serves as a cost function. Work space analysis has suggested a number of interesting results, the hand size should be as small as possible [Gupta86b], the last three joints should be revolute with axes which are mutually orthogonal and intersect at a point [Vijayk86], the Denavit-Hartenberg parameters should contain a minimal number of non-zero offsets with the twist angles consisting of either 0, $+\frac{\pi}{2}$, or $-\frac{\pi}{2}$, and the upperarm and forearm lengths for a two link manipulator with revolute joints should be equal [Yoshik85]. These findings are supported by evolutional history of commercially available manipulators. Work space analysis also identifies the difficult regions such as singular points [Uchiya79, Litvin86, Lai86], but it is performed off-line. In an inverse kinematics algorithm, It is often desirable to measure manipulability on-line so that the nearness of singularities can be detected and steps can be taken to avoid or anticipate their effects.

2.2.2 Singular Region

Singularities are an undesirable artifact of a manipulator's structure. At a singularity, one or more degrees of freedom are lost and the manipulator can no longer move in

certain directions. As mentioned in the previous section, the Jacobian becomes singular or rank deficient generating a mapping of finite work space changes onto infinite joint space changes. Singularities occur in a number of ways, the axes of two joints can become aligned making one joint redundant, for example the first and last wrist joints, or a joint can reach the limit of its motion, such as a fully extended elbow. Since for an n degree of freedom arm there are 2^n potential inverse kinematics solutions [Litvin86], there then can be n possible singularities as a singularity forms a boundary between alternate solutions. For most commercially used manipulators, there are only three singular configurations. For example, the Puma's singularities are, when the wrist's origin is over the shoulder such that the axes of joint 2 and joint 6 are parallel, when the elbow, joint 3, is fully extended, and when the axes of joint 4 and joint 6 are parallel.

A number of approaches have been taken to address the singularity problem. Uchiyama [Uchiya79] suggests detection and avoidance of singularities. He claims this approach is only viable when a small portion of the work space is used and alternative areas can be found. A side effect of this approach is that it adds holes to the work space which need to be recognized and avoided by the path planner thereby coupling the inverse kinematics and path planning problems. Vijaykumar [Vijayk86] shows that the majority of singularities can be eliminated by judicious choice of joint limits. Another approach is to use redundancy to avoid the singularies or place them outside the joint limits such as Trevelyan's ET wrist [Trevel86] or Yoshikawa's 4 joint wrist [Yoshik85]. Carefully avoiding the problem by introducing redundancy is not a viable approach when the inverse kinematics must be found for an existing manipulator. Waldron's selection of joint limits does not address the "wrist over shoulder" singularity therefore this approach does not provide a complete solution. What must be accepted in singular regions is that highly accurate solutions are not obtainable due to the ill-conditioned Jacobian amplifying numerical errors. A useable solution is desirable in such regions to ensure that an inverse

kinematics solution is robust and stable.

2.2.3 Out of Reach Region

The out of reach region consists of all targets that either violate a joint limit or cannot be reached without moving the manipulator's base and must be considered since a human operator could potentially direct a teleoperator there. Although work space analysis can describe a manipulator's boundaries, this information is not readily stored for on-line use. As with singular regions, out of reach solutions need to be handled sensibly by the inverse kinematics solution since the algorithm, while attempting to drive the error below a given tolerance, can only achieve a minimum. The algorithm must recognize that the target is out of reach and act accordingly. The problem is further compounded by the minimum error solution often being at a singular position. In summary, the inverse kinematics algorithm should be robust for out of reach targets and produce a minimum error solution.

2.3 Calculation Methods

Some of the methods used to solve the direct and inverse kinematics problems will be presented, as well as the notation and mathematical tools used.

2.3.1 Direct Kinematics

Although there are many *ad hoc* methods for describing the direct kinematics, the method due to Denavit and Hartenberg [Denavi55] has the most widespread use. Denavit-Hartenberg parameters describe the relationship between the coordinate frames of two joints in a serial, rigid multi-link, open chain manipulator. This relationship between two joint coordinate frames i and $i-1$ is defined by four parameters: the offset a_i, the

displacement d_i, the twist α_i and the rotation θ_i. See Paul [Paul81] for details on how these joint coordinate frames are defined. For revolute joints the variable is θ_i and for prismatic joints it is d_i. This sequence of transformations is said to form an A matix. Such an A matrix for revolute joints is shown in Equation 2.7.

$$\begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

Multiplying the A matrices together will produce a matrix which describes the end effector position and orientation with respect to the base coordinates as shown in Equation 2.8

$$T^6 = A_1A_2A_3A_4A_5A_6 \tag{2.8}$$

Other systematic methods exist for specifying the direct kinematics such as Gupta's Zero Reference Position Method [Gupta86a], a variant of the Denavit-Hartenberg approach.

2.3.2 Inverse Kinematics

Closed Form Solutions

Finding the inverse kinematics solution is not systematic as was the direct kinematics. Although closed form solutions specify all possible solutions for the inverse kinematics problem, the solutions become ill conditioned near singularities. Due to the multiple solutions and the difficulties near singularities, a number of different cases must be carefully sifted through. Also, care must be taken to detect targets that are outside the work space. Closed form solutions are computationally compact, but they do not consider the target manipulator deviating from its mathematical model. Errors in the model due to

machining tolerances, wear, link bending and other geometric and non-geometric irregularities will be reflected as a position error. As a result, closed form inverse kinematics will produce a solution that is repeatable but not extremely accurate. Many solutions have been described in the literature but only the methods due to Pieper, Paul and Featherstone will be discussed.

Pieper shows in his Ph.D. dissertation [Pieper69] that a general solution to the inverse kinematics problem for a six degree of freedom manipulator is insoluble. However, He claims that "the existence of 3 revolute axes intersecting at a point leads to a soluble class" [Pieper69 p29] of manipulators. Having three intersecting axes reduces the problem to finding the roots of a fourth degree equation. While the root finding can be done numerically, the degree of the equation can be reduced and a closed form solution determined if the manipulator has a simple geometry such as a_i, d_i equal to zero and/or $\alpha_i = n\frac{\pi}{2}$. Pieper has found a number of solutions for manipulators falling within this classification (3 revolute joint axes intersecting), setting a precedent for the approach to this problem.

Paul [Paul81] has proposed another approach. He expresses the end effector as shown in Equation 2.8 and then generates an additional set of 5 matrix equations by sequentially multiplying T^6 by A_i^{-1} as shown in Equation 2.9. The multiplication by A_i^{-1} transforms the end effector position and orientation to joint i's coordinate frame.

$$\begin{aligned} A_1^{-1}T^6 &= T_6^1 \\ A_2^{-1}A_1^{-1}T^6 &= T_6^2 \\ A_3^{-1}A_2^{-1}A_1^{-1}T^6 &= T_6^3 \\ A_4^{-1}A_3^{-1}A_2^{-1}A_1^{-1}T^6 &= T_6^4 \\ A_5^{-1}A_4^{-1}A_3^{-1}A_2^{-1}A_1^{-1}T^6 &= T_6^5 \end{aligned} \tag{2.9}$$

By equating each element in all 6 matrix equations and representing T^6 symbolically, 72

equations are created. A solution is attempted by sorting through the equations, searching for easy solutions and exploiting the accumulated knowledge. Paul demonstrates his method on the Stanford manipulator and a Puma-like manipulator that has no offsets, the "elbow" manipulator.

Featherstone [Feathe83] has also published a closed form inverse kinematics solution for a manipulator similar to Paul's elbow manipulator. His approach does not suggest a systematic method that could be applied to other manipulators as the solution is aided by the manipulator's geometrical properties. The solution is obtained by using the Cosine Law and `ATAN2()` to find the first three joint angles and spherical geometry to find the remaining three. This solution is interesting as suggestions are made for an on-line implementation. Also, the conditions where the solution is ill conditioned are discussed and alternatives proposed. Featherstone also addresses testing for targets that are out of reach.

Some of the approaches to closed form inverse kinematics have been briefly described here. It is seen that closed form solutions can only be found for a limited class of manipulators to which, fortunately, most industrial manipulators belong. The approach to finding a solution is manipulator specific and a general approach does not exist. Furthermore, a closed form solution does not consider the manipulator deviating from its ideal model which could arise, for example from wear or inconsistent fabrication producing an erroneous solution.

Iterative Solutions

Iterative solutions attempt to minimize the error between the target and end effector after each iteration cycle and continue until a set of termination conditions are met. In contrast to closed form solutions, only the solution that is closest to the present manipulator pose is found and more calculations are required to obtain a solution. However, iterative

solutions can use sensor information for locating the end effector position and orientation. This reduces the effect of modelling errors since the algorithm is driven by the measured distance between the end effector and target. Thus, problems that would be encountered by a closed form solution are circumvented. Unfortunately, the rate of convergence is not fixed for iterative solutions and varies according to the manipulator's location. An approximate upper bound on the convergence rate is suggested by performing a number of experiments about difficult manipulator poses.

Most iterative inverse kinematics solutions are based on the Jacobian. As the Jacobian can be calculated for any manipulator, inverse kinematics solutions for manipulators that have no closed form solution can be obtained. Since the Jacobian maps joint space changes onto work space changes, its inverse is needed, which can only be found if the Jacobian is full rank or non-singular. Iterative solutions based on the Jacobian work well when the manipulator is well away from its singular points, but a number of researchers have attempted solutions based on the Jacobian that produce useable results in singular regions. The Jacobian based inverse kinematic solutions due to Whitney, Goldenberg and Wampler will be presented as well as the non-Jacobian based solutions due to Milenkovic and Poon.

The first references for using the Jacobian to address the inverse kinematics problem are attributed to Whitney [Whitne69, Whitne72] who calculated the Jacobian by partial differentiation of the forward kinematics as well as by using vector notation and working directly with the Cartesian and angular velocities. Recognizing that calculating the Jacobian was time expensive, Whitney suggested precomputing a number of J^{-1} matrices at various values of q and then interpolating. The singularity problem was identified, discussed and the suggested solution, where possible, was to delete from the Jacobian the rows and columns associated with the singular joints. This strategy does not provide an adequate solution to the overall singularity problem, (for example, how the manipulator

would reach a target located at a full reach singularity). Otherwise, no modification to the Jacobian was made and it was assumed to be invertable so that $\dot{q}$ could be obtained from $\dot{q} = J^{-1}\dot{x}$. The joint angle solution was then obtained by integrating the joint velocity solution.

Goldenberg *et al.* [Golden85] uses an iterative minimization method, a hybrid of Newton-Raphson and steepest descent, which is regulated by imposing a maximum step size on the joint increments. If the set of joint increments obtained by the inverse Jacobian is less than the maximum step size, then the Newton-Raphson procedure is used. Otherwise, the Newton-Raphson solution and a negative multiple of the gradient are combined such that their norm is equal to the step size. Another iteration is started if the resulting solution is montonicaly decreasing and none of the joint limits are exceeded. Should one of these conditions not hold, the step size is reduced and the process is repeated. Enforcing a maximum step size in such a manner is very computationally intensive, forming a series of iterations within an iteration. Powell, the developer of this minimization algorithm, outlines the conditions for its use "However the actual procedure we have defined is less elegant than the one used by Marquardt, so here it is appropriate to repeat the remark that the method that we prefer is chosen because it economizes on the number of computer operations, when J [the Jacobian] is approximated numerically."[Powell70] Examination of Goldenberg's method suggests that a more stream-lined approach should be pursued along the lines of Levenberg/Marquardt stabilization [Lawson74] if the goal of on-line inverse kinematics is to be accomplished.

The approach of Levenberg/Marquardt as applied to inverse kinematics was developed independently by researchers Wampler [Wample85, Wample88] and Nakamura/Hanafusa [Nakamu86]. Whereas Goldenberg *et al.* did not address the singularity issue, these researchers did. The name given to the implementation of the Levenberg/Marquardt method by Wampler is damped least squares since the objective is to minimize the error

function $\|\dot{x}-J\dot{q}\|+\alpha^2\|\dot{q}\|$. The first term is the basic least squares criterion, requiring that the solution has a small error and the second term requires that the solution be feasible. Thus, the two terms orchestrate a trade-off between a minimum error solution and a feasible solution. For high manipulability poses the second term is of no consequence and only adds to the solution error. But, for singular poses, the second term dominates and allows a useable solution to be obtained. The joint velocity solution using damped least squares is shown in Equation 2.10.

$$\dot{q} = (J^T J + \alpha^2 I)^+ J^T \dot{x} \tag{2.10}$$

Wampler refers to the term $(J^T J + \alpha^2 I)^+ J^T$ as the damped pseudoinverse which is the generalized inverse [Stewar77] that minimizes the error function. The α^2 term, called the damping factor, provides damping to the solution and was fixed in Wampler's work. Nakamura/Hanafusa went one step further and used a variable damping factor. A variable damping factor has the advantage that it can be minimized when it is not needed, such as in a high manipulability region, and then increased as the manipulator approaches a singularity. Chan [Chan87] developed a variable damping method which was regulated by the distance between the end effector and target. Damped Least Squares allows a useable solution to be obtained in singular regions.

Iterative solutions which are not based on the Jacobian have also been found. Two such solutions are Functional Joint Control [Poon88a] and the Discrete Linkage Method [Milenk83]. Functional Joint Control (FJC) assumes that each manipulator joint has a specific function that needs to be fulfilled. Under this approach each joint attempts to fulfill its function, independent of the actions of the other joints. The method works well in singular regions but suffers when the manipulator can not be divided into major and minor linkages [Poon88a, Poon88b]. The Discrete Linkage Method also assumes that

the manipulator is separable into major and minor linkages with its major linkage being much bigger than its minor linkage. The manipulator's position is solved by moving the major linkage and the orientation is solved for by moving the minor linkage. The solution is then obtained by iterating until the desired tolerance is met. Chan [Chan87] implemented a version of this technique and observed some instances where the major and minor linkages oscillated in opposition such that a solution was not obtained.

2.4 Desired Properties of an Inverse Kinematics Solution

To achieve the goal of a readily portable, on-line inverse kinematics solution suitable for teleoperation, a Jacobian based algorithm appears to present a viable approach in light of its properties and the findings of other researchers. Closed form solutions, although more computationally efficient than iterative ones, are not transferable between manipulators and suffer when modelling errors are present. With iterative algorithms, the effect of modelling errors can be reduced by using sensor information for locating the end effector and calculating the distance to the target. Despite the fact that the Jacobian can be readily calculated for any manipulator, the calculations still have to be performed, impeding the algorithm's transfer to other manipulators. For on-line operation of a Jacobian based iterative algorithm, the bulk of the calculation is in the following three steps which must be performed for each iteration: calculate the Jacobian, stablize it and then solve for the joint rates using linear algebra tools, such as, Gaussian elimination. Each step in this procedure must have a minimal number of calculations if on-line performance is to be optimal. In summary, the properties of a Jacobian based inverse kinematics algorithm should have a minimal solution time, be robust in singular regions and be accurate within a specified error tolerance.

Chapter 3

Inverse Jacobian Estimation

The approach to inverse kinematics proposed in this thesis is to estimate the Jacobian's inverse using recursive least squares. Several benefits can be realized from this approach: the Jacobian does not need to be calculated, a numerical matrix inversion routine is no longer required and no *a priori* knowledge of the manipulator's kinematic parameters is necessary. This implies that the algorithm is readily portable among manipulators and will adapt to changes in the manipulator's kinematics. As the Jacobian's inverse is estimated, a usable relation is created in regions where the analytic inverse would have been undefined. The results from the work due to Lobbezoo *et al.* [Lobbez88] on a two link manipulator suggest that estimating the inverse Jacobian as an inverse kinematics strategy is viable.

Before an inverse kinematics algorithm based on recursive least squares can be outlined, some specific problems encountered by an estimator in this application must be considered. This chapter addresses the modelling and properties issues that arise when the Jacobian is estimated. An understanding of what results can be obtained using estimation must be established before the algorithm, based on estimation, can be presented and evaluated.

3.1 Inverse Jacobian Model Selection

Before an estimation scheme can be implemented, a model of the system to be estimated must be defined. In the present case, the system of interest is the inverse Jacobian. What

must be considered is how to formulate this system in a model that can be used by an estimator. Of the several models that are used by estimators, ARMA, state space and linear difference models, to name a few, the linear difference model was selected because only input and output data need to be known. When applying the linear difference equation to model the inverse Jacobian, data from previous sampling intervals is not explicitly considered and no time-delay is assumed. (Data from previous sampling intervals will be considered by the estimator though.) The inverse Jacobian, as seen in the previous chapter, is a multiple input multiple output (MIMO) system. The inputs are the components of the work space vector $\dot{x}$ and the outputs are the joint space vector components $\dot{q}$. For modelling purposes, the inverse Jacobian can be considered as being made up of a series of multiple input single output (MISO) systems as shown in Equation 3.11.

$$\begin{aligned}
\dot{q}_1 &= J_{11}^{+}\dot{x}_1 + J_{12}^{+}\dot{x}_2 + \cdots + J_{1m}^{+}\dot{x}_m \\
\dot{q}_2 &= J_{21}^{+}\dot{x}_1 + J_{22}^{+}\dot{x}_2 + \cdots + J_{2m}^{+}\dot{x}_m \\
\dot{q}_3 &= J_{31}^{+}\dot{x}_1 + J_{32}^{+}\dot{x}_2 + \cdots + J_{3m}^{+}\dot{x}_m \\
&\vdots \\
\dot{q}_n &= J_{n1}^{+}\dot{x}_1 + J_{n2}^{+}\dot{x}_2 + \cdots + J_{nm}^{+}\dot{x}_m
\end{aligned} \tag{3.11}$$

The parameters J_{ij}^{+} $i = 1, n,\ j = 1, m$ are to be estimated by some recursive least-squares estimation scheme. Note that the MIMO model for the inverse Jacobian can be treated as a matrix with the same dimensions as the transposed analytic Jacobian. No simplifications can be made to this matrix, such as zeroing elements or making specific elements constant, since no information about the target manipulator is available other than the number of joints.

When a manipulator with six degrees of freedom is considered, the Jacobian's column

dimension can vary according to the orientation representation used. To avoid estimator tracking problems the orientation representation must be carefully selected. Of the orientation representations examined, Euler parameters were found to produce the most stable results when used in conjunction with an estimator. Although orientation angles such as Euler angles and Roll-Pitch-Yaw angles provide the minimal number of variables needed to specify an arbitrary orientation, they are both singular at one particular configuration [Wample85]. Near an orientational singularity the orientation parameters change rapidly and discontinously preventing the estimator from tracking them so a non-singular orientation method is required. N-O-A vectors are one such method. However, a total of nine parameters are needed introducing 6 redundant parameters involving the estimation of an additional 36 parameters over the orientation angles case. Estimating redundant parameters is undesirable as indicated by the Parsimony principle of Ljung and Söderström [Ljung83], which states that cumulative estimation error increases with each redundant parameter. Another, more desirable method, is Euler parameters which consist of four components that are nonsingular, but non-unique. Although redundancy is undesirable it is compensated for by the non-singular nature of Euler parameters. The non-uniqueness can be readily solved by consistently selecting the positive solution for the scalar quantity. Four components for specifying orientation implies that a 6×7 matrix with 42 parameters must be estimated, 6 parameters over the theoretical minimum for the six degree of freedom arm.

3.2 Estimation Method Selected

A MIMO estimator is required to determine the inverse Jacobian relationship between the work space and joint space. As the relationship is required in real time, on-line estimation methods are necessary. Recursive least squares estimation is an on-line method

that requires minimal information about the system being estimated and is obtained by restructuring the inputs to existing recursive least squares estimators. Also, it forms the basis of many more detailed estimation schemes such as extended least squares or recursive maximum likelihood method [Åström77]. A MIMO estimator can be formed by merging several MISO estimators together. Since each estimator shares the same error covariance matrix, a computationally efficient MIMO estimator can be formed [Toivon77]. The MISO estimator has the same structure as the basic SISO recursive least squares estimator allowing a MIMO recursive least squares estimator to be readily implemented.

One recursive least squares algorithm that is more numerically stable and accurate than the conventional algorithm is UD factorization. This algorithm which was developed by Bierman [Bierma77] updates the error covariance matrix using the Agee-Turner algorithm [Agee72] as modified by Carlson [Carlso73]. The UD factorization algorithm updates the square root of the error covariance matrix. Kaminski [Kamins71] has shown that algorithms that operate on a matrix's square root have a condition number that is the square root of the matrix's condition number. From this observation, claims have been made that square root algorithms such as UD factorization have twice the numerical precision of the conventional algorithm [Kamins71, Carlso73]. Error analysis by Gentleman [Gentle73] and Bierman [Bierma77] indicate that UD factorization has good numerical accuracy and stability which is comparible to Householder orthogonal transformations. UD factorization is also computationally efficient. Unlike other square root algorithms such as Potter's algorithm[Potter64], it does not require square roots to be explicitly calculated thereby improving the computational performance. Bierman [Bierma77] presents several tables showing the computations of UD factorization in comparision to other algorithms. His findings are that the UD factorization algorithm is more efficient than the Potter and Carlson square root algorithms and it is almost as efficient as the Kalman algorithm [Kalman63] but does not require *a priori* knowledge

of the noise variances. Table 3.1 compares the computations required by MIMO UD factorization and conventional recursive least squares to estimate a $n \times n$ model matrix.

MIMO Recursive Least Squares Method	$\times$	+/-	$\div$
Conventional	$5.5n^2 + 2n$	$6n^2 + 3n$	$n^2 + 1$
UD factorization	$3.5n^2 + 2.5n$	$3.5n^2 + 0.5n$	3n

Table 3.1: Comparision of Recursive Least Squares Computations

From Table 3.1 it is observed that UD factorization has less computationally expensive operations, such as division and multiplication, than conventional recursive least squares. For the purpose of estimating the inverse Jacobian the UD factorization algorithm is preferred over conventional recursive least squares as it is both numerically stable and more computationally efficient.

Another complication when estimating the inverse is its variable nature. The rate at which the inverse varies is joint space dependent. In high manipulability regions the rate is low whereas in singular positions the rate is high, which in the limit approaches infinity. This implies that, in the worst case, the estimator must be able to track multiple parameters that are rapidly varying. Traditionally, slowly varying parameters were tracked by using a forgetting factor which allowed older data to be weighted more lightly in the estimation. However, this factor is usually a constant and will not work well for rapidly varying parameters. One method for ensuring good tracking is to perform a series of estimations about a fixed joint space pose before moving to another pose. The computations required would be prohibitive for an on-line implementation of the estimator as the direct kinematics would then be required. However, such an approach is robust as stable estimates would be obtained before any control would be performed. Another

approach is to restrict the joint space movement per sampling interval which has proved to adequately handle the tracking problem in this application.

3.3 Estimated Inverse Jacobian Properties

Using estimation, the calculation, stabilization and inversion of the Jacobian is performed in one step, which is more computationally efficient than performing these steps individually. Experimental findings indicate that recursive least squares estimation (RLSE) accomplishes these three steps and produces a stabilized inverse Jacobian with properties similar to damped least squares (DLS). Proving this experimentally obtained result in analytical terms is difficult. An analytical approach is only tractable for simple manipulators, such as one and two link manipulators. This is because the equations being studied are non-linear and multi-dimensional. While numerical measures can be used to characterise the estimate's properties, they are of limited utility when comparing results from Jacobian methods on more interesting manipulators since finding an unbiased data representation is arduous. A thorough discussion of estimated inverse Jacobian's properties, independent of the proposed inverse kinematic algorithm's structure, is important because the estimate separates the proposed algorithm from other Jacobian based methods.

3.3.1 Comparision of Least Squares Methods

It has been suggested in the previous section that the estimated inverse Jacobian obtained from RLSE has properties similar to DLS. By examining the cost function minimized by the respective methods, some qualitative comparisions of RLSE and DLS will be made. In Table 3.2 the equations describing both least squares methods are shown. DLS requires that the Jacobian is known at the desired joint pose as well as the work space velocity

or position change, whereas RLSE estimation requires that several pairs of joint/work space data vectors are known.

	Damped Least Squares	Recursive Least Squares Estimation
Relation	$\dot{q} = (J^T J + \alpha I)^{-1} J^T \dot{x}$	$\Delta q = J^+_{est} \Delta x + e$
Cost Function	$\|\dot{x} - J\dot{q}\| + \alpha\|\dot{q}\|$	$\|\Delta q - J^+_{est}\Delta x\|$
Independent variables	$J,\ \alpha,\ \dot{x}$	$\Delta q,\ \Delta x$
Dependent variables	$\dot{q}$	J^+_{est}

Table 3.2: Comparision of Least Squares Methods

It is seen from Table 3.2 that the cost function for DLS consists of two components. Minimizing the first component, $\|\dot{x} - J\dot{q}\|$, is the basic least squares approach which is shared by RLSE as well. However, the second component in the DLS Cost Function, $\alpha\|\dot{q}\|$, provides damping or Levenberg-Marquart stabilization [Lawson74]. The DLS Cost Function seeks a tradeoff in the solution between minimum error and solution size. Unfortunately, this damping term is not readily added to the RLSE Cost Function since Δq is an independent variable as opposed to being a dependent variable in DLS. However, it does suggest that controlling the size of Δq data could implement this additional constraint. This supposition is expanded on in Section 3.3.2. Also, it is seen that if α is set to zero then DLS reduces to least squares and the result is the inverse. By examining the cost functions and inputs to DLS and RLSE similarities between the two methods can be seen.

3.3.2 Estimation Using Kinematic Position Data - An Approximate Model

The inverse Jacobian can be obtained by least squares estimation provided that the joint and work space data is related by the Jacobian. If a differential change in the joint space

is reflected to the equivalent differential change in the work space by the manipulator's direct kinematics, the two changes are also related by the Jacobian since the Jacobian can be formed from partial differentiation of the direct kinematics. Therefore, least squares estimation of such data should produce the Jacobian as seen in the previous section. However, if non-differential spatial differences are used by the estimator, it is not clear that the Jacobian relationship will still be estimated. An approximate analytic model of estimation using kinematic position data will be developed and examined.

A one link manipulator is used here as a basis for an approximate model. Although a one link manipulator is neither practical nor interesting, it does provide an effective vehicle for developing a clear and simple model of the results from the estimation process. This model will provide insight when estimation is applied to more useful manipulators and will aid the interpretation of the numeric results presented in the sections below.

The direct kinematics of the one link manipulator shown in Figure 3.2 is described by Equation 3.12.

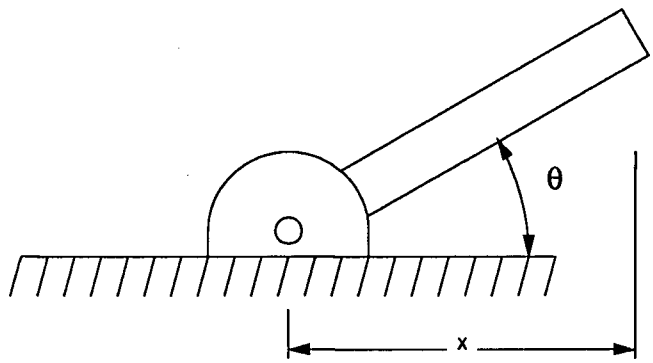

Figure 3.2: One Link Manipulator

$$x = \cos\theta \tag{3.12}$$

Differentiating Equation 3.12 provides the inverse Jacobian relationship shown in Equation 3.13.

$$\dot{\theta} = -\frac{1}{\sin\theta}\dot{x} \quad \theta \neq 0 \tag{3.13}$$

The damped least squares result due to Wampler [Wample85] is given by Equation 3.14.

$$\dot{\theta} = \frac{1}{-\sin\theta + \alpha^2}\dot{x} \tag{3.14}$$

The Jacobian was estimated using a simple average of n data points $(\Delta\theta, \Delta x)$ as described by Equation 3.15. The inverse Jacobian can then be obtained by taking the reciprocal. In later sections the inverse Jacobian is estimated; however, the Jacobian was estimated here to simplify the mathematics.

$$J_{est} = \frac{1}{n}\sum_{i=1}^{n}\frac{\Delta x_i}{\Delta\theta_i} \tag{3.15}$$

The joint space position difference, $\Delta\theta_i$ in Equation 3.15, is obtained by generating a Gaussian random number with mean zero and variance σ^2 (Equation 3.16). This allows changes about a specific joint angle to be examined. Given $\Delta\theta_i$, the work space position difference, Δx_i, is then calculated using Equation 3.17.

$$\Delta\theta \in N(0, \sigma^2) \tag{3.16}$$

$$\Delta x = \cos(\theta + \Delta\theta) - \cos\theta \tag{3.17}$$

Equation 3.17 can be simplified by using the trigonometric identity.

$$\cos(A + B) = \cos A \cos B - \sin A \sin B$$

in conjunction with the following approximations for $\sin\Delta\theta$ and $\cos\Delta\theta$ knowing that $\Delta\theta$ is small. Higher order terms in the sin/cos approximations are neglected since they approach zero more rapidly.

$$\begin{aligned} \sin\Delta\theta &\approx \Delta\theta \\ \cos\Delta\theta &\approx 1 - \frac{\Delta\theta^2}{2} \end{aligned}$$

It is found that Equation 3.17 reduces to

$$\Delta x_i \approx -\Delta\theta_i \sin\theta - \frac{\Delta\theta_i^2}{2}\cos\theta \tag{3.18}$$

Therefore the estimated Jacobian in Equation 3.15 can be approximated as follows:

$$J_{est} \approx -\sin\theta - \frac{\overline{\Delta\theta}}{2}\cos\theta \quad \textit{where} \quad \overline{\Delta\theta} = \frac{1}{n}\sum_{i=1}^{n}\Delta\theta_i \tag{3.19}$$

The inverse Jacobian in this simple case is the reciprocal and is shown in Equation 3.20.

$$J_{est}^{-1} = \frac{1}{-\sin\theta - \frac{\overline{\Delta\theta}}{2}\cos\theta} \tag{3.20}$$

An examination of Equation 3.20 reveals several interesting properties. It is seen that the $\frac{\overline{\Delta\theta}}{2}\cos\theta$ term is functionally equivalent to the α^2 term in damped least squares, Equation 3.14. This term provides the estimation method with damping and will be referred to as 'the damping term'. Intuitively, the damping term should be zero since the sample mean, $\overline{\Delta\theta}$, is the sum of Gaussian random numbers with mean zero. However, it is observed in numerical trials that this does not occur, primarily because the number of samples taken to form the mean is not very large. The sample mean is a number near zero which is influenced by the size of the variance and the number of data points used.

A range over which the sample mean would reasonably lie is needed in order to suggest a range for the damping factor. If an interval is considered in which $\overline{\Delta\theta}$ will be found with probability 0.99, a 99% confidence interval, the following bounds are defined.

$$-2.57583\frac{\sigma}{\sqrt{n}} \leq \overline{\Delta\theta} \leq 2.57583\frac{\sigma}{\sqrt{n}}$$

This will provide an upper limit on the damping factor. A reasonable non-zero lower limit is also required to indicate a miminum damping factor. The probabilty that the sample mean is exactly zero is zero but the sample mean will most likely to be found outside a small region about zero. Assuming that this region, having probability 0.01, is sufficiently small, then an approximate lower limit can be placed on the sample mean. Such a bound would be

$$\overline{\Delta\theta} \leq -0.01253\frac{\sigma}{\sqrt{n}} \;\; and \;\; 0.01253\frac{\sigma}{\sqrt{n}} \leq \overline{\Delta\theta}$$

An interval in which the sample mean will lie with probability 0.98 can be defined as follows.

$$0.01253\frac{\sigma}{\sqrt{n}} \leq |\overline{\Delta\theta}| \leq 2.57583\frac{\sigma}{\sqrt{n}} \tag{3.21}$$

The range of $\overline{\Delta\theta}$ can be modified by the standard deviation σ and the number of data samples n as seen from Equation 3.21. Assuming that n is fixed, then the damping factor can be tuned by varing the standard deviation.

Another property arising from Equation 3.20 is the damping factor's dependence on the joint pose θ, which is due to the $\cos\theta$ term. When the one link manipulator is in a high manipulability region ($\theta = \frac{\pi}{2}$), the damping is zero, but when it is in a singular region ($\theta = 0$), the damping is maximized. So, the damping factor resulting from estimation is not only tunable, it is also variable according to the manipulator's pose. This maximizes the damping where it is most needed and removes it where it is not.

The model, as derived for the one link manipulator, suggests that several desirable properties arise from estimation. When difference data generated from the direct kinematics is used to estimate the Jacobian, an estimate whose reciprocal is functionally equivalent to damped least squares results. The amount of damping can be tuned by adjusting the variance of the joint angle changes, since the variance is proportional to the damping as seen by Equation 3.21. The damping term is also dependent on the joint pose where high manipulability poses have no damping and singular poses have the maximum amount of damping. This property, labeled as "variable damping", was not conclusively found when estimation was applied to more sophisticated manipulators. The properties suggested by this model indicate that least squares estimation provides a basis for a robust inverse kinematics algorithm which is similar to Damped Least Squares.

3.3.3 Numerical Measures of Estimator Properties

An analytic model for a more practical manipulator is not viable since the Jacobian becomes dependent on a number of variables that are not readily separable due to non-linearities. Therefore, modeling of more interesting manipulators must be abandoned due to the complexity involved, suggesting that a numerical approach should be taken. A numerical approach would identify salient features which could be readily compared to other Jacobian based iterative methods. However, the results from such comparisons will be largely qualitative. Although the potential for insight is lost when a numerical approach is taken, a numerical measure can be made independent of the manipulator's structure. As discussed in the previous chapter, researchers have used a numerical result, the manipulability measure, to investigate the quality of a manipulator's work space as a function of the Jacobian. This approach can also be applied to determine how modifications to the Jacobian would affect the manipulability, provided that the comparisions are made over the same spatial region. As an example, the manipulability of a two link

manipulator will be measured by calculating the condition number of the three Jacobian methods: RLSE, DLS, as well as the basic, unmodified inverse Jacobian. Furthermore, the elements of the modified Jacobian for the three methods will also be compared.

In the following simulation work, several constraints have been placed on the inverse Jacobian estimation procedure in order to obtain typical behaviour. The data used in the estimation process is noise-free and is generated by perturbing the manipulator at a given joint pose by a vector of Gaussian random variables with zero mean and fixed variance. The work space perturbations resulting from this random vector are determined by using the manipulator's direct kinematics. The generated joint and work space perturbations are passed as inputs to the estimator until a stable estimate is obtained. This procedure is repeated a number of times, typically 100, after which the estimates from each step are averaged. Overall, the aim of this process is to average out anomalous behavior of the estimator so that its typical behavior can be observed.

A two link manipulator was used to investigate the estimator's properties, numerically, because its closed form solution can be calculated, its results can be readily verified, and it allows immediate comparison of RLSE to DLS and J^{-1}. (For a description of the Denavit-Hartenberg parameters of the two link manipulator see Appendix A.) The data is collected from a number of manipulator poses ranging from high manipulability poses to singular ones. It is known that the second revolute joint creates the performance problems that must be considered by any robust Jacobian method whereas the first joint does not, so it will be arbitrarily set to zero. The two link manipulator provides an effective environment for performing numerical comparisions of the inverse kinematics algorithms.

If the condition numbers for J^{-1}, DLS and RLSE are recorded as joint two is varied from $\frac{\pi}{2}$ to zero, Figure 3.3 results.

It is noted that the condition number is smallest at $\theta_2 = \frac{\pi}{2}$ and largest at $\theta_2 = 0$. The

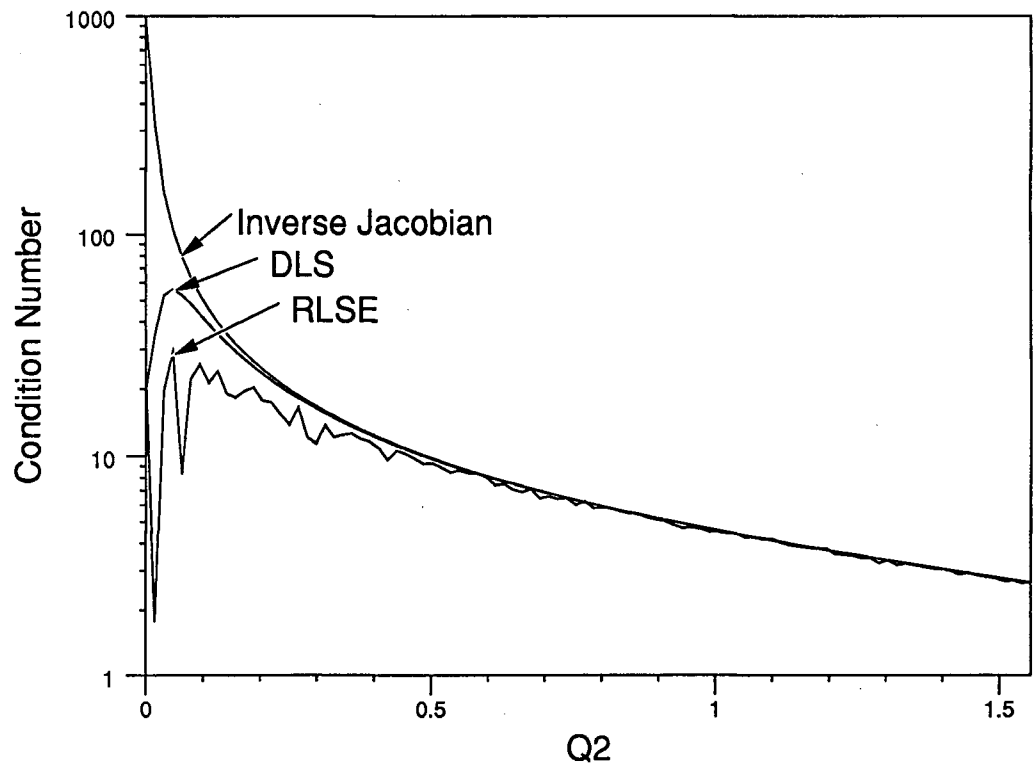

Figure 3.3: Condition Number over range of θ_2 for J^{-1}, DLS, and RLSE

trade-off in the DLS cost function between the correct answer and a feasible answer can be seen here. For θ_2 near $\frac{\pi}{2}$, a high manipulability region, the DLS and RLSE condition numbers are very close to the strict inverse Jacobian. As θ_2 approaches 0, the singular region, the condition numbers for DLS and RLSE diverge from the strict inverse Jacobian, reach a maximum and then decrease. The condition numbers divergence and subsequent restriction is the result of damping as compared to the strict inverse Jacobian's condition number which grows without bound. The behavior of DLS and RLSE condition number can be explained by examining the cost function for DLS from Table 3.2. As the Jacobian becomes increasingly singular the size of the solution for $\dot{q}$ or Δq becomes

regulated by the $\alpha\|\dot{q}\|$ term, thereby constraining the solution's size. The condition number then decreases due to the switch in emphasis in the cost function. It is suggested from examining Figure 3.3 that RLSE has the same characteristics as DLS, as presented in the approximate model of the previous section.

Since RLSE appears to damp the Jacobian near singularities in a manner similar to DLS, it should be determined if this damping can be controlled. This property was investigated by recording the condition number for several variances of $\Delta\theta_2$ over the range $0 \leq \theta_2 \leq \frac{\pi}{2}$. The results from the strict inverse and DLS Jacobian were included as well for reference, with the damping term for DLS arbitrarily set to $\sigma^2 = 10^{-3}$. Figure 3.4 shows that the condition number's peak is compressed as the variance of $\Delta\theta_2$ is increased. In contrast, small variances, 10^{-5} for example, have a condition number curve that closely follows the strict inverse Jacobian for most of the range of θ_2 which supports the model's hypothesis that, given differential data, RLSE closely approximates the strict inverse Jacobian, J^{-1}. The main feature to be observed from Figure 3.4 is that the RLSE condition number curve changes with the variance of $\Delta\theta_2$ suggesting that the damping is a function of the variance. This result is further supported by examining the individual RLSE parameters over the prescribed range. These parameters are presented in Figures 3.5, 3.6, 3.7, 3.8 shown below along with the DLS and strict inverse Jacobian parameters. Note that the inverse Jacobian parameters J_{12}^{-1} and J_{22}^{-1} are constant, a result from $\theta_1 = 0$. From Figures 3.5 thru 3.8 it is seen that the RLSE and DLS parameters are very close to the strict inverse Jacobian when θ_2 is near $\frac{\pi}{2}$ and diverge as θ_2 approaches the singular point at $\theta_2 = 0$. Parameters J_{11} and J_{21} for the strict inverse Jacobian increase without bound whereas the corresponding DLS and RLSE parameters are bounded. It is seen that the bounding is a function of variance and that the damping impacts on the DLS and RLSE parameters J_{12}^{+}, J_{22}^{+} near zero, even though these inverse Jacobian parameters are fixed. By examination of the parameters and the condition number, it is

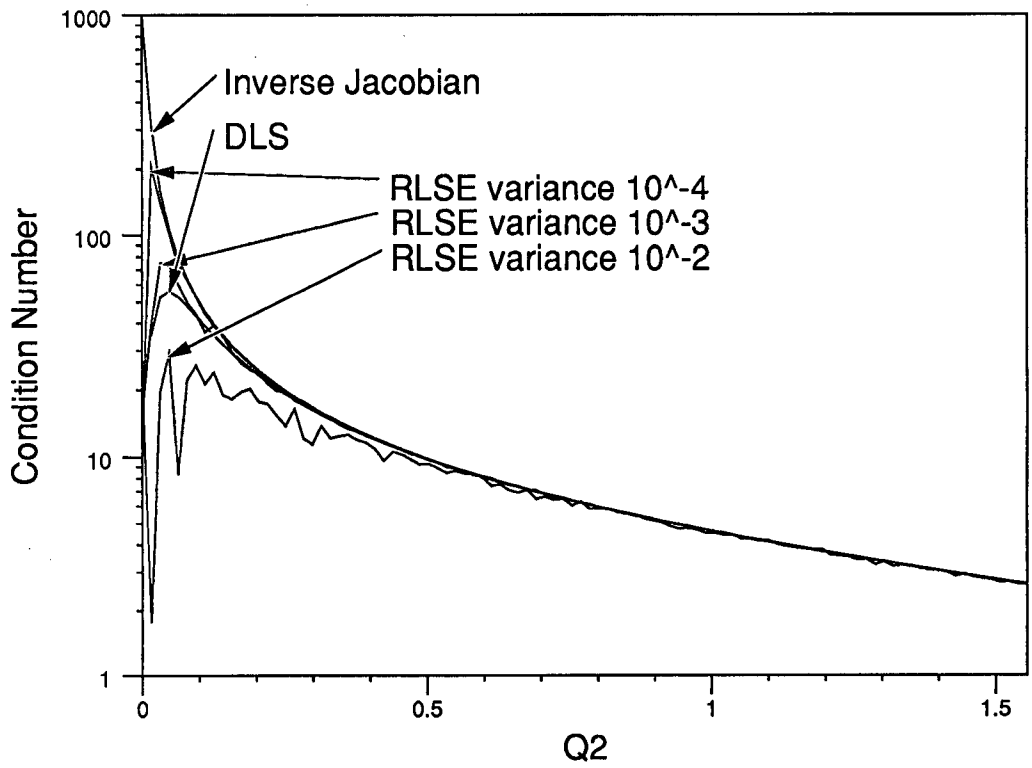

Figure 3.4: Tuneable damping shown by condition number for range of θ_2 at 3 variances of $\Delta\theta_2$

seen that RLSE has damping that is tuneable by adjusting the variance.

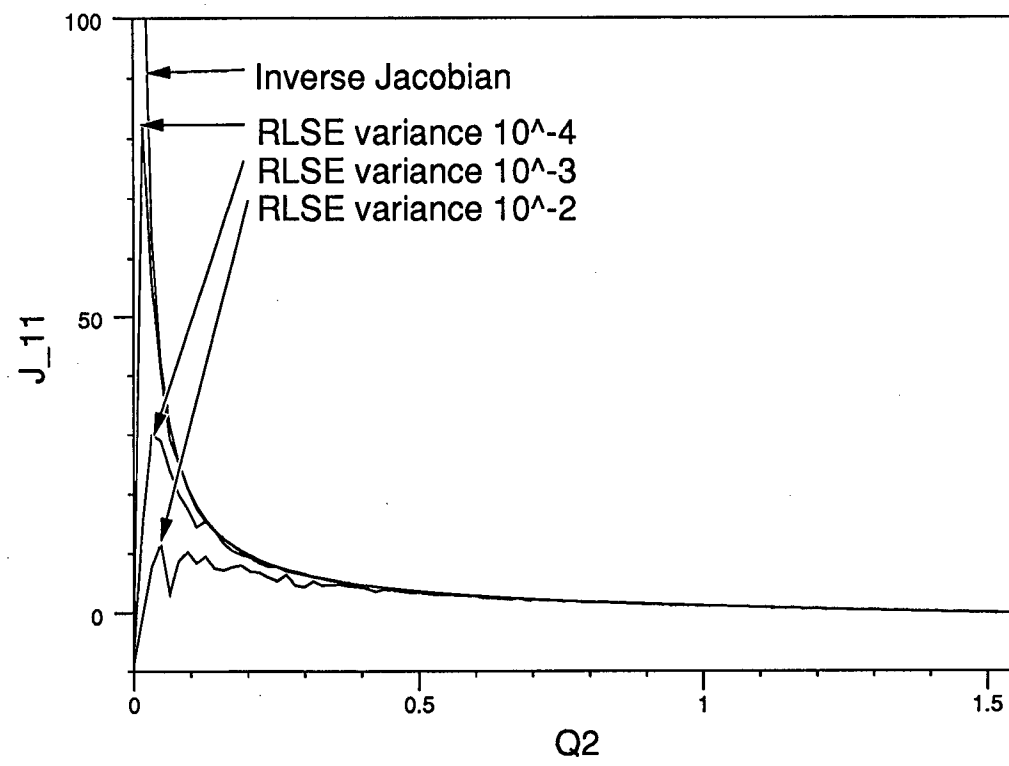

Figure 3.5: Tuneable Damping effect on J_{11}^{+}

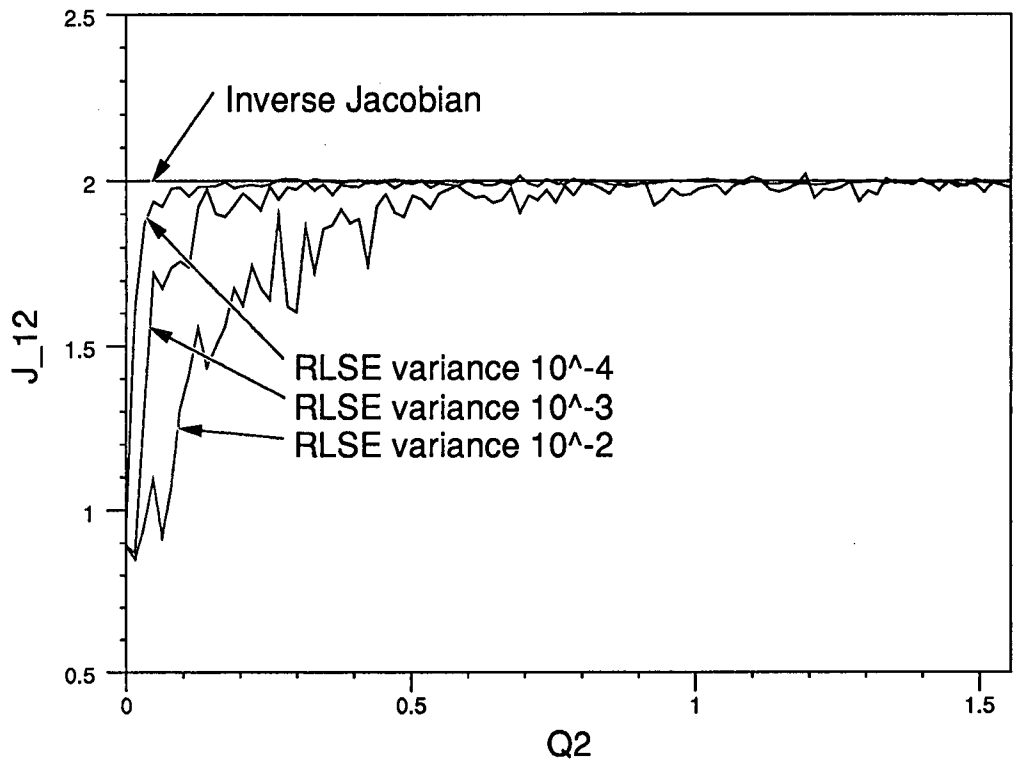

Figure 3.6: Tuneable Damping effect on J_{12}^{+}

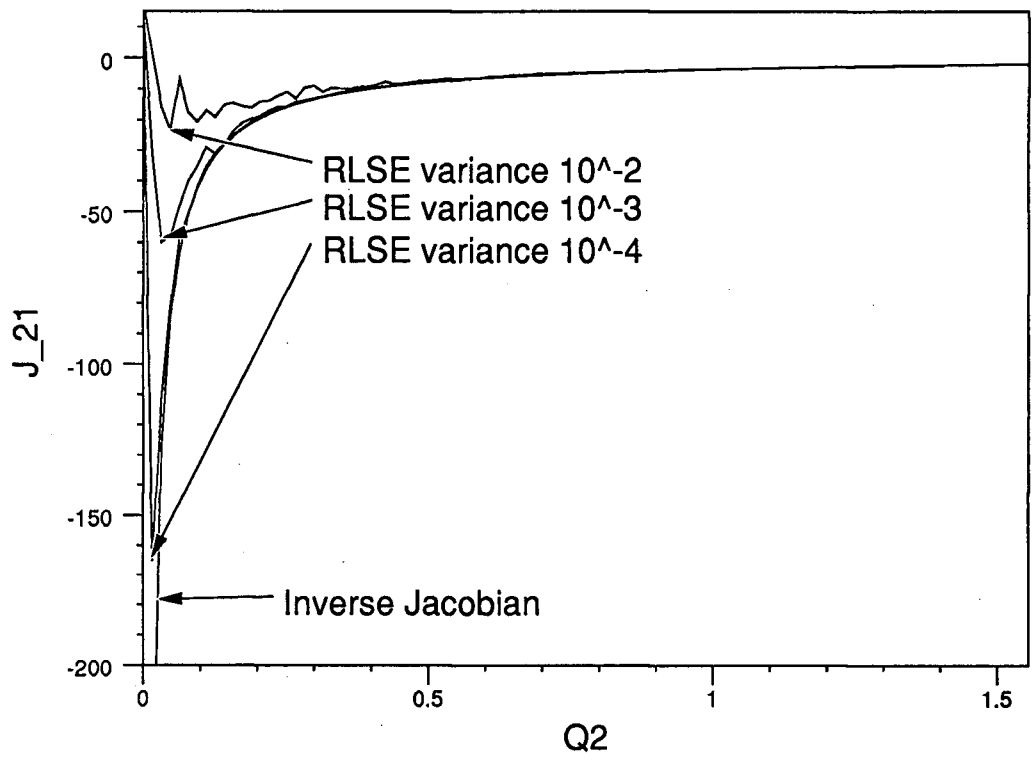

Figure 3.7: Tuneable Damping effect on J_{21}^{+}

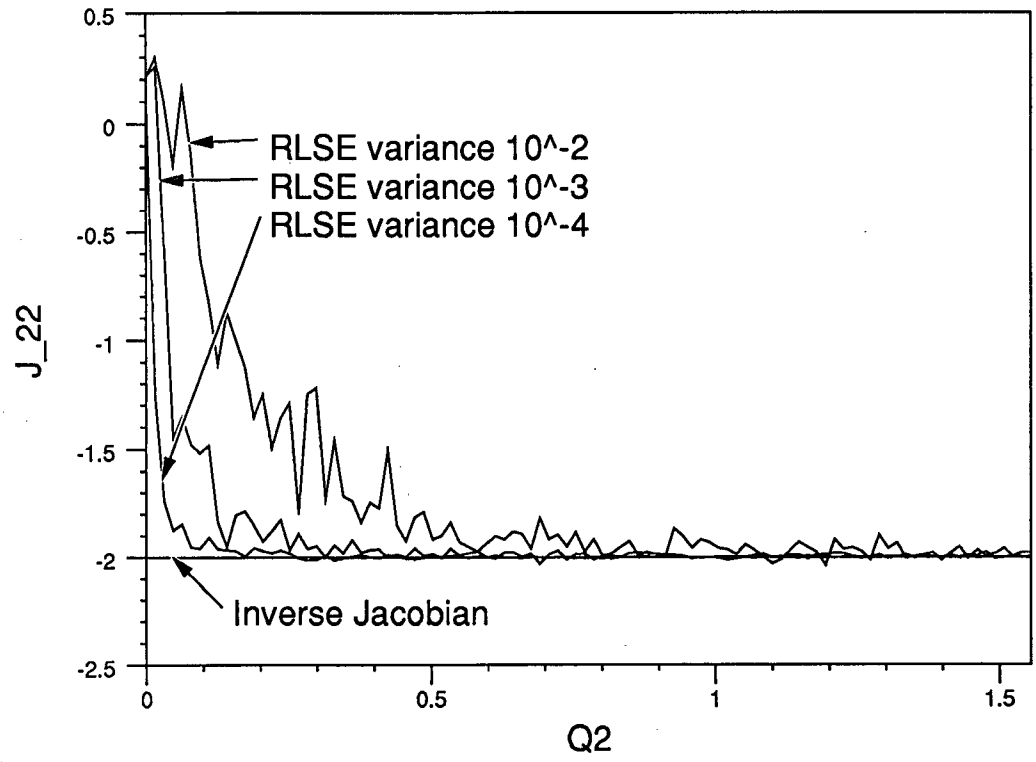

Figure 3.8: Tuneable Damping effect on J_{22}^{+}

Chapter 4

Proposed Inverse Kinematics Algorithm

4.1 Algorithm Description

The proposed algorithm, as shown schematically in Figure 4.9, consists of two main parts: the estimation component and the control component. A target is presented to the algorithm in work space coordinates along with the current end-effector pose and their difference is determined, forming an error which drives the algorithm. The error is converted from the work space coordinates to the joint space coordinates by the estimated inverse Jacobian. After minor processing, the result is then applied to the manipulator. The manipulator moves according to the commanded joint space motion, the corresponding work space motion is monitored by sensors such as a vision system, and then fed back to update the error. The algorithm terminates when one of the following conditions are satisfied: the error tolerances are met, a preset iteration limit exceeded, or an out of reach target is detected, otherwise one iteration cycle of the algorithm has been completed and another iteration is initiated.

The property that distinguishes this algorithm from other Jacobian based iterative algorithms is that the inverse Jacobian is estimated rather than calculated analytically from the manipulator's geometry. This is where the benefits lie as the algorithm requires minimal geometric information and can be used on-line with sensor data if some overshoot is acceptable. The resulting benefit from on-line operation is that the algorithm will adapt to manipulator structural changes. Or, the algorithm can be used iteratively to obtain

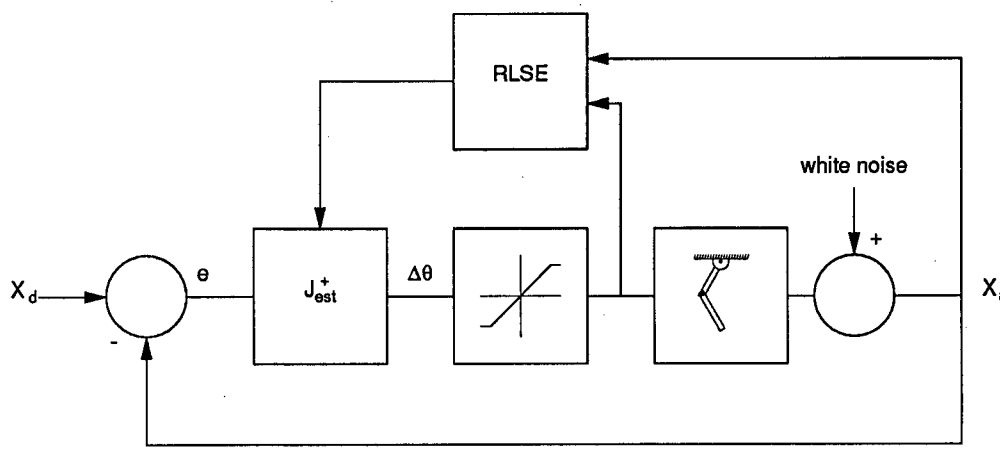

Figure 4.9: Schematic of the Proposed Algorithm

a solution before directing the manipulator to move, thereby avoiding the overshoot problem. The estimator gathers the actual joint space change and resulting work space change to update the damped inverse Jacobian estimate as seen in Figure 4.9.

For proper operation of the estimator several processing steps atypical of a standard Jacobian based algorithm must be performed. The joint space motion must be limited to prevent the Jacobian's parameters from changing too rapidly. If the parameters change too rapidly the estimator will not be able to track them and chaos will result. Another problem inherent to estimators that use forgetting factors for tracking purposes is estimator wind-up. This condition occurs if the data posed to the estimator is not "sufficiently rich", the data describes only a limited region of the function that generated it. When this occurs the gain used to adjust the estimates, the error covariance matrix, grows exponentially indicating that the estimates are more and more uncertain. Steps must be taken to ensure that the signal is sufficiently rich. This cannot be guaranteed by the feedback loop used in the algorithm so white noise is added to the work space

sensor data. The total noise magnitude is an order of magnitude smaller than the error tolerance specifications. Although these problems must be considered when estimating the inverse Jacobian, the benefit of not needing to know the manipulator's geometry is substantial.

4.2 Algorithm Characteristics

4.2.1 Overview

Evaluation of the algorithm was performed using a computer simulation package developed by the author. The simulation package was written using the C programming language on a Hewlett-Packard 9050 computer. The package produces 2d wireframe graphics allowing the manipulator's response to the algorithm's commands to be observed. As no dynamics were considered in the simulation package, the manipulator moves exactly as commanded by the algorithm. The simulation package allows the user to vary many of the parameters affecting the algorithm and specify start and target poses in a variety of ways.

The start and target poses used in the simulations were very selective and a limited distance apart. When joint space targets were generated the maximum distance per joint space variable was $|0.2|$ rads and in the work space 10 cm. This constraint is based on the assumption that, in practice, targets will be inputted to the algorithm from a path planner which will interpolate long distances into a sequence of shorter ones. Also, as exhaustive simulation is neither practical nor informative, specific areas of interest were investigated to determine the algorithm's range of performance. This simplification is valid because most of a manipulator's working volume consists of poses that do not present any difficulty to iterative algorithms. These assumptions make the evaluation process managable, concentrating the efforts on where the algorithm has difficulties.

Of great interest for measuring performance is the time that the algorithm requires to obtain a solution. Unfortunately this is a hard measurement to quantify due to a variety of factors that are dependent on the algorithm's implementation such as the computing hardware used and how the algorithm is programmed. The time measurement used here will be iterations per solution. An iteration reflects a fixed number of calculations that the algorithm must perform in order to determine its status and issue a move command to the manipulator. The number of calculations per iteration can be outlined and an approximate value for time of solution can be determined. This makes a measure which is independent of the implementation details. An actual implementation of the algorithm will require carefull consideration of a variety of conflicting parameters in order to obtain a minimal solution time.

Characterization of the algorithm's properties was performed in several stages working from specific traits to the general ones. First, how tuning the algorithm's parameters affects its performance was examined. Once a guideline for what parameters to use was established, the algorithm's dependence on manipulator pose was investigated followed by a comparision of how the algorithm performed on different manipulators. The evaluation concluded by executing the algorithm on other researcher's examples and comparing the results.

4.2.2 Investigation of Algorithm Parameters

The algorithm has several parameters that can be tuned to alter the algorithm's performance. These parameters fall into two basic categories: those associated with the estimator and those associated with the control algorithm. The estimator parameters affect the estimator initialization, the forgetting factor, λ, which allows the estimator to track the inverse Jacobian and the maximum permissible joint increment per iteration, $\Delta Q_{i_{max}}$, which regulates how much change in the inverse Jacobian is permitted before

the estimator is updated. The parameters associated with control are primarily concerned with detecting when the algorithm should terminate, such as how the position and orientation errors are calculated, what error tolerances were used and how an out of reach target is detected. Once it was established how these various parameters affect the algorithm's performance, suitable values were specified.

Estimator Parameters - Estimate Initialization

As the algorithm is based on estimating the inverse Jacobian parameters, proper initialization of the parameters must be performed before the algorithm is allowed to proceed. For purposes of simulation this initialization is done by generating Gaussian distributed, random joint increments with mean zero and variance typically 0.001. The variance of the Gaussian random number generator determines how coarse or fine the initial estimate is as outlined earlier in Section 3.3.2. The random joint increments are added to the current joint space pose and then the associated work space pose is determined via the direct kinematics equations. Typically, 25 sets of the random data are used for estimator initialization since experiments have shown that the estimate does not improve too much after this point. The same sequence of "random" numbers was used for all estimator initializations which is necessary for obtaining repeatable results. In practice, when the direct kinematics are not known, this initialization procedure is not viable. However, the manipulator can be deployed from its stowed position to some point in space under joint control and the sensors can provide joint space/work space pairs of data to initialize the estimator.

Estimator Parameters - Forgetting Factor Selection

Once the estimator has been initialized the next most important parameter to consider is the forgetting factor. A forgetting factor with a value of one indicates that all data used

by the estimator is equally weighted, whereas, values less than one weight older data less heavily. A smaller forgetting factor causes faster forgetting. The damped inverse Jacobian's parameters are variable in nature so the estimator must use a forgetting factor less than unity to track them. Properly choosing the forgetting factor value is important for proper operation of the algorithm.

As the forgetting factor's function is to facilitate tracking of the varying inverse Jacobian, a series of experiments were performed to determine what would be a suitable value. A Puma manipulator was placed in both a good pose as well as a singular pose and 100 random targets were selected about those poses. The same set of targets were run for different values of the forgetting factor and the average number of iterations per converging solution was calculated. Since the inverse Jacobian varies the most in singular regions, the effect of the forgetting factor needs to be studied there. But, the effect of changing the forgetting factor will also be studied in a high manipulability region to establish a reference. The results of these experiments are shown in Table 4.3.

About the good pose the forgetting factor does not dramatically alter the average number of iterations to obtain a solution, however, there is a noticeable change for the singular pose experiments. In the singular pose experiments the average number of iterations per solution incorporates a fixed value of 50 for all solutions that exceeded that limit rather than simply ignoring the non-converging solutions. It is seen that the average number of iterations per solution decreases with decreasing λ until it reaches 0.5 for the singular region experiments. In the high manipulability region a plateau is reached for $\lambda = 0.6$. Subjective evaluation suggests that λ should not be made as small as 0.5 since the algorithm tends to make the manipulator oscillate more about the target than if the forgetting factor was a slightly larger value. In consideration of this point and the data described in Table 4.3, a forgetting factor of 0.6 was selected.

	good pose		singular pose	
λ	average	number of converging targets	average	number of converging targets
1.00	5.56	100	42.43	29
0.95	5.55	100	40.43	35
0.90	5.53	100	39.37	40
0.85	5.50	100	36.65	49
0.80	5.50	100	33.45	69
0.75	5.48	100	30.78	91
0.70	5.45	100	28.02	89
0.65	5.44	100	24.29	95
0.60	5.45	100	22.03	96
0.55	5.45	100	20.99	98
0.50	5.45	100	19.70	97

Table 4.3: Solution Time as a Function of the Forgetting Factor

Estimator Parameters - $\Delta Q_{i_{max}}$ Selection

Since the estimator has to track varying parameters, precautions must be taken to limit the manipulator's motion. If the manipulator moves too rapidly the estimator may not be able to accurately track the damped inverse Jacobian. Should the estimator lose itself, the algorithm will flail the manipulator about in a chaotic manner. Motion limiting may occur in practice due to actuator saturation and the manipulator's inertia; however, this problem should be guarded against. There are two ways to achieve this which should be used in concert. First, start and target poses should be relatively close so that the inherent error in the estimate is not unduly amplified by a large distance between the target pose and end effector. Also, the described conditions result in requests for large joint changes. When such joint changes are limited, the end-effector describes an erratic path in the work space. Although the algorithm can handle large distances it is better

suited to relatively small distances, 10 to 15 cm separation in position, and maximum orientation changes of order |0.2| rad per variable. Second, a limit should be placed on how much a joint can move in an iteration. This forces the estimator to update itself when a joint changes rapidly. In practice, this will be a function of the sensor sampling rate. The algorithm for limiting joint change per iteration was that of Figure 4.10.

```
if (|ΔQ_i| > ΔQ_imax)
   ΔQ_i = sign(ΔQ_i)ΔQ_imax
```

Figure 4.10: Joint Change Limiting Algorithm

As with the forgetting factor, a best value for $\Delta Q_{i_{max}}$ was determined. If $\Delta Q_{i_{max}}$ is too small the algorithm's time performance will be reduced. Likewise if it is too big the changes in the damped inverse Jacobian will not be adequately limited. In practice $\Delta Q_{i_{max}}$ will be defined by actuator saturation and the rate at which the sensors can be sampled, however, for purposes of simulation a value of 0.05 rads was found to be suitable.

Control Parameters

The control parameters are primarily concerned with detecting under what conditions the algorithm should terminate. The position error is measured in terms of the Euclidean distance between the end-effector and target pose whereas the orientation errors are expressed in terms of NOA vectors regardless of the orientation representation used by the estimator. Specifying the orientation in this manner permits the simulation data to be compared with that of my colleagues, Steve Chan and Joseph Poon. Position and orientation error tolerances were typically chosen to be 0.1 mm and 0.01 radians respectively. If the error tolerances are not met after a preset number of iterations have

elapsed the algorithm terminates. For simulation purposes this threshold was made quite large, 100 iterations, so that the convergence properties of the algorithm would not be biased by too low a threshold, although it could be lowered to 50 iterations. In summary, the control parameters consist of what error tolerances are specified and the maximum allowable iterations per solution. The detection of out of reach targets also falls under this category but its description will be defered until the next section.

4.2.3 Investigation of Pose Dependence

A manipulator's targets can be found in any of three regions: high manipulability region, singular region or out of reach region. Most iterative algorithms perform extremely well in the high manipulability region as the Jacobian provides a very good mapping between joint space and work space changes. However, the regions of real interest are singular regions and out of reach regions. In a singular region the Jacobian no longer provides a good mapping and out of reach regions do not permit an algorithm to drive its error to zero. Thus, these regions are a real test of an inverse kinematics algorithm's robustness whereas performance in the high manipulability region indicates an algorithm's typical behavior.

High Manipulability Region

The Puma manipulator was placed in the high manipulability pose with joint angles (0.0, -0.785, 3.14, 0.0, -0.785, 1.57) as depicted in Figure 4.11.

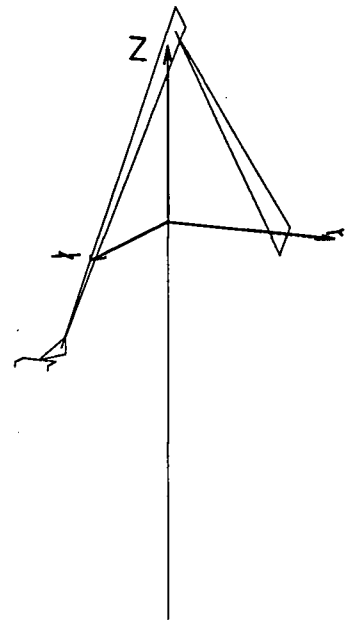

Figure 4.11: Puma in High Manipulability Pose

Two experimental approaches for generating random targets were used to evaluate the algorithm in the high manipulability region, one based in the work space the other based in the joint space. The work space experiments consisted of generating 100 random targets that lie on the surface of a sphere with radius 10cm, centered on the end-effector position given the start pose described above. Whereas, the joint space experiment consisted of generating 100 random targets by adding uniformly distributed random numbers with a specified range to the described start pose. Then, the target pose was converted from the joint space to the work space using the Puma's direct kinematics equations. Two ranges for the uniformly distributed random numbers were used $|0.1|$ radians and $|0.2|$ radians which were then converted to variances that could be used by the random number generation routine. For each experiment, the average number of iterations to reach the target was calculated. The results are shown in Table 4.4 and

confirm that this algorithm, like other iterative algorithms, performs well in the high manipulability region.

Target Type	Target Range	average number of iterations per solution
Work space	10cm	6.07
Joint space	0.1 rad	3.21
Joint space	0.2 rad	5.46

Table 4.4: Algorithm Average Solution Times in High Manipulability Region

Figures 4.12 and 4.13 show for a typical target how the error is reduced after each iteration as well as how the manipulator's joints are moved. Figure 4.13 shows that the commanded joint angles are smooth and monotonic.

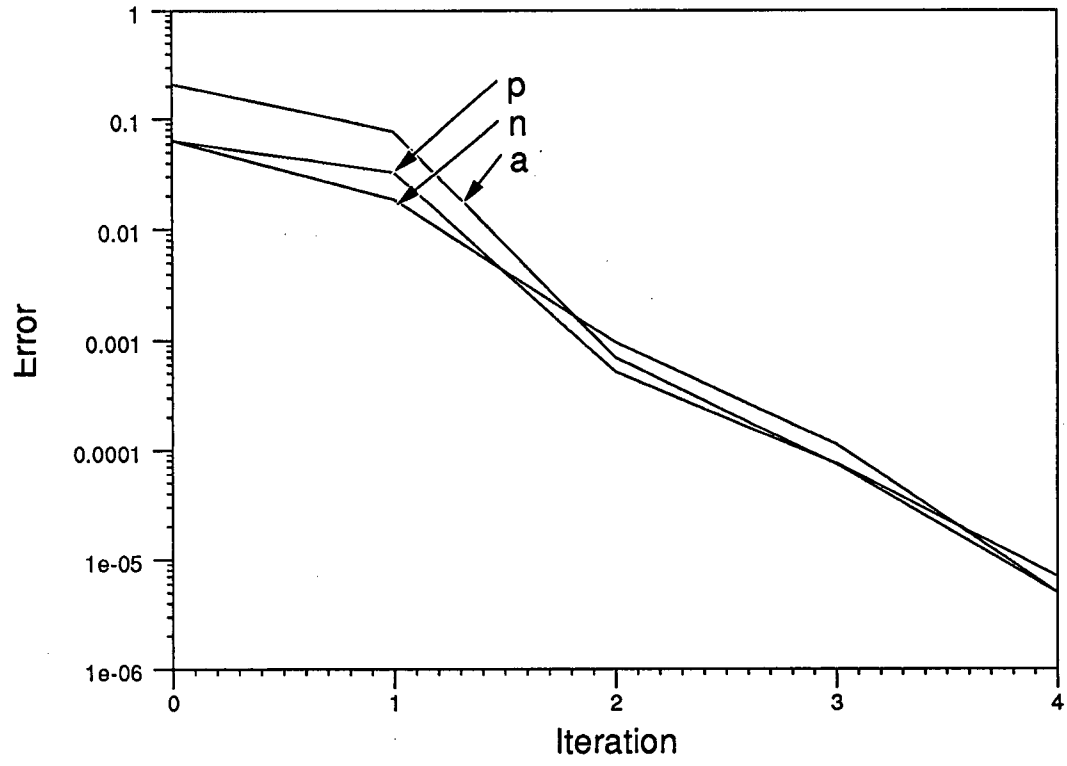

Figure 4.12: Error reduction - Puma in High Manipulability Pose

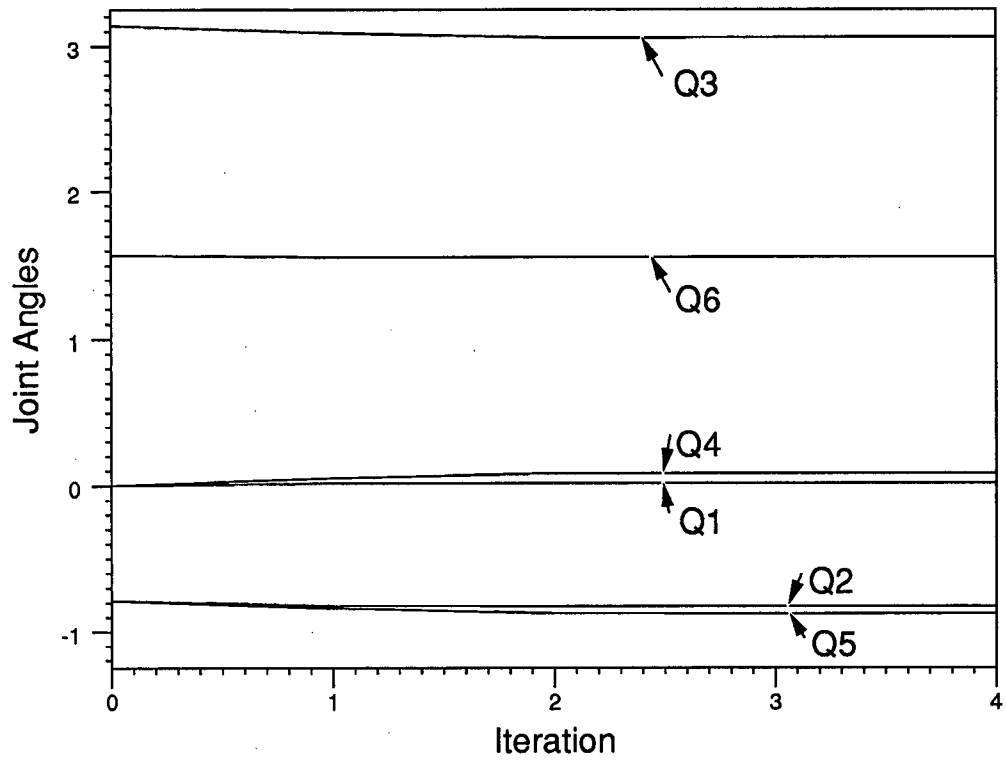

Figure 4.13: Joint Angles - Puma in High Manipulability Pose

Singular Region

The algorithm's performance in a singular region will now be examined. The Puma manipulator was placed in a pose such that both the wrist, joints 4 through 6, and the elbow, joint 3, are singular. Such a pose is the joint angle vector (0.0, 0.0, 1.57, 0.0, 0.0, 1.57) as depicted in Figure 4.14.

Iterative algorithms based on the Jacobian have great difficulty leaving from a singular pose as the Jacobian relationship no longer provides a good relationship between changes in the joint space and work space. The algorithm behavior on exitting a singular pose

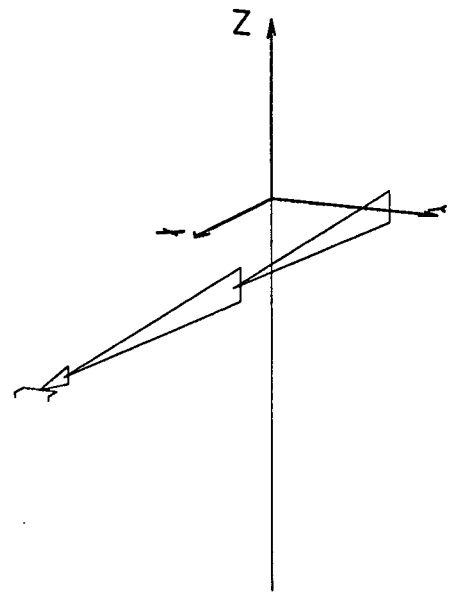

Figure 4.14: Puma in Singular Pose

was examined by the joint space experimental method described in the previous section. Using targets generated from joint space data, ensures that the targets are within the manipulator's reach. 100 targets were used. Two ranges of the random increments were used $|0.1|$ and $|0.2|$ radians which permits the average number of iterations per solution to be compared with the results from the high manipulability region. For completeness, the targets will be reused as start positions for a second set of experiments whose target is the singular pose described above allowing the properties of the algorithm approaching a singular pose to be studied. The data from these experiments is shown in Table 4.5.

Target Pose	Target Range	average number of iterations per solution
near singular	0.1 rad	18.27
near singular	0.2 rad	21.93
singular	0.1 rad	10.41
singular	0.2 rad	16.16

Table 4.5: Algorithm Average Solution Times in Singular Region

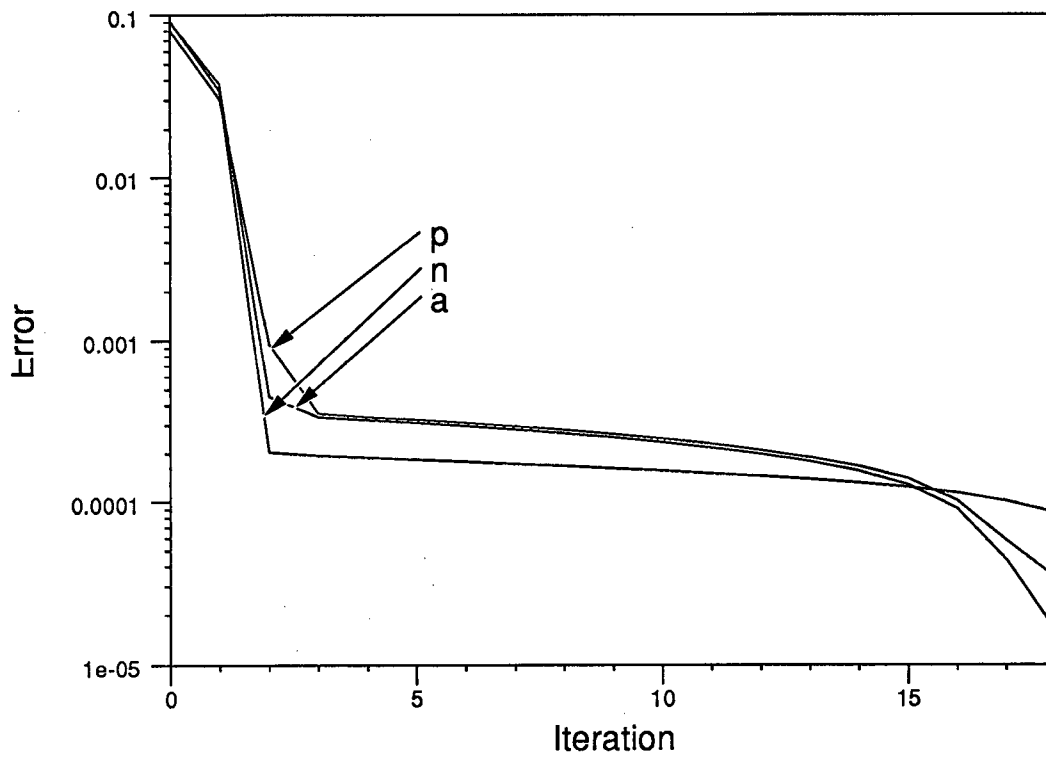

Figure 4.15: Error Reduction - Puma in Singular Pose

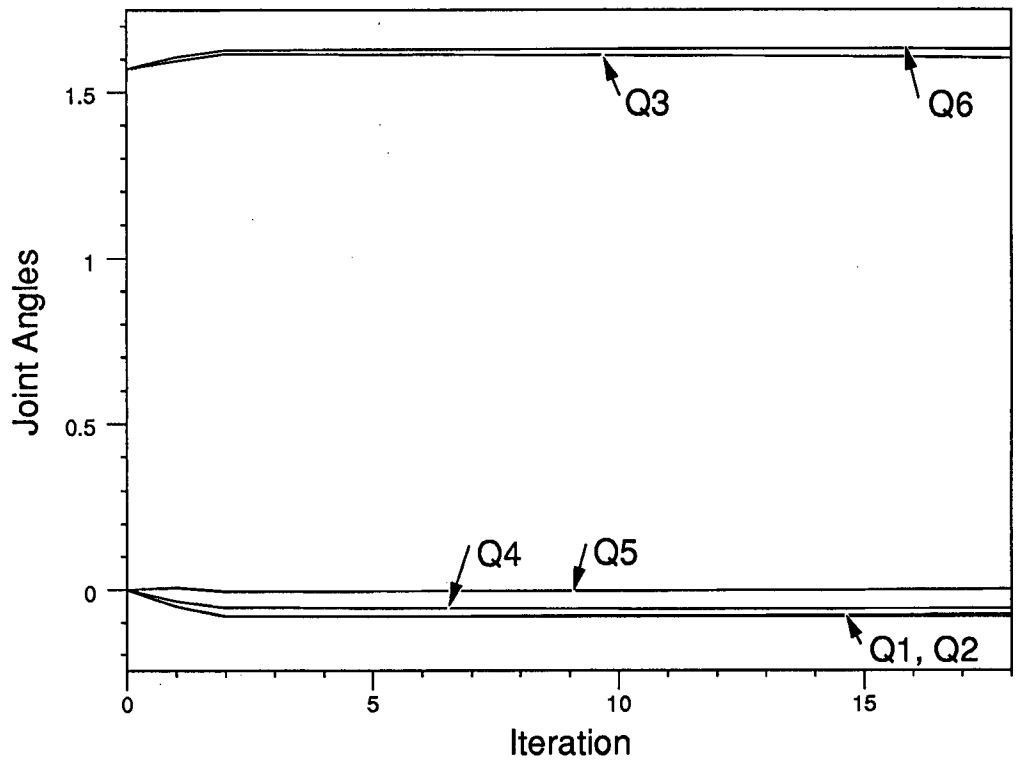

Figure 4.16: Joint Angles - Puma in Singular Pose

As expected the algorithm took longer to reach near singular targets than for high manipulability targets and the algorithm had more difficultly leaving a singular region than entering one. Since the damping on the estimate is more pronounced the closer the starting pose is to the singularity, the algorithm will take longer to find a solution when it starts at a singular pose and moves to a near singular pose. Conversely, if the manipulator starts near a singular pose the damping on the estimate will not be as great, therefore shorter solution times result despite the target being a singular pose.

Figures 4.15 and 4.16 show for a typical target how the error was reduced after each iteration as well as how the manipulator's joints were moved. It is seen that the error

initially decreases rapidly but then slows down as a result of the damping. Also, it is seen that the joint angles change in a smooth manner.

Out of Reach Region

The out of reach region encompasses two categories: targets that can not be reached by the manipulator without moving its base and targets that are unreachable due to restrictions placed on the manipulator's range of motion. If a target is beyond a manipulator's reach the best solution that can be obtained is one of minimal error. Likewise, if a joint limit is reached other solutions which do not violate the limits should be tried. The proposed algorithm is able to handle the first category by using a heuristic but is unable to handle the second category.

Targets Outside The Workspace

If the algorithm is to use no prior knowledge of the manipulator's geometry it can not directly determine when a target is out of reach. The only information that can be used to resolve this is the position error trend over time. When a target is out of reach, the position error can only be reduced to a minimum which is, in general, well above the desired tolerance. Unfortunately, the algorithm does not simply stop itself at such a minimum. The remaining distance to the target erroneously drives the algorithm to find a further minimum in a process shown in Figure 4.17 producing a position error that is oscillatory with time and does not terminate until the iteration limit is reached.

Ideally, for targets that are reachable, the position error should monotonically decrease until the target pose is reached which suggests that the solution would be to terminate the algorithm when the error is no longer non-decreasing. Unfortunately, in some cases the the error may actually increase before decreasing, a phenomenon resulting from $\Delta Q_{i_{max}}$ movement restrictions and errors in the inverse Jacobian estimate. Considering these

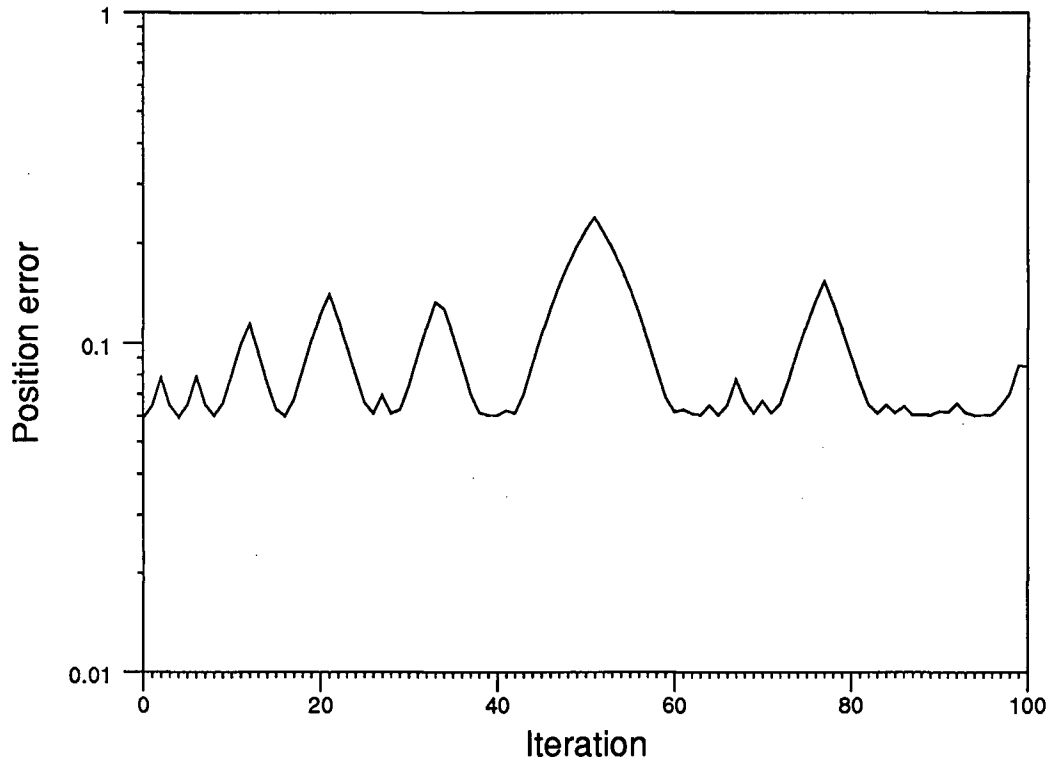

Figure 4.17: Position Error For an Out of Reach Target

vague conditions, a heuristic is needed to identify out of reach targets. The heuristic is: look for a minimum error that has not decreased for a set number of iterations. Once a minimum is detected look for a maximum. As with the detection of the minimum, the maximum must not increase for a set number of iterations. Once a maximum is detected the target is assumed to be out of reach. The purpose behind waiting a number of iterations before declaring a minimum or maximum is to filter out local, temporary extremum, however care must be taken in selecting the limits. Experiments show that 15 iterations for detecting a minimum and maximum was adequate. The heuristic was prone to false alarms as sometimes it would trigger for a target that is known to be

within the manipulator's reach. Such false alarms are rare and approximately occur for 2 targets out of 100 where the targets are in singular regions. Using the data presented in Figure 4.17 above as an example, a minimum would be detected on start up, after iteration 15 this would be confirmed and the heuristic would look for a maximum which does not decrease. Such a maximum was found at iteration 21 since no other error value was greater than it before the iteration limit of 15 was reached. An out of reach target was then detected at iteration 36 and the algorithm terminated.

Joint Limits

Although joint limits can be easily detected, how the algorithm should proceed beyond the detection stage is a considerably more difficult problem. The algorithm must have knowledge of the manipulator's geometry at least in the form of an approximate inverse kinematics solution to suggest what alternatives can be investigated. But, the proposed algorithm's very strength is that it needs no geometric information thereby ruling out this possibility. With the present algorithm implementation, once a joint limit was reached, an error remained between the current pose and the target and was treated as being outside the manipulator's reach. To sensibly handle joint limits, more information than what is currently used by this algorithm is required. A job which is best handled at a higher control level.

4.2.4 Investigation of Manipulator Dependence

Since no knowledge of the manipulator's geometry is required, the algorithm was readily portable and required no restructuring. How the algorithm's performance is dependent on what manipulator it is implemented on will be investigated in this section. Specifically, how the performance of the algorithm fares when the target manipulator is less than kinematically ideal is of interest. The results of two experiments will be presented here.

A Puma manipulator was degraded by adding successive offsets and displacing the twist angles from a multiple of $\frac{\pi}{2}$. Also, the algorithm's performance on a kinematically good manipulator (Puma) was compared with its performance on a kinematically poor one (Kodiak).

Degraded Puma

A series of degradations to a Puma manipulator were outlined by Joseph Poon [Poon88a] for defining how his inverse kinematics algorithm, Functional Joint Control, degrades for increasingly less ideal manipulators. How the proposed inverse kinematics algorithm behaves under such a set of degradations was examined. Manipulator 1 is the unmodified Puma manipulator with Denavit-Hartenberg parameters as described in Appendix A. Degradation was performed over five steps by adding the following changes to the Denavit-Hartenberg parameters.

a_i i=1,2,3 1cm/step

a_i i=4,5,6 1mm/step

α_i i=1,...,6 1deg/step

Two trajectories were studied. One in the high manipulability region with coordinates given in the joint space, starting joint angles (0.9, 0.4, 0.9, 0.9, 0.9, 0.9) to target joint angles (1.0, 0.5, 1.0, 1.0, 1.0, 1.0). The other trajectory was in a singular region as described by the joint coordinates, starting joint angles (0.0, 0.0, 1.57, 0.0, 0.0, 0.0) to target joint angles (0.1, 0.1, 1.4, 0.1, 0.1, 0.1).

Manipulator	singular pose	high manipulability pose
1	14	4
2	15	4
3	16	4
4	18	4
5	23	4
6	24	4

Table 4.6: Results from Puma Degradation Experiments

From Table 4.6 it is seen that the algorithm is not affected by the degradation when the trajectories are in the high manipulability region, but, it gradually and gracefully degrades for trajectories in the singular region.

Puma vs Kodiak

The procedure described in Section 4.2.3 was repeated using the Kodiak manipulator instead of the Puma in both the high manipulability and singular regions. Appendix A contains the Denavit-Hartenberg parameters for the Kodiak manipulator. The joint angles for the high manipulability pose are (0.0, 0.785, -1.57, 0.785, 1.57, -1.57) and for the singular pose (0.0, 0.0, 0.0, 0.0, 0.0, -1.57). As seen from Figure 4.18 the manipulators are in roughly the same poses allowing the performance of the algorithm on the two manipulators to be compared. Note that for the Kodiak's singular pose that its end effector needs to be at right angles to its forearm for the wrist to be singular. The results from the Kodiak and Puma are compared in Table 4.7.

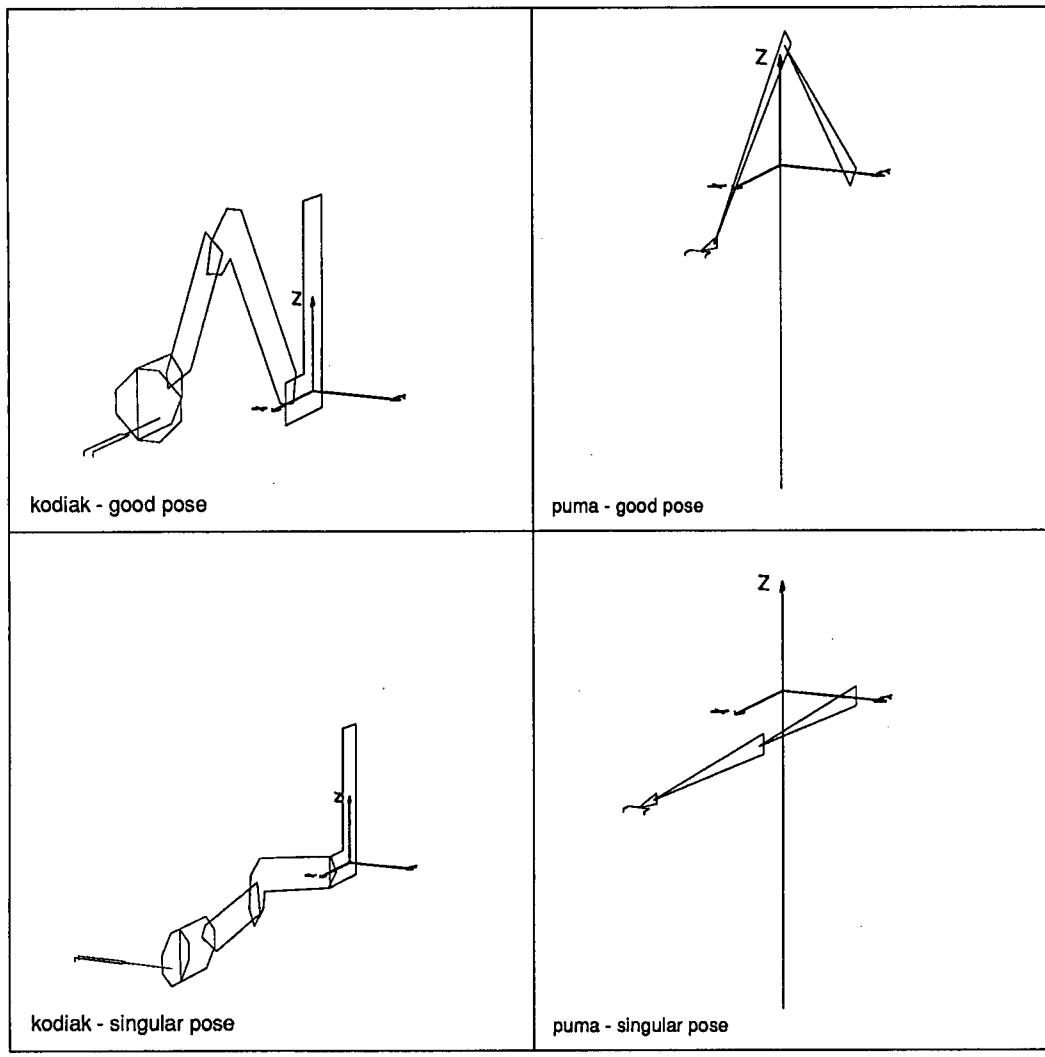

Figure 4.18: Comparision of Kodiak and Puma Poses Used

			Kodiak	Puma
Pose	Target Type	Target Range	average solution time (iterations)	average solution time (iterations)
good	Work Space	10 cm	6.04	6.07
	Joint Space	0.1 rad	3.40	3.21
	Joint Space	0.2 rad	5.65	5.46
sing to ns	Joint Space	0.1 rad	10.47	18.27
	Joint Space	0.2 rad	17.63	21.93
ns to sing	Joint Space	0.1 rad	7.31	10.41
	Joint Space	0.2 rad	12.30	16.16

Table 4.7: Comparison of Algorithm Performance Between the Puma and Kodiak

The performance of the algorithm is very similar for both manipulators about the high manipulability pose. As seen in Section 4.2.3 the algorithm has more trouble starting from a singular position than approaching one. It is interesting to note that the algorithm produced better results for the Kodiak in the singular region than for the Puma despite the Puma being more kinematically streamlined than the Kodiak. Since no modifications were made to the algorithm when the Kodiak replaced the Puma, and comparable results were obtained, the algorithm should be readily portable among various six degree of freedom manipulators.

4.3 Comparision To Other Researcher's Work

The inverse kinematics algorithms published in the open literature do not include examples with enough detail for a fair comparision to other algorithms. The computational requirements per iteration of the proposed algorithm will be compared with those of Wampler [Wample85]. This comparision is fairly sterile of implementation details however it does not address the issue of how many iterations are required to obtain a solution. The number of iterations required to obtain a solution for various manipulator poses will

be achieved by using the examples cited in the published work of my colleages Joseph Poon [Poon88a, Poon88b] and Stephen Chan [Chan87].

4.3.1 Computations per Iteration

The computations per iteration of Wampler's Damped Least Squares using the vector method of calculating the Jacobian are compared with the proposed algorithm. The operations compared are $\times, \pm, \div$. Wampler does not list the calls to transcendental functions which are required for his coordinate transformations. At most he would need 2n sine and cosine calls. The proposed algorithm makes no transcendental function calls.

Algorithm	$\times$	$\pm$	$\div$
Wampler's	$\frac{1}{6}n^3 + \frac{15}{2}n^2 + \frac{49}{3}n - 8$	$\frac{1}{6}n^3 + \frac{21}{4}n^2 + \frac{151}{12}n - 5$	n
Proposed	$3mn + \frac{3}{2}n^2 + \frac{5}{2}n + 7$	$3mn + \frac{3}{2}n^2 - m + 5$	$3n$
Wampler's $n = 6$	395	296	6
Proposed $m = 6$, $n = 7$	224	216	21

Table 4.8: Comparison of Calculations per Iteration Between the Proposed Algorithm and Wampler's

In table 4.8 the number of computations as a function of the number of joints n is given. The quantity m reflects the number of work space vector components. In Wampler's case m is the same as n and equals 6, however because the proposed algorithm uses Euler parameters to describe the orientation $m = 7$. From table 4.8 it is seen that the number of multiplications required by the proposed algorithm is a little more than half of that required by Wampler's algorithm. It is also seen that the proposed algorithm requires more divisions which is a more time expensive operation than multiplication. If it is assumed that a division takes twice as long as a multiply then the proposed algorithm's computation still requires less time to compute.

4.3.2 Functional Joint Control

Joseph Poon's published examples [Poon88a, Poon88b] were used in Section 4.2.4 for examining how the proposed algorithm behaved when the Puma manipulator was degraded. The data from that section is repeated here and compared with the data obtained from using Functional Joint Control as shown in Tables 4.9 and 4.10.

Manipulator	RLSE			FJC		
	×	±	÷	×	±	÷
1	3136	3024	294	920	720	210
2	3360	3240	315	1288	1008	294
3	3584	3456	336	1196	936	273
4	4032	3888	378	1288	1008	294
5	5152	4968	483	2116	1656	483
6	5376	5184	504	—	—	—

Table 4.9: Comparision of RLSE with FJC - Singular Region

Manipulator	RLSE			FJC		
	×	±	÷	×	±	÷
1	896	864	84	1012	792	231
2	896	864	84	1196	936	273
3	896	864	84	1196	936	273
4	896	864	84	1196	936	273
5	896	864	84	1196	936	273
6	896	864	84	1196	936	273

Table 4.10: Comparision of RLSE with FJC - High Manipulability Region

As shown in Table 4.10 the proposed algorithm fares much better in the high manipulability region since the solution provides coordinated control of the joints whereas Functional Joint Control controls each joint independently. Although the proposed algorithm's performance degrades in the singular region it does so gracefully and is able to provide a

solution for manipulator 6 where Functional Joint Control could not. However it requires more operations to obtain a solution in the singular region than Functional Joint Control as seen in Table 4.9.

4.3.3 Damped Least Squares with Variable Damping

Stephen Chan has published examples [Chan87] that can also be used for comparision. There were two trajectories for the Kodiak manipulator, both involving singular regions with joint angles (1, -1, 1, 0, 1, 1) to (0, 0, 0, 0, 1, 1) and (0, 0, 1, 0, 1, 0) to (-1, -1, 1, 1, 0, 0). The distance between the start and target poses are too large to be efficiently handled by the proposed algorithm whose performance suffers due to the necessity of limiting the maximum change per joint. The error tolerances used for these trajectories were the same as those used by Chan, 1mm and 0.001 radians.

Trajectory	RLSE			DLS with Variable Damping		
	×	±	÷	×	±	÷
singular elbow	17696	17064	1659	2765	2072	42
singular wrist	6720	6480	126	6320	4736	96

Table 4.11: Comparision of RLSE with Damped Least Squares with Variable Damping

As seen in Table 4.11 the proposed algorithm had difficulty solving for the singular elbow target and the path that the end-effector followed was erratic. The algorithm made a series of mistakes before it finally reached the target. But, the algorithm had no difficulties with the singular wrist target which was the greater distance of the two. These results support the assumption that the distance between the start and target should be relatively small. In terms of performance, the Damped Least Squares with Variable Damping algorithm can be expected to perform better than the proposed algorithm as the estimated result will inevitably have some error.

Chapter 5

Conclusions

5.1 Summary

An iterative inverse kinematics algorithm based on the Jacobian has been presented. Although the Jacobian can be calculated for any manipulator, it must be explicitly derived for each manipulator to which the algorithm is applied. Furthermore, most Jacobian based algorithms are computationally intensive because the Jacobian must be calculated, stabilized, and then inverted for each iteration step. In this thesis a Jacobian based algorithm was developed which utilizes recursive least squares estimation on joint space/work space data pairs to estimate the inverse Jacobian. Estimating the inverse Jacobian places the calculation, stabilization and inversion of the Jacobian into one step which is computationally compact and efficient.

Showing that estimation accomplishes these three steps, especially stabilization, is difficult. Given joint space and work space data explicitly related by the Jacobian, it has been shown that the inverse Jacobian results from least squares estimation. Data from the two spatial regions was then obtained by mapping randomly generated joint space changes onto their corresponding work space changes using the current manipulator pose and the direct kinematics rather than the Jacobian. With data derived from this process on a one link manipulator, and a simple estimation procedure, the Jacobian was estimated and its inverse determined. The resulting inverse was similar to Wampler's damped least squares method which suggests that if estimation was performed using joint

space/work space changes with constrained joint space changes, the estimated inverse would be damped. This supposition was investigated on a two link manipulator by comparing, over the same spatial range, the recursive least squares estimate and damped least squares condition numbers, as well as the inverse Jacobian parameters. For both methods it was observed that the condition number and the inverse Jacobian parameters had identical traits suggesting equivalency.

It has been demonstrated that when recursive least squares estimation is used in this application, the estimator must be properly initialized and the forgetting factor, λ, must be carefully selected to ensure that the estimator properly tracks the Jacobian. To further aid estimator tracking a sensible maximum joint change per iteration must be defined which, when exceeded, will cause the estimator to update.

The proposed inverse kinematics algorithm, incorporating the estimator, was tested for a number of random targets at different manipulator poses, both high manipulablity and singular. A heuristic for out of reach targets was presented. The proposed algorithm's portability to other manipulators was investigated, both by transfering the unmodified algorithm to the Kodiak manipulator and by degrading the Puma manipulator over a series of steps. Finally, the proposed algorithm was compared to the work of other researcher's.

5.2 Conclusions

The proposed algorithm presented in this thesis shows that Lobbezoo *et al*'s [Lobbez88] results, developed for a two link manipulator, can be extended to six degree of freedom manipulators provided that a non-singular orientation method is used.

It has also been shown that the algorithm based on recursive least squares estimation with joint space limiting has properties similar to Wampler's damped least squares

method but is more computationally efficient. However, the proposed algorithm's solution time markedly degrades for large start and target pose separations ($>$ 10 cm, 0.5 rads).

The proposed algorithm was successfully applied, unmodified, to the Kodiak manipulator which suggests, in light of the minimal information required by the estimator, that it can be readily ported among manipulators.

Since the inverse Jacobian's parameters vary according to the manipulator's pose, the estimator must be able to track these changes. This problem can be addressed by forcing an update once a preset maximum change per joint is exceeded and the proposed algorithm is adaptable to structural changes provided that the end effector information is obtained by a sensor system. Adaptability is evident in the algorithm's performance when successive offsets are added to the Puma manipulator.

Furthermore, a robust, useable inverse kinematics solution can be determined by the proposed algorithm throughout the manipulator's work space albeit the performance is degraded in singular regions. This solution is a cohesive unit, independent of other robotics problems such as path planning. When compared to Functional Joint Control, the proposed algorithm performs better in the high manipulablitiy region and produces comparable results in the singular region.

5.3 Recommendations

Methods for improving the MIMO estimator's tracking ability should be explored. Currently, the algorithm's performance is limited by the fixed maximum joint change per iteration needed in singular regions. If this constraint could be optimized according to the manipulator's pose, for instance a greater maximum for high manipulability regions and a smaller one for singular regions, the algorithm's performance would be improved.

A solution to this problem could be obtained by monitoring the manipulator's pose by a suitable manipulability measure that could be readily determined on-line, such as a computationally efficient singular value decomposition algorithm. Such a manipulability measure could then act as a control input to alter the maximum joint change per iteration. The MIMO estimator's tracking performance can also be altered by the forgetting factor or the error covariance matrix.

While this simulation work has suggested some interesting results, it should be confirmed by implementing the algorithm on an actual manipulator with a vison system. An implementation would characterize what speeds the algorithm can achieve, whether an on-line solution is viable using currently available computing hardware and what omissions from the algorithm need to be considered.

References

[Agee72] Agee, W. S. and Turner, R. H., "Triangular Decomposition of a Positive Definite Matrix Plus a Symmetric Dyad With Application to Kalman Filtering", White Sands Missile Range Technical Report No. 38, 1972.

[Åström77] Åström, K. J., Borisson, U., Ljung, L., and Wittenmark, B.,"Theory and Applications of Self-Tuning Regulators", *Automatica*, Vol. 13, pp. 457–476, 1977.

[Bejczy83] Bejczy, A. K., and Handlykken, M., "Generalization of Bilateral Force-Reflecting Control of Manipulators" 4^{TH} *Symposium on Theory and Practice of Robots and Manipulators, CISM-IFToMM*, 1983, pp. 242-255.

[Bierma77] Bierman, G. J., *Factorization Methods for Discrete Sequential Estimation*, New York, Academic Press, 1977.

[Carlso73] Carlson, N. A., "Fast Triangular Formulation of the Square Root Filter", AIAA Journal, Vol. 11, No. 9, 1973, pp. 1259-1265.

[Chan87] Chan, S. K. C., "An Iterative General Inverse Kinematics Solution with Variable Damping" M. A. Sc. Thesis, Department of Electrical Engineering, University of British Columbia, 1987.

[Chen87] Chen, M. J., and Norton, J. P., "Estimation technique for tracking rapid parameter changes", *International Journal of Control*, Vol. 45, No. 4, 1987, pp. 1387–1398

[Denavi55] Denavit, J., and Hartenberg, R. S., "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices", *ASME Journal of Applied Mechanics*, June 1955, pp. 215-221.

[Euler] Euler, L., Formulea generales pro translatione quacunque corporum rigidorum, *Imperatorskaia akademiia nauk* (Russia) *Novi commentarii Academiae Scientiarum Imperialis Petropolitane* 20, 189-207, 1775/1776.

[Eykhof74] Eykhoff, P., *System Identification*, New York, Wiley, 1974.

[Feathe83] Featherstone, R., "Position and Velocity Transformations Between Robot End Effector Coordinates and Joint Angles" *The International Journal of Robotics Research*, Vol. 2, No. 2, 1983, pp. 35-45.

[Foley82] Foley, J. D., and Van Dam, A., *Fundamentals of Iteractive Computer Graphics*, Addison-Wesley, Reading, Massachusetts, 1982.

[Gentle73] Gentleman, W. M., "Least Squares Computations by Givens Transformations Without Square Roots", *J. Inst. Maths Applics*, Vol. 12, 1973, pp. 329-336.

[Golden85] Goldenberg, A. A., Benhabib, B., and Fenton, R. G., "A Complete Generalized Solution to the Inverse Kinematics of Robotics", *IEEE Journal of Robotics and Automation*, Vol. RA-1, No. 1, March 1985, pp. 14-20.

[Golub83] Golub, G. H.,and Van Loan, C. F., *Matrix Computations*, The Johns Hopkins University Press, Balitmore, Maryland, 1983, p 27.

[Gupta86a] Gupta, K.C., "Kinematic Analysis of Manipulators Using the Zero Reference Position Description" *The International Journal of Robotics Research*, Vol. 5, No. 2, 1986, pp. 5-13.

[Gupta86b] Gupta, K. C., "On the Nature of Robot Workspace", *The International Journal of Robotics Research*, Vol. 5, No. 2, 1986, pp. 112-121.

[Ishii87] Ishii, M., Sakane, S., Kakikura, and M., Mikami, Y., "A 3-D Sensor System for Teaching Robot Paths and Environments", *The International Journal of Robotics Research*, Vol. 6, No. 2, 1987, pp. 45-59.

[Kalman63] Kalman, R. E., "New Methods in Wiener Filtering Theory", *In Proc. 1^{ST} Symp. Eng. Appl. Random Function Theory Probability*, J.L. Bogdanoff and F. Kozin eds., Wiley, New York, 1963, pp. 270-388.

[Kamins71] Kaminski, P. G., Bryson, A. E., and Schmidt, S. F. "Discrete Square Root Filtering: A Survey of Current Techniques", *IEEE Transactions on Automatic Control*, Vol. AC-16, No. 6, 1971, pp. 727-735.

[Kane83] Kane, T. R., Likins, P. W., and Levinson, D. A., *Spacecraft Dynamics*, McGraw-Hill, New York, 1983.

[Klein87] Klein, C. A., and Blaho, B. E.,"Dexterity Measures for the Design and Control of Kinematically Redundant Manipulators", *The International Journal of Robotics Research*, Vol. 6, No. 2, 1987, pp. 72-83.

[Klema80] Klema, V. C., and Laub, A. J., "The Singular Value Decomposition: Its Computation and Some Applications", *IEEE Transactions on Automatic Control*, Vol. AC-25, No. 2, 1980, pp. 164-176.

[Korein85] Korein, J. U., *A Geometric Investigation of Reach*, The MIT Press, Cambridge, Massachusetts, 1985.

[Kumar86] Kumar, A., and Patel, M. S.,"Mapping the Manipulator Workspace Using Interactive Computer Graphics",*The International Journal of Robotics Research*, Vol. 5, No. 2, 1986, pp. 122-130.

[Litvin86] Litvin, F. L., Yi, Z., Castelli, V. P., and Innocenti, C., "Singularities, Configurations, and Displacement Functions for Manipulators", *The International Journal of Robotics Research*, Vol. 5, No. 2, Summer 1986, pp. 52-65.

[Lai86] Lai, Z. C., and Yang, D. C. H., "A New Method for the Singularity Analysis of Simple Six-link Manipulators", *The International Journal of Robotics Research*, Vol. 5, No. 2, Summer 1986, pp. 66-74.

[Lawson74] Lawson, C. L. and Hanson, R. J., *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, N.J., 1974.

[Ljung83] Ljung, L., and Söderström, T., *Theory and Practice of Recursive Identification*, Cambridge, Massachusetts, The MIT Press, 1983.

[Lobbez88] Lobbezoo, A. J., Bruijn, P. M., Davies, M. S., Dunford, W. G., Lawrence, P. D., and van Nauta Lemke, H. R., "Robot Control Using Adaptive Transformations", *IEEE Journal of Robotics and Automation*, Vol. 4, No. 1, 1988, pp. 104-108.

[Milenk83] Milenkovic, V., and Huang, B. "Kinematics of Major Robot Linkage" 13^{TH} *International Symposium on Industrial Robots and Robots 7*, Vol. 2, 1983, pp. 16.31-16.47.

[Milenk84] Milenkovic, V., "Effect of robot wrist singularity on path control", *14th International Symposium on Industrial Robots, 7th International Conference on Industrial Robot Technology*, 1984, pp. 279-286.

[Nakamu86] Nakamura, Y., and Hanafusa, H., "Inverse Kinematic Solutions With Singularity Robustness for Robot Manipulator Control", *ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 108, 1986, pp. 163-171.

[Paul81] Paul, R. P., *Robot Manipulators*, The MIT Press, Cambridge, Massachusetts, 1981.

[Paul83] Paul, R. P., and Stevenson, C. N., "Kinematics of Robot Wrists", *The International Journal of Robotics Research*, Vol. 2, No. 1, 1983, pp. 31-38.

[Pieper69] Pieper, D. L., *The Kinematics of Manipulators Under Computer Control*, Ph.D. Thesis, Stanford University, 1969.

[Poon88a] Poon, J. K., *Multiprocessor-compatible Inverse Kinematics and Path Planning For Robots*, Ph.D. Thesis, Department of Electrical Engineering, University of British Columbia, 1988.

[Poon88b] Poon, J. K., and Lawrence, P. D., "Manipulator Inverse Kinematics Based on Joint Functions", *The 1988 IEEE International Conference on Robotics and Automation*, 1988, pp. 669-674.

[Potter64] Battin, R. H., *Astronautical Guidance*, McGraw-Hill, New York, 1964, pp. 338-339.

[Powell70] Rabinowitz, P., *Numerical Methods for Non-linear Algebraic Equations*, Gordon and Breach, New York, 1970, pp. 87-114.

[Spring86] Spring, K. W., "Euler Parameters and the Use of Quaternion Algebra in the Manipulation of Finite Rotations: A Review" *Mechanism and Machine Theory*, Vol. 21, No. 5, 1986, pp. 365-373.

[Stewar77] Stewart, G. W., "On the Perturbation of Pseudo-inverses, Projections and Linear Least Squares Problems", SIAM Review, Vol. 19, No. 4, 1977, pp. 634-662.

[Toivon77] Toivonen, H., and Westerlund, T., "The Identification of Linear, Discrete Time Multivariable System By the Least-squares Method", *Sähkö - Electricity in Finland* 50, No. 11, , 1977, pp. 387-394.

[Trevel86] Trevelyan, J. P., Kovesi, P. D., Ong, M., and Elford, D., "ET: A Wrist without Singular Positions", *The International Journal of Robotics Research*, Vol. 4, No. 4, 1986, pp. 71-85.

[Uchiya79] Uchiyama, M., "A Study of Computer Control of Motion of a Mechanical Arm - Report 1", *Bulletin of the JSME*, Vol. 22, No. 173, 1979, pp. 1640-1647.

[Vijayk86] Vijaykumar, R., Waldron, K., and Tsai, M., "Geometric Optimization of Serial Chain Manipulator Structures for Working Volume and Dexterity" *The International Journal of Robotics Research*, Vol. 5, No. 2, 1986, pp. 91-103.

[Wample85] Wampler, C. W., "Computer Methods In Manipulator Kinematics, Dynamics, and Control: A Comparative Study", Ph. D. thesis, Stanford

University, 1985.

[Wample88] Wampler, C. W., and Leifer, L. J., "Applications of Damped Least Squares Methods to Resolved-Rate and Resolved-Acceleration Control of Manipulators", *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 110, 1988, pp. 31-38.

[Whitne69] Whitney, D. E., "Resolved Motion Rate Control of Manipulators and Human Prostheses", *IEEE Transactions on Man-Machine Systems*, Vol. MMS-10, No. 2, 1969, pp. 47-53.

[Whitne72] Whitney, D. E., "The Mathematics of Coordinated Control of Prosthetic Arms and Manipulators", *ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 122, 1972, pp. 303-309.

[Yoshik85] Yoshikawa, T.,"Manipulability of Robotic Mechanisms", *The International Journal of Robotics Research*, Vol. 4, No. 2, 1985, pp. 3-9.

Appendix A

Denavit-Hartenberg Parameters

The Denavit-Hartenberg parameters and coordinate frame assignments for the Two Link, Puma, and Kodiak manipulators discussed in this thesis are presented below. The manipulators are shown in their "zero position", where all joint angles and distances are zero.

A.1 Two Link Manipulator

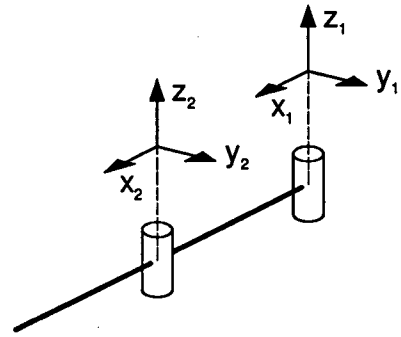

Figure A.2: Coordinate Frames of a Two Link Manipulator

Joint	α_i (rad)	a_i (m)	d_i (m)
1	0	1	0
2	0	1	0

Table A.2: Two Link Denavit-Hartenberg Parameters

A.2 Puma Manipulator

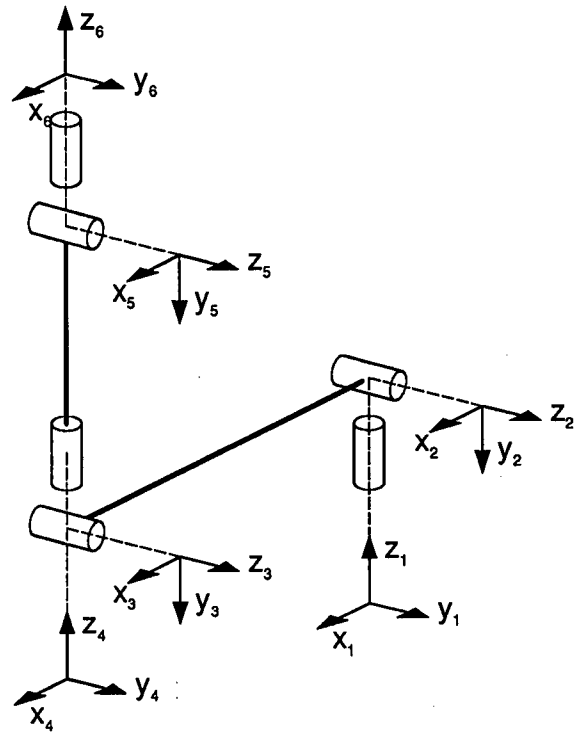

Figure A.3: Coordinate Frames of a Puma Manipulator

Joint	α_i (rad)	a_i (m)	d_i (m)
1	-90.0	0.0	0.0
2	0.0	0.432	0.1495
3	90.0	0.0	0.0
4	-90.0	0.0	0.432
5	90.0	0.0	0.0
6	0.0	0.0	0.0565

Table A.3: Puma Denavit-Hartenberg Parameters

A.3 Kodiak Manipulator

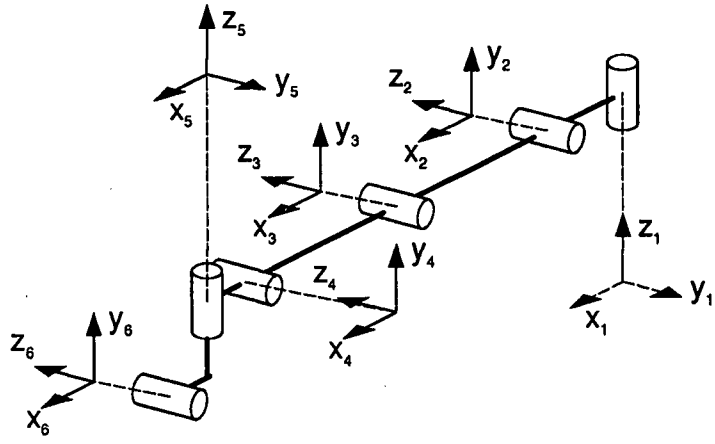

Figure A.4: Coordinate Frames of a Kodiak Manipulator

Joint	α_i	a_i	d_i
1	90.0	0.10	0.0
2	0.0	0.56	0.0
3	0.0	0.305	0.0
4	-90.0	0.15	0.0
5	90.0	0.05	-0.05
6	0.0	0.0	0.3

Table A.4: Kodiak Denavit-Hartenberg Parameters

Appendix B

Direction Cosine - Euler Parameter Conversions

The source for two functions, noa_EulerParam(), EulerParam_noa(), written in the programming language **c** are given in this appendix. noa_EulerParam() converts orientation expressed in $\vec{n}, \vec{o}, \vec{a}$ parameters to Euler parameters. Likewise, EulerParam_noa() converts orientation expressed in Euler parameters to $\vec{n}, \vec{o}, \vec{a}$ parameters. Because Euler parameters have two solutions per orientation representation, the solution closest to the target orientation was consistently selected. Changing from one solution to the other is accomplished by simply changing the sign of the parameters and is performed outside these routines.

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

#define  N  0
#define  O  1
#define  A  2

#define  X  0
#define  Y  1
#define  Z  2

#define  PHI  0
#define  THETA 1
#define  SI 2

#define  SOLN1  0
#define  SOLN2  1
#define  SOLN3  2
#define  SOLN4  3
```

B.1 noa_EulerParam()

```
/*
    noa_EulerParam()

    This function converts the Direction Cosine portion of the
    noap matrix (4x4) to Euler Parameters.  The solutions used
    originate from C. W. Wampler's Ph.D. thesis "Computer Methods in
    Manipulator Kinematics, Dynamics, and Control: A Comparative Study"
    pp. 34-36.  There are four solutions that can be used to
    find the Euler Parameters.

    Solution Used

    From Table 4.3 of Wampler

  solution 1       solution 2       solution 3  solution 4

  e1       del1/4       (C12+C21)/del2  (C31+C13)/del3  (C32-C23)/del4
  e2   (C12+C21)/del1       del2/4      (C23+C32)/del3  (C13-C31)/del4
  e3   (C31+C13)/del1   (C23+C32)/del2      del3/4      (C21-C12)/del4
  e4   (C32-C23)/del1   (C13-C31)/del2  (C21-C12)/del3      del4/4

    where
          del1 = +/- 2*sqrt(1 + C11 - C22 - C33)
          del2 = +/- 2*sqrt(1 - C11 + C22 - C33)
          del3 = +/- 2*sqrt(1 - C11 - C22 + C33)
          del4 = +/- 2*sqrt(1 + C11 + C22 + C33)

          e1, e2, e3, e4 are the respective Euler Parameters

         | C11  C12  C13 |     | Nx  Ox  Ax |
         | C21  C22  C23 |  =  | Ny  Oy  Ay |
         | C31  C32  C33 |     | Nz  Oz  Az |

    The four values of del are calculated (del1, del2, del3, del4)
    and the largest del i defines which solution number is selected.
    The solution is calculated.

    Parameters Passed

    noa[4][4] - 4x4 Dennavit-Hartenberg Ai or T matrix
    ep        - 4 element Euler parameter vector (order e1,e2,e3,e4)
*/
```

```
noa_EulerParam(noa, ep)
double noa[4][4];
double ep[];
{
  double del[4], MaxDel;
  register int i;
  int solution;

  del[0] = (1 + noa[X][N] - noa[Y][O] - noa[Z][A]);
  del[1] = (1 - noa[X][N] + noa[Y][O] - noa[Z][A]);
  del[2] = (1 - noa[X][N] - noa[Y][O] + noa[Z][A]);
  del[3] = (1 + noa[X][N] + noa[Y][O] + noa[Z][A]);

  MaxDel = -1.0;
  solution = -1;
  for (i=0; i < 4; i++)
  {
    if (del[i] <= 0.0)
      del[i] = 0.0;
    else
      del[i] = 2*sqrt(del[i]);

    if ( MaxDel < del[i])
    {
      MaxDel = del[i];
      solution = i;
    }
  }

  switch(solution)
  {
    case SOLN1:
      ep[0] = MaxDel/4;
      ep[1] = (noa[X][O] + noa[Y][N])/MaxDel;
      ep[2] = (noa[Z][N] + noa[X][A])/MaxDel;
      ep[3] = (noa[Z][O] - noa[Y][A])/MaxDel;
      break;
    case SOLN2:
      ep[0] = (noa[X][O] + noa[Y][N])/MaxDel;
      ep[1] = MaxDel/4;
      ep[2] = (noa[Y][A] + noa[Z][O])/MaxDel;
      ep[3] = (noa[X][A] - noa[Z][N])/MaxDel;
      break;
    case SOLN3:
      ep[0] = (noa[Z][N] + noa[X][A])/MaxDel;
      ep[1] = (noa[Y][A] + noa[Z][O])/MaxDel;
      ep[2] = MaxDel/4;
      ep[3] = (noa[Y][N] - noa[X][O])/MaxDel;
```

```
      break;
    case SOLN4:
      ep[0] = (noa[Z][O] - noa[Y][A])/MaxDel;
      ep[1] = (noa[X][A] - noa[Z][N])/MaxDel;
      ep[2] = (noa[Y][N] - noa[X][O])/MaxDel;
      ep[3] = MaxDel/4;
      break;
    default:
      fprintf(stderr,
              "ERROR: noa_EulerParam() results in divide by zero\n");
      exit(1);
      break;
  }
}
```

B.2 EulerParam_noa()

```
/*
    EulerParam_noa()

    This function converts Euler Parameters ( e1, e2, e3, e4) to
    the Direction Cosine matrix used in Dennavit-Hartenberg Ai or Ti
    matrices (noap).  The solution used originates from C. W. Wampler's
    Ph.D. thesis "Computer Methods in Manipulator Kinematics, Dynamics,
    and Control: A Comparative Study" pp. 34-36.

    Solution Used

    Equation (29) of p 35.

 Nx = 1-2(e2^2+e3^2)  Ox = 2(e1e2-e3e4)    Ax = 2(e3e1+e2e4)
 Ny = 2(e1e2+e3e4)    Oy = 1-2(e3^2+e1^2) Ay = 2(e2e3-e1e4)
 Nz = 2(e3e1-e2e4)    Oz = 2(e2e3+e1e4)   Az = 1-2(e1^2+e2^2)

    Parameters Passed

    ep          - pointer or 4 element vector (order e1,e2,e3,e4)
    noa[4][4]   - 4x4 Dennavit-Hartenberg Ai or T matrix
 */

EulerParam_noa(ep,noa)
double *ep;
double noa[4][4];
{
  double e1, e2, e3, e4;

  e1 = *ep;
  e2 = *(ep+1);
  e3 = *(ep+2);
  e4 = *(ep+3);

  noa[X][N] = 1 - 2*(e2*e2 + e3*e3);
  noa[Y][N] = 2*(e1*e2 + e3*e4);
  noa[Z][N] = 2*(e3*e1 - e2*e4);

  noa[X][O] = 2*(e1*e2 - e3*e4);
  noa[Y][O] = 1 - 2*(e3*e3 + e1*e1);
  noa[Z][O] = 2*(e2*e3 + e1*e4);

  noa[X][A] = 2*(e3*e1 + e2*e4);
  noa[Y][A] = 2*(e2*e3 - e1*e4);
  noa[Z][A] = 1 - 2*(e1*e1 + e2*e2);
} /* EulerParam_noa() */
```