Geometric Modelling and Accuracy Enhancement of Parallel Manipulators

Andrew B Lintott

A thesis presented for the degree of Doctor of Philosophy in Mechanical Engineering at the University of Canterbury, Christchurch, New Zealand.

1 February 2000
To my parents.
(Top) The University of Canterbury Delta Robot. (Bottom) The Kiwibot with Tim Jones (left), Andrew Cree, and Andrew Lintott (right).
CONTENTS

2.4.3 Open Loop Derivative with Respect to All Geometric Parameters . 31
2.4.4 Verification of Open Loop Jacobian .......................... 35
2.5 Closed Loop Jacobian ........................................... 35
2.5.1 Chain Rule Method ......................................... 35
2.5.2 Nullspace Decomposition Method ............................ 40
2.6 Nominal Geometric Model for the Delta Robot .................. 45
2.6.1 Direct Geometric Solution ................................. 46
2.6.2 Geometric Solution of R. Clavel .......................... 46
2.6.3 Numerical Stability of Clavel’s Solution .................. 48
2.6.4 A Variation on Clavel’s Geometric Solution ............... 49
2.6.2 Inverse Geometric Solution .............................. 50

3 The Calibration Problem ........................................ 53
3.1 Identification .................................................. 54
3.2 Implementation ................................................ 55
3.3 A Survey of Non-Linear Minimisation Techniques ............... 55
3.3.1 Line Search .................................................. 55
3.3.2 Bounding and Non-Feasible Points ......................... 56
3.3.3 Non-Unique Solutions ..................................... 57
3.3.4 Least Squares Problems ................................... 58
3.3.5 Newton’s Method ........................................... 58
3.3.6 Gauss-Newton ............................................. 59
3.3.7 Quasi-Newton (Variable Metric) ......................... 59
3.3.8 Levenberg-Marquardt ...................................... 60
3.4 Application of Non-Linear Minimisation Techniques ........... 60
3.5 The Influence of Noise on the Solution ....................... 63
3.5.1 Linear Error Analysis .................................... 64
3.5.2 Interpretation of Covariance and Confidence Intervals ... 64
3.5.3 Testing the Assumption of Normality ...................... 66

4 Application ......................................................... 69
4.1 Introduction .................................................... 69
4.2 Design of System Components ................................ 69
4.2.1 Implementation of Modelling, Visualisation, and Calibration
4.2.2 Mechanical Design of the Canterbury University Delta Robot 71
4.2.1 Major Components .......................................... 75
4.2.2 Support Structures ......................................... 75
4.2.1 Base Platform ............................................... 76
4.2.2 Reduction Housing ......................................... 77
4.2.2 Actuated Arms ............................................. 78
4.2.2 Ball Joints .................................................. 78
4.2.2 Parallelogram Links ....................................... 79
4.2.2 Nacelle ..................................................... 79
4.2.3 Control System Design .................................... 79
List of Figures

1.1 Cartoon exaggeration of the mobile mass problem. .................................. 2
1.2 Large Stewart Platform for automobile driving simulation. ...................... 2
1.3 Representation of joint space and task space parameters and the geometric solutions used to convert between them. ................................. 3
1.4 Kinematic arrangement of the 3-dimensional 4-bar closed loop mechanism. 17
1.5 Kinematic arrangement of the Delta robot (left) and the Kiwibot (right). 17

2.1 Frame transformations in the Modified Complete and Parametrically Continuous (MCPC) model. .......................................................... 21
2.2 Closure of a 4 bar chain mechanism. ..................................................... 27
2.3 Logical structure of the connect.m algorithm. ....................................... 28
2.4 A 2 dimensional closed 4 bar chain. .................................................... 37
2.5 Example mechanisms with differing conditioning and constraints. .......... 44
2.6 Nominal geometric model of the Delta robot (direct geometric solution). 46
2.7 Nominal geometric model of the Delta robot (inverse geometric solution). 51

3.1 Distribution of results from pseudo-random normally distributed multivariate data in a chi-square plot. .............................................. 67

4.1 Formation of the identification Jacobian. ............................................. 72
4.2 Logical structure of the calibration algorithm. ...................................... 73
4.3 The Kiwibot 3 degree-of-freedom robot. ............................................. 74
4.4 Major components of the Delta robot. ................................................ 75
4.5 Column support structure for Delta robot ........................................... 76
4.6 The base platform and mounting brackets. .......................................... 81
4.7 Physical set-up of the control system hardware. .................................. 81
4.8 Overall schematic layout of robot control system. ............................... 83
4.9 Schematic layout of serial communication program for interfacing with the IFX motor controller. ....................................................... 83
4.10 The simulation process ........................................ 85
4.11 Arrangement of the kinematic model for simulation of the 4 bar closed
   loop mechanism ............................................. 89
4.12 Residuals of Monte Carlo tests with constant actuator noise ........ 91
4.13 Residuals of tests with constant measurement noise ................ 91
4.14 Residuals of tests with varying number of measurement points ...... 92
4.15 Computation time for identification algorithm ................... 92
4.16 Example of a mechanism that corresponds to an outlier ............. 93
4.17 Detection of outliers in Monte Carlo simulations on the 4 bar chain
   mechanism .................................................. 93
4.18 Marginal distributions of Monte Carlo results for run 3 ............. 94
4.19 Chi-square plot for run 3 along with comparable normally distributed data
   sets ......................................................... 95
4.20 Comparison of the magnitude of confidence ellipsoid major axes to sensi-
   tivity along the axis ....................................... 95
4.21 Arrangement of the kinematic model for simulation of the Kiwibot ..... 99
4.22 The measurement grid used for simulated Kiwibot calibration ....... 101
4.23 Positioning error, for first simulation of the Kiwibot, before calibration on
   the measurement grid ...................................... 102
4.24 Positioning error, for first simulation of the Kiwibot, after calibration .. 102
4.25 Orientation error, for first simulation of the Kiwibot, before calibration . 103
4.26 Orientation error, for first simulation of the Kiwibot, after calibration .. 103
4.27 Position error after implementation for second simulation of the Kiwibot 104
4.28 Orientation error after implementation for second simulation of the Kiwibot 104
4.29 Position error after implementation for third simulation of the Kiwibot 105
4.30 Orientation error, for first simulation of the Kiwibot, after calibration . 105
4.31 Arrangement of the kinematic model for simulation of the Delta robot . 109
4.32 Kinematic model of the Delta robot with kinematic loops closed .... 109
4.33 Positioning error before calibration on the measurement grid ....... 112
4.34 Positioning error after calibration for simulation 1 ................. 112
4.35 Orientation error before calibration ................................ 113
4.36 Orientation error after calibration for simulation 1 .................. 113
4.37 Positioning error after calibration for simulation 2 ................... 114
4.38 Orientation error after calibration for simulation 2 ................... 114
4.39 Positioning error after calibration for simulation 3 ................... 115
4.40 Orientation error after calibration for simulation 3 ................... 115
LIST OF FIGURES

4.41 Anticipated positioning error after calibration. .......................... 119
4.42 Anticipated orientation error after calibration. ......................... 119

A.1 Comparison of the accuracy of 2 methods for converting a rotation matrix 
to a set of Euler parameters. ...................................................... 129

B.1 Graphical output from drawgeom.m. An open 4 bar linkage. ........... 141
B.2 Graphical output from drawgeom.m. A closed 4 bar linkage whose free 
joint angles were determined by connect.m ................................. 141
List of Tables

3.1 Summary of minimisation techniques. ........................................ 61
3.2 Performance of non-linear minimisation algorithms. .................... 63

4.1 MCPC parameters laid out in mcpc format with connectivity and con­
straint information in the last 3 columns. ................................. 71
4.2 Input data for Monte Carlo simulation experiments on the 4 bar chain
mechanism. ................................................................. 90
4.3 Main diagonals and eigenvalues of $\Sigma_\alpha$ and $\Sigma^*_\alpha$ for comparison. .... 96
4.4 Nominal model definition matrix for the Kiwibot. ......................... 99
4.5 Results of simulated calibrations on the Kiwibot. ....................... 100
4.6 Nominal model definition matrix for the Delta robot. .................. 110
4.7 Results of calibrations on the Delta robot. .............................. 111

A.1 Typical values of normalised difference metric $\bar{e}(A; A + kE)$. ......... 134
Abstract

The geometric analysis and accuracy enhancement of parallel topology robots is of theoretical interest as well as having potential for application in industrial settings where parallel robots are used for tasks in which positional accuracy is important. This work presents new techniques for modelling of closed loop mechanisms and applies these experimentally in the calibration of a real parallel topology robot: The Delta robot. This work extends the current literature on serial robot calibration into the realm of parallel robots and presents a systematic approach which does not require the model or solution technique to be adjusted for the particular geometry of the robot under investigation.

The geometric modelling technique is quite general although in its current implementation it is only capable of modelling mechanisms that have rotary joints. Theory is presented for modelling of prismatic joints and the model can be adapted to handle joints of any type. The model analyses the structure as if it were a tree of bodies, each connected by a rotary or prismatic joint. A method of calculating the derivatives of the body frame positions with respect to the geometric parameters is also given.

A defining characteristic of parallel topology mechanisms is that the kinematic chains form closed loops. Finding the joint configuration that has all loops properly closed is a non-linear minimisation problem referred to as the closure problem. This is solved using the Levenberg-Marquardt technique.

For analysis of errors in a robot that is already assembled, an experimental calibration procedure is necessary. This procedure compares measured endpoint positions with those predicted by the geometric model and attempts to find a set of parameters that minimises these differences. The calibration procedure that was developed was tested on a number of simulated robots and a working Delta robot, which was designed and built specially for the calibration experiment. The mechanical design of the robot, software and hardware design of the robot controller, and the software implementation of the modelling and calibration procedures are described.

It was found that the modelling, identification, and implementation methods worked successfully on the robots examined, but that the implementation process was too slow for use in a practical controller because of the need to perform multiple direct geometric solutions. The computational effort required for the implementation procedure was considerable, but use of a compiled computer language and optimised code would provide significant improvement.
Acknowledgements

I gratefully acknowledge the encouragement and assistance of Dr Reg Dunlop, principal supervisor. I also thank my co-supervisor Dr Ian Coope for his invaluable contribution. I would like to thank Dr Chris Damaren for his contributions to the theoretical content of the work. During the project I have taken useful advice from a number of people including Dr Murray Smith of the Mathematics Department and also Prof. Harry McCallion.

I would like to thank those workshop staff and technicians who were of assistance during the project. In particular I would like to mention Mr Scott Amies, who built the Delta robot and Mr Andrew Cree who made it work.

I acknowledge my fellow postgraduate students who have been the source of many ideas on a wide range of topics and good companions for the duration of the project. In particular I would like to acknowledge Dr Tim Jones with whom I have worked closely for several years.

Finally I would like to thank my parents for their support and patience during the years it has taken to finish this project. I certainly could not have undertaken the project without them.
Publications

Refereed Publications


Other Publications


Notation

General Notation

When new terminology is introduced, it is written in emphasised text and defined in the glossary. The last word in this sentence is emphasised. Sections of the text that contain long derivations have the final result summarised at the beginning in a boxed paragraph headed “Key Result”.

The following notational conventions are used for mathematical expressions: Scalars are represented in italic type as in

$$x = -b \pm \frac{\sqrt{b^2 - 4ac}}{2a}$$  \hspace{1cm} (1)

Matrices are represented in boldface type. In general an upper case letter represents a matrix of more than one column and a lower case letter represents a single column matrix as in

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \end{bmatrix}$$ \hspace{1cm} (2)

$$p = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$ \hspace{1cm} (3)

Composite matrices that consist of an array of sub-matrices are represented in tableau as

$$P = \begin{bmatrix} A & B \\ C & D \end{bmatrix}.$$ \hspace{1cm} (4)

It may be assumed that the sizes of the sub-matrices are compatible.

A distinction is made between a vector and a column matrix representing the components of a vector. A vector is a directed quantity written in bold with an arrow over the symbol as in $\vec{r}$. The vector cannot in this notation be written as a column matrix $r$ until it is expressed in a frame as follows

$$\vec{r} = \mathbf{F} \cdot r$$ \hspace{1cm} (5)
where the symbol $\vec{F}$ represents a vectrix. A vectrix may be described as a 'column matrix of vectors'. It contains the unit vectors corresponding to the axes of a coordinate frame as in

$$\vec{F}_b = \begin{bmatrix} \hat{1}_b \\ \hat{2}_b \\ \hat{3}_b \end{bmatrix}. \tag{6}$$

The same vector may be written as a different column matrix if expressed in a different frame but the vector itself remains invariant. This notation was introduced by Hughes [1986].

A unit vector is represented with a 'hat' or caret above the symbol. In particular, the principal unit vectors are written $\hat{1}, \hat{2}, \hat{3}$, and the column matrices representing the principal unit vectors are written $\hat{1}, \hat{2}, \hat{3}$. Coordinate frames are represented by the symbol $\mathcal{F}$. A coordinate frame $\mathcal{F}_a$ always possesses a vectrix $\vec{F}_a$ describing the orientation of the frame and a position vector $\vec{r}_a$ representing the position of the frame's origin. If a frame $\mathcal{F}_a$ is defined in the text, then $\vec{F}_a, \vec{r}_a$, and $r_a$ are understood to exist without being explicitly defined. Subscripts are used to identify related symbols. Occasionally the frame in which the vector is expressed is indicated with a preceding superscript as in $\mathcal{F}^b_a$.

The identity matrix is denoted $I$. The size of the matrix is sometimes specified as in $I_n$ for an $n$-square identity matrix. The zero matrix is denoted $0$. The notation $0_{n \times m}$ specifies an $n$ by $m$ zero matrix.

The vector cross product is conventionally represented as an operator,

$$\vec{r} = \vec{p} \times \vec{q} \tag{7}$$

however it may also be represented as a transformation of a column matrix

$$\vec{p}^\times = \begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix} \tag{8}$$

such that

$$\vec{r} = \vec{p} \times \vec{q} = \vec{p}^\times \vec{q}. \tag{9}$$

The Kronecker product of $A$ and $B$ is written $A \otimes B$ and is defined as follows:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix}. \tag{10}$$
An orthonormal rotation matrix that transforms a vector expressed in frame $F_b$ to the same vector expressed in a different frame $F_a$ would be denoted $C_{ab}$. For example

$$r_a = C_{ab}r_b.$$ (11)

The principal rotation matrices as defined in section A.1 represent rotations about each of the axes of a coordinate frame and are denoted $C_1$, $C_2$, and $C_3$. Homogeneous transformation matrices that define the rotation and translation of a coordinate frame from joint a to joint b are denoted $T_{ab}$ where

$$T_{ab} = \begin{bmatrix} C_{ab}^T & r_{ab} \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$ (12)

Functions that return scalar values are written $f(x)$, $f(x)$, or $f(y; x)$ while functions that return matrix values are shown in boldface type as in $f(x)$, $f(x)$, or $f(Y)$. Geometrical entities such as points, axes and planes, are written in italics. Points are written in upper case while lines, axes, planes and spheres are written using lower case symbols. For example; A point $P$ lies on a plane $p$ while axis $a$ defines the axis of rotation of sphere $s$.

A Gaussian or Normally distributed quantity $x$ having mean $\mu$ and variance $\sigma$ is defined

$$x \sim N(\mu, \sigma).$$ (13)

For multivariate Gaussian distributions, a subscript indicates the number of dimensions, the mean becomes column matrix $\mu$, and the covariance matrix is $V$ so that

$$x \sim N(d \mu, V).$$ (14)

**Units**

Unless explicitly stated otherwise, quantities expressing length are given in meters and angular quantities are given in radians. Model parameters are given without units in order to save space. Some quantities, such as the value of a cost function, are composed of the sum of quantities with unlike units. These are shown without units.
Nomenclature

\(a\) Column matrix containing the parameters of a geometric model (§2.4.3).

\(D_p\) The open loop Jacobian of frame \(F_p\) with respect to all model parameters (§2.4.3).

\(f(\theta_i; a)\) The geometric model of a mechanism with endpoint position and orientation expressed in terms of measurement device coordinates.

\(f_*\) The matrix of all \(f(\theta_i; a)\) stacked into a column (§1.1.1).

\(g(\theta_i; a)\) The geometric model of a mechanism with endpoint position and orientation expressed in 7 parameters (3 cartesian parameters, 4 Euler parameters) (§2.4, 2.3).

\(J_{id}\) The identification Jacobian (§1.1.1, 4.2.1).

\(J_{ol}\) The collective matrix of all open loop derivatives for all frames in a mechanism (§2.4.3).

\(M_i\) The transformation that performs the mapping \(g(\theta_i; a) \rightarrow f(\theta_i; a)\).

\(N\) The number of joints in a given geometric model.

\(n\) The number of parameters in a given geometric model.

\(n_c\) The number of joint closures in a given geometric model.

\(m\) The number of measurements taken in a calibration experiment or simulation.

\(W_c\) Weighting matrix for closure problem.

\(W\) Weighting matrix for calibration problem.

\(y_i\) Measurement data in the form returned by the measurement device.

\(y_*\) The matrix of all measurements \(y_i\) stacked into a column.
Trademarks

- Borland, Borland Turbo C, and Borland C++ are registered trademarks of Borland International Inc.

- LabVIEW is a registered trademark of National Instruments Corporation.

- MATLAB is a registered trademark of The MathWorks Inc.

- Digiplan, and IFX are registered trademarks of Parker Hannifin Corporation.

- Optotrak is a registered trademark of Northern Digital Inc, Waterloo, Ontario.

- Pentium is a registered trademark of Intel Corporation.

- Tracker4000 is a registered trademark of SMX Corporation, Pennsylvania, USA
Glossary

actuator A motion producing device capable of responding to control signals. Most often an electric motor or hydraulic ram.

auxiliary frame A frame in a geometric model that does not represent a body, but represents a convenient reference point on the body represented by its predecessor frame. For example, tool frames are defined as auxiliary frames.

base frame A frame in a geometric model that represents the base of a robot. Movement of the base frame results in movement of the entire robot as a rigid body.

closure joint If a closed-loop mechanism is analysed as an open-loop chain of bodies, it must be “broken” at some joint in each loop. This joint is termed the closure joint. See section 2.3 for a fuller explanation.

closure problem The mathematical problem of finding the joint configuration of a mechanism that properly closes all kinematic closed loops.

direct geometric solution A model the converts joint space coordinates into task space coordinates.

end effector The terminal component of a robot, sometimes a gripper or some variety of tool.

forward calibration A method of calibration in which the direct geometric solution is used to compute an expected pose which is compared to the measured pose.

gain of mobility A type of mechanical singularity in which a mechanism gains freedom of movement at a certain configuration.

geometric calibration Calibration of a robot with respect to its geometry (link lengths and fixed joint angles) as well as joint offsets.

hybrid parallel robot A subclass of parallel robots that have complex kinematic chains connecting the base and nacelle.

identification The process of finding values for the parameters of a model that adequately reproduce the measurements obtained from the measurement process.
identification Jacobian  A matrix of derivatives that relates the rate of change of the
task space coordinates of the robot to the rate of change of the model parameters.

implementation  The process of finding the joint space coordinates that produce a given
set of task space coordinates in a real robot.

implicit calibration  A method of calibration in which the geometric solution consists
of a set of equations that express kinematic consistency (correct closure, for ex­
ample) but does not explicitly compute the direct or inverse geometric solution.
Measurements may be made of endpoint pose, actuator settings, passive joint set­
tings, or some combination of these.

inverse calibration  A method of calibration in which the endpoint is moved to a known
position and the joint settings are measured.

inverse geometric solution  A model that converts task space coordinates into joint
space coordinates.

joint level calibration  Determining the correct relationship between the joint offsets
and the corresponding joint offset transducers.

joint space (coordinates)  The space of actuator settings of a robot. Actuator settings
includes the angles of the rotary joints and the extensions of the prismatic joints.

Mahalanobis radius  The characteristic dimension of a hyper-ellipsoid. The radius of
the sphere obtained by scaling the axes of a hyperellipsoid with the inverse of its
positive-definite scaling matrix (cf. §3.5.3).

modelling  The act of producing a mathematical model to describe the relationship
between task space and joint space coordinates given the geometric and/or non­
geometric parameters of the robot.

measurement  The act of taking a set of measurements of a real mechanism in order to
characterise the relationship between its joint space and task space coordinates.

measurement space  The space of coordinates returned by a measurement device. For
example, a theodolite may return azimuth and altitude readings, thus the measure­
ment space is in a bounded region of $\mathbb{R}^2$.

nacelle  The mobile platform of a parallel topology robot or mechanism.

nominal parameter set  The set of geometric model parameters that represent the
robot as designed without geometric errors.

non-geometric calibration  Calibration of a robot with respect to joint offsets, geo­
metric parameters, and additional parameters not related to the geometry of the
robot, such as elasticity, or temperature effects.
off-line (programming) Programming a robot to perform a given task using a kinematic model rather than physically referencing task space positions. The task space coordinates specified in the program are assumed to be achieved by the robot during operation.

on-line (programming) Programming a robot by recording the robot configuration at reference positions in the task space. The robot is moved to the required position and the joint space coordinates are stored for later reference by the program.

parallel topology robot A robot that consists of a base connected to a nacelle by multiple kinematic chains.

perturbed parameter set In simulated calibration: The set of model parameters that represent the physical robot with geometric errors. In practical calibration, this parameter set is unknown. In reality the true robot may not be adequately represented by the model so that there is no perturbed parameter set that could represent the real robot accurately.

prismatic joint A joint whose motion results in a linear translation. Occasionally, in the literature, this term describes a joint whose motion is a combination of rotation and translation but no such joints are considered in this work.

serial topology robot A robot that consists of a base connected to an end-effector by a single kinematic linkage.

simple parallel robot A subclass of parallel robots that have the base connected to the nacelle by kinematic chains each consisting of a single link.

task space (coordinates) The position and orientation space of the end effector of a robot. The task space coordinates of a robot are usually the cartesian coordinates and euler angles describing the position and orientation of the end effector.

teach-pendant A piece of equipment often used in on-line programming of robots. The teach-pendant is typically a small hand-held console with which the programmer is able to control the robot manually and store the robot’s configuration in a memory device.

tool frame An auxiliary frame in a geometric model that represents the critical point of an end effector, perhaps the tool tip.

trust-region In non-linear minimisation: A locale around the current search point in which the function is considered likely to be well modelled by a quadratic approximation.

Wolfe conditions In non-linear minimisation: A set of conditions that guarantee adequate progress in a line search. The “sufficient decrease condition” specifies that the
decrease in the value of the cost function must be larger than a linear extrapolation from the start point of the line search. The gradient of the linear extrapolation is proportional to the slope along the direction of the line search. The gradient condition specifies that the slope at the test point must be flatter than the slope at the start point.

work-space The volume of space that a specified point on a manipulator is capable of occupying.
Chapter 1

Introduction

The word "robot" was initially derived from the Czech word "robota" which means forced labour or slavery. This is the role that robots currently play: they are servants performing work that is uneconomic, too tedious, too dangerous, too difficult, or too precise for humans to perform.

By far the most robots produced and in use today are serial topology robots. A serial robot consists of a single chain of rigid links connected by rotary or prismatic joints. These robots have found favour because they are (generally) more dexterous and possess a larger work-space than parallel topology robots. They are used for automated assembly, machining, materials transfer and palletising, welding, machine loading, and a wide variety of miscellaneous uses. Much work has been done on the kinematic, geometric, and dynamic behaviour of serial robots. There have also been considerable advances in the theory and componentry of control systems.

Despite this, serial robots are not necessarily the best choice for all tasks. The serial nature of the mechanism has inherent disadvantages. One of these disadvantages is that much of the mass of a serial robot is mobile. The situation is demonstrated in Figure 1.1. The base motor must carry the weight and the inertia of all of the succeeding members so it must be very large and powerful in relation to the power actually required to manipulate the tool. A typical example of this is a robot weighing 1100 kg that has a rated payload of only 50 kg.

Another problem with serial robots is that joint positioning errors are additive. If each joint has an error of 10 arc minutes, then a 6 jointed robot will have an orientation error of up to 1 degree. Worse than this is the positional error which may be ±1 mm. Apart from joint inaccuracy, the robot may flex under load. The effect of this is to increase the positional uncertainty further. Serial robot designers are forced to compromise the stiffness of the robot in order to decrease the mobile mass. This increases deflections due to gravity and inertial loads in the process.

Parallel topology robots have distinctive advantages where a high payload-to-mobile-mass ratio and high stiffness are required. A parallel topology robot consists of multiple chains of members that are interconnected. Having the nacelle supported by multiple
Figure 1.1 Cartoon exaggeration of the mobile mass problem (courtesy Hamish Trolive).

Figure 1.2 Large Stewart Platform for automobile driving simulation. Parallel robots are often used in applications where large payloads are involved (Rexroth Newsletter).
(individually actuated) members means that the actuators can work together to support the payload. An example of this is the Stewart Platform [Gough and Whitehall, 1962; Stewart, 1965] which has been used as a 6 degree of freedom motion simulator, supporting loads of several tonnes. Figure 1.2 shows an application where a serial robot would be impracticable.

Parallel robots can also offer advantages such as high speed and high accuracy. High accelerations are potentially possible because the actuators work cooperatively to move the payload. Positioning errors are not additive with parallel robots, so the effect of actuator inaccuracy is not as large.

A further distinction can be made between parallel robots whose base and nacelle members are connected by serial chains and the simpler structures that are connected by a single link (such as the Stewart platform). Robots that are connected by multiple serial chains can be referred to as hybrid parallel robots. Examples of hybrid parallel robots are shown in Figure 1.5 (p. 17). Hybrid parallel robots are distinct from simple parallel robots in that they are more complex to model geometrically.

In order to control a robot, mathematical models of the structure are required. A model that relates the joint space coordinates to the task space coordinates of a robot is referred to as a direct geometric solution. The converse model relating the task space coordinates to the joint space coordinates is called the inverse geometric solution (see Figure 1.3). Both models require the fixed geometric parameters of the robot such as the lengths of the links and the relative orientation of successive joints.

![Figure 1.3](image.png)  
Figure 1.3  Representation of joint space and task space parameters and the geometric solutions used to convert between them.
The direct geometric solution for a serial robot is relatively straightforward, being the product of a series of homogeneous transformation matrices. In contrast the inverse geometric solution is complex with no completely general solution method. The converse is true for simple parallel robots. For example the inverse geometric solution for the Stewart platform is a simple matter of closing the loop equation between the end effector and the base platform while the direct geometric solution involves finding the roots of high order polynomials and was only solved in general form in 1994\(^1\). Both the direct and inverse geometric solutions for hybrid parallel robots are relatively complex.

1.1 Accuracy Enhancement and Calibration of Manipulators

Most robots have excellent repeatability (of the order of 0.1 mm for a serial robot) but suffer from poor accuracy (in the order of millimetres). The repeatability is dependent on the accumulated indeterminacy of the mechanical components of the robot. If the final position of a component is dependent on some stochastic factor (friction, for example) then the position will carry some uncertainty. Any small force or torque will produce component distortion, and no joint is without friction. No motor will reproduce exactly the same position every time, and no control signal is without noise or distortion. With good mechanical design and high quality componentry, the positional errors associated with such effects can be made very small indeed.

Inaccuracy can be said to be a result of the differences between the mathematical model of the robot system and the physical robot. If the various effects due to member deflection, manufacturing tolerances and clearances, temperature effects, and the myriad other sources of error could be modelled precisely then the accuracy of the robot could be improved to the same order of magnitude as the repeatability. The task of accuracy enhancement, then, becomes one of improving the mathematical model of the robot.

Calibration becomes a worthwhile task when a robot is being programmed off-line and relatively high accuracy is required. On-line techniques such as teach-pendant programming are not subject to modelling inaccuracy but calibration or re-teaching may be necessary if the robot is repaired or replaced. If tasks are shared between more than one robotic work station then calibration may be desirable in order to avoid significant differences between components produced by different work stations.

Many of the causes of inaccuracy are difficult to model, however significant improvement has been achieved in serial robot accuracy by modelling geometric errors in the kinematic chain. It is reasonable to expect that similar improvements are possible with parallel topology robots. The improvement in accuracy may well be even more marked for parallel topology robots because the effect of elastic deflections is much less than for

---

\(^1\)The general solution of the Stewart platform involves finding the roots of a polynomial of order 40. The coefficients of the polynomial are derived analytically, but the roots must be found by numerical methods [Dasgupta and Mruthyunjaya, 1994].
serial robots. Most of the previous work on robot calibration has been primarily focused on serial robots. Parallel robots have not yet made a significant impact in industrial applications and this is probably why there has been relatively little work on the calibration of fully parallel manipulators. Much of the work is based on the Stewart Platform and little experimental work has been published (cf. §1.1.2).

1.1.1 The Calibration Process

Roth et al. [1987] neatly encapsulate the calibration process into 3 levels of complexity and 4 distinct phases. This framework has been widely used by subsequent authors. A level 1 calibration or joint level calibration determines the correct relationship between the actuated joint offsets and the corresponding joint transducer signals. No account is taken for geometric errors in the links. A level 2 calibration or geometric calibration attempts to correct for geometric errors in the links of the robot as well as joint transducer errors. Links are assumed to be rigid and the joints are assumed to rotate or translate along their nominal axes with no movement in planes that the joint is not designed to move in. Level 3 calibration or non-geometric calibration describes models that account for errors in addition to those already described such as static deflections, irregularities in joint movements, errors due to payload variation, temperature effects, and any other possible sources of error. The level of complexity chosen reflects the degree of accuracy required and the availability of suitable models and measuring apparatus for the purpose.

The 4 phases of calibration occur in all levels of calibration. The phases are modelling, measurement, identification, and implementation.

Modelling

Modelling involves the development of a mathematical model for the mechanism that accounts for the types of errors observed. Most authors have used modifications to the standard direct and inverse geometric models to include error parameters. Denavit and Hartenberg [1955] proposed a method of direct geometric analysis for mechanisms based on homogeneous transformation matrices. This basic method was used (with modifications) by several researchers to calibrate serial robots. The method suffers from numerical instability in situations where consecutive rotation axes are almost parallel and most of the modifications to the method involve alternative formulations for situations where the axes are nearly parallel. Trevelyan and Sultan [1992] proposed an alternative method of formulation that was not based on homogeneous transformation matrices and applied it to a spherical wrist manipulator. Trevelyan et al. [1992] also proposed a method of calibration of serial robots using imaginary extra actuators to represent error parameters. A novel approach called the “Zero-Reference Model” [Mooring, 1983] represents joint axes with unit vectors and locates them with points expressed in a reference frame, thus avoiding the Denavit-Hartenberg difficulties.
A lesser used approach is to construct a function in the joint space that represents
the actuator offsets required to place the end effector in the correct pose (for example, a
spline surface in $n$ dimensions where the robot has $n$ joints). Such an approach requires
the identification of a large number of parameters to fix the shape of the surface and
is therefore more suitable for calibration in some relatively small sub-region of the task
space.

Calibration models for parallel mechanisms have generally been specific to a particu-
lar mechanism. Bennett and Hollerbach [1991], used single closed loops for calibration
of serial robots while Mooring et al. [1991] discuss calibration models for mechanisms
with single closed loops. Zhuang and Roth [1993b] and Masory et al. [1993] calibrated
the Stewart platform using its standard inverse geometric model while Hollerbach and
Lokhorst [1995], and Wampler et al. [1995] have published on the calibration of the
RSI 6-DOF Hand Controller using a specialised geometric model. Maurine and Dombre
[1996] and Vischer [1996] have successfully calibrated the Delta robot using a standard
kinematic model.

Some more general models for calibration of parallel mechanisms have been proposed
and these are discussed in §1.1.2.

**Measurement**

Taking of accurate position measurements in order to extract enough data to identify
the geometric parameters of the robot is not a trivial task. A variety of techniques and
devices have been used from simple joint position transducers to sophisticated laser and
magnetic tracking devices. The form of the measurement produced by the measurement
device adds another level of complication to the identification procedure. For example,
a theodolite may only return an elevation and yaw angle which means that the output
of the geometric model must be transformed into an anticipated elevation and yaw for
identification, in addition the theodolite gives no absolute scale to the measurements
necessitating that at least one dimension of the robot be measured by some other means.
Selection of a measurement system is often a trade off between cost and accuracy. More
expensive measurement equipment is often more accurate but it may not necessarily be
easy to set up or use. Simple systems that can obtain large numbers of measurements at
lower accuracy may be advantageous if the identification process is capable of filtering
measurement noise. Few measurement systems are capable of returning full 6 DOF
position and orientation data. Fortunately this is not necessary in order to achieve
identification.

Whitney et al. [1986] used the single theodolite method to measure the endpoint
position of a Puma 560 robot. Some of the shortcomings of the single theodolite method
may be overcome by using 2 theodolites to triangulate the position of the endpoint
[Judd and Knasinski, 1990]. In New Zealand, Otago University's Industrial Measurement
Centre has a twin theodolite system capable of measurement accuracies of 0.001 mm under good conditions. In a non-ideal environment the accuracy is 0.025 mm. Stone and Sanderson [1987] used a GP-8-3D Sonic Digitizer. The device consists of 4 ultrasonic transducers which independently detect the time of flight of a sound wave generated by an ultrasonic "sparker". Sonic systems suffer from temperature and pressure variations. These may be controlled in a laboratory environment but would be very problematic on a factory floor. Nevertheless, accuracies of 0.3 mm have been achieved.

Several authors report the use of coordinate measuring machines (e.g. Borm and Menq [1989]). While high accuracies are achievable, the machines are not particularly portable or practical for use in industrial settings. Some very sophisticated systems are available such as the Optotrag™ Laser Measurement System which is capable of measuring the position of a target in motion with accuracies of better than 0.5 mm. Other advanced metrology systems, such as the SMX Tracker4000™ laser tracker, claim to be capable of measuring 3 components of position with accuracies of ±25 µm on targets within 5 m of the device. The companies Leica and API produce similar devices, all of which operate on principles developed by Lau and Hocken [1987].

Some novel devices have been used. Canepa et al. [1994] used a triaxial accelerometer to extract geometric information from a serial robot. Everett and Ives [1996] developed a sensor based on photodiodes to detect the position of an array of spheres gripped by a serial robot. Low cost and low complexity measurement systems have also been tried. van Albada et al. [1992] use a system in which a CCD camera gripped by the robot is pointed at a flat plate on which a pattern of circles is printed. The accuracy of this system is approximately 0.1 mm and the system setup is relatively simple.

Identification

Identification is the process of finding values for the model parameters from the measurement data. Generally this involves some variety of minimisation procedure. There are several complicating factors in the identification process which make it a difficult problem. Often the solution is non-unique (either there are several discreet solutions or a subspace of parameter combinations that satisfy the solution constraints). Not all parameters are necessarily observable from the measurements (It is possible to construct measurement sets that have very little or no influence from a particular model parameter, thus rendering it impossible to estimate). The problems are usually of large scale with many model parameters and large numbers of observations, thus making computation a time consuming process. Measurement noise can affect the result, particularly when some model parameters are highly sensitive. Generally an identification is considered to be successful if it results in a parameter set which models the measurement data to within some acceptable accuracy regardless of whether it reflects the true geometry of the mechanism.
Several researchers have used variations of linear least squares techniques on serial robots with success. The minimum variance method is capable of minimising the effects of measurement noise on the result given estimates of the noise variance (Mooring et al. [1991], van Albada et al. [1994], Yao and Wu [1995]). Circle point analysis is a technique sometimes applied to serial robots. All joints are fixed except for 1, and position measurements are taken of the end effector while the free joint is moved throughout its range. The points lie on a circle and the orientation of the circle can be used to recursively determine the joint axes (Canepa et al. [1994], Zhuang and Roth [1993a]).

Some unconventional methods have been tried: Zhuang and Roth [1993a] presented a 2 step procedure in which the rotational parameters are identified independently from the translational parameters. The rotational parameters were solved recursively using circle point analysis. After each joint's rotational parameters have been determined individually, the translational parameters may be solved as a linear system of equations. The method was found to be inferior to the more common model based methods in terms of residual error because the 2 step procedure did not guarantee optimality for the parameter set as a whole. Legnani and Trevelyan [1996] applied an extended Kalman filter to the identification problem and reported several advantages using that method including rapid convergence, explicit representation of estimation error in the covariance matrix, and the ability to handle redundancy in the model parameters.

Assuming that the model is sufficiently linear about the solution, the Identification Jacobian has been used to glean information about the condition of an identification problem. The Identification Jacobian relates changes in the robot pose as measured by the measurement device to changes in the model parameters as follows

\[
\delta f_* = \begin{bmatrix}
\delta f(\theta_1; a) \\
\delta f(\theta_2; a) \\
\vdots \\
\delta f(\theta_m; a)
\end{bmatrix} = J_{ID} \delta a.
\]  

(1.1)

where \(\delta f(\theta_i; a)\) represents a small change in the values returned by the measurement device at pose \(i\) and \(\delta a\) is a small change in the model parameters.

The condition number of \(J_{ID}\) has been used as an indication of the solvability of the general non-linear problem. Menq and Borm [1988] investigated the solvability of calibration problems and concluded that the quantity

\[
O_{MB} = \sqrt[2]{\sigma_1 \sigma_2 \ldots \sigma_n} / \sqrt{m}
\]  

(1.2)

where \(\sigma_i\) is the \(i\)th singular value of \(J_{ID}\) was a useful indicator to use, larger values indicating better observability. Later, Nahvi and Hollerbach [1996] proposed that the Noise
1.1 ACCURACY ENHANCEMENT AND CALIBRATION OF MANIPULATORS

Amplification Index

\[ O_{NA} = \frac{\sigma_n^2}{\sigma_1} \]  

was superior to Menq and Borm's index because it penalised situations when the least singular value was small while \( O_{MB} \) can still be large when the least singular value is small. A small least singular value would imply that some parameter or combination of parameters was nearly unobservable.

A commonly applied technique is to use a non-linear minimiser or least squares solver to match the model output to pose measurement data. Mauricio et al. [1997] applied the Levenberg-Marquardt method (§3.3) in a simulated calibration of a Puma 560 robot. They found that it was necessary to reduce the number of parameters in their model from 27 to 25 in order to achieve identifiability. Nahvi et al. [1994] use the Gauss Newton Method for identifying parameters of a parallel topology shoulder joint mechanism. It is explained in §3.3 that both these techniques can suffer convergence problems due to approximation of the Hessian matrix of the parameters, however neither author reports encountering such problems.

Implementation

A robot controller typically receives the desired endpoint pose \( \mathbf{g} \) as input and uses the nominal inverse geometric solution to determine the actuator settings \( \theta \). Most calibration techniques find parameters for a direct geometric model. Except for the simplest of mechanisms, the inverse geometric model is not easy to derive from the direct geometric model.

Mooring et al. [1991] characterise the problem as one of matching the output of 2 robots, A and B, such that

\[ \mathbf{g}(\theta_A; a_A) = \mathbf{g}(\theta_B; a_B) \]  

where robot A may be the nominal machine and robot B is the calibrated robot. The problem is then to find a \( \theta_B \) that satisfies equation 1.4. This approach may also be applied for situations where a robot is being replaced in an industrial application by another with slightly different geometry. Typically an exact analytical solution cannot be found so the problem becomes a minimisation problem of the form

\[ \min \left\{ (\mathbf{g}_A - \mathbf{g}_B)^T \mathbf{W}_c (\mathbf{g}_A - \mathbf{g}_B) \right\} \]  

or

\[ \min \left\{ |\mathbf{W}_c (\mathbf{g}_A - \mathbf{g}_B)|_\infty \right\} \]  

where \( \mathbf{g}_A = \mathbf{g}(\theta_A; a_A) \) and \( \mathbf{g}_B = \mathbf{g}(\theta_B; a_B) \). Equation 1.5 minimises the squared sum of errors while equation 1.6 minimises the largest component of the error. Since the implementation problem must be solved in real time, the solution must be computed
quickly. Veitschegger and Wu [1987] proposed 2 methods. Both methods involved a single step calculation of the actuator settings. Method 1 involved a linearisation of the problem, effectively driving the standard controller to a position $p_d - \delta g$ where $\delta g$ is the pose error between the desired pose $p_d$ and the calibrated model $g_B$.

$$\delta g = g_B - p_d.$$  \hspace{1cm} (1.7)

Method 2 amounts to a single Newton-Raphson step to solve equation 1.4. Vuskovic [1989] used a technique based on the sensitivity of $g_B$ to the identified error parameters. For serial robots, the sensitivities can be calculated relatively simply: linear parameters translate directly to sensitivities and rotational parameters can be converted using cross products. This does not apply for parallel mechanisms because the forward kinematic solution is itself a non-linear problem.

### 1.1.2 Previous Work

Relatively little has been written on the subject of parallel robot calibration. Several papers present methods that are specific to a particular mechanism and not able to be generalised, some of which are summarised below. For others see Kugiumtzis and Lillekjendlie [1994], Jagadeesh and Kurien Issac [1995], and Koseki et al. [1998].

Everett and Lin [1988] presented a method for calibrating manipulators with closed loops. Around a closed kinematic loop consisting of joint frames $1,2,\ldots,k$

$$T_c = T_{1,2}T_{2,3} \cdots T_{k-1,k}T_{k,1}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$  \hspace{1cm} (1.8)

while the pose error at the end-effector is characterised by

$$\delta T = T_0^{-1}(T_m - T_0)$$

$$= \begin{bmatrix} 0 & -\delta z & \delta y & dx \\ \delta z & 0 & -\delta x & dy \\ -\delta y & \delta x & 0 & dz \\ 0 & 0 & 0 & 0 \end{bmatrix}$$  \hspace{1cm} (1.9)

where $T_m$ is the homogeneous transformation matrix representing the measured end-effector pose and $T_0$ is the expected end-effector pose. The 6 independent elements of $\delta T$ are extracted into a column matrix by the operator $S(\delta T)$ so that the cost function
1.1 ACCURACY ENHANCEMENT AND CALIBRATION OF MANIPULATORS

is

\[ \rho = \sum_{i=1}^{m} S(\delta T_i)^T S(\delta T_i) \]  (1.10)

Adding a modification term to the cost function enforces the closure constraint so that

\[ \rho = \sum_{i=1}^{m} S(\delta T_i)^T S(\delta T_i) + \lambda^T S \left( T_{c,i}^{-1}(I - T_{c,i}) \right) \]  (1.11)

where \( \lambda \) is a vector of Lagrange multipliers. The cost function is minimised by forming a set of equations of the form

\[ \frac{\partial}{\partial p_j} (\rho) = 0 \quad j=1\ldots n \]  (1.12)

where \( p_j \) is the \( j \)’th geometric parameter of the model. The authors demonstrated a simulated calibration of a mechanism containing a 5 bar actuated closed loop. The resulting system of equations was relatively complex, even for a simple mechanism but the biggest problem in terms of practical implementation is the requirement for full pose measurement in order to find \( T_c \).

Zhuang and Roth [1993b] presented a method for calibrating a Stewart platform from measurements of the endpoint position. Measurements were taken in 6 sets, each with a different actuator fixed. This allows each actuator’s parameters to be identified separately. The geometric model was simplified by assuming the robot to have perfect ball joints. Each ball joint then had 3 parameters that described its position relative to the base frame of the platform to which it was attached. The loop closure equations give the actuator lengths:

\[ l_i = |b_i - C_{bp} p_i - t| \quad i=1,2,\ldots,6 \]  (1.13)

where \( b_i \) is the position vector of base ball joint \( i \) relative to the base frame, \( p_i \) is the position vector of the nacelle ball joint \( i \) relative to the nacelle frame, and \( t \) is the position vector of the nacelle frame relative to the base frame. The length of actuator \( i \) is therefore represented by \( l_i \). For 2 measurements, \( j \) and \( k \), taken with \( l_i \) constant,

\[ |b_i - C_{bp}^j p_i - t^j| = |b_i - C_{bp}^k p_i - t^k|. \]  (1.14)

This allows the identification of \( b_i \) and \( p_i \) independently of \( l_i \) (provided \( j \neq k \)) given 7 independent observations of \( t \). In simulation studies, the Gauss-Newton method was used to solve the non-linear equation set on measurements with normally distributed noise added. It was found that the Gauss-Newton method typically converged to a solution in less than 5 steps. The leg length errors \( \delta l_i \) can easily be computed afterwards. Because the parameters are identified separately, it cannot be guaranteed that they are globally optimal thus making the method sensitive to measurement noise.
Masory et al. [1993] revisited calibration of a Stewart platform using an error model based approach. The individual kinematic chains from the base platform to the end-effector were modelled as serial kinematic chains using the Denavit and Hartenberg [1955] convention. The direct geometric solution was then performed iteratively. The identification Jacobian was evaluated by the finite difference technique which necessitated performing the direct geometric solution repeatedly. This was very computationally expensive. Despite being computationally demanding, successful results were demonstrated in simulations. Runs were performed in which measurement noise was simulated. It was shown that more pose measurements were required in order to obtain accurate results from noisy measurements however the accuracy ceased to improve as the number of measurements increased beyond 20. No mention is made of the problems associated with modelling ball joints as sets of 3 rotary joints in series. This is questionable because the motions of a real ball joint with geometric errors (ball and cup irregularities, for example) are unlikely to be simulated accurately by 3 rotary joints in series.

Zhuang et al. [1995] tried a modified approach to Stewart platform calibration, this time the need to perform a direct geometric solution was avoided. Full pose measurements are taken using a single theodolite and the expected leg lengths are compared with the leg length returned by the joint transducers. The cost function was defined as follows:

\[ \rho = \sum_{j=1}^{m} \sum_{i=1}^{6} \left( l_i + \delta l_i \right)^2 - \left( b_i - C_{bp}^j p_i - v^j \right)^T \left( b_i - C_{bp}^j p_i - v^j \right). \]  

Equation 1.15 is minimised using a gradient based minimisation algorithm. The authors report that the minimisation problem was well conditioned and that they were able to experimentally improve the positioning accuracy of a Stewart platform to near its repeatability.

Besnard and Khalil [1999] recently presented a calibration method for Stewart platforms using 2 inclinometers mounted on the nacelle member. The expected inclination of the nacelle is compared to the measured inclination and an error model with 29 error parameters is fitted to the results using the Levenberg-Marquardt algorithm. Simulated results showed that the accuracy of the inclinometers was important to the accuracy of the calibration. Using an inclinometer with a precision of $10^{-3}$ degrees, a positioning accuracy of 0.4 mm was achieved in simulations on a Stewart platform with a nominal leg length (at home position) of 850 mm.

Hollerbach and Lokhorst [1995] calibrated a 6-DOF hand controller that consisted of 2 closed loop kinematic chains. They calibrated the mechanism's joint angle offsets and sensor gains. This was a difficult task because of 2 complicating factors: Firstly, each closed loop had 6 unsensed degrees of freedom which had to be eliminated from the geometric equations, and secondly the gain parameters had a trivial solution at zero which exerted a large region of attraction over the space of the cost function. The non-
trivial solution was attainable only if the starting point for the solution algorithm was close to the final solution.

Innocenti [1995] considered the calibration of generalised Stewart platforms. He presented 2 algorithms for calibration of parallel mechanisms consisting of 2 members connected by 6 ball jointed rods. The interesting feature of his methods was that the geometry was solved directly without referring to a nominal model. With 7 observations of the endpoint pose and the corresponding link lengths a set of nonlinear equations were generated of the form

\[ l_k^2 - \left( b_k - C_{by}^i p_k - v^j \right)^T \left( b_k - C_{by}^i p_k - v^j \right) = 0 \]  \tag{1.16}

where \( k = 1, 2, \ldots, 6 \) is the index of the actuator and \( j = 1, 2, \ldots, 7 \) is the index of the observation. Subtracting the seventh equation from the previous 6, the system is reduced to the following form:

\[
\sum_{k=1}^{6} \sum_{j=1}^{7} \sum_{u=0}^{3} \sum_{v=0}^{3} e_{u,v,j,k} t_{u,k} p_{u,k} = 0 \]
\[ t_{0k} = p_{0k} = 1 \]  \tag{1.17}

This system can be formulated analytically as a set of 6 second order polynomials. The problem reduces to finding the roots of a polynomial of order 20 and back-substituting the roots to determine all parameter values. The second method is similar but with the capacity to calculate actuator length errors using an eighth observation. These techniques are of theoretical interest, however it is commonly accepted that redundant measurements are required to smooth out the effects of measurement noise.

There have been several attempts at calibrating the Delta robot (Zobel and Clavel [1993], Maurine and Dombre [1996], and Vischer [1996]). All used incomplete error models to identify a reduced parameter set in order to simplify the process. Zobel and Clavel used an 18 parameter model and took measurements by positioning a locating profile into a series of endpoint fixtures. The positioning error was reduced from ±7.7 mm to ±1.5 mm. Maurine and Dombre used a laser range finder mounted on the nacelle to locate points on an accurately machined plate. They identified 6 parameters in stage 1 of the procedure and another 3 in a second stage which used a different measurement scheme. The results in terms of positioning accuracy were not given. Vischer used 2 models containing 24 and 54 parameters respectively. He obtained full pose measurements of a small delta robot using a coordinate measurement machine with an attachment for measuring angular deviations. The identification problem was solved using the Levenberg-Marquardt method. The accuracy of the 24 parameter model was improved by a factor of approximately 10 (although orientation accuracy was not considered in the model), while the 54 parameter model showed similar improvement for positional errors.
but orientation errors were only improved by a factor of 3.7.

Vischer also compared several different schemes of calibration. Forward calibration is the method adopted here in this work (cf. chapter 3). The direct geometric problem is used to generate an endpoint pose estimate which is compared to the measured endpoint pose. Inverse calibration is when the endpoint is moved to known positions and the actuator settings are recorded. Implicit calibration is when the geometric model expresses kinematic consistency but doesn't necessarily produce an endpoint pose as output. The measurement devices may measure actuator settings, endpoint settings, passive joint settings, or some combination of these. Vischer concluded that implicit calibration is advantageous for parallel robot calibration because the need to evaluate the direct geometric solution can be avoided. It may be that a generalised implicit calibration procedure is possible. While such an approach would be of interest, it is likely to be somewhat more difficult to develop a systematic calibration method using implicit calibration, thus the forward calibration method was adopted.

While reasonable results were being obtained using methods that are customised to particular mechanisms, the problem of a general method of calibrating closed loop mechanisms remained unresolved. Wampler et al. present a versatile method of implicit calibration which is capable of handling both serial and parallel manipulators. The expression of the geometric model is dependent on the mechanism in question, but is represented generically by

$$f(x_i; a) = 0 \quad i = 1, 2, \ldots, N$$

(1.19)

where $x$ is the set of all joint space and task space parameters of the robot and $a$ is the set of all geometric parameters including parameters relating to the measurement device. If measurement noise exists then the model becomes

$$f(x_i + \delta_i; a + \delta) = 0.$$  

(1.20)

If the mechanism is physically feasible then equation 1.20 will always have a solution. Assuming Gaussian noise, with covariances $\Sigma_a$ which is known a-priori and $\Sigma_x = E(\delta_i \delta_i^T)$, the maximum likelihood estimate is the value of $\delta$ that minimises

$$\chi^2 = \sum_{i=1}^{N} \delta_i^T \Sigma_x^{-1} \delta_i + \delta^T \Sigma_a^{-1} \delta$$

(1.21)

subject to equation 1.20. The method of Lagrange multipliers is used to perform the minimisation.

The method is demonstrated on the previously mentioned 6-DOF hand controller [Hollerbach and Lokhorst, 1995] and Stewart platform mechanism. The method calculates the variance of the parameter estimates and provides a goodness of fit estimate.
which can indicate deficiencies in the geometric model.

1.2 Scope and Outline of Thesis

The object of this project is to present both a standardised representation for geometric modelling of parallel mechanisms and a method of calibrating those mechanisms. The method is formulated to account for static geometric errors only, but with adaptation may be applicable to more complex non-geometric error models. The formulations given are for mechanisms composed of rigid bodies that are connected by rotary, prismatic, or spherical joints.

The performance of the methods is examined by simulation on a variety of mechanisms and also by experimental application of the method on a Delta robot which was designed and built as part of the project. Of interest is the effect of measurement noise on the result of the calibration and the amount of improvement in measurement accuracy attainable as a function of the measurement noise.

Chapter 2 presents a general geometric model for parallel mechanisms. Beginning with a description of the “Modified Complete and Parametrically Continuous” modelling convention for rigid body chains, the modelling convention is applied to “tree structured” mechanisms. A method for achieving closure of the “branches” of the tree results in a practical method of modelling parallel structures. The motion and mobility of the mechanisms is revealed by the Jacobian matrix. Jacobians are given for both unconstrained “tree” mechanisms and closed mechanisms. Nominal direct and inverse models for the robots under investigation are presented.

Chapter 3 defines and discusses the calibration problem. A variety of non-linear minimisation techniques are summarised and discussed in terms of applicability and the management of noise in the measurements is discussed.

Chapter 4 discusses the practical application of the previously presented theoretical results. The design of the system components including the Canterbury University Delta robot, the control system and software, and the Modelling, Visualisation, and Calibration software packages are discussed. This is followed by a description of the simulation series with results. Finally the results of an experimental calibration of the Delta robot are presented.

Robots that are Modelled

Three robots are investigated in this project: The single closed loop 4 bar chain mechanism, the Delta robot, and the Kiwibot. The 4 bar chain mechanism represents the simplest closed loop mechanism that is mobile. Since the mechanism is 3 dimensional, it requires 7 degrees of freedom to be mobile according to Kutzbach’s mobility equation
The mobility is expressed by

\[ M = \sum_{i=1}^{N} n_j^i - 6L \]  

(1.22)

where \( n_j^i \) is the number of freedoms in joint \( i \) and \( L \) is the number of closed loops in the mechanism. The arrangement of the rotary joints is illustrated in Figure 1.4. The Delta robot was first proposed by Clavel [1985, 1991]. The structure is a simplification of a 6-degree of freedom robot first proposed by Hunt [1983] which in turn bears a resemblance to the Stewart platform. The robot consists of 2 platforms: a base platform and a nacelle. The platforms are connected by 3 identical kinematic chains (cf. Figure 1.5). Each chain consists of a loop of 4 spherical joints in the form of a parallelogram connected in series to a rotary jointed actuating arm. The parallelogram section of the chain is passive (meaning that it is not driven by any form of actuator). If the actuating arms are fixed in position then the nacelle cannot translate or rotate about any axis unless the robot is in a singular configuration. The sum of joint freedoms \( \sum n_j^i \) is 39 and there are 5 independent closed loops in the mechanism giving a Kutzbach mobility of 9. Six of the mobilities are free rotations of the parallelogram arms about the ball joint centres leaving 3 translational freedoms.

The Kiwibot is a 3 degree of freedom robot based on a constant velocity joint proposed by Hunt [1983]. It consists of 3 actuated arms which are arranged identically to those of the Delta robot. Three passive arms are attached by rotary joints to the nacelle member that sits above the base platform (Figure 1.5). The upper and lower sub-mechanisms are connected by 3 spherical joints. The homokinetic plane is defined by the centres of the 3 spherical joints and this defines a plane of symmetry in the mechanism. Kutzbach’s mobility equation gives a mobility of 3 \( (\sum n_j^i = 15, L=2) \) which are expressed as 2 degrees of rotational freedom and a “plunging freedom”. The plunging freedom allows the distance between the base and nacelle platform to be changed while the orientation of the nacelle remains constant. The mechanism is suitable for satellite dish aiming applications (cf. Dunlop and Jones [1997, 1998]) and with a suitable geometry it is capable of scanning an entire hemisphere without encountering a mechanical singularity.
1.2 SCOPE AND OUTLINE OF THESIS

Figure 1.4  Kinematic arrangement of the 3-dimensional 4-bar closed loop mechanism.

Figure 1.5  Kinematic arrangement of the Delta robot (left) and the Kiwibot (right).
Chapter 2

Modelling

2.1 Introduction

A geometric model is necessary in order to describe the relationship between the actuator coordinates and the resulting configuration of the robot. This chapter presents a general geometric model for parallel robots along with the open loop and closed loop differential geometric models. The model is general in the sense that any closed or open loop mechanism composed of rigid bodies and connected by rotary, prismatic, or spherical joints can be represented.

Section 2.2 discusses methods of representing rigid body chains for calibration purposes and describes the Modified Complete and Parametrically Continuous (MCPC) modelling convention. In §2.3 the MCPC modelling convention is applied to describe the geometry of mechanisms consisting of multiple interconnected rigid body chains. Closing the kinematic chains of the mechanism is achieved by solving a set of nonlinear closure equations. This is described in §2.4 along with the open loop differential model which is required by the closure algorithm. The closed loop differential model describes the effect of individual geometric errors on the closed mechanism. This is developed in §2.5 and used in the identification phase of the calibration process.

Nonlinear iterative solvers are much more likely to converge to the correct solution if the starting point of the solution is close to the correct solution to begin with. Since geometric errors are small by nature, then the nominal or ‘zero error’ state is the logical choice for the start point. Often a simplified geometric model is available to provide such a start pose. Section 2.6 gives nominal direct and inverse geometric solutions for the Delta Robot.

2.2 Rigid Body Chains

Modelling a robot as a series of rigid bodies is not advisable if elastic deflections are significant. However, rigid body models are routinely applied to serial mechanisms successfully. Because calibration techniques ‘best fit’ the model parameters to the measured
data, the effect of elastic deflections is partially compensated for. For example, Judd and Knasinski [1990] found that parameters included to model gear train errors in an Automatix AID-900 robot also compensated for gravitational deflections. Parallel robots are generally much stiffer than serial robots and therefore the use of a rigid body model is even more appropriate. It is conceivable that the model could be adapted to contain parameters for first mode elastic deflections.

The object is to specify the position and orientation of a joint axis relative to another with the minimum number of parameters. Generally, it takes 6 parameters to describe the position and orientation of a body with respect to another but it is possible to describe the transformation from one joint axis to another in 4 parameters. The price paid for this is the loss of 2 freedoms in the placement of the new frame. In the models described below, these freedoms correspond to the position of the origin of the new frame along the joint axis and the orientation of the new frame about the joint axis.

Consider two axes \( a \) and \( b \) that are placed arbitrarily with respect to each other (cf. Figure 2.1). A coordinate frame \( F_a \) has its origin on \( a \) and its 3 axis aligned along \( a \). A set of principal rotations or translations is performed on \( F_a \) so that its origin lies on \( b \) and its 3 axis is aligned with \( b \). Starting with \( F_a \) on axis \( a \), a rotation about the 1 axis (angle \( \alpha_1 \)) is followed by a rotation about the new 2 axis (angle \( \alpha_2 \)) so that the 3 axis of the rotated \( F_a \) is parallel to axis \( b \). This new frame will be referred to frequently in the following sections and denoted \( F_b \). Next, the frame is translated along its new 1 and 2 axes (distances \( d_1 \) and \( d_2 \) respectively) to lie on axis \( b \). The combined transformation is

\[
\begin{align*}
\dot{F}_a &= C_2(\alpha_2)C_1(\alpha_1) \dot{F}_a \\
&= C_{\dot{a}a} \dot{F}_a 
\end{align*}
\tag{2.1}
\]

and

\[
r_b = C_{\dot{a}a}r_a + r_{ab}
\tag{2.3}
\]

where

\[
r_{ab} = \begin{bmatrix} d_1 \\ d_2 \\ 0 \end{bmatrix}
\tag{2.4}
\]

and \( r_b \) is expressed in frame \( F_a \). The model does not yet contain any reference to the joint angle (or offset if the joint is prismatic). The joint transformation is applied afterwards as follows. For a rotary joint equation 2.1 is rewritten

\[
\begin{align*}
\dot{F}_b &= C_{b\dot{a}}C_{\dot{a}a} \dot{F}_a \\
\dot{F}_b &= C_3(\theta)C_2(\alpha_2)C_1(\alpha_1) \dot{F}_a
\end{align*}
\tag{2.5}
\]
Figure 2.1 Frame transformations in the Modified Complete and Parametrically Continuous (MCPC) model.
and for a prismatic joint equation 2.4 is rewritten

\[ r_{ab} = \begin{bmatrix} 0 \\ 0 \\ \theta \end{bmatrix} \]  

(2.6)

where \( \theta \) is the joint setting (angle or offset, depending on context). Note that the parameters \( d_1 \) and \( d_2 \) are not used in equation 2.6 because they are redundant. There are 4 parameters that describe the fixed physical relationship between consecutive joint axes and 1 parameter describing the variable joint setting.

Zhuang [1995] proposed an identical model which he dubbed the 'Modified Complete and Parametrically Continuous' or 'MCPC' model. The term MCPC is adopted to describe the model defined above. Zhuang distinguished the difference between position dependent and position independent parameters. A position dependent parameter is one that changes depending on the joint setting, such as angular error due to gear tooth variation. A position independent error is a fixed offset whose value does not change depending on the actuator setting. The MCPC model neatly separates position dependent parameters from position independent errors so that the model can be written

\[ \text{\( \mathbf{F}_b = QV \mathbf{F}_a \)} \]  

(2.7)

where \( V \) is the transformation due to position independent parameters (equation 2.1) and \( Q \) is the transformation due to the position dependent parameters. Expressed in the notation used in equations 2.5–2.6,

\[ Q = \begin{cases} C_3(\theta) & \text{for a rotary joint,} \\ I_3 & \text{for a prismatic joint.} \end{cases} \]  

(2.8)

\[ V = C_2(\alpha_2)C_1(\alpha_1). \]  

(2.9)

The joint offset vector becomes

\[ r_{ab} = q_{ab} + v_{ab} \]  

(2.10)

where

\[ q = \begin{bmatrix} 0 \\ 0 \\ \theta \end{bmatrix}^T \]  

(2.11)

for a rotary joint,

\[ v = \begin{bmatrix} d_1 \\ d_2 \\ 0 \end{bmatrix}^T. \]  

(2.12)

Under special conditions as described in section 2.4 equations 2.8–2.12 alter slightly.
2.3 A General Geometric Model for Parallel Mechanisms

Adopting the MCPC modelling convention, it is possible to specify any parallel mechanism as a tree structure by disconnecting the mechanism at selected joints until there are no closed loops in the mechanism. The optimal way to disconnect the mechanism is an open question, however it can be stated that spherical joints are convenient joints at which to disconnect the chains. This is because loop closure at a spherical joint is achieved when the position vectors of the branch ends coincide. The orientation of the joint frames need not coincide at a spherical joint and therefore fewer constraint equations need to be satisfied.

Once all loops in the mechanism have been opened, then the position and motion of each branch may be considered separately. Closure of the mechanism and mobility constraints need not be considered at this stage. Choosing a particular chain and numbering its joint frames consecutively starting at zero for the base frame, the position vector of the \( p \)’th joint frame in a serially connected chain is

\[
\vec{r}_{0,p} = \sum_{i=1}^{p} \vec{r}_{i-1,i}.
\]  

(2.13)

Vector \( \vec{r}_{i-1,i} \) is best expressed in frame \( i \) (§2.2) because it has no component in the 3 axis direction of this frame,

\[
\vec{r}_{0,p} = \sum_{i=1}^{p} C_{0,i} \vec{r}_{i-1,i}.
\]  

(2.14)

Adopting the MCPC convention,

\[
C_{0,i} = \begin{cases} 
\left( \prod_{j=1}^{i-1} V_j^T Q_j^T \right) V_i^T & \text{for } i > 1, \\
V_i^T & \text{for } i = 1 
\end{cases}
\]  

(2.15)

where

\[
V_j = C_2(\alpha_{2j})C_1(\alpha_{1j})
\]  

(2.16)

\[
Q_j = \begin{cases} 
C_3(\theta_j) & \text{for rotary joints}, \\
I_3 & \text{for prismatic joints}
\end{cases}
\]  

(2.17)

and

\[
\vec{r}_{i-1,i} = \begin{cases} 
\begin{bmatrix} d_{1i} & d_{2i} & 0 \end{bmatrix}^T & \text{for rotary joints}, \\
\begin{bmatrix} 0 & 0 & \theta_j \end{bmatrix}^T & \text{for prismatic joints}
\end{cases}
\]  

(2.18)

Equations 2.16–2.18 assume that the physical robot’s joints are ideal in the sense that a rotary joint will perform a pure rotation and a prismatic joint will perform a pure translation. The joint parameters are independent of the joint position parameter.
under this assumption. For a geometric calibration using position independent parameters, any multi-degree-of-freedom joint can be constructed from a combination of rotary and prismatic joints. If more complex joints are modelled, then a joint specific transformation with specialised parameters could be substituted for equations 2.16–2.18. In particular, a ball joint can be modelled as 3 intersecting rotary joints however the model will fail to span the space of possible errors if position-dependent parameters (ball and cup irregularities, for example) are included in the model.

2.4 The Closure Problem

Closing the loops of the mechanism is achieved by an iterative numerical procedure. The joint settings that correspond to actuators \( \theta_{\text{act}} \) are considered to be fixed while the remainder \( \theta_{\text{free}} \) are allowed to vary. Collectively the set of all joint settings, fixed and free, is denoted \( \theta \).

Consider 2 chains of joints that normally form a closed loop, denoted a and b. The terminal joint of chain a corresponds to the same physical joint as the terminal joint of b. These joints are termed closure joints. Denoting the terminal joint frame of chains a and b respectively as \( \mathcal{F}_a \) and \( \mathcal{F}_b \), the condition for closure is

\[
\mathcal{F}_a = \mathcal{F}_b
\]

which implies

\[
\vec{r}_a = \vec{r}_b
\]

and

\[
C_{0a} = C_{0b}
\]

which, in turn, implies

\[
C_{0a} = C_{0b}
\]

If the closure joint is spherical then equation 2.22 becomes defunct and closure is achieved when equation 2.23 is satisfied.

Since \( C_{0a} \) and \( C_{0b} \) are both 3 by 3 orthonormal matrices, there are 9 interdependent parameters that must be matched. The orthonormality condition implies that there are actually only 3 independent parameters that define the orientation of the frame. These could be extracted from the rotation matrices thus reducing the number of constraint equations. Possible choices for a parameterisation of orientation are Euler angles or Euler parameters (§A.1). Any set of Euler angles will possess a singularity at some orientation that will prevent the rotation matrix being decomposed into 3 parameters. Because of this, Euler parameters are used. Euler parameters are non-singular in all
configurations with the drawback that they are a 4 parameter representation. This adds a single constraint equation to the closure constraints, this being that the norm of the parameter set is always equal to 1.

Due to the 2 constraints on the position of consecutive joint axes mentioned in §2.2, some extra parameters are required at closures. For rotary joints, there is no way to specify where along the terminal joint axis the joint frame will lie, or at what orientation about the axis. There is no reason that chains a and b will place the terminal frame at the same position and orientation. To remedy this, an extra 2 parameters are added to joints that are closure joints. These extra parameters are an offset along the joint axis \( d_3 \) and a rotation about the 3 axis \( \alpha_3 \) to make the terminal frames coincident. For prismatic joints, the terminal axes may be parallel but not collinear and their orientation about the 3 axis may differ. In these situations, 4 extra offset parameters are specified to make the axes coincident: 3 offsets \((d_1, d_2, d_3)\) and a rotation about the 3 axis \( \alpha_3 \). The extra parameters need only be specified for one of the chains involved in the closure.

The nature of parallel mechanisms means that there may be several possible configurations of the mechanism where closure can be achieved. Ideally a nominal solution will be available in order to provide a reasonable estimate of the favoured configuration as a starting point for the closure algorithm.

2.4.1 The Closure Algorithm

The problem is solved using the Levenberg-Marquardt technique. This technique is well suited to the problem since the residuals after closure are small and because the algorithm can be forced to find the nearest local minimum by suitable manipulation of the initial trust region.

The Levenberg-Marquardt method requires first derivative information about the changes in frame position with respect to the free joint settings \( \theta_{\text{free}} \). The derivation for the open loop Jacobian matrix with respect to the joint settings \( \theta \) is given in section 2.4.2. The closure algorithm picks out those columns that correspond to \( \theta_{\text{free}} \). A more general differential model for calibration purposes is given in section 2.4.3. This gives the changes in frame position with respect to all model parameters.

The algorithm converges rapidly and reliably in practice. It is implemented in the MATLAB environment using the routine \texttt{levmrq.m}. This routine takes as arguments the names of 2 m-files (\texttt{connecfn.m} and \texttt{connecj a.m}) that provide, respectively, an evaluation of the cost function, and an evaluation of the open loop Jacobian, given a column matrix of joint settings as the input argument.

Let

\[
g_p (\theta; \alpha) = \begin{bmatrix} r_{0,p} \\ \varepsilon_p \\ \eta_p \end{bmatrix}
\]  

(2.24)
denote the general geometric model of a mechanism with parameters expressed in work­
space coordinates, where \( \varepsilon_p \) and \( \eta_p \) are the Euler parameters (§A.1) corresponding to \( C_{0,p} \). The cost function is calculated as follows

\[
c = \sum_{i=1}^{n_c} \left( g_{p_i} - g_{q_i} \right)^T W_c \left( g_{p_i} - g_{q_i} \right)
\]

(2.25)

where \( n_c \) is the number of closures and joints \( p_i \) and \( q_i \) form the \( i \)'th closure pair. The weighting matrix \( W_c \) scales the linear and rotational parameters so that they are comparable in magnitude. In practise the algorithm converged successfully for all mechanisms tried using \( W_c = I_T \) with linear units in metres and Euler parameters for rotational units. The structure of the algorithm is demonstrated in Figure 2.3.

The free parameters in the minimisation are those corresponding to the joint freedoms except for those that are actuators. For rotary joints, it is \( \alpha_j \) and for prismatic joints it is \( \delta_j \) where \( j = 1 \ldots N \). Currently the software does not support spherical joints at any place other than joint closures. A spherical joint can easily be simulated at some other point in the chain with 3 rotary joints in series. This requires additional constraints to be added during the calibration phase to prevent the axes of the simulated spherical joint from separating. The constraints are added manually by passing a matrix containing the indices of constrained parameters to connect.m.

If the starting point for the minimisation is badly chosen, connect.m may converge at a configuration that is not within the primary workspace, or may not be able to close at all. If an approximate solution exists for the direct kinematic solution then it can be used to provide a good estimate of the desired solution and used as a starting point. The residual is checked after the minimisation has converged and a warning is generated if its norm is larger than \( 10^{-10} \). This is an effective check for failure to achieve closure. Figure 2.2 illustrates the progress of connect.m as it attempts to close a 3 dimensional 4 bar chain mechanism.

2.4.2 Open Loop Derivative with Respect to Joint Settings

\[
\frac{d}{ds} g_p(\theta) = \begin{bmatrix}
- \sum_{i=1}^{P} l_{0,p} C^T_{0,i-1} C^T_{i=1} C^T_{2i} \hat{3} \frac{d\theta_i}{ds} \\
Z_{Ed}^{-1} \sum_{i=1}^{P} C^T_{0,i-1} \left( C^T_{1i} C^T_{2i} \hat{3} \right)^x C^T_{i,p} \frac{d\theta_i}{ds}
\end{bmatrix}
\]

(2.26)

The open loop derivative of a frame \( p \) in a serially connected linkage with respect to the joint setting parameters (assuming all joints are rotary joints) is equivalent to
Figure 2.2 Closure of a 4 bar chain mechanism. The numerals next to the terminal frame indicate the iteration number. Closure was achieved in 8 iterations with a residual error less than $10^{-10}$. The graphic was generated by the function `dravgeom.m`. 
Figure 2.3 Logical structure of the connect.m algorithm. The diagram illustrates the hierarchy of subroutines but exact details of the algorithm are omitted.
2.4 THE CLOSURE PROBLEM

The position and orientation of frame $F_p$ may be characterised numerically by the components of its position vector $\mathbf{r}_p$ expressed in the base frame and the rotation matrix that converts vectors expressed in $\mathbf{F}_p$ to vectors expressed in $\mathbf{F}_0$

$$\mathbf{r}_{0,p} = \mathbf{F}_0^T \mathbf{F}_p \mathbf{r}_p$$

(2.27)

$$\mathbf{F}_0 = C_{0p} \mathbf{F}_p.$$  

(2.28)

Recalling equation 2.14

$$\mathbf{r}_{0,p} = \sum_{i=1}^{p} C_{0,i} \mathbf{r}_{i-1,i}$$

(2.29)

where

$$C_{0,i} = \left( \prod_{j=1}^{i-1} V_j^T Q_j^T \right) V_i^T$$

$$= \left( \prod_{j=1}^{i-1} C_{1j}^T (\alpha_{1j}) C_{2j}^T (\alpha_{2j}) C_{3j}^T (\theta_j) \right) V_i^T.$$  

(2.30)

For clarity, the following shorthand notation will be used

$$C_{1j} = C_1 (\alpha_{1j})$$

(2.31)

$$C_{2j} = C_2 (\alpha_{2j})$$

(2.32)

$$C_{3j} = C_3 (\theta_j).$$

(2.33)

Differentiating equation 2.29, gives the expression

$$\frac{d\mathbf{r}_{0,p}}{ds} = \sum_{i=1}^{p} \frac{dC_{0,i}}{ds} \mathbf{r}_{i-1,i}.$$  

(2.34)

From equation 2.30 and using the chain rule

$$\frac{dC_{0,i}}{ds} = \sum_{j=1}^{i-1} C_{0,j-1} \frac{dC_{j-1,i}}{ds} C_{j,i-1} V_i^T.$$  

(2.35)

Given the individual rates at which the $\theta_i$ vary with respect to $s$,

$$\frac{dC_{j-1,i}}{ds} = C_{1j}^T C_{2j}^T \frac{d}{d\theta_j} C_{3j}^T (\theta_j) \frac{d\theta_j}{ds}$$

$$= C_{1j}^T C_{2j}^T \frac{d}{d\theta_j} C_{3j}^T \frac{d\theta_j}{ds}$$

$$= \left( C_{1j}^T C_{2j}^T \frac{d}{d\theta_j} \right) \frac{d}{ds} C_{j-1,i} \frac{d\theta_j}{ds}.$$  

(2.36)
From equation 2.35

\[
\frac{dC_{0,i}^{T}}{ds} = \sum_{j=1}^{i-1} C_{0,j-1}^{T} \left( C_{1j}^{T} C_{2j}^{T} \mathbf{3} \right)^{\times} C_{j-1,i-1}^{T} V_{i}^{T} \frac{d\theta_{j}}{ds}
\]

\[
= \left( \sum_{j=1}^{i-1} C_{0,j-1}^{T} C_{1j}^{T} C_{2j}^{T} \mathbf{3} \frac{d\theta_{j}}{ds} \right)^{\times} C_{0,i}^{T}.
\]  

(2.37)

Returning to equation 2.34,

\[
\frac{dr_{0,p}}{ds} = \sum_{i=1}^{p} \left( \sum_{j=1}^{i-1} C_{0,j-1}^{T} C_{1j}^{T} C_{2j}^{T} \mathbf{3} \frac{d\theta_{j}}{ds} \right)^{\times} C_{0,i}^{T} r_{i-1,i}
\]

\[
= - \sum_{i=1}^{p} (C_{0,i}^{T} r_{i-1,i})^{\times} \left( \sum_{j=1}^{i-1} C_{0,j-1}^{T} C_{1j}^{T} C_{2j}^{T} \mathbf{3} \frac{d\theta_{j}}{ds} \right).
\]  

(2.38)

Expanding and regrouping equation 2.38,

\[
\frac{dr_{0,p}}{ds} = - \sum_{i=1}^{p} \mathbf{0}_{0,i}^{\times} C_{0,i-1}^{T} C_{1i}^{T} C_{2i}^{T} \mathbf{3} \frac{d\theta_{i}}{ds}.
\]  

(2.39)

Instead of representing the derivative of the orientation of \( F_{p} \) with a rotation matrix, it is more convenient to use Euler parameters (§A.1). The angular rate of change matrix is mapped to the 4 Euler parameters by

\[
\frac{d}{ds} \begin{bmatrix} \epsilon_{p} \\ \eta_{p} \end{bmatrix} = Z_{Eul}^{-1} \frac{d}{ds} C_{0,p}^{T}
\]

(2.40)

where

\[
C_{0,p}^{T} = \prod_{i=1}^{p} C_{i-1,i}^{T}
\]

(2.41)

and the derivative

\[
\frac{d}{ds} C_{0,p}^{T} = \sum_{i=1}^{p} C_{0,i-1}^{T} \left( C_{1i}^{T} C_{2i}^{T} \mathbf{3} \right)^{\times} C_{i,p}^{T} \frac{d\theta_{i}}{ds}.
\]  

(2.42)
### 2.4.3 Open Loop Derivative with Respect to All Geometric Parameters

#### Key Result

The open loop derivative of a frame $p$ in a serially connected linkage with respect to all geometric parameters is equivalent to

$$
\frac{d}{ds} g_p(a) = D_p \frac{da}{ds} = \sum_{i=1}^{p} \left[ \begin{array}{c} \Delta_{i}^{1,1} \\ \Delta_{i}^{2,1} \\ \Delta_{i}^{1,2} \\ \Delta_{i}^{2,2} \end{array} \right] T_i \frac{da_i}{ds}. 
$$

(2.43)

where

$$
\begin{align*}
\Delta_{i}^{1,1} &= -\gamma_{i,p} \times C_{0,i-1} Z_{i-1,i} - \gamma_{i-1,i} \times C_{0,i-1} Z_{i-1,i} T_z \\
\Delta_{i}^{2,1} &= C_{0,i} \\
\Delta_{i}^{2,1} &= Z_{Eul}^{i-1} C_{0,i-1} Z_{i-1,i} \\
\Delta_{i}^{2,2} &= 0_{4,3}, 
\end{align*}
$$

(2.44)

$s$ is an arbitrary parameter, and $a$ is the set of all model parameters including the additional closure parameters described at the beginning of §2.4.

The following derivation is for the derivatives of a frame $F_p$ in a chain of rigid bodies with no constraints. All geometric parameters are considered to be free including the lengths and twist angles of the members. The matrix $D_p$ is referred to as the open loop Jacobian matrix for frame $p$ and it will be used often in subsequent sections. If $j = 1 \ldots N$ and $N$ is the number of joints in the model,

$$
a_j = \begin{cases} 
[\alpha_{1j} \alpha_{2j} \alpha_{3j} \quad d_{1j} \quad d_{2j}]^T & \text{if joint } j \text{ is a rotary joint}, \\
[\alpha_{1j} \alpha_{2j} \quad d_{3j}]^T & \text{if joint } j \text{ is prismatic}, \\
[\alpha_{1j} \alpha_{2j} \alpha_{3j} \quad d_{1j} \quad d_{2j} \quad d_{3j}]^T & \text{if joint } j \text{ is a closure joint},
\end{cases} 
$$

(2.45)

$$
a = [a_1^T \quad a_2^T \quad \ldots \quad a_N^T]^T.
$$

(2.46)

For rotary joints the $\alpha_{3j}$ parameter corresponds to the joint setting and for prismatic joints $d_{3j}$ corresponds to the joint setting freedom.

Because each type of joint has a different number of parameters, the $T_j$ matrix is introduced to map the $a_j$ to a standard size for use in the expressions to follow. It is
defined as

\[
T_j = \begin{cases} 
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
\end{cases}
\]

if \( a_j \) is a rotary joint,

\[
T_j = \begin{cases} 
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{cases}
\]

if \( a_j \) is a prismatic joint,

\[
T_j = \begin{cases} 
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
\end{cases}
\]

if \( a_j \) is a closure joint.

The position and orientation of frame \( \mathcal{F}_p \) may be characterised numerically by the components of its position vector \( \vec{r}_{0,p} \) expressed in the base frame and the rotation matrix that converts vectors expressed in \( \mathcal{F}_p \) to vectors expressed in \( \mathcal{F}_0 \)

\[
r_{0,p} = \mathcal{F}_0^T \mathcal{F}_p \vec{r}_{0,p} \\
\mathcal{F}_0 = C_{0,i} \mathcal{F}_p.
\]  

Recalling equation 2.14,

\[
r_{0,p} = \sum_{i=1}^{p} C_{0,i} \vec{r}_{i-1,i} 
\]

where

\[
C_{0,i} = \left( \prod_{j=1}^{i-1} \vec{V}_j^T \vec{Q}_j^T \right) \vec{V}_i^T. 
\]

Differentiating equation 2.50, gives the expression

\[
\frac{dr_{0,p}}{ds} = \sum_{i=1}^{p} \frac{dC_{0,i}}{ds} \vec{r}_{i-1,i} + \sum_{i=1}^{p} C_{0,i} \frac{d\vec{r}_{i-1,i}}{ds}. 
\]

The second term in equation 2.52 contains the derivative of column matrix \( \vec{r}_{i-1,i} \),

\[
\frac{d\vec{r}_{i-1,i}}{ds} = \sum_{j=1}^{p} \frac{\partial \vec{r}_{i-1,i}}{\partial a_j} \frac{da_j}{ds}
\]
which is

\[
\frac{\partial r_{i-1,i}}{\partial a_j} = \begin{cases} 
0 & \text{if } j \neq i \\
0 & \text{if } j = i,
\end{cases}
\]

so that

\[
\frac{dr_{i-1,i}}{ds} = [0_3 \quad I_3] T_i \frac{da_i}{ds}.
\] (2.55)

The first term in equation 2.52 contains the derivative of a rotation matrix, which is derived in section A.2. The term can be written

\[
\frac{dC_{0,i} r_{i-1,i}}{ds} = \left( \sum_{j=1}^{i-1} C_{0,j} \Omega_{j-1,j} + C_{0,i} \Omega_{i-1,i} \right) \times C_{0,i} r_{i-1,i}
\]

\[
= - (C_{0,i} r_{i-1,i}) \times \left( \sum_{j=1}^{i-1} C_{0,j} \Omega_{j-1,j} + C_{0,i} \Omega_{i-1,i} \right)
\]

\[
= - (C_{0,i} r_{i-1,i}) \times \left( \sum_{j=1}^{i-1} C_{0,j} Z[I_3 \quad 0_3] T_j \frac{da_j}{ds} \right.
\]

\[
+ C_{0,i} Z T_z[I_3 \quad 0_3] T_i \frac{da_i}{ds} \right)
\] (2.56)

where

\[
T_z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.
\] (2.57)

The quantity \( \Omega_{j-1,j} \) is somewhat analogous to angular velocity but since \( da_j/ds \) is not a time derivative, it is more correctly termed an \textit{angular rate of change}. It is expressed in frame \( j - 1 \) and is the angular rate of change of frame \( j \) due to changes in the elements of \( a_j \). The \( Z_{j-1,j} \) matrix maps the derivative \( da_j/ds \) to \( \Omega_{j-1,j} \). Substituting into equation 2.55 the expression becomes

\[
\frac{dr_{0,i}}{ds} = \sum_{i=1}^{p} \left( C_{0,i} r_{i-1,i} \right) \times \left( \sum_{j=1}^{i-1} C_{0,j} Z_{j-1,j} [I_3 \quad 0_3] T_j \frac{da_j}{ds} \right)
\]

\[
- (C_{0,i} r_{i-1,i}) \times C_{0,i-1} Z_{i-1,i} T_z[I_3 \quad 0_3] T_i \frac{da_i}{ds}
\]

\[
+ \sum_{i=1}^{p} \left( C_{0,i} [0_3 \quad I_3] T_i \frac{da_i}{ds} \right).
\] (2.58)

Expanding and regrouping equation 2.58,

\[
\frac{dr_{0,i}}{ds} = \sum_{i=1}^{p} \left( \sum_{j=i+1}^{p} (C_{0,j} r_{j-1,j}) \times \right) C_{0,i-1} Z_{i-1,i} [I_3 \quad 0_3] T_i \frac{da_i}{ds}
\]
Instead of representing the derivative of the orientation of $\mathcal{F}_i$ with a rotation matrix, it is more convenient to use Euler parameters. The angular rate of change vector is mapped to the 4 Euler parameters by (cf. §A.1)

\[
\frac{d}{ds} \begin{bmatrix} \varepsilon_i \\ \eta_i \end{bmatrix} = Z_{\text{Eul}}^{-1} C_{0,i-1} \Omega_i,
\]

\[
= Z_{\text{Eul}}^{-1} C_{0,i-1} Z_i \begin{bmatrix} I_3 & 0_3 \end{bmatrix} T_i \frac{da_i}{ds}.
\]

The results from equations 2.58 and 2.60 can be combined into a single expression

\[
\frac{d}{ds} \varphi_p(\alpha) = \frac{d}{ds} \begin{bmatrix} \varphi_{0,p} \\ \eta_p \end{bmatrix} = \sum_{i=1}^{p} \begin{bmatrix} \Delta_{i,1}^{1,1} & \Delta_{i,2}^{1,2} \\ \Delta_{i,1}^{2,1} & \Delta_{i,2}^{2,2} \end{bmatrix} T_i \frac{da_i}{ds}.
\]

where

\[
\Delta_{i,1}^{1,1} = -q_{i-1,1}^T C_{0,i-1} Z_{i-1,i} - q_{i-1,1}^T C_{0,i-1} Z_{i-1,i} T_2,
\]

\[
\Delta_{i,1}^{1,2} = C_{0,i}
\]

\[
\Delta_{i,1}^{2,1} = Z_{\text{Eul}}^{-1} C_{0,i-1} Z_{i-1,i}
\]

\[
\Delta_{i,1}^{2,2} = 0_{4,3}.
\]

The frame is now parameterised with 7 parameters consisting of the 3 components of $r_{0,p}$ and the 4 Euler parameters defining the rotation matrix $C_{0,i}$ (see equation A.14). The open loop Jacobian can be expressed as a single composite matrix

\[
D_p = \begin{bmatrix}
\Delta_{1,1}^{1,1} & \Delta_{1,2}^{1,2} \\
\Delta_{1,1}^{2,1} & \Delta_{1,2}^{2,2}
\end{bmatrix} \cdots \begin{bmatrix}
\Delta_{N,1}^{1,1} & \Delta_{N,2}^{1,2} \\
\Delta_{N,1}^{2,1} & \Delta_{N,2}^{2,2}
\end{bmatrix}.
\]
The matrix of open loop derivatives for all frames is denoted

\[
J_{ol} = \begin{bmatrix}
D_1 \\
D_2 \\
D_3 \\
\vdots \\
D_N
\end{bmatrix}
\]  

(2.64)

### 2.4.4 Verification of Open Loop Jacobian

The open loop Jacobian matrix was formed for each frame of a geometric model and compared to the equivalent matrix evaluated using a finite difference technique. The model used was the general Delta robot model which has 30 body frames and 124 geometric parameters. The resulting Jacobian matrix was 210 by 124.

Comparison of the analytic and finite difference Jacobians yielded a maximum difference of \(2 \times 10^{-5}\). The condition number of the Jacobian matrix is 101 which implies that the closure problem has a low sensitivity to small errors in the Jacobian.

### 2.5 Closed Loop Jacobian

When all loops are closed, there are extra constraints on the motion of the mechanism. The constraints arise from the condition that the closure joints must remain closed and may be linearised for small motions. If all of the MCPC transformation parameters are included in the closed-loop derivative then the mechanism has a very large space of possible motions, as if all links are capable of extending, bending, and twisting.

Two methods of computing the closed loop Jacobian are presented. The first has the advantage of speed while the second reveals information about the constraint and physical feasibility of the mechanism.

#### 2.5.1 Chain Rule Method

**Key Result**

The Jacobian of a frame \(p\) in a closed mechanism with respect to all joint parameters is equivalent to

\[
\frac{\partial f}{\partial \alpha} \bigg|_{\nu=0} = \frac{\partial f}{\partial \alpha} - \frac{\partial f}{\partial \theta_{\text{free}}} \left( \frac{\partial \nu}{\partial \theta_{\text{free}}} \right)^{-1} \frac{\partial \nu}{\partial \alpha}
\]  

(2.65)

where \(\nu\) is as defined in equation 2.67.

The constraints on the motion of the mechanism are expressed by the conditions for closure (equations 2.22–2.23). Given a particular geometry \(\alpha\) and actuator setting \(\theta_{\text{act}}\),
the closure conditions define the value of the free joints $\theta_{\text{free}}$. This can be written

$$\theta_{\text{free}} : \quad v(\theta_{\text{act}}; \theta_{\text{free}}; a) = 0.$$  \hspace{1cm} (2.66)

The geometric model $g(\theta_{\text{act}}, \theta_{\text{free}}; a)$ of the mechanism gives the endpoint position and orientation as in equation 2.24. Let

$$v = \begin{bmatrix}
  g_{p_1} - g_{q_1} \\
  g_{p_2} - g_{q_2} \\
  \vdots \\
  g_{p_n} - g_{q_n}
\end{bmatrix}$$  \hspace{1cm} (2.67)

where joints $p_i$ and $q_i$ are a closure pair and $n_c$ is the number of closures in the mechanism. Spherical joints are a special case because the rotational freedoms are not constrained. For spherical joints the elements of $g_{p_i}$ and $g_{q_i}$ that correspond to rotational freedoms are discarded.

The endpoint position as measured by the measurement device is $f(\theta_{\text{act}}; \theta_{\text{free}}; a)$ and its partial derivative with respect to $a$ may be split into orthogonal components using chain rule as follows

$$\frac{\partial f}{\partial a} = \left. \frac{\partial f}{\partial a} \right|_{v=0} + \frac{\partial f}{\partial v} \frac{\partial v}{\partial a}$$

$$= \left. \frac{\partial f}{\partial a} \right|_{v=0} + \frac{\partial f}{\partial \theta_{\text{free}}} \left( \frac{\partial v}{\partial \theta_{\text{free}}} \right)^{-1} \frac{\partial v}{\partial a}.$$  \hspace{1cm} (2.68)

Rearranging gives

$$\left. \frac{\partial f}{\partial a} \right|_{v=0} = \frac{\partial f}{\partial a} - \frac{\partial f}{\partial \theta_{\text{free}}} \left( \frac{\partial v}{\partial \theta_{\text{free}}} \right)^{-1} \frac{\partial v}{\partial a}.$$  \hspace{1cm} (2.69)

The components of equation 2.69 can all be evaluated from the open loop Jacobian (§2.4.3). In particular,

$$\frac{\partial v}{\partial a} = \begin{bmatrix}
  D_{p_1} - D_{q_1} \\
  D_{p_2} - D_{q_2} \\
  \vdots \\
  D_{p_n} - D_{q_n}
\end{bmatrix}$$  \hspace{1cm} (2.70)

and $\frac{\partial v}{\partial \theta_{\text{free}}}$ can be formed by extracting columns of $\frac{\partial v}{\partial a}$ that correspond to the same freedoms as $\theta_{\text{free}}$. Again, for spherical joints, the rows of $D_{p_i}$ and $D_{p_i}$ that correspond to rotational freedoms are discarded.

The following example illustrates the use of this method on a simple 2 dimensional 4 bar closed mechanism as illustrated in Figure 2.4. Joint 1 is a rotary actuator and its value is fixed for the purposes of this evaluation of the closed loop Jacobian. The free joints are joints 4, 5 and 6, with joints 2 and 6 forming the loop closure. In the current configuration the mechanism is closed. The measurement device returns the position of
Figure 2.4 A 2 dimensional closed 4 bar chain. Joint 1 is actuated and the loop closure is between joints 2 and 6.
joint 5 in the form of 2 cartesian components. The geometry is defined by the matrix $\alpha$ which has a length parameter $a_j$ and angular offset parameter $\phi_j$ for each joint $j$. Thus $\alpha$ is defined as

$$\alpha = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & \phi_1 & \phi_2 & \phi_3 & \phi_4 & \phi_5 & \phi_6 \end{bmatrix}^T$$  (2.71)

and

$$\theta_{act} = \theta_1$$  (2.72)
$$\theta_{free} = [\theta_4 \ \theta_5 \ \theta_6]^T.$$  (2.73)

The geometric model gives

$$g_2 = \begin{bmatrix} a_1 \cos(\theta_1 + \phi_1) \\ a_1 \sin(\theta_1 + \phi_1) \\ \phi_2 + \theta_2 \end{bmatrix}$$  (2.74)
$$g_5 = \begin{bmatrix} a_3 \cos(\phi_3) + a_4 \cos(\theta_4 + \phi_4) \\ a_3 \sin(\phi_3) + a_4 \cos(\theta_4 + \phi_4) \\ \phi_5 + \theta_5 \end{bmatrix}$$  (2.75)
$$g_6 = \begin{bmatrix} a_3 \cos(\phi_3) + a_4 \cos(\theta_4 + \phi_4) + a_5 \cos(\theta_5 + \phi_5) \\ a_3 \sin(\phi_3) + a_4 \sin(\theta_4 + \phi_4) + a_5 \sin(\theta_5 + \phi_5) \\ \phi_6 + \theta_6 \end{bmatrix}$$  (2.76)

and because the measurement device is cartesian,

$$f_5 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} g_5$$
$$= \begin{bmatrix} a_3 \cos(\phi_3) + a_4 \cos(\theta_4 + \phi_4) \\ a_3 \sin(\phi_3) + a_4 \sin(\theta_4 + \phi_4) \end{bmatrix}.$$  (2.77)

The constraint equations for closure are

$$v = g_2 - g_6$$
$$= \begin{bmatrix} a_1 \cos(\theta_1 + \phi_1) - a_3 \cos(\phi_3) - a_4 \cos(\theta_4 + \phi_4) - a_5 \cos(\theta_5 + \phi_5) \\ a_1 \sin(\theta_1 + \phi_1) - a_3 \sin(\phi_3) - a_4 \sin(\theta_4 + \phi_4) - a_5 \sin(\theta_5 + \phi_5) \\ \phi_2 + \theta_2 - \phi_6 - \theta_6 \end{bmatrix}$$
$$= 0.$$  (2.78)

An additional constraint is required at the joint closure because any value of $\theta_6$ satisfying $\theta_6 = \phi_2 + \theta_2 - \phi_6$ will satisfy the constraint. The constraint is applied by fixing the value of $\theta_6$ and zeroing the columns of $\frac{\partial v}{\partial \alpha}$ that correspond to joints 2 and 6 (the affected
elements are marked with an asterisk in equation 2.81).

The required derivatives are

\[
\frac{\partial f_4}{\partial a} = \begin{bmatrix}
0 & 0 & \cos(\phi_3) & \cos(\theta_4 + \phi_4) & 0 & 0 & 0 & 0 & -a_3 \sin(\phi_3) \\
0 & 0 & \sin(\phi_3) & \sin(\theta_4 + \phi_4) & 0 & 0 & 0 & a_3 \cos(\phi_3) \\
& & -a_4 \sin(\theta_4 + \phi_4) & 0 & 0 \\
ar_4 \cos(\theta_4 + \phi_4) & 0 & 0
\end{bmatrix}
\] (2.79)

\[
\frac{\partial f_4}{\partial \theta_{\text{free}}} = \begin{bmatrix}
-a_4 \sin(\theta_4 + \phi_4) & 0 & 0 \\
a_4 \cos(\theta_4 + \phi_4) & 0 & 0
\end{bmatrix}
\] (2.80)

\[
\frac{\partial \mathbf{v}}{\partial a} = \begin{bmatrix}
c(\theta_1 + \phi_1) & 0 & -c(\phi_3) & -c(\theta_4 + \phi_4) & -c(\theta_5 + \phi_5) & 0 \\
s(\theta_1 + \phi_1) & 0 & -s(\phi_3) & -s(\theta_4 + \phi_4) & -s(\theta_5 + \phi_5) & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\] (2.81)

\[
\frac{\partial \mathbf{v}}{\partial \theta_{\text{free}}} = \begin{bmatrix}
a_4 \sin(\theta_4 + \phi_4) & a_5 \sin(\theta_5 + \phi_5) & 0 \\
a_4 \cos(\theta_4 + \phi_4) & -a_4 \cos(\theta_5 + \phi_5) & 0 \\
0 & 0 & 1
\end{bmatrix}
\] (2.82)

The closed loop Jacobian is

\[
\left. \frac{\partial f_4}{\partial a} \right|_{\mathbf{v}=0} = \frac{\partial f_4}{\partial a} - \frac{\partial f_4}{\partial \theta_{\text{free}}} \left( \frac{\partial \mathbf{v}}{\partial \theta_{\text{free}}} \right)^{-1} \frac{\partial \mathbf{v}}{\partial a}
\] (2.83)

and using

\[
a = \begin{bmatrix} 0.95 & 0 & 0.9 & 1.1 & 0.9 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T
\] (2.84)

\[
\theta_{\text{act}} = \theta_1 = \frac{\pi}{2} + 0.4
\] (2.85)

\[
\theta_{\text{free}} = \begin{bmatrix} 1.9270 \\
3.3158 \\
0
\end{bmatrix}
\] (2.86)

the result is

\[
\frac{\partial f_4}{\partial a} \big|_{\mathbf{v}=0} = \begin{bmatrix}
-0.2134 & 0 & 0.0614 & -0.1762 & 0.9530 & 0 \\
-0.0794 & 0 & -0.3492 & 1.0014 & 0.3545 & 0 \\
& & -0.8823 & 0 & -0.1486 & 0 & 0 & 0 \\
& & -0.3282 & 0 & 0.8447 & 0 & 0 & 0
\end{bmatrix}
\] (2.87)

which may be verified by finite difference approximation, or the nullspace decomposition method. Note that \( \mathbf{v} = 0 \) at the given configuration.
2.5.2 Nullspace Decomposition Method

**Key Result**

The Jacobian of a frame \( p \) in a closed mechanism with respect to all joint parameters is equivalent to

\[
\frac{\partial f}{\partial \alpha} \bigg|_{\nu=0} = MD_p N \psi^{-1}
\]

(2.88)

where

\[
N = \text{null} \left( \frac{\partial \nu}{\partial \alpha} \right)
\]

(2.89)

\( \psi \) is an orthonormal transformation matrix defined below, and \( \nu \) is as defined in equation 2.67.

If a suitable \( \theta_{\text{free}} \) has been found that satisfies the conditions for closure, then the closure constraints can be expressed

\[
\frac{\partial \nu}{\partial \alpha} = 0
\]

(2.90)

where \( \delta \alpha \) is an arbitrary column matrix in \( \mathbb{R}^n \). To find the nullspace of equation 2.90, either singular value decomposition (§A.3) or QR decomposition (§A.4) may be used. Using SVD,

\[
\frac{\partial \nu}{\partial \alpha} = USV^T
\]

(2.91)

where \( V \) is an \( n \times n \) orthonormal matrix and

\[
\text{diag}(S) = \begin{bmatrix} \sigma_1 & \sigma_2 & \cdots & \sigma_r & 0 & \cdots & 0 \end{bmatrix}^T.
\]

(2.92)

Diagonal elements of \( S \) that are zero or near zero correspond to the closed-loop freedoms of the mechanism. If \( r \) is the index of the last significant singular value then the last \( n - r \) columns of \( V \) form the required nullspace \( N \).

Using QR decomposition,

\[
\frac{\partial \nu}{\partial \alpha} = QRE^T
\]

(2.93)

where \( R \) is arranged such that \( |r_{ii}| \geq |r_{i+1,i+1}| \). If \( r \) is the index of the last significantly non-zero diagonal element of \( R \) then the last \( n - r \) columns of \( Q \) form the required nullspace \( N \). Note that the SVD and QR methods will not in general produce the same \( N \) but this is unimportant since the result in either case is a basis for the nullspace of equation 2.90 and in the next step, \( N \) is transformed into a unique ‘canonical’ form.

The rows of \( N \) correspond to the full set of MCPC parameters in sequence. Since the columns of \( N \) form a basis for the range of possible movements of the mechanism it is desirable to apply an orthogonal transformation so that the influence of the \( n - r \) independent parameters is separately discernable. This is done by post-multiplying \( N \)
by the inverse of an orthonormal matrix $\psi$ so that

$$N' = N\psi^{-1}$$ (2.94)

where $\psi$ is formed from the $n - r$ rows of $N$ that correspond to controlled freedoms. The resulting matrix has the structure of an identity matrix with $r$ additional rows interspersed that describe the allowable motions of the passive joints.

Column matrix $a \in \mathbb{R}^n$ is as defined in equation 2.45. Let $a_c \in \mathbb{R}^{n-r}$ be the column matrix of all controlled parameters, that is, $a$ with parameters corresponding to $\theta_{free}$ removed:

$$\frac{da}{ds} = N'\frac{da_c}{ds}.$$ (2.95)

The motions of some joint frame $p$ may then be found by multiplying equation 2.95 by the open-loop derivatives

$$\frac{df_p}{ds} = MD_pN'\frac{da_c}{ds},$$ (2.96)

and the closed loop Jacobian is

$$\left.\frac{\partial f_p}{\partial a}\right|_{\psi=0} = MD_pN'.$$ (2.97)

Returning to the example of the 2 dimensional 4 bar closed mechanism and using the same configuration and geometry defined in equations 2.84–2.86, the SVD of equation 2.81 gives

$$\text{diag}(S) = \begin{bmatrix} 2.0262 & 1.9045 & 0 \end{bmatrix}^T$$ (2.98)

thus $r = 2$, $n = 12$, and the last $n - r$ columns of $V$ are

$$N = \begin{bmatrix} -0.8523 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0000 & 0.6199 & 0.0585 & 0 & 0.0777 & 0 & 0.5401 & -0.0601 & -0.5576 & 0 & 0 & 0 \\ 0.1164 & 0.1042 & 0.4922 & 0 & -0.4524 & 0 & -0.0631 & 0.5294 & -0.0138 & 0 & 0 & 0 \\ -0.3211 & 0.6188 & -0.0213 & 0 & -0.0619 & 0 & -0.3367 & 0.0533 & 0.3453 & 0 & 0 & 0 \\ -0.0628 & -0.0707 & 0.8340 & 0 & 0.1449 & 0 & -0.0107 & -0.1713 & 0.0372 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0089 & -0.0143 & 0.1555 & 0 & 0.8468 & 0 & -0.0625 & 0.1769 & 0.0388 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.2694 & -0.3210 & 0.0337 & 0 & -0.1020 & 0 & 0.6999 & 0.1027 & 0.2903 & 0 & 0 & 0 \\ -0.0652 & -0.0019 & -0.1813 & 0 & 0.1746 & 0 & 0.0562 & -0.7976 & -0.0283 & 0 & 0 & 0 \\ 0.2834 & 0.3371 & -0.0080 & 0 & 0.0806 & 0 & 0.3064 & -0.0772 & 0.6902 & 0 & 0 & 0 \end{bmatrix}. (2.99)

The controlled freedoms are $a_c = [a_1 \ a_2 \ a_3 \ a_4 \ a_5 \ a_6 \ \phi_1 \ \phi_2 \ \phi_3 \ \phi_6]$ and these correspond to rows 1–9 and 12 of the nullspace matrix $N$. The orthonormal matrix $\psi$ is composed of rows 1–9 and 12 of $N$ and
Finally, from equations 2.79 and 2.97,

$$
\frac{\partial f_i}{\partial \mathbf{a}} \bigg|_{\mathbf{a}=0} = \frac{\partial f_i}{\partial \mathbf{a}} N' = \begin{bmatrix}
-0.2134 & 0 & 0.0614 & -0.1762 & 0.9530 & 0 \\
-0.0794 & 0 & -0.3492 & 1.0014 & 0.3545 & 0 \\
-0.8823 & 0 & -0.1486 & 0 & 0 & 0 \\
-0.3282 & 0 & 0.8447 & 0 & 0 & 0
\end{bmatrix}
$$

which is identical to equation 2.87.

Using this method, it is possible to gain some insight into the conditioning and physical feasibility of the mechanism. The 3-dimensional 4-bar closed loop mechanism will be used as an example. Firstly, consider the configuration shown in Figure 2.5a. In this situation,

$$
\text{diag} \left( \mathbf{S} \right) = \begin{bmatrix} 2.8971 & 2.8336 & 2.7281 & 0 & 0 & 0 \end{bmatrix}^T
$$

$$
\mathbf{U} \in \mathbb{R}^{7 \times 7}
$$

$$
\mathbf{V} \in \mathbb{R}^{36 \times 36}
$$

The open loop Jacobian matrix as defined in section 2.4.3 has varying numbers of columns for each joint frame depending on the type of joint. The computer program actually returns 6 columns per joint frame with the missing columns zeroed, thus the dimension of \( \mathbf{V} \) is \( \mathbb{R}^{6N \times 6N} \). This has no effect on the results.

For a mechanism specified in MCPC format the following scheme is used to determine which parameters are controlled and which are free: Firstly, begin with the set of all joints. Next remove all joints that are actuated (it is assumed that the mechanism has full mobility and that the correct number of actuated joints are present). Next, one joint from each closure pair is removed from the set in order to constrain the extra freedom introduced by defining the same joint twice. For closures that correspond to spherical joints, both joints of the closure pair are removed from the set because the orientation
constraint is not enforced allowing both joints to rotate independently. Finally, if any extra reference frames are defined such as a tool frame for example, these joints are removed from the set. The remaining joints are the free joints of the mechanism.

Next consider the set of all of the parameters of the model including the ones that correspond to the joint freedoms. The controlled parameters are all of the parameters in this set except for the ones that correspond to the joint freedom of the free joints.

In this example, there are 6 joints. Joint 1 is the actuator and joints 2 and 6 form a spherical closure pair. There are no tool frames defined in this model so the remaining joints 3, 4, and 5 are the free joints. The parameters corresponding to rotation about axis 3 for joints 3, 4, and 5 are removed from the set of all parameters and the remaining parameters define \( a_c \).

From equation 2.102, the number of significant non-zero singular values is \( r = 3 \). Which agrees with the number of free joints. Matrix \( \psi \) has condition number 4.0227. This mechanism has the correct mobility, and is in a non-singular configuration.

Next consider the same mechanism in the configuration shown in Figure 2.5b. The actuator angle \( \theta_1 = 0 \) and the geometry is such that joints 2 and 4 are coincident. This is known as a gain of mobility singularity because joints 2 and 4 are free to rotate. This time,

\[
\text{diag}(S) = \begin{bmatrix} 2.8479 & 2.3766 & 2.2361 & 0 & 0 & 0 \end{bmatrix}^T
\]

\( r = 3 \) \hspace{1cm} (2.105)

\[
\text{cond}(\psi) = 1.9879 \times 10^{12}.
\]

(2.106)

The poor conditioning of \( \psi \) is because some model parameters now act in the same sense and thus the corresponding columns of \( \psi \) are nearly identical.

A similar situation is when the actuator angle \( \theta_1 = \pi \) shown in Figure 2.5c. In this singularity, it is uncertain how the mechanism will behave when it is driven out of this configuration. There are 2 possibilities: One where the mechanism remains collapsed and one where it opens into a parallelogram. Once again \( r = 3 \) and the condition number of \( \psi \) is very high.

Figure 2.5d shows the same mechanism as the previous examples except that the length of the actuator link is now 0. This time \( r = 3 \) and the condition number of \( \psi \) is just 3.7417. The only indication that the mechanism is immobile is that column 3 of the closed loop Jacobian matrix is zero except for the rotational elements corresponding to joints 1 and 2. In contrast, Figure 2.5e shows the mechanism with the length of the link between joints 4 and 5 set to 0. Here \( r = 3 \) and condition number of \( \psi \) is once again very high.

If both joints 1 and 3 are defined to be actuators, as in Figure 2.5f then the mechanism is overconstrained. The free joints are 4 and 5, however \( r = 3 \). This mismatch may be
Figure 2.5 Example mechanisms with differing conditioning and constraints. The mechanisms are 2 dimensional with the loop closure occurring at joint 2.
used to detect overconstraint. Another example of overconstraint is shown in Figure 2.5g. An extra member effectively turns the mechanism into a truss. There are 4 free joints numbered 3, 4, 5 and 7 and

\[
\text{diag}(S) = \begin{bmatrix} 3.00 & 3.00 & 2.92 & 2.64 & 2.28 & 2.18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \end{bmatrix}^T
\]

(2.108)

\[
U \in \mathbb{R}^{14 \times 14}
\]

(2.109)

\[
V \in \mathbb{R}^{48 \times 48}.
\]

(2.110)

Over-constraint is detected by recognising that \( r = 6 \) but there are only 4 free joints.

Finally, addition of an extra non-actuated link results in an underconstrained mechanism (see Figure 2.5h) with

\[
\text{diag}(S) = \begin{bmatrix} 3.2462 & 3.2014 & 3.1036 & 0 & 0 & 0 & 0 \\ \end{bmatrix}^T
\]

(2.111)

\[
U \in \mathbb{R}^{7 \times 7}
\]

(2.112)

\[
V \in \mathbb{R}^{42 \times 42}.
\]

(2.113)

There are 4 free joints numbered 3, 4, 5 and 6 but \( r = 3 \). This means that there is a column in the nullspace for which there is no controlled freedom.

In summary, these examples have shown that it is possible to detect conditions such as underconstraint and overconstraint as well as singularities by analysis of the closed loop Jacobian. If \( n_f \) is the number of free joints then

- \( r > n_f \) indicates overconstraint,
- \( r = n_f \) indicates proper constraint,
- \( r < n_f \) indicates underconstraint,

and poor conditioning of \( \psi \) indicates poor physical conditioning of the mechanism.

### 2.6 Nominal Geometric Model for the Delta Robot

Various geometric solutions for the Delta robot have been presented by a number of authors. The earliest geometric solution was presented by Sternheim [1987]. Pierrot and Fournier [1990] presented a geometric solution that facilitates faster computation and later improved on their inverse geometric solution (cf. Pierrot et al. [1991]). In addition Clavel [1991] has presented a geometric solution in his doctoral dissertation.

Two direct geometric solutions are presented here. The first is Clavel’s solution with comments about the numerical stability of the equations, while the second is a variation on Clavel’s solution that has better numerical stability and is based on a more geometrical interpretation. A non-singular inverse geometric solution is then presented.
2.6.1 Direct Geometric Solution

Geometric Solution of R. Clavel

With reference to Figure 2.6: Given the three actuator angles, $\theta_1$, $\theta_2$, and $\theta_3$ it is possible to find the position of the end effector $P$ given the base platform radius $r_b$, the nacelle radius $r_n$ and the arm lengths $m$ and $n$. This is done by determining the position of points $B_1$, $B_2$, and $B_3$ in the base reference frame and constructing three spheres of radius $m$ originating at the three $B_i$ points ($i = 1, 2, 3$). The three edges of the nacelle touch the spheres at the three points $C_i$. Since $OA_i$ is parallel to $PC_i$, the three $C_i$ are related to the point $P$ by linear translations and it is possible to solve the resulting equations in terms of the components of point $P$.

![Figure 2.6 Nominal geometric model of the Delta robot (direct geometric solution).](image)

The base reference frame has origin $O$, and is situated at the centre of the base platform with the laxis aligned to pass through point $A_1$. The 3 actuated arms are arranged symmetrically about $O$ at 120 degrees to each other. In this frame, the position
of the points $B_i$ is expressed as follows:

$$b_i = \begin{bmatrix} (r_b + n \cos(\theta_i)) \cos(\psi_i) \\ (r_b + n \cos(\theta_i)) \sin(\psi_i) \\ -n \sin(\theta_i) \end{bmatrix}$$  \quad (2.114)

where

$$\psi_i = (i - 1) \frac{2\pi}{3}. \quad (2.115)$$

If point $C_i$ has coordinates $[c_{i1} \ c_{i2} \ c_{i3}]$ and point $B_i$ has coordinates $[b_{i1} \ b_{i2} \ b_{i3}]$ where $i$ refers to the linkage under consideration then

$$m^2 = (c_{i1} - b_{i1})^2 + (c_{i2} - b_{i2})^2 + (c_{i3} - b_{i3})^2. \quad (2.116)$$

Unless the mechanism has passed through a mechanical singularity, the nacelle is constrained to lie in the horizontal plane hence the position of the point $C_i$ is

$$c_i = \begin{bmatrix} p_1 + r_n \cos(\psi_i) \\ p_2 + r_n \sin(\psi_i) \\ p_3 \end{bmatrix}. \quad (2.117)$$

Substituting equation 2.117 into equation 2.116

$$m^2 = (p_1 - e_i)^2 + (p_2 - f_i)^2 + (p_3 - g_i)^2 \quad (2.118)$$

where

$$\begin{bmatrix} e_i \\ f_i \\ g_i \end{bmatrix} = \begin{bmatrix} b_{i1} - r_n \cos(\psi_i) \\ b_{i2} - r_n \sin(\psi_i) \\ b_{i3} \end{bmatrix}. \quad (2.119)$$

Physically this corresponds to a problem of finding the intersection of three generally positioned spheres of equal diameter. There are in general two real solutions,

$$\begin{align*}
p_1 &= \frac{h_5}{h_2} p_3 - \frac{h_4}{h_2} \\ p_2 &= \frac{h_1}{h_2} p_3 - \frac{h_3}{h_2} \\ p_3 &= \frac{-M \pm \sqrt{M^2 - 4LN}}{2L} \quad (2.122)
\end{align*}$$

where

$$\begin{align*}
h_1 &= e_1 g_2 - e_1 g_3 + e_2 g_3 - e_2 g_1 + e_3 g_1 - e_3 g_2 \\ h_2 &= e_1 f_3 - e_1 f_2 + e_2 f_1 - e_2 f_3 + e_3 f_2 - e_3 f_1 \\ h_3 &= e_1 D_3 - e_1 D_2 + e_2 D_1 - e_2 D_3 + e_3 D_2 - e_3 D_1
\end{align*} \quad (2.123 \text{ to } 2.125)$$
\[ h_4 = f_1D_2 - f_1D_3 + f_2D_3 - f_2D_1 + f_3D_1 - f_3D_2 \] (2.126)
\[ h_5 = f_1g_3 - f_1g_2 + f_2g_1 - f_2g_3 + f_3g_2 - f_3g_1 \] (2.127)

and

\[ D_i = \frac{m^2 - e_i^2 - f_i^2 - g_i^2}{2} \] (2.128)
\[ L = \frac{h_2^2 + h_1^2}{h_2^2} + 1 \] (2.129)
\[ M = -2 \left( \frac{h_5h_4 + h_3h_1}{h_2^2} \right) + h_5e_1 + h_1f_1 + g_i \] (2.130)
\[ N = \frac{h_2^2 + h_3^2}{h_2^2} + 2 \left( \frac{h_4e_1 + h_3f_1}{h_2} + D_i \right). \] (2.131)

**Numerical Stability of Clavel’s Solution**

Clavel’s geometric solution works perfectly well for robots where the length of the actuated arms is smaller than the size of the base platform. If the ratio

\[ K = \frac{r_b - r_n}{n} \] (2.132)

is greater than 1.0 then the solution is non singular. For geometries in which the ratio \( K \) is less than 1.0 a problem occasionally arises because \( h_2 \) may be zero causing a singularity in any expression using \( h_2 \) as a denominator. To show this, substitute equations 2.114, 2.119, and 2.132 into equation 2.124 to give

\[
\begin{align*}
  h_2 &= \sin(\psi_1) (K + \cos(\theta_1)) ([K + \cos(\theta_2)] \cos(\psi_2) - [K + \cos(\theta_3)] \cos(\psi_3)) \\
  &+ \sin(\psi_2) (K + \cos(\theta_2)) ([K + \cos(\theta_3)] \cos(\psi_3) - [K + \cos(\theta_1)] \cos(\psi_1)) \\
  &+ \sin(\psi_3) (K + \cos(\theta_3)) ([K + \cos(\theta_1)] \cos(\psi_1) - [K + \cos(\theta_2)] \cos(\psi_2))
\end{align*}
\] (2.133)

which is quadratic in \( K \). Let \( K_{\text{crit}} \) be the value of \( K \) where \( h_2 = 0 \). Therefore

\[ K_{\text{crit}} = \frac{-q_1 \pm \sqrt{q_1^2 - 4q_2}}{2}. \] (2.134)

The quantities \( q_1 \) and \( q_2 \) are functions of the actuator coordinates \( \theta_1, \theta_2 \) and \( \theta_3 \). Let the \( K \) value of a Delta robot with a certain geometry be \( K_r \). If, at some point in actuator space, \( K_{\text{crit}} = K_r \) then \( h_2 \) is zero and the geometric equations break down. Being quadratic, equation 2.134 always has a maximum so that if \( K_r \) is beyond the maximum, \( h_2 \) cannot become zero at any point in the workspace. The maximum value of \( K_{\text{crit}} \) is \( K_{\text{crit}} = 1 \).
### 2.6 NOMINAL GEOMETRIC MODEL FOR THE DELTA ROBOT

#### A Variation on Clavel's Geometric Solution

Each point $B_i$ (see equation 2.114) is the centre of a sphere of radius $m$ at the end of each actuated arm. The three spheres, radius $m$, centred on the $B_i$ touch the triangular nacelle. The condition that the nacelle must remain horizontal gives rise to a finite number of positions that the nacelle can occupy. The problem is greatly simplified by translating all three spheres inwards toward the centre of the nacelle by the nacelle radius $r_n$ such that the three points $C_i$ become coincident. This reduces the problem to finding the intersection of three spheres.

A linear translation is applied to the $B_i$ points

$$b'_i = b_i + \begin{bmatrix} -r_n \cos(\psi_i) \\ -r_n \sin(\psi_i) \\ 0 \end{bmatrix}. \quad (2.135)$$

A coordinate frame $\mathcal{F}_s$ is established with origin at $B'_3$ that is oriented so that $B'_1$ lies on the 1-axis and $B'_2$ lies on the 1-2 plane. The unit basis vectors that describe the frame $\mathcal{F}_s$ are

$$\hat{1}_s = \frac{(b'_1 - b'_2)}{|b'_1 - b'_2|} \quad (2.136)$$

$$\hat{3}_s = \frac{(b'_1 - b'_3) \times (b'_2 - b'_3)}{|(b'_1 - b'_3) \times (b'_2 - b'_3)|} \quad (2.137)$$

$$\hat{2}_s = \hat{3} \times \hat{1}. \quad (2.138)$$

The three spheres have coordinates $[a \ 0 \ 0]^T$, $[b \ c \ 0]^T$, and $[0 \ 0 \ 0]^T$ in frame $\mathcal{F}_s$.

$$a = (b'_1 - b'_3)^T \hat{1}_s \quad (2.139)$$

$$b = (b'_2 - b'_3)^T \hat{1}_s \quad (2.140)$$

$$c = (b'_2 - b'_3)^T \hat{2}_s. \quad (2.141)$$

The point of intersection in frame $\mathcal{F}_s$ is described by $p_s$ with components

$$p_1 = \frac{a}{2} \quad (2.142)$$

$$p_2 = \frac{c}{2} + \frac{(b^2 - ab)}{2c} \quad (2.143)$$

$$p_3 = -\sqrt{m^2 - \frac{a^2}{4} - \left(\frac{c}{2} + \frac{b^2 - ab}{2c}\right)}. \quad (2.144)$$

The expression for $p_3$ is the negative solution of a quadratic equation. The positive solution also exists and corresponds physically to the upper intersection of the spheres.
Expressing $p_s$ in the base frame

$$p = b_3' + C_{0s}p_s$$  \hspace{1cm} (2.145)

where

$$C_{0s} = \mathcal{F}_0 \mathcal{F}_s^T.$$  \hspace{1cm} (2.146)

### 2.6.2 Inverse Geometric Solution

Several inverse geometric solutions have been proposed for the Delta robot, notably those of Clavel [1991], Codourey [1991] and Pierrot et al. [1991]. The following solution has no numerical singularities if the endpoint position $P$ is physically attainable.

For geometric compatibility (or closure) on any of the three linkages between base and nacelle, the distance between points $B$ and $C$ must be equal to $m$. Closure is calculated for each linkage separately by defining a frame $\mathcal{F}_u$ (see Figure 2.7) so that the 3axis is coincident to the base frame’s 3axis with the 1axis pointing toward $A_i$ ($i = 1, 2, 3$). Expressing the endpoint position $P$, point $C_i$ and point $A_i$ in $\mathcal{F}_u$, the closure constraint equations are used to solve for actuator angle $\theta_i$. Interpreted geometrically, the point $B_i$ will lie on the intersection of the circular locus of the actuator arm and the spherical locus of the attachment point on the parallelogram.

The coordinate transformation matrix from the base reference frame to $\mathcal{F}_u$ is $C_{u0} = C_3(\phi_i)$ (see section A.1). If the position vectors for points $P$, $A_i$, $B_i$, and $C_i$ expressed in $\mathcal{F}_u$ are $p_u$, $a_{ui}$, $b_{ui}$, and $c_{ui}$ respectively, then

$$a_{ui} = \begin{bmatrix} r_b & 0 & 0 \end{bmatrix}^T$$  \hspace{1cm} (2.147)

$$p_u = C_{u0}p_0$$  \hspace{1cm} (2.148)

$$c_{ui} = p_u + \begin{bmatrix} r_n & 0 & 0 \end{bmatrix}^T$$  \hspace{1cm} (2.149)

Point $C_i$ is the centre of a sphere of radius $m$. The locus of the point $B_i$ intersects this sphere. The components of $b_{ui}$ and $c_{ui}$ are $b_1$, $b_2$, $b_3$ and $c_1$, $c_2$, $c_3$ respectively thus

$$m^2 = (b_1 - c_1)^2 + (b_2 - c_2)^2 + (b_3 - c_3)^2$$  \hspace{1cm} (2.150)

Point $A_i$ is the centre of a circle in the 1-2 plane of radius $n$ representing the locus of point $B_i$. If the components of $a_{ui}$ are written $a_1$, $a_2$, and $a_3$ then

$$n^2 = (b_1 - a_1)^2 + (b_3 - a_3)^2$$  \hspace{1cm} (2.151)

There exist in general two real solutions to the intersection of a circle and a sphere. Solving equations 2.150 and 2.151 simultaneously with $a_{ui} = \begin{bmatrix} r_b & 0 & 0 \end{bmatrix}^T$ yields the com-
Figure 2.7 Nominal geometric model of the Delta robot (inverse geometric solution).
ponents of $b_{ui}$,

$$b_1 = \frac{(r_b + FG) \pm \sqrt{(r_b + FG)^2 - (G^2 + 1) (r_b^2 + F^2 - n^2)}}{(G^2 + 1)}$$  \hspace{1cm} (2.152)$$

$$b_2 = 0$$  \hspace{1cm} (2.153)$$

$$b_3 = F - Gb_1$$  \hspace{1cm} (2.154)$$

where

$$F = \frac{n^2 - m^2 - r_b^2 + \mid u_{ui} \mid^2}{2c_3}$$  \hspace{1cm} (2.155)$$

and

$$G = \frac{c_1 - r_b}{c_3}.$$  \hspace{1cm} (2.156)$$

The actuator joint angle is given by

$$\theta_i = \arccos \left( \frac{b_1 - r_b}{n} \right)$$  \hspace{1cm} (2.157)$$

or

$$\theta_i = \arcsin \left( \frac{b_3}{n} \right).$$  \hspace{1cm} (2.158)$$

Two solutions for $b_1$ are obtained by taking either the positive or negative root in equation 2.152. The preferred solution is that which gives the smallest (acute) value of $\theta_i$. This corresponds to the positive solution of the radical.
Chapter 3

The Calibration Problem

This chapter presents a set of techniques necessary for solving the calibration problem. Calibration of robots is the process of finding a set of parameters for a suitable geometric model that predict the pose of a real robot over some defined part of its work-space. This is done using data obtained from experimental pose measurements of the real robot. Roth et al. [1987] divide the robot calibration process into 4 distinct phases and this framework has been widely adopted by subsequent authors. The 4 phases are: modelling, measurement, identification, and implementation (cf. §1.1.1 for a full description).

Modelling involves the development of a mathematical model for the mechanism that accounts for the types of errors observed. Such a model has been presented in §2.3. Details of the measurement process used for the experimental calibration of the Delta robot are given in §4.4.2. Identification is the process of obtaining the best fit of the model parameters to the measurement information. If the model is appropriate for the physical situation and the global best fit is obtained by some identification process, then the model parameters will represent the true geometry of the mechanism. In practice the result is made much less reliable by measurement noise and there is also the possibility of multiple solutions to the best-fit criterion. Nevertheless, a model using properly identified parameters will generally simulate the actuator-setting to endpoint-position relationship of the real mechanism well enough to improve positioning accuracy over most of the measured work-space. The identification problem is discussed in detail in §3.1. The final phase of calibration, implementation, involves changing the robot controller to account for and correct the measurement errors. This involves altering the geometric model contained in the controller (usually the nominal geometric model) or replacing it completely with a more advanced model. This is discussed in §3.2.

The identification and implementation phases of the calibration process both require the solution of non-linear systems of equations. In addition, the selection of sets of measurement points can be optimised relative to some observability criterion. Because of this, several non-linear optimisation techniques were investigated. These are summarised in §3.3, while §3.4 discusses 3 different non-linear problems that must be solved for calibration and explains the reasoning behind the solution methods chosen.
The pose measurement data always contains noise, which affects the result of the identification problem. Techniques for investigating and quantifying the influence of measurement noise are given in §3.5.

### 3.1 Identification

An identification or model-fitting problem generally has the following structure. An experiment is performed in which \( m \) position measurements are obtained from some measuring device. The \( i \)’th measurement is denoted \( y_i \) and it is taken when the joints are set to a configuration described by \( \theta_i \). A geometric model \( f \) exists that has \( n \) geometric parameters, collectively denoted \( a \). The model is designed to output values that anticipate the output of the measurement device so that

\[
\varepsilon_i = y_i - f(\theta_i; a)
\]

quantifies the discrepancy between the measurement and the model prediction. The aim is to minimise some combination of the \( \varepsilon_i \) by altering the components of \( a \) in order to fit the output of the model to the measurements. The simplest cost function for the solution of the calibration problem is the weighted sum of squares which is expressed:

\[
\rho = \sum_{i=0}^{m} \varepsilon_i^T W \varepsilon_i
\]

where \( W \) is a constant weighting matrix which scales the components of \( \varepsilon_i \) so that the contribution of each component to the cost function is roughly the same magnitude.

It is likely that the model is not capable of exactly representing the true mechanism. If every point in the workspace was measured, the set of model parameters that provided the best fit would be the closest representation that the geometric model can generate. This set is denoted \( \tilde{a} \). In practice only a discrete subset of measurement points can be measured. The set of model parameters that provide the best fit to the given set of measurements is denoted \( \hat{a} \) and

\[
\hat{a} = \lim_{m \to \infty} \tilde{a}(m).
\]

In general the models will be non-linear and therefore only non-linear solution techniques will be examined. All non-linear solution techniques are iterative meaning that they begin with an initial estimate of the solution \( a_0 \) and generate successively improved estimates \( a_k \) until some stopping condition is satisfied. Assuming that the solution is unique and that \( a_k \) is within the convex region about \( \hat{a} \) then

\[
\hat{a} = \lim_{k \to \infty} a_k.
\]
The software implementation of the calibration process is described in §4.2.1.

3.2 Implementation

The identification process provides a set of identified model parameters considered to be sufficiently close to \( \hat{a} \), while the nominal model has parameters \( a_{\text{nom}} \). The desired pose may be expressed in either task space coordinates \( g_{\text{des}} \) or in joint space coordinates which are then converted to task space through the direct geometric solution

\[
g_{\text{des}} = g(\theta_{\text{des}}; a_{\text{nom}}). \tag{3.5}
\]

For implementation, the task is to find a set of joint coordinates \( \theta \) that satisfies

\[
\min \left\{ (g_{\text{des}} - g(\theta, \hat{a}))^T W_c (g_{\text{des}} - g(\theta, \hat{a})) \right\}. \tag{3.6}
\]

Because the speed of the solution is critical, approximate methods are necessary. A reasonable result can be obtained by conducting a single line minimisation in the gradient direction. This involves 1 gradient evaluation and a small number of function evaluations. Because the function evaluations are themselves quite time consuming, a quadratic interpolation method is more effective. This involves evaluating the cost function 3 times at different offsets in the gradient direction and fitting a quadratic to the result. The minimum of this quadratic is used for \( \theta \).

3.3 A Survey of Non-Linear Minimisation Techniques

A variety of non-linear minimisation techniques exist and methods vary in effectiveness depending on the nature of the problem to be solved. All non-linear techniques are iterative and most make some sort of assumption about of the local topology of the cost function. Some ideas such as line searching are common to most of the techniques considered. The following sections explain relevant ideas followed by an overview of several non-linear techniques.

Line Search

Most algorithms implement a line search of some variety. Usually a search direction \( s \) is determined by some process and points are sampled along the line \( a_k + \alpha s \) in the direction of decreasing cost. The objective of a line search is not necessarily to find the exact minimum of a cost function along the search direction but to find a point that represents a sufficient decrease, often with conditions on the gradient at that point as well. After that point has been found a new search direction is determined and the process repeated.
'Sufficient decrease' is more stringent than simply stipulating that

\[ \rho(a_k + \alpha s) < \rho(a_k) \]  

(3.7)

because this does not exclude the possibility of infinitesimal reductions in \( \rho \) at each step. Conditions that guarantee sufficient decrease (known as the Wolfe conditions [Fletcher, 1987]) are:

\[ \rho(a_k + \alpha s) \leq \rho(a_k) + \alpha \gamma \rho'(a_k) \]  

(3.8)

\[ \rho'(a_k + \alpha s) \geq \sigma \rho'(a_k) \]  

(3.9)

where \( \rho' \) is the slope along the search direction

\[ \rho'(a_k) = \frac{\partial \rho}{\partial a}. \]  

(3.10)

Typically \( \gamma \) is relatively small (\( \approx 0.01 \)) so as not to force the steps to be impractically short. The value of \( \sigma \) depends on the exactness required of the line search. A value of 0.9 is a weak line search while a value of 0.1 requires the selected point to be much closer to the exact minimum along the search direction.

Bounding and Non-Feasible Points

There are common-sense bounds on the extent of an error. For example, the link lengths of a Puma 560 robot are unlikely to be 50 mm different from their nominal values. Similar bounds exist for all error parameters. It is possible (and quite common) for an unconstrained minimiser to find a solution that is well beyond these boundaries but which returns a reasonably low residual. Such a solution fits the measurement data but is unlikely to model the mechanism well over points in the workspace that were not measured. Constrained minimisation techniques limit the search space of the minimiser to lie within prescribed bounds. The simplest type of constraint is a bound constraint of the form

\[ l_i \leq a_i \leq u_i, \quad 1 \leq i \leq n \]  

(3.11)

where each element of \( a \) is constrained to lie between scalar limits. A constrained minimiser acts like an unconstrained minimiser within the search space but does not sample points outside. It may return a solution that lies on the boundary of the search space.

In robot calibration a solution on the boundary of the search space is very probably an error and so little is gained from using the more complex constrained minimisation techniques. The approach used for the calibration problems was to artificially increase the cost function when the minimiser sampled points outside the search space in order to direct the search back within bounds. This allows the use of unconstrained techniques, which are simpler.
3.3 A SURVEY OF NON-LINEAR MINIMISATION TECHNIQUES

For some cost functions, in particular those involving closure of mechanisms, the cost function is not defined over all \( a \). If the mechanism cannot be closed for some geometry \( a_k \) and configuration \( \theta_i \) then \( a_k \) is called a non-feasible point\(^1\).

Whenever a bound constraint is violated or a non-feasible point is sampled the cost function is adjusted to a very large value. This has the effect of driving the line search parameter \( \alpha \) back towards zero. The problem with this approach is that it violates the assumption of continuity, however in practice the method was shown to work satisfactorily.

Non-Unique Solutions

Many functions exhibit multiple minima and any algorithm that does not perform an exhaustive search of the entire parameter space can only claim to have found a local minimum. For calibration problems the solution is typically close to the start point and if bounding is implemented then many of the possible local minima are excluded from the search space. Problems occur if there are multiple minima within the bounds of the search space. Physically, any of the local minima represent possible geometries that reproduce the measurement data, and differentiating between minima with equal \( \rho \) is impossible without further measurement information from the physical robot. A similar but more difficult situation is when the minimum is not a point but a straight line or even a flat multi-dimensional surface. In this situation there is an entire subspace of the model parameters that satisfy the conditions for a solution.

It is possible to force uniqueness on the solution by superimposing a bias function onto \( \rho \). The simplest and most logical such function is a quadratic centered on the start point \( a_0 \) making

\[
\rho_b(a) = \rho(a) + \mu (a - a_0)^T (a - a_0) \tag{3.12}
\]

\[
\rho'_b(a) = \rho'(a) + 2\mu (a - a_0) \tag{3.13}
\]

where \( \mu \) is chosen to be small enough not to distort the cost function greatly but significant enough to have a detectable gradient at the numerical precision used for the calculations. This forces the minimisation algorithm to terminate at a point \( a_* \) which is near to \( \min \rho(a) \) and as close as possible to the start point \( a_0 \). From here the minimisation is re-started from \( a_* \) with a smaller \( \mu \) in order to tidy the solution. When a decrease in \( \mu \) changes the solution point by a negligible amount, the process is stopped. Obviously the result of such a process will not in general correspond to the geometry of the physical robot, but if the residual is small enough then the solution is likely to model the behaviour of the physical robot with reasonable accuracy.

\(^1\)Alternatively, a "non-feasible geometry": \( a_k \) is a point in \( \mathbb{R}^n \).
Least Squares Problems

A variety of specialised techniques exist for the solution of non-linear least-squares model-fitting problems. Such techniques are a sub-class of general non-linear minimisation techniques. The least squares criterion imposes structure on the derivatives of the cost-function and many techniques take advantage of this. If the cost function is a weighted sum of squares

$$\rho = \sum_{i=1}^{m} (\varepsilon_i^T W \varepsilon_i)$$

(3.14)

then

$$\frac{\partial \rho}{\partial a} = 2 \sum_{i=1}^{m} \left( \frac{\partial \varepsilon_i}{\partial a}^T W \varepsilon_i \right)$$

(3.15)

$$\frac{\partial^2 \rho}{\partial a \partial a^T} = 2 \sum_{i=1}^{m} \frac{\partial \varepsilon_i}{\partial a}^T W \frac{\partial \varepsilon_i}{\partial a} + 2 \sum_{i=1}^{m} \frac{\partial^2 \varepsilon_i}{\partial a \partial a^T} W \varepsilon_i.$$  

(3.16)

The Hessian matrix is often approximated by the first right-hand term in equation 3.16 obviating the need to derive the second derivative of the model function. In many practical problems the neglected term is ignorably small, particularly if the residual near the solution is close to zero. Problems can occur if neither the $\varepsilon_i$ nor the second derivative tend to zero at the solution. This can cause the solver to converge slowly because the approximate Hessian is inaccurate. In such situations a Quasi-Newton approach is likely to perform better. In the following sections, the approximate Hessian is represented by

$$\bar{H} = \sum_{i=1}^{m} \frac{\partial \varepsilon_i}{\partial a}^T W \frac{\partial \varepsilon_i}{\partial a}.$$  

(3.17)

Newton’s Method

In Newton’s method, the function is assumed to be nearly quadratic about the current sample point $a_k$ and the next sample point is chosen along a direction that lies towards the minimum of this quadratic. If $\nabla^2 \rho(a_k)$ is the Hessian of the cost function evaluated at sample point $a_k$, and $\nabla \rho(a_k)$ is the Jacobian of the cost function at $a_k$, the next sample point is chosen to minimise

$$\rho(s) = \rho(a_k) + \nabla \rho(a_k) s + \frac{1}{2} s^T \nabla^2 \rho(a_k) s$$

(3.18)

by solving

$$\nabla^2 \rho(a_k) s = -\nabla \rho(a_k)$$

(3.19)

and making the next sample point

$$a_{k+1} = a_k + \alpha s.$$  

(3.20)
Convergence to a local minimum is guaranteed if $\nabla^2 \rho(a_k)$ is positive definite. The rate of convergence is quadratic meaning that

$$|a_{k+1} - \bar{a}| \leq \beta |a_k - \bar{a}|^2$$

(3.21)

where $\beta$ is some positive constant.

Most implementations of Newton's method modify the basic method to improve performance when $\nabla^2 \rho(a_k)$ is poorly conditioned. The constant $\alpha$ is calculated either by employing line-search methods or by a trust-region approach. Inside the trust-region the discrepancy between the quadratic approximation and the actual function is considered small enough to neglect. The size of the region is adjusted between steps according to some heuristic rule, based on the size of the discrepancy (cf. Moré and Wright [1993, chapter 2]).

Newton's method can only be applied to problems where $\nabla^2 \rho(a_k)$ is available and this proves to be a major drawback. Many problems exist where it is impractical to compute the Hessian. The method forms the basis of several alternative methods which use some sort of approximation or alternative to the Hessian, some of which are described below.

**Gauss-Newton**

This method makes use of the approximated Hessian $\bar{H}$ (equation 3.17) instead of the $\nabla^2 \rho(a_k)$ term found in Newton's method. The resulting system of equations is

$$\sum_{i=1}^{m} \frac{\partial \varepsilon_i}{\partial a} W \frac{\partial \varepsilon_i}{\partial s} = -\sum_{i=1}^{m} \frac{\partial \varepsilon_i}{\partial a} \varepsilon_i$$

(3.22)

which must be solved for $s$.

When the residuals $\varepsilon_i$ are small at the solution, the neglected term in equation 3.16 becomes insignificant and—like Newton's method—the method converges quadratically. This may also occur if the second derivatives of the $\varepsilon_i(a)$ are small. Otherwise, the method converges linearly.

If the approximate Hessian is rank deficient, a variety of techniques are available to obtain an approximate inverse such as the Moore-Penrose pseudo-inverse [Penrose, 1954].

**Quasi-Newton (Variable Metric)**

A more sophisticated approximation to the Hessian matrix may be obtained by using information gleaned from gradients evaluated during previous steps. The Broyden-Fletcher-Goldfarb-Shanno (cf. Fletcher [1987 §3.2]) updating formula successively improves an initial estimate of the Hessian matrix over several steps. Since the Hessian is inverted
in order to evaluate $s$, it is more common to estimate either the inverse Hessian or the Cholesky decomposition of the Hessian directly thus

$$H_{k+1}^{-1} = \left( I - \frac{s_k y_k^T}{y_k^T s_k} \right) H_k^{-1} \left( I - \frac{y_k s_k^T}{y_k^T s_k} \right) + \frac{s_k s_k^T}{y_k^T s_k}$$

(3.23)

where

$$y_k = \nabla \rho (a_{k+1}) - \nabla \rho (a_k).$$

(3.24)

As for Newton's method,

$$s_{k+1} = -H_k^{-1} \nabla \rho (a_k).$$

(3.25)

Some practical matters relating to the application of this method are discussed in §4.3.3.

Levenberg-Marquardt

When the local topology of the cost function is poorly approximated by a quadratic, a steepest descent approach will often perform better. Levenberg-Marquardt is a solution method that has the property of behaving like a steepest descent method further from the minimum and like a second order method closer to the minimum. Essentially, the method is similar to the Gauss-Newton method except that the main diagonal of the approximate Hessian $\tilde{H}$ is augmented by the factor $\lambda$. A large $\lambda$ forces the step towards the direction of steepest descent while a small $\lambda$ produces something like a Gauss-Newton step

$$(\tilde{H} + \lambda I \text{ diag } (\tilde{H})) s = -\sum_{i=1}^{m} \frac{\partial \varepsilon_i}{\partial a} \varepsilon_i.$$  

(3.26)

The ratio $r$ expresses the ratio of the actual decrease in cost function between $a_{k+1}$ and $a_k$ to the predicted decrease

$$r = \frac{\rho(a_{k+1}) - \rho(a_k)}{\frac{1}{2} s^T \left( -\sum_{i=1}^{m} \frac{\partial \varepsilon_i}{\partial a} \varepsilon_i + \lambda s \right)}.$$  

(3.27)

The value of $\lambda$ is adjusted according to a simple heuristic rule. If $r < 0.25$, the function is not much like a quadratic so $\lambda$ is increased. If $r > 0.75$, the function is close enough to the quadratic approximation so $\lambda$ is decreased. If $r$ is zero or positive then the step is taken, otherwise the step is not taken and another trial step is evaluated with the new value of $\lambda$.

3.4 Application of Non-Linear Minimisation Techniques

Table 3.1 summarises the characteristics of the various solution methods. In terms of robot modelling and calibration there are 3 distinct problems to be solved with different
<table>
<thead>
<tr>
<th>Method</th>
<th>Performs well with large residual</th>
<th>Performs a line search</th>
<th>Seeks a local optimum</th>
<th>Convergence rate</th>
<th>Gradient required</th>
<th>Hessian</th>
<th>Rank Deficiency</th>
<th>Routines</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newton's Method</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Required</td>
<td>Required</td>
<td>A variety of methods are used</td>
<td>MATLAB PROC NLP NAG and many others</td>
<td>Not a least squares method</td>
<td></td>
</tr>
<tr>
<td>Gauss-Newton</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Required</td>
<td>Approx.</td>
<td>Uses pseudo-inverse</td>
<td>DFNLP, NAG MATLAB OPTIMA TEN-SOLVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Variable Metric)</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Superlinear</td>
<td>Required</td>
<td>Approx. with Broyden's update</td>
<td>Can be avoided by applying curvature conditions</td>
<td>PROC NLP DFNLP LANCELOT NLSSOL Num. recipes</td>
<td></td>
</tr>
<tr>
<td>Levenberg-Marquardt</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Required</td>
<td>Approx.</td>
<td>May be avoided</td>
<td>MINPACK IMSL MATLAB ODRPACK PROC NLP</td>
<td>Trust region approach</td>
<td></td>
</tr>
</tbody>
</table>
characteristics:

1. Closure
   - low residuals
   - no constraints on parameters
   - multiple optima (some with low residuals)
   - function evaluation relatively easy
   - gradient available
   - Hessian possible but complex to evaluate

2. Identification
   - high residuals
   - bounds on parameters may be applied
   - multiple optima or optimal sub-spaces
   - function evaluation costly in terms of computation and involves solution of
     the closure problem
   - non-feasible points will occur if the closure algorithm fails
   - gradient available
   - Hessian not available

3. Implementation
   - low residuals
   - unconstrained parameters
   - multiple optima possible but unlikely
   - function evaluation involves solution of the closure problem
   - gradient available
   - Hessian not available
   - requirement for high speed computation

The closure problem is a good candidate for any gradient based method. Levenberg-
Marquardt, and Gauss-Newton are especially well suited because it is a low residual
problem and both of these methods will converge quadratically. MATLAB's leastsq.m
function implements both. The problem of multiple minima can often be avoided by
using an estimate of the closed configuration obtained from some nominal model.

The identification problem does not necessarily have low residuals near the solu-
tion and as a result the Levenberg-Marquardt and Gauss-Newton methods are likely to
3.5 THE INFLUENCE OF NOISE ON THE SOLUTION

<table>
<thead>
<tr>
<th>Method</th>
<th>Problem</th>
<th>Closure</th>
<th>Identification</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fn</td>
<td>Grad</td>
<td>MFlops</td>
<td>Fn</td>
</tr>
<tr>
<td>Gauss-Newton (leastsq.m)</td>
<td>45</td>
<td>14</td>
<td>3.35</td>
<td>failed to converge</td>
</tr>
<tr>
<td>Quasi-Newton (bfgs.m)</td>
<td>66</td>
<td>67</td>
<td>9.24</td>
<td>162</td>
</tr>
<tr>
<td>Quasi-Newton (fminu.m)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>188</td>
</tr>
<tr>
<td>Lev.-Marq. (levmrq.m)</td>
<td>9</td>
<td>9</td>
<td>2.63</td>
<td>-</td>
</tr>
<tr>
<td>Lev.-Marq. (leastsq.m)</td>
<td>52</td>
<td>15</td>
<td>3.69</td>
<td>failed to converge</td>
</tr>
</tbody>
</table>

Table 3.2 Performance of non-linear minimisation algorithms. The table shows the number of function and gradient evaluations and the total number of floating point operations required to come to a solution using different methods. MFlops is short for “Million Floating point operations”. The model used for the closure problem and the implementation problem was the Delta robot with 30 parameters, and 125 calibration measurements. (cf. §4.3.3). The 4 bar chain with 6 frames and 26 parameters was used for the identification problem. While the performance of each algorithm is dependent on the efficiency of the code and the initial settings of the solver parameters, these figures are indicative of relative performance.

perform poorly. This was verified experimentally (cf. Table 3.2). Quasi-Newton techniques are not sensitive to significant residuals near the solution because they build up an estimate of the Hessian progressively from local gradients. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) method was tried and it was able to solve the problem while the Levenberg-Marquardt and Gauss-Newton methods failed to come to a solution. The BFGS algorithm performed correctly when non-feasible points were encountered.

The implementation problem has many of the features of the closure problem however the requirement for high speed computation is significant. The Levenberg-Marquardt method was applied and found to work satisfactorily however the computation time was too long for use in a practical controller. It is unlikely that the Matlab environment would be used to solve this problem and a compiled language would probably perform the computation about 30 times faster. Nevertheless the requirement that the closure problem be computed several times before convergence remains a major drawback.

3.5 The Influence of Noise on the Solution

Noise is defined for the purposes of the following discussion as errors in the measurement data. The errors arise due to imperfect accuracy of the measurement device and are assumed to conform to some time-independent probability distribution. The effect of noise is to alter the result of the identification process $\hat{a}$ such that it becomes a random variable conforming to some probability distribution. The variance-covariance matrix (also referred to as the covariance matrix) characterises the spread of $\hat{a}$, and the construction of confidence intervals based on the covariance matrix gives an indication of the accuracy with which each parameter has been estimated. The construction of confidence intervals implies knowledge of the probability density function. Commonly the distribution is assumed to be Gaussian for simplicity, but it is worthwhile testing the validity of this assumption.
3.5.1 Linear Error Analysis

If the effect of nonlinearities is small for the noise levels encountered in the calibration process, then a linear analysis will be suitable for predicting the covariance matrix of the identified parameters. At the solution, small changes in the residual are related to changes in the model parameters

$$\delta y_s \approx J_{dp}\delta a.$$  \hspace{1cm} (3.28)

The noise may be modelled as coming from 2 sources: measurement noise and actuator (or joint) repeatability, thus the covariance matrix of the measurements

$$\Sigma_y = \Sigma_n + J_{id}\Sigma_\theta J_{id}^T$$  \hspace{1cm} (3.29)

where $\Sigma_n$ is the covariance of the measurement device noise and $\Sigma_\theta$ is the covariance of the actuator repeatability. Assuming linearity, the covariance of the parameter estimate is

$$\Sigma_a = \left\{ \left( J_{id}^T J_{id} \right)^{-1} J_{id}^T \right\} \Sigma_y \left\{ \left( J_{id}^T J_{id} \right)^{-1} J_{id}^T \right\}^T.$$  \hspace{1cm} (3.30)

Unfortunately, for all but the simplest models, the identification Jacobian is usually rank deficient.

An estimate of the covariance matrix $\Sigma_a$ can be obtained from the Hessian at the solution

$$\Sigma_a \approx H^{-1}.$$  \hspace{1cm} (3.31)

or

$$\Sigma_a \approx \tilde{H}^{-1}.$$  \hspace{1cm} (3.32)

In particular, if a quasi-Newton method is used, the inverse Hessian is estimated directly.

3.5.2 Interpretation of Covariance and Confidence Intervals

Once estimates of the solution $\tilde{a}$ and covariance matrix $\Sigma_a$ have been obtained by whatever means from a sample of measurements, it is desirable to make statements about the relationship between the estimate $\tilde{a}$ and the true model. This statement takes the form of a confidence region. A confidence region is defined as a region about the estimate that may be expected to contain the true model with probability $1 - \alpha$. Such a region is termed a \textit{\('1 - \alpha\) confidence region}. If $\tilde{a}$ is the value that minimises equation 3.2 then, assuming linearity, the probability density will be chi-square distributed about $\tilde{a}$.

Because $\tilde{a}$ has dimension $n$, we can define a confidence region for any particular element of $\tilde{a}$ or any linear combination of the elements of $\tilde{a}$. Let $z$ be a linear combination of the elements of $\tilde{a}$.

$$z = w^T \tilde{a}$$  \hspace{1cm} (3.33)
3.5 \ THE \ INFLUENCE \ OF \ NOISE \ ON \ THE \ SOLUTION

The covariance of \( z \) is

\[
\text{Var}(z) = s_z = w^T S w.
\]

(3.34)

Because \( z \) is univariate the confidence region is

\[
z \pm \sqrt{\chi_1^2 (1 - \alpha)} \sqrt{s_z}
\]

(3.35)

or

\[
w^T \hat{a} \pm \sqrt{\chi_1^2 (1 - \alpha)} \sqrt{w^T Sw}
\]

(3.36)

where \( \chi_1^2 (1 - \alpha) \) denotes the value of the chi-square statistic with 1 degree of freedom that has probability \( 1 - \alpha \). If, for example, \( w \) is

\[
w = [1 \ 0 \ \cdots \ 0]^T
\]

(3.37)

then the confidence interval in equation 3.36 becomes the region between two planes perpendicular to the \( a_1 \) axis. The geometrical interpretation of this in terms of robot calibration is that the probability that the true error parameter \( \hat{a}_1 \) lies within the bounds given by equation 3.36 is \( 1 - \alpha \). Similarly, if we choose another axis

\[
w = [0 \ 1 \ \cdots \ 0]^T
\]

(3.38)

then the probability that \( \hat{a}_2 \) lies within the bounds given by equation 3.36 is \( 1 - \alpha \).

The above results are useful if one wishes to make statements about individual parameters. Unfortunately, the probability of both sets of bounds containing the true mean simultaneously is not \( 1 - \alpha \). To illustrate, consider the simple case where all components of \( \hat{a} \) are independent. The covariance matrix \( S \) is

\[
S = \begin{bmatrix}
s_{11} & 0 & \cdots & 0 \\
0 & s_{22} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & s_{nn}
\end{bmatrix}
\]

(3.39)

and the probability of \( k \) confidence intervals holding simultaneously is

\[
P(\text{intervals } 1 \ldots k) = P(\text{interval } 1)P(\text{interval } 2) \ldots P(\text{interval } k)
\]

\[
= (1 - \alpha)(1 - \alpha) \ldots (1 - \alpha)
\]

(3.40)

\[
= (1 - \alpha)^k.
\]

In robot calibration, (under the assumption that the error model fully spans the error space) there is the true set of error parameters \( \hat{a} \) and the estimated set \( \tilde{a} \). The size of the confidence region gives an indication of how close the estimate is likely to
be to the true set and it is desirable that all parameters are included in the confidence statement simultaneously. Therefore, the confidence region which contains the true set with probability $1 - \alpha$ for all $w$ is the most useful. This is the region where the $\chi^2$ parameter is less than a certain value $\chi^2_n$. For a linear model, the confidence region turns out to be ellipsoidal and its boundary is described by

$$
(a - \bar{a})^T S^{-1} (a - \bar{a}) = \chi^2_n (1 - \alpha).
$$

(3.41)

Because the units of the parameters in the covariance matrix may not be homogeneous, the principal axes of the confidence ellipsoid say little about the limits on each individual parameter. To find these limits, the confidence ellipsoid is projected onto each axis and the bounds are derived from the projection. The projected confidence interval for axis $i$ is

$$
\bar{a}_i \pm \sqrt{\chi^2_n (1 - \alpha)/s_{ii}}.
$$

(3.42)

### 3.5.3 Testing the Assumption of Normality

Well established methods exist for determining the normality of univariate distributions of points such as the Kolmogorov-Smirnov test (cf. Press et al. [1994]) or the Quantile-Quantile plot (cf. Johnson and Wichern [1982]). For multivariate distributions there is more uncertainty because there are many ways in which non-normality can be obscured. For example, it is possible for marginal distributions to appear to be normally distributed while strong non-normality exists in some other dimension. Despite this, the following test (cf. Johnson and Wichern [1982 §4.6]) will indicate non-normality given "well behaved" data.

The Mahalanobis distance of sample point $x_i$ from the mean $\bar{x}$ is defined as

$$
d^2_i = (x_i - \bar{x}) S^{-1} (x_i - \bar{x}).
$$

(3.43)

If the $x_i$ are taken from a multivariate Gaussian distribution then $d^2_i$ will be chi-square distributed with $n$ degrees of freedom. Comparison of a sample of data points with the chi-square distribution is achieved by plotting the sorted $d^2_i$ against the $i$th quantile of the chi-square distribution $\chi^2_n \left( \left( i - \frac{1}{2} \right) / p \right)$. If the points lie along a straight line then it is likely that the sample is Gaussian. Points that lie a comparatively long way from the line do not conform to the Gaussian distribution. The plot gives a visual sense of how well the sample conforms. Figure 3.1 shows the behaviour of 100 sets of pseudo-random normally distributed multivariate data in a chi-square plot. The data were generated with the MATLAB version 5.3 function randn.m using $n = 26$, $p = 100$. 
Figure 3.1 Distribution of results from pseudo-random normally distributed multivariate data in a chi-square plot. The dark line is ideal.
Chapter 4

Application

4.1 Introduction

This section describes the experimental and practical part of the research that used real measurement data to test the solution techniques and models developed. Section 4.2 describes the software design, mechanical design of the robot, and control system design. Section 4.4 describes the modelling, measurement, identification and implementation of the calibration process along with details of the simulations performed.

The experimental part of the research was limited by the equipment available. A lack of appropriate measurement gear meant that the Delta robot used had to be made large enough to make errors detectable. An error large enough to detect was artificially introduced into the mechanism by making one of the 6 parallel arms 20 mm longer than its nominal length. Although an error this large is very unlikely, it represents only 2.2% of the nominal length of the arm. In general, smaller geometric errors are easier to identify as long as the measurement noise is proportionately smaller. The experimental procedure and results are discussed in §4.4.

The research required large amounts of computation. Most of the code was written for the MATLAB environment and executed on a variety of machines and operating systems. It is worthwhile to note that the same code, when executed on different operating systems can produce slightly different results due to slight differences in the handling of floating point calculations. The systems used were a dual 350MHz Pentium II machine running SMP Linux version 2.21, a variety of Pentium based machines running Windows 98 and Windows NT, and a dual 450Mhz Sun Ultra Enterprise running Solaris 2.6.

4.2 Design of System Components

4.2.1 Implementation of Modelling, Visualisation, and Calibration Package

The analysis described in chapters 2 and 3.1 was implemented in the MATLAB environment as 3 separate suites of software: rbk, mcpc, and modelfit. The code was initially
written under the MATLAB 4 environment but finished under MATLAB version 5.2. No particular effort was made to ensure backward compatibility so complete functionality is only available under the later version.

The functions and data structures for modelling rigid body chains and trees of linked bodies form the rbk (Rigid Body Kinematics) suite. The most important function is dirgeom.m which calculates the direct geometric solution of a model given a model definition matrix and a set of actuator coordinates. The model definition matrix contains the joint transformation in the form of a 3 by 3 rotation matrix and a 3 element offset vector as well as the index of the previous link in the chain. If the joint is a closure joint then the index of the member that it closes with is also stored. The data format used for the model definition matrix is referred to as dim format. The rbk suite also provides the function drawgeom.m which displays the rigid body chain in a figure window with mini-axes representing the body frames.

The modelfit suite contains algorithms for least-squares fitting of data to a model and multivariate minimisation. The function lmmin.m performs the Levenberg-Marquardt method. It proved to be quite successful when used for the closure of parallel mechanisms but was not very efficient for the calibration problem. This is probably because the closure problem is a small residual problem which is ideal for the Levenberg-Marquardt method. The quasi-Newton function bfgs_idc.m performed much better for the calibration problem.

The mcpc suite contains functions for interfacing with the rbk suite, closing the kinematic loops on parallel mechanisms, and performing calibration. The MCPC parameters were far more convenient to tabulate and manipulate than the same information in dim format. The parameters were arranged in a matrix as shown in Table 4.1 and the function mcpc2dim.m was used to convert the parameters from this format into dim format. In Table 4.1, each row defines one link of the robot. The first 6 columns contain the MCPC parameters as defined in equation 2.45. Note that only special frames require all 6 parameters to be defined. Unused parameters not containing 0 will cause an error to be generated by mcpc2dim.m.

Links are defined relative to preceding links, forming a tree structure. The tree structure originates at the base link (link 0). For link i, column 7 of row i holds the index of the preceding link in the tree. Column 8 of row i, if non-zero, defines the link's body frame as a closure frame and contains the index of the link that it forms a closure with. Finally, column 9 of row i defines the type-number of joint i: 0 for rotary joints, 1 for prismatic joints, 2 for spherical joints, and 3 for auxiliary frames. Auxiliary frames are frames requiring special constraints during the closure process. It is necessary to define a frame that defines the position of the endpoint relative to the last joint in the mechanism (called the tool frame) and a frame that relates the base of the robot to an arbitrary base coordinate frame (called the base frame). Both these frames should be
4.2 DESIGN OF SYSTEM COMPONENTS

\[
\begin{bmatrix}
\alpha_{11} & \alpha_{21} & \alpha_{31} & d_{11} & d_{21} & d_{31} & Prd_1 & Clo_1 & Typ_1 \\
\alpha_{12} & \alpha_{22} & \alpha_{32} & d_{12} & d_{22} & d_{32} & Prd_2 & Clo_2 & Typ_2 \\ 
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\alpha_{1n} & \alpha_{2n} & \alpha_{3n} & d_{1n} & d_{2n} & d_{3n} & Prd_n & Clo_n & Typ_n
\end{bmatrix}
\]

Table 4.1 MCPC parameters laid out in mcpc format with connectivity and constraint information in the last 3 columns.

defined as type 3. Each of the joint types defined above is constrained differently during the calibration process.

The function `connect.m` invokes a nonlinear least-squares solver (`lm.m`) to minimise the difference between position and orientation parameters in connected frames (referred to as the closure problem, §2.4). The minimiser requires a matrix of derivatives expressing the freedoms of the open loop mechanism with respect to the joint settings. This is calculated by the function `dirjack0.m`.

The function `calib.m` invokes the nonlinear solver for calibration. The solver could be any of the solvers supplied with the MATLAB Optimisation Toolbox, the NAG library, or a custom written routine such as `bfgs_idc.m`. The function `bfgs_idc.m` was used for all of the simulation and experimental work. The solver in turn calls `funf.m` and `funj.m` in order to evaluate its cost function and identification Jacobian. The cost function is evaluated as in equation 3.2 and involves calling `connect.m` for each iteration of the summation. As a result the computation involved is considerable. The closed loop derivative matrix is calculated using the method described in §2.5. The identification Jacobian is formed by extracting and transforming sections of the the closed loop derivative matrix of the measurement frame for each measurement pose and assembling the derivatives into a single matrix. This is illustrated in Figure 4.1. The structure of `calib.m` is illustrated in Figure 4.2.

4.2.2 Mechanical Design of the Canterbury University Delta Robot

The design and control of the Canterbury University Delta robot was a major undertaking. The design and construction was done jointly as part of 2 separate but related doctoral projects: this project, and an investigation into the kinematics and rigid body dynamics of the Kiwibot conducted by Dr T. P. Jones [Jones, 1997]. Design tasks were performed in close cooperation with Dr Jones and the design of components common to both mechanisms was shared.

The mechanical design of the robot attempts to utilise the advantages of the robot’s parallel structure. These are, specifically, greater rigidity and a higher payload to mobile-mass ratio. These features potentially allow the creation of a highly accurate robot that is capable of high end-effector velocities and high payloads. Constraints on the
Figure 4.1 Formation of the identification Jacobian. The assembled open-loop Jacobian is converted to the closed-loop Jacobian (by a null space transformation in this case §2.5). Next, the rows corresponding to the measurement frame (frame \(f\)) are extracted and assembled into the identification Jacobian. Since there are \(m\) measurements, \(m\) closed-loop Jacobians are required. The transformation \(M\) represents the measurement device transformation which converts task space coordinates to coordinates in measurement space.
4.2 DESIGN OF SYSTEM COMPONENTS

Nominal Model \( a_0 \)  
Measurement Data \( y \)  
Sample Point Data \( \theta \)

\textbf{calib.m:}\n
wrapper function for solver e.g. bfgs_idc.m

\textbf{Solver:}\n
e.g. bfgs_idc.m, fminu.m...
solves \( \min(e^TWe) \) where

\[
\varepsilon = \begin{bmatrix}
    f(\theta_1; a) - y_1 \\
    f(\theta_2; a) - y_2 \\
    \vdots \\
    f(\theta_n; a) - y_n
\end{bmatrix}
\]

Termination tolerance on \( a \) is \( 1 \times 10^{-10} \).
Termination tolerance on gradient is \( 1 \times 10^{-10} \).

\textbf{funf.m:}\n
evaluates \( \varepsilon \)

\textbf{funj.m:}\n
evaluates \( \frac{\partial \varepsilon}{\partial \theta} \) from closed loop Jacobian \( J_{CL} \)

\textbf{connect.m:}\n
solves \( c = \min \left( \sum_{i=1}^{n_c} \left| g_{pi} - g_{qi} \right|^2 \right) \)

where joints \( [p_i q_i] \) are a closed pair, and \( n_c \) is the number of closure joints.

Termination tolerance on \( \theta \) is \( 1 \times 10^{-10} \).
Termination tolerance on gradient is \( 1 \times 10^{-10} \).

\textbf{calibfn.m:}\n
evaluates \( f(\theta; a) \)

\textbf{measdev.m:}\n
simulates measurement device \( g(\theta; a) \)

\textbf{cljack.m:}\n
evaluates \( J_{CL} \) from \( J_{OL} \) under constraint \( c = 0 \)

\textbf{dirgeom.m:}\n
evaluates \( g(\theta; a) \)

\textbf{measdev.m:}\n
simulates measurement device, \( g(\theta; a) \rightarrow f(\theta; a) \), \( \frac{\partial g(\theta; a)}{\partial \theta} \rightarrow \frac{\partial f(\theta; a)}{\partial \theta} \)

\textbf{dirjack.m:}\n
evaluates \( J_{CL} = \frac{\partial g(\theta; a)}{\partial \theta} \)

Figure 4.2 Logical structure of the calibration algorithm.
design were cost of manufacture and the workshop facilities that were available. As a result, the components were designed to be rigid, manufacturable, inexpensive, and accurate. Standard materials such as steel and aluminium were used and the components are able to be made on standard machine tools such as a lathe and milling machine. Another requirement of the design is that it accommodates the needs of two similar but functionally different robot structures: the Delta robot and the Kiwibot (cf. Jones [1997]). The requirements of both robots had to be balanced. Sometimes the design was a compromise between the best solution for each robot while at other times the requirements were contradictory and some careful conceptual design was required in order to accommodate both.

![Image of the Kiwibot](image)

**Figure 4.3** The Kiwibot 3 degree-of-freedom robot.

The Kiwibot (see Figure 4.3) is a 3 degree of freedom spherical positioning robot. It's base and actuated arms are very similar to those of the Delta robot but it's upper section is kinematically a mirror image of the lower section. The resulting motion of the robot has 2 rotational degrees of freedom and a plunging freedom. The cartesian translations of the endpoint are coupled to these freedoms. Such a mechanism was found to be advantageous for satellite dish positioning applications.

The starting point for the design was the power of the robot. Three brushless DC motors rated at 2 kW each were available and the robot was scaled to suit the motors.
Also available were some Digiplan motor drives and 3 Parker Hannafin IFX brushless DC motor controllers were purchased.

### Major Components

The Delta robot is composed of 7 major sub-structures (cf. Figure 4.4). These are the support structure, the base platform, the reduction housing, the actuated arms, parallelogram linkage, the ball joints, and the nacelle.

![Diagram of Delta robot components](image)

**Figure 4.4** Major components of the Delta robot.

### Support Structures

For any robot, the mobile mass is the critical component of the design. Static mass such as mass contained in support structures need not be minimised, indeed there are advantages in having a stiff and solid base and support. The simplest and probably the stiffest support structure would be a large diameter steel or aluminium tube that could be bolted to a substantial overhead structure. Figure 4.5 shows this type of arrangement.
However, because the Kiwibot works in the space above the base platform, this could not be used. Instead, a support that protrudes minimally into the workspace of either robot was chosen.

![Figure 4.5 Column support structure for Delta robot](image)

The design chosen consisted of tubular aluminium legs that protruded from the base platform at 45 degrees from horizontal (see Figure 4.4). The legs were each bolted to a flange on the base member and terminated at the opposite end in a vertical tubular section with an attachment for a leg extension. The leg extension is not used when the robot is in the Kiwibot configuration in order to keep the robot lower to the floor. The Delta robot uses the leg extensions to raise the robot so that the robot arm linkages cannot reach the floor. This prevents the nacelle from hitting the floor when there is a sudden power failure.

**Base Platform**

The major consideration in the design of the base platform was that it would locate the mounting brackets for the reduction housings accurately. It was constructed from 15 mm thick steel plate, welded and then stress relieved to prevent distortion during subsequent machining. Rotational symmetry was preserved by mounting the base platform on a rotary table for machining of the mating surfaces for the supporting legs and reduction housing mounting brackets. The mounting brackets were welded and stress relieved. The mating surface with the base platform was machined flat and one edge was used as a reference while the holes for the reduction housing were line bored. The brackets were positioned relative to the base platform by clamping them in place and testing their relative position with a wiggler.
The Kiwibot sometimes has a central ball-jointed strut that is mounted in the base. As a result the base platform has a large hole in its top surface and a mount for a central ball joint. There are also holes for trunking of power wires to the motors and wires for sensors. To prevent damage to the actuated arms if the robot collapses, 3 rubber buffers are bolted to the underside of the base platform. These prevent the arms from rotating so far that they collide with the base platform.

Reduction Housing

The torque requirement at the actuated joints was mostly decided by the loading requirements of the Kiwibot. The requirement was roughly reckoned to be the torque needed to support the weight of the Kiwibot assembly entirely on one arm. The motors were rated at 2 kW and nominally capable of 6 Nm torque output between 0 and 400 RPM. Given the torque and speed requirements, a speed reduction of approximately 100:1 was necessary. Because of cost and availability constraints, the final design was slightly under-rated for this purpose, however there was no fixed lower limit for the Delta robot and the reduction housings are perfectly adequate for its requirements.

A more important requirement for the Delta robot was to minimise backlash and flexibility in the drive system. Several possible designs were rejected on this criterion. Worm gear arrangements typically possess backlash unless they are preloaded in one direction. Preloading increases friction between the gear elements and lowers transmission efficiency. Harmonic drives are now commonly used for rotary joints in robotics applications. They possess favourable properties such as a backlash of only 3 arc seconds and power transmission efficiencies of 75-90%. Harmonic drives work by deforming a flexible spline (or Flexspline) which mates with a fixed outer spline (termed Circular spline). The splines have slightly different numbers of teeth so that in one revolution of the deforming element (termed the “wave generator”) indexes around by the difference in spline tooth numbers. Despite their large reduction ratios, Harmonic drives are backdriveable. The unit used was a Harmonic Drive Technologies HDIC 50 with 80:1 reduction ratio and 520 Nm rated load. Harmonic drives may be used at up to 140% of rated load with a consequent reduction of fatigue life.

By transmitting the output of the Harmonic drive to the rotary arms through a stiff, large diameter torque tube, elastic deflections are reduced. The Harmonic drive unit may be configured in a number of ways; For speed reduction, the wave generator is driven and either the Flexspline is used as the output (with Circular spline fixed) or the Circular spline is used as output (with Flexspline fixed). In the final design, the Circular spline was used as output and the Harmonic drive unit was housed inside the torque tube which acts as a housing. The drive configuration altered the output ratio to 81:1.

The housing rotated on 2 deep groove ball bearings. The outer races of the bearings served as location surfaces for the actuated arm. The inner races were fitted onto mount-
ing flanges which were bolted to a supporting bracket. Location of the entire housing was ensured by a close fit between the mounting flange and the bracket. There was provision for a rotary encoder to be mounted to the driving shaft at the opposite end to the motor. It was essential that the wave generator be concentric to the outer spline. The deep groove ball bearings possess a radial running clearance of 36 microns which makes it unlikely that the required tolerance was achieved. A carefully machined shim was inserted between the opposing flange and the inner race of the adjacent bearing to reduce the clearance. In hindsight, angular contact ball bearings would have been a better choice because these can be preloaded against each other to reduce running clearance.

Actuated Arms

The actuated arms contribute to the mobile mass of the robot and were therefore designed to be lightweight. A pair of 'C' shaped forks were designed to slide onto the ends of the reduction housing and were located on the outer races of the bearings which support the reduction housing. The arms were assembled from bolted aluminium sections. Bolts were used because it was difficult to prevent distortion during welding. A pair of mounts for ball joints were bolted to the sides of the box section. These were located accurately with a pair of spigots.

Ball Joints

The parallel arm linkages of the robot were connected with 3 degree of freedom joints. It was found that the angular range of the joints was the factor that determined the extent of the robot's reachable workspace. A 3 dimensional drawing package was used to determine the angular range required and it was determined that a range of $125^\circ$ would be adequate. Although it is difficult to make a ball joint that possesses such a large angular range, it was much simpler than creating a multi-axis type virtual ball joint for this purpose.

A 50 mm diameter ball was cut by rotating the stock about an oblique axis while a fly cutter rotated about the vertical axis of a milling machine. The two motions together generated a spherical ball. A tapered stem was left on the ball which had a thread and an accurate location surface. The ball sits in a phosphour bronze cup which was pressed into a steel housing. The housing features a spigot for accurately locating the ball joint and an external thread so that the outer housing could be screwed down onto it. The outer housing located a phosphour bronze retaining ring which enclosed the ball in the cup.

The retaining ring contacted the ball along a thin contact area meaning that the ball joint was much weaker in tension than in compression. Fortunately the ball joints are mainly exposed to side load so that the cup takes the greater portion of the load. In
order to achieve the required range of movement, the thin end of the tapered stem of the ball was only 11.5 mm thick.

**Parallelogram Links**

The parallelogram links consisted of a pair of mounts for the ball joint spigots at either end of a steel box section. Steel was chosen in order to reduce costs but ideally a lighter material would be preferable. A linkage between the parallelogram links restrains the free rotations about the axial direction of the link preventing the links from rotating far enough to hit the ball joint stems.

**Nacelle**

The nacelle was made from a 10 mm thick aluminium plate. Solid square sections provide location surfaces for the ball joint stems. A central location hole is used to mount some variety of end effector or a reflector for a theodilite measuring device.

**4.2.3 Control System Design**

As for the mechanical design of the robot, the control system design was performed cooperatively.

There are several possible approaches to the problem of providing a control interface between the robot and the user. Many systems use a dedicated operating system with a specialised robotic control language. This approach is a major undertaking in itself. Another approach that has merit is to use a proprietary software package that provides customised input/output in an interactive environment (MATLAB or LabVIEW for example). A third approach is to write an extension library for one of the established programming languages.

Most robotic control programs spend the majority of their time in routine mathematical, logical, and branching operations (as opposed to I/O with the actual robot). An extension library for a standard language would provide the I/O functions while the other functions are provided by the familiar standard programming language. This is the approach that was implemented.

The control software was written in the C language and an IBM compatible personal computer was used to communicate commands to the motor drivers through an RS232 serial link. The control system was designed to be modular and adaptable. Replacement of modules in the system (including hardware components) with improved versions should be greatly simplified as a result. The system is designed so that it may be adapted to other robots, other motor controllers, and different high level user interfaces.
Control Hardware

The three step motors used to drive the robot arms are controlled and driven by Digiplan IFX step motor drivers. An IBM compatible PC is linked to the Digiplan controller which is sent ASCII commands via an RS-232 serial communication port. The command language is referred to by the manufacturer as X Code. The Digiplan controller has no input/output data flow control (no 'hand-shaking' or 'XON/XOFF' capabilities) and therefore the flow control must be handled by the computer software only. The computer relies entirely on interrupt signals from the serial communication port to signal the reception or transmission of data. Figure 4.7 shows the basic set-up of the system hardware.

The serial link is full duplex, meaning that different lines are used for the two signal directions, hence with the use of this system, data can be transmitted and received at the same time. When a character is sent to the controller, the character is echoed back to the computer. The advantage of this scheme is that the user gets verification that the correct character was received by the controller.

Both the transmission and reception of data by the computer is interrupt driven to allow the computer to carry out other supporting tasks as well as the primary task of transmitting and receiving data. Some of these tasks include computation of way points when computing the path of the end effector, generation of X Code for subsequent moves, and maintenance of buffers inside the computer memory.

The Parker Hannifin Digiplan IFX indexer is a multi-axis controller for step motors. It has a number of limitations which limit its usefulness as a robot controller. The controller is not capable of servo control, only endpoint verification. Endpoint verification means that movement commands are executed open loop and the endpoint position as measured by the rotary encoder is compared with the expected position. If a difference is detected then a corrective movement is executed. Such a simple scheme held few advantages since a missed step by the motor generally resulted in total collapse of the mechanism. As a result the motors were run open loop. The X Code control language as implemented on the controller was ill equipped for simultaneous coordination of multiple axes. As a result, synchronisation of multi-axis movements had to be performed at the end of a movement when the robot is at rest. The serial communication link to the controller was primitive and awkward. No hand-shaking of any kind is supported so frequent buffer status requests must be made to ensure that the controller's command buffer is not overfilled. If more than 1 status command is sent to the controller at a time the returned information is interleaved therefore a control logic was implemented in the PC to prevent this happening.
Figure 4.6 The base platform and mounting brackets.

Figure 4.7 Physical set-up of the control system hardware.
Sensors

If the actuated arms of the robot are driven too far, they will collide with the base. The forces induced in the Harmonic drives by the sudden deceleration could well damage them. To detect when the arms have reached the end of their travel, magnetic reed switches were used. The switches were mounted on plates that were attached to the mounting brackets. The switches were triggered by small magnets that were attached to the harmonic drive housing. The limit switches activate the limit switch inputs on the IFX controller causing the controller to stop the movement immediately. If the load has enough inertia then the motors will probably continue to rotate and drop out of synchronisation causing the stopping torque to drop to zero. Rubber buffers in the base platform should absorb the impact as the arm collides with the base.

In addition to end-of-travel sensors, home position sensors are necessary to synchronise the arms. These were optical sensors that are triggered by a disc that was bolted to the reduction housing. The opto-switch was mounted on the same plate as the end-of-travel reed switches.

Control Program

The control program is implemented in the C language. Details of the software and control system design may be found in Lintott and Jones [2000]. The Borland Turbo C version 2.0 compiler and development environment were used and the source code should be compatible with any later Borland C or C++ compiler.

The control program was written such that the routines are divided into a structured hierarchy of levels, each level calling a lower level through a clearly defined protocol. The design facilitates interchangeability of parts so that if, for example, a new motor controller is used then the driver section can be replaced leaving the rest of the program unchanged. Alternatively, a better high level programmer interface could be used and the driver level could remain unchanged as long as the programmer level interfaces correctly to the driver. The scheme is shown in Figures 4.8 and 4.9.

Bottom up design requires a standard set of functions that each module must implement. The prototypes of the functions are specified in a schedule document and any module that complies with the schedule should interface correctly with its input and output modules. The kinematics routines are in the form of a module that can be linked into the compilation. The kinematics module contains routines that implement the direct and inverse geometric solutions, and also (optionally) some collision checking routines.
Figure 4.8 Overall schematic layout of robot control system.

Figure 4.9 Schematic layout of serial communication program for interfacing with the IFX motor controller.
4.3 Simulation

Simulation provides a tool for investigating the stability and reliability of the experimental results. The principal areas of interest are the stability of the solution under the influence of measurement noise and the effect of changing the parameterisation of the model. This section describes the simulation of a variety of geometric models with varying numbers of model parameters and varying levels of noise. The models investigated are the 4 bar chain mechanism, the Delta robot, and the Kiwibot.

For simulation the nominal parameter set represents the robot without geometric errors. A second set of parameters termed the perturbed parameter set represents the physical robot. The aim is to reproduce the perturbed parameters from a set of simulated measurements. Figure 4.10 schematically illustrates the measurement data simulation process. The nominal inverse geometric solution (§2.6.2) is used to convert the nominal measurement positions into nominal actuator coordinates. Normally distributed random noise with variance $\Sigma_o$ and zero mean is added to the actuator coordinates to simulate the finite repeatability of the actuators. The actuator coordinates with added noise and the set of perturbed geometric parameters are passed to the direct geometric model (§2.3) to yield a set of endpoint coordinates. The measurement device model transforms the endpoint coordinates into a set of "raw" data for analysis. The assumption is made that the measurement device adds an unbiased normally distributed random error with covariance $\Sigma_m$ to the readings.

The accuracy of the simulation is limited by the assumptions made about the distribution of measurement noise and repeatability, the complexity of the geometric model, and the behaviour of the measurement device. The assumption of normally distributed noise is justified by the Central Limit Theorem which states that the sum of many independent random errors tends to be normally distributed as long as they are of approximately the same scale. For small errors the geometric model may be linearised, thus it is reasonable to expect the probability distribution of the endpoint pose to be Gaussian.

The geometric model developed in §2.3 cannot span the space of all possible errors. For example, the errors introduced by ball and cup irregularities in spherical joints cannot be perfectly modelled with 3 coincident rotary joints as in the Delta model. Similarly, running errors in rotary joint bearings, gravitational effects, thermal effects, and many other lesser sources of error are not simulated. The measurement device may also introduce non Gaussian noise into the experimental data.
Figure 4.10: The simulation process.
4.3.1 Four Bar Chain Simulation

The tables and figures referred to in the following paragraphs appear on pages 89–96.

The model used for the simulation of the 4 bar chain mechanism is shown in Figure 4.11. The perturbed model was generated by adding random numbers to the geometric parameters (cf. Tables 4.2a, 4.2b) and various amounts of Gaussian noise were added (cf. Table 4.2c). In order to assess the probability distribution of the resulting parameter vector a Monte Carlo type approach was used. This involves performing multiple simulations, each with a different sample from the random noise distribution. A linear analysis of the parameter variance (as described in §3.5.1) was also carried out for comparison.

A useful quantity to examine is the residual error after identification

\[
   r = \begin{bmatrix}
      W\varepsilon_1 \\
      W\varepsilon_2 \\
      \vdots \\
      W\varepsilon_n
   \end{bmatrix}
\]

where \( \varepsilon_i \) is defined in equation 3.1. The 2-norms of the residuals before and after calibration are compared in Figures 4.12 and 4.13. Obviously the magnitude of the measurement and actuator noise affects the norm of the residual; higher noise levels lead to larger residuals. Figure 4.12 shows the residual norm as a function of measurement noise. Traces for actuator noise with variance \( s_\theta = 0 \) and \( s_\theta = 1 \times 10^{-6} \) are shown. The norm of the residual is predominantly a function of the actuator noise until the magnitude of the measurement noise becomes comparable at approximately \( s_m = 1 \times 10^{-7} \). Figure 4.13 shows the residual norm as a function of actuator noise. Traces for measurement noise with variance \( s_m = 0 \) and \( s_m = 2.5 \times 10^{-7} \) are shown. When measurement noise is present the curve remains flat for low levels of actuator noise. This indicates that measurement noise is the dominant influence. When the actuator noise becomes significant in comparison with the measurement noise (at \( s_\theta \approx 1 \times 10^{-7} \)), the norm of the residual increases accordingly. It can be concluded that both measurement and actuator noise are significant factors in the final accuracy that can be obtained for this mechanism and that an error analysis should model both sources of error.

The effect on the residual of varying the number of measurement points is shown in Figure 4.14. For noiseless measurements, the norm of the residual is small as long as there are enough measurements to make a fully determined set of equations. For noisy measurements the norm of the residual decreases monotonically with the number of measurements. This is probably because the net effect of the errors tends towards zero as the sample size grows. The improvement in accuracy must be balanced against the disadvantage of significantly increased computation time (cf. Figure 4.15).

It was found that the Monte Carlo results contained the occasional outlier.
indicates that some caution is necessary when accepting the result of an identification process. Outliers are detectable because the parameter values are far from their nominal values. Figure 4.16 shows the physical mechanism that corresponds to one of the outliers. The physical distortion is typically large enough to detect by eye and thus the solution is obviously invalid. The effect of the outliers is to badly distort the covariance matrix and thus it is better to remove them from the results. Figure 4.17 shows how outliers are detectable from the expression

\[ r_o = |\bar{a} - a_0| . \] (4.2)

The distribution of the results (with outliers removed) from a Monte Carlo analysis (run 3) is shown in Figure 4.18. Inspection by eye indicates that the marginal distribution of results for each parameter is roughly normal without marked distortion. A more advanced test is the chi-square plot (cf. §3.5.3). Figure 4.19 shows the chi-square plot for run 3 along with some comparable plots of pseudo-random data sets having the same covariance. It is apparent that the Monte Carlo results deviate from normality. The trend in Figure 4.19 indicates that the results are centrally biased relative to a normal distribution within an elliptical region of Mahalanobis radius\(^1\) \(\sqrt{31}\). Outside this region the results are spread more sparingly than the normal distribution would predict. Using the method of §3.5.2 (equation 3.41), the 95% confidence ellipsoid has Mahalanobis radius

\[ d = \sqrt{\chi^2_{26} \left( \left( \frac{95 - \frac{1}{2}}{100} \right) / 100 \right)} \] (4.3)

\[ = 6.2 \] (4.4)

however it only contains 86% of the points. The ellipsoid containing 95% of points has Mahalanobis radius 7.1. Note that the number of test points in the Monte Carlo analysis is too few to make firm inferences on 26 parameters, however the computation time needed to generate the number of points required for a firm inference is prohibitive.

Comparing the covariance of the Monte Carlo results \(\Sigma_a\) to the covariance matrix obtained by linear analysis \(\Sigma_a\) (§3.5.1), it can be seen that the main diagonals of the covariance matrices appear to correspond reasonably well (cf. Table 4.3). The similarity metric from §A.5 is \(\bar{c}(\Sigma_a, \Sigma_a^*) = 0.35\), which indicates poor similarity. The eigenvalues of the 2 matrices (also tabulated) agree except for the fourth which is shown in the fifth column of Table 4.3. The eighth element of the fourth eigenvector accounts for most of that eigenvector's magnitude. This corresponds to the eighth parameter of the model which happens to be \(\alpha_2\) for joint 2 (denoted \(\alpha_{2,2}\)). Physically this parameter corresponds to a rotation about the axis of the bar joining frames 1 and 2 (shown as a black arrow in Figure 4.11). Because frame 2 is a closure frame for a ball joint, such a rotation has no

---

\(\text{1 The term Mahalanobis radius is used here to define the characteristic dimension of a hyperellipsoid and is defined by equation 3.43, p. 66}\)
effect on the kinematics of the model and should perhaps have been constrained in the Monte Carlo analysis. This would eliminate the discrepancy. It is concluded that the covariance matrix derived by linear analysis makes a reasonable approximation to the results of the Monte Carlo analysis as long as redundant parameters are constrained.

Confidence intervals are constructed using equation 3.42 ($\S$3.5.2) and the widths of the confidence intervals for each parameter in each run are tabulated in Table 4.3. Note that some confidence intervals are relatively large. Investigation of the sensitivity of the cost function to combinations of parameter perturbations reveals that there are directions of low sensitivity. The confidence hyperellipsoid is elongated in these directions. To demonstrate this, consider the eigenvectors $e_j$ of the covariance matrix $\Sigma_a$. Each represents a principal axis of the confidence hyperellipsoid, thus the sensitivity of the cost function $\rho$ to perturbations in $a$ along the eigenvector $e_j$ is

$$\frac{\delta \rho}{\delta \lambda_j} = e_j^T J_{ID}^T J_{ID} e_j$$

These sensitivities are shown relative to the extent of the confidence hyperellipsoid generated by Monte Carlo analysis for run 3 (cf. Figure 4.20 and Table 4.3). Note the inverse relationship between the sensitivity and the confidence interval width.

The Monte Carlo analysis demonstrates that the result of the calibration process is relatively robust under the influence of noise and that a linear error analysis can be used to estimate the variance of the parameters as long as noise levels are relatively small.
Figure 4.11  Arrangement of the kinematic model for simulation of the 4 bar closed loop mechanism. The mini-axes represent joint frames. The blue axes (with arrow heads) represent the axis of rotation for rotary joints. The numerals represent the joint number. The only closure is between joints 2 and 6 and joint 6 is oriented differently to joint 2 to illustrate the relaxed conditions for closure at ball joints. The circular arrow represents parameter $\alpha_{2,2}$ which was erroneously left unconstrained during the Monte Carlo analysis.
### a. Nominal parameter set

<table>
<thead>
<tr>
<th>Body</th>
<th>$\alpha_1$</th>
<th>$\alpha_2$</th>
<th>$\alpha_3$</th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>Predecessor</th>
<th>Closure</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>1.0000</td>
<td>0.0000</td>
<td>1</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>1.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>-1.5708</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>3</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1.5708</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>1.0000</td>
<td>0.0000</td>
<td>4</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>-1.5708</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-1.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>5</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

### b. Perturbed parameter set

<table>
<thead>
<tr>
<th>Body</th>
<th>$\alpha_1$</th>
<th>$\alpha_2$</th>
<th>$\alpha_3$</th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>Predecessor</th>
<th>Closure</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0095</td>
<td>0.0046</td>
<td>0.0020</td>
<td>0.0092</td>
<td>0.0041</td>
<td>0.0000</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.0023</td>
<td>0.0002</td>
<td>0.0000</td>
<td>0.0074</td>
<td>1.0089</td>
<td>0.0000</td>
<td>1</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0.0061</td>
<td>0.0082</td>
<td>0.0000</td>
<td>1.0018</td>
<td>0.0006</td>
<td>0.0000</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>-1.5659</td>
<td>0.0044</td>
<td>0.0000</td>
<td>0.0041</td>
<td>0.0035</td>
<td>0.0000</td>
<td>3</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1.5797</td>
<td>0.0062</td>
<td>0.0000</td>
<td>0.0094</td>
<td>1.0081</td>
<td>0.0000</td>
<td>4</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>-1.5632</td>
<td>0.0079</td>
<td>0.0000</td>
<td>-0.9908</td>
<td>0.0001</td>
<td>0.0060</td>
<td>5</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

### c. Noise covariances applied in Monte Carlo tests

<table>
<thead>
<tr>
<th>Run</th>
<th>Measurement ($s_m$)</th>
<th>Actuator ($s_\theta$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$1 \times 10^{-4}$</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>$2.5 \times 10^{-6}$</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>$2.5 \times 10^{-8}$</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>$2.5 \times 10^{-9}$</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>$2.5 \times 10^{-11}$</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>$1 \times 10^{-4}$</td>
<td>$1 \times 10^{-6}$</td>
</tr>
<tr>
<td>7</td>
<td>$2.5 \times 10^{-6}$</td>
<td>$1 \times 10^{-6}$</td>
</tr>
<tr>
<td>8</td>
<td>$2.5 \times 10^{-8}$</td>
<td>$1 \times 10^{-6}$</td>
</tr>
<tr>
<td>9</td>
<td>$2.5 \times 10^{-9}$</td>
<td>$1 \times 10^{-6}$</td>
</tr>
<tr>
<td>10</td>
<td>$2.5 \times 10^{-11}$</td>
<td>$1 \times 10^{-6}$</td>
</tr>
</tbody>
</table>

Table 4.2 Input data for Monte Carlo simulation experiments on the 4 bar chain mechanism. In Table 4.2c, the covariance matrices $\Sigma_m$ and $\Sigma_\theta$ are calculated from the table using $\Sigma_m = W_\xi s_m$ and $\Sigma_\theta = W_\zeta s_\theta$ where $s_m$ and $s_\theta$ are the appropriate table entries and $W_\xi$ is a weighting matrix that scales the linear and rotational parameters to approximately the same magnitude (for these tests, $W_\xi = I$).
Figure 4.12 Residuals of Monte Carlo tests with constant actuator noise.

Figure 4.13 Residuals of tests with constant measurement noise.
Figure 4.14 Residuals of tests with varying number of measurement points.

Figure 4.15 Computation time for identification algorithm run on a 400 MIPS Pentium III computer running Windows NT.
Figure 4.16 Example of a mechanism that corresponds to an outlier.

Figure 4.17 Outliers are easily detected by examining the difference between the individual solutions of the Monte Carlo tests and the nominal solution. The dashed line represents the rejection threshold.
Figure 4.18 Marginal distributions of Monte Carlo results for run 3.
Figure 4.19 Chi-square plot for run 3 along with comparable normally distributed data sets.

Figure 4.20 Comparison of the magnitude of confidence ellipsoid major axes to sensitivity along the axis. Both are normalised. The confidence ellipsoid is derived from the Monte Carlo results, while the sensitivity is derived from the linear analysis. High sensitivities generally correspond to narrow confidence bands although parameter 4 is an exception.
Table 4.3 Main diagonals and eigenvalues of $\Sigma_a$ and $\Sigma^*_a$ for comparison. Column 5 contains the eigenvector corresponding to the fourth eigenvalue. Column 6 contains the sensitivities of the cost function to parameter perturbations along each eigenvector. Columns 6 and 7 are plotted against each other for comparison in Figure 4.20. Bold elements are discussed p. 87.

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
<th>Column 4</th>
<th>Column 5</th>
<th>Column 6</th>
<th>Column 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Sigma_a$</td>
<td>$\Sigma^*_a$</td>
<td>$\lambda_j = \text{eig}(\Sigma_a)$</td>
<td>$\lambda_j^* = \text{eig}(\Sigma^*_a)$</td>
<td>Eigenvector $e_d(\Sigma^*_a)$</td>
<td>$\frac{\delta \rho}{\delta \lambda_j}$</td>
<td>Confidence interval width</td>
</tr>
<tr>
<td>$2.01 \times 10^{-5}$</td>
<td>$1.58 \times 10^{-5}$</td>
<td>$4.25 \times 10^{-5}$</td>
<td>$3.97 \times 10^{-5}$</td>
<td>0.018</td>
<td>0.023</td>
<td>0.0265</td>
</tr>
<tr>
<td>$1.9 \times 10^{-5}$</td>
<td>$1.58 \times 10^{-5}$</td>
<td>$3.47 \times 10^{-5}$</td>
<td>$3.67 \times 10^{-5}$</td>
<td>-0.0173</td>
<td>0.0072</td>
<td>0.0266</td>
</tr>
<tr>
<td>$1.5 \times 10^{-5}$</td>
<td>$1.4 \times 10^{-5}$</td>
<td>$1.89 \times 10^{-5}$</td>
<td>$1.78 \times 10^{-5}$</td>
<td>0.308</td>
<td>0.03</td>
<td>0.0233</td>
</tr>
<tr>
<td>$3.67 \times 10^{-6}$</td>
<td>$3.67 \times 10^{-6}$</td>
<td>$5 \times 10^{-7}$</td>
<td>$1.38 \times 10^{-5}$</td>
<td>-0.122</td>
<td>139</td>
<td>0.0119</td>
</tr>
<tr>
<td>$3.67 \times 10^{-6}$</td>
<td>$3.35 \times 10^{-6}$</td>
<td>$2.25 \times 10^{-7}$</td>
<td>$1.92 \times 10^{-6}$</td>
<td>-0.179</td>
<td>2.2</td>
<td>0.0114</td>
</tr>
<tr>
<td>$4.25 \times 10^{-8}$</td>
<td>$4.37 \times 10^{-8}$</td>
<td>$2.22 \times 10^{-7}$</td>
<td>$6.07 \times 10^{-7}$</td>
<td>-0.00676</td>
<td>1.5</td>
<td>0.0013</td>
</tr>
<tr>
<td>$3.47 \times 10^{-7}$</td>
<td>$2.94 \times 10^{-7}$</td>
<td>$1.73 \times 10^{-7}$</td>
<td>$2.58 \times 10^{-7}$</td>
<td>0.0163</td>
<td>1.4</td>
<td>0.00338</td>
</tr>
<tr>
<td>$1.01 \times 10^{-9}$</td>
<td>$\textbf{1.45 \times 10^{-8}}$</td>
<td>$1.04 \times 10^{-7}$</td>
<td>$1.67 \times 10^{-7}$</td>
<td>$\textbf{0.906}$</td>
<td>7</td>
<td>0.0237</td>
</tr>
<tr>
<td>$2.58 \times 10^{-7}$</td>
<td>$2.31 \times 10^{-7}$</td>
<td>$9.76 \times 10^{-8}$</td>
<td>$1.59 \times 10^{-7}$</td>
<td>-0.0279</td>
<td>2.6</td>
<td>0.003</td>
</tr>
<tr>
<td>$2.47 \times 10^{-15}$</td>
<td>$1.73 \times 10^{-14}$</td>
<td>$5.9 \times 10^{-8}$</td>
<td>$1.19 \times 10^{-7}$</td>
<td>$4.62 \times 10^{-6}$</td>
<td>4.2</td>
<td>8.19 $\times 10^{-7}$</td>
</tr>
<tr>
<td>$2.33 \times 10^{-14}$</td>
<td>$1.67 \times 10^{-14}$</td>
<td>$4.98 \times 10^{-8}$</td>
<td>$9.62 \times 10^{-8}$</td>
<td>$-5.23 \times 10^{-6}$</td>
<td>5.1</td>
<td>8.05 $\times 10^{-7}$</td>
</tr>
<tr>
<td>$4.26 \times 10^{-8}$</td>
<td>$3.76 \times 10^{-8}$</td>
<td>$2.13 \times 10^{-8}$</td>
<td>$7.49 \times 10^{-8}$</td>
<td>0.0126</td>
<td>16</td>
<td>0.00121</td>
</tr>
<tr>
<td>$3.23 \times 10^{-8}$</td>
<td>$1.45 \times 10^{-6}$</td>
<td>$1.64 \times 10^{-8}$</td>
<td>$5.32 \times 10^{-8}$</td>
<td>0.122</td>
<td>15</td>
<td>0.0075</td>
</tr>
<tr>
<td>$1.3 \times 10^{-7}$</td>
<td>$1.34 \times 10^{-7}$</td>
<td>$1.11 \times 10^{-8}$</td>
<td>$4.47 \times 10^{-8}$</td>
<td>0.00222</td>
<td>23</td>
<td>0.00228</td>
</tr>
<tr>
<td>$3.18 \times 10^{-8}$</td>
<td>$1.4 \times 10^{-6}$</td>
<td>$7.32 \times 10^{-9}$</td>
<td>$1.52 \times 10^{-8}$</td>
<td>0.1</td>
<td>36</td>
<td>0.00738</td>
</tr>
<tr>
<td>$3.81 \times 10^{-7}$</td>
<td>$3.45 \times 10^{-7}$</td>
<td>$2.68 \times 10^{-9}$</td>
<td>$1.14 \times 10^{-8}$</td>
<td>0.00793</td>
<td>93</td>
<td>0.00366</td>
</tr>
<tr>
<td>$3.94 \times 10^{-8}$</td>
<td>$2.29 \times 10^{-15}$</td>
<td>$1.2 \times 10^{-9}$</td>
<td>$8.18 \times 10^{-9}$</td>
<td>$-1.17 \times 10^{-6}$</td>
<td>209</td>
<td>2.98 $\times 10^{-7}$</td>
</tr>
<tr>
<td>$3.15 \times 10^{-9}$</td>
<td>$9.57 \times 10^{-9}$</td>
<td>$7.9 \times 10^{-10}$</td>
<td>$4.99 \times 10^{-9}$</td>
<td>$-2.27 \times 10^{-6}$</td>
<td>316</td>
<td>1.93 $\times 10^{-7}$</td>
</tr>
<tr>
<td>$1.06 \times 10^{-8}$</td>
<td>$1.09 \times 10^{-8}$</td>
<td>$6.63 \times 10^{-10}$</td>
<td>$2.85 \times 10^{-9}$</td>
<td>$-0.00168$</td>
<td>423</td>
<td>0.00965</td>
</tr>
<tr>
<td>$8.95 \times 10^{-8}$</td>
<td>$9.55 \times 10^{-8}$</td>
<td>$2.27 \times 10^{-22}$</td>
<td>$1.91 \times 10^{-9}$</td>
<td>$-0.00316$</td>
<td>$9.7 \times 10^{-22}$</td>
<td>0.00133</td>
</tr>
<tr>
<td>$8.72 \times 10^{-8}$</td>
<td>$7.93 \times 10^{-8}$</td>
<td>$7.72 \times 10^{-24}$</td>
<td>$1.34 \times 10^{-9}$</td>
<td>0.00422</td>
<td>3.0 $\times 10^{-25}$</td>
<td>0.00176</td>
</tr>
<tr>
<td>$1.74 \times 10^{-5}$</td>
<td>$1.76 \times 10^{-5}$</td>
<td>$5.36 \times 10^{-24}$</td>
<td>$5.12 \times 10^{-10}$</td>
<td>0.0647</td>
<td>1.1 $\times 10^{-25}$</td>
<td>0.0262</td>
</tr>
<tr>
<td>$2.95 \times 10^{-5}$</td>
<td>$2.99 \times 10^{-7}$</td>
<td>$1.3 \times 10^{-24}$</td>
<td>$2.36 \times 10^{-14}$</td>
<td>0.0287</td>
<td>1.1 $\times 10^{-24}$</td>
<td>0.00341</td>
</tr>
<tr>
<td>$1.71 \times 10^{-5}$</td>
<td>$1.72 \times 10^{-5}$</td>
<td>$1.05 \times 10^{-24}$</td>
<td>$7.13 \times 10^{-16}$</td>
<td>$-0.0057$</td>
<td>1.7 $\times 10^{-25}$</td>
<td>0.0258</td>
</tr>
<tr>
<td>$2.08 \times 10^{-8}$</td>
<td>$1.67 \times 10^{-8}$</td>
<td>$7.92 \times 10^{-25}$</td>
<td>$1.14 \times 10^{-16}$</td>
<td>0.00325</td>
<td>1.5 $\times 10^{-25}$</td>
<td>0.000805</td>
</tr>
<tr>
<td>$3.15 \times 10^{-8}$</td>
<td>$3.87 \times 10^{-8}$</td>
<td>$4.82 \times 10^{-25}$</td>
<td>$1.49 \times 10^{-17}$</td>
<td>$-0.00139$</td>
<td>6.7 $\times 10^{-25}$</td>
<td>0.00123</td>
</tr>
</tbody>
</table>
4.3 Simulation

4.3.2 Kiwibot Simulation

The tables and figures referred to in the following paragraphs appear on pages 99-105.

Because the Kiwibot model (with 16 bodies and 2 joint closures, cf. Figure 4.21) is more complex than the 4 bar chain mechanism, a Monte Carlo approach would require excessive computation time. In order to verify that the method works effectively for this model, 3 simulations were carried out. The first simulation used full pose measurements in 160 positions approximately spanning a hemisphere (cf. Figure 4.22). The second simulation used orientation measurement only, as might be returned by some variety of inclinometer mounted on the nacelle. The third simulation introduced simulated measurement noise to the data used for the second simulation.

Figure 4.21 shows the geometric model used for the Kiwibot simulation. One of the 3 ball joints is simulated by 3 coincident rotary joints while the other 2 ball joints constitute closure frames. Frame 10 is considered to be the tool frame, and it is the only frame that the measurement device senses. Table 4.4 gives the definition matrix for the model. A number of model parameters were artificially constrained in order to remove redundancy and decrease the complexity of the model. Frame 1 is fixed and coincident with the ground frame and must be completely immobile. The model could easily be defined without this frame but it makes a useful reference and enables the entire model to be re-oriented or translated easily. Frame 5 exists in order to allow frame 6 to be oriented with its 3 axis along the actuator arm as in the physical mechanism. Frames 6, 7, and 8 constitute a simulated ball joint, and therefore do not need any freedom for identification; they are free to rotate about their 3 axes as free joints, but their MCPC parameters remain fixed. Frame 10 is the tool frame and it also acts as the reference point for the nacelle. Since the nacelle is a single solid body, it must be fixed. The closures between joints 13-15 and 14-16 need only have 3 freedoms because they are modeled as spherical joints, therefore the $d_2$ freedoms are constrained. Finally, all type 2 joints (spherical closure joints such as joints 15 and 16) must have their $d_3$ freedom constrained.

The results for all 3 simulations are shown in Table 4.5. The results are shown as differences from the perturbed model. In the first simulation, the norm of the residual was reduced from $1.54 \times 10^{-2}$ to $7.54 \times 10^{-10}$. The identified model is not identical to the perturbed model that was used to generate the simulated measurements, but it reproduces the measured positions with high accuracy. The differences between the identified and perturbed model are not due to the solver terminating before a true minimum was found. Rather, both the perturbed model and the identified solution are valid solutions to the identification problem.

Because the mechanism has only 3 degrees of freedom, implementation cannot correct for positional and orientation errors simultaneously. The robot is primarily designed for beam aiming applications so orientation errors are more significant than positional errors. Accordingly, the implementation procedure was adjusted to minimise orientation
errors only. The positioning and orientation error, before and after calibration, is shown in Figures 4.23-4.26. The positioning errors are shown relative to the nominal grid (cf. Figure 4.22). Rotational parameters are shown in a similar manner. The lines represent the axis of rotation, and the length of the line is proportional the angle of rotation about the axis.

It was initially found that in attempting to reduce orientation errors, the positional errors became extremely large. This is most likely because the mechanism can achieve the same orientation in a wide variety of poses. The term "strut length" describes the distance between the centroids of the base platform and the nacelle\(^2\). A given orientation can be achieved at any mechanically feasible strut length. The implementation routine initially had no constraint governing the strut length and therefore found solutions that were up to 600 mm from the starting position. A bias function based on the strut length was added to the cost function which solved this problem. The solver was essentially constrained to find a solution that maintained the distance between base and nacelle centres.

The second simulation used only the orientation of the platform as measurement data. The same perturbed model was applied, and the norm of the residual was reduced from \(1.23 \times 10^{-2}\) to \(1.19 \times 10^{-8}\). Once again the model was not identical to the perturbed model, but it did converge to a solution that was similar to that of the first simulation. The pose errors after implementation are shown in Figures 4.27 and 4.28.

Measurement noise of variance \(1.28 \times 10^{-9}\) and repeatability noise of variance \(1.31 \times 10^{-9}\) was added to the measurements to simulate realistic noise levels in the third simulation. The norm of the residual was reduced from 0.0124 to \(8.66 \times 10^{-4}\). The pose errors after calibration, are shown in Figures 4.29 and 4.30.

Overall, the results show that the Kiwibot can be successfully calibrated using orientation measurements despite realistic levels of measurement noise. The improvement in pose accuracy is significant and of great advantage for large antenna beam aiming applications.

\(^2\)i.e. The distance between frames 1 and 10.
Figure 4.21 Arrangement of the kinematic model for simulation of the Kiwibot. The mini-axes represent joint frames, the blue axes represent the axis of rotation for rotary joints, and the numerals represent the joint number. The model is shown with both closure joints disconnected.

Table 4.4 Nominal model definition matrix for the Kiwibot. Key: • indicates free parameters in the model, ○ indicates artificially constrained parameters, unmarked parameters are automatically constrained.
<table>
<thead>
<tr>
<th>Param.</th>
<th>Nominal Model</th>
<th>Perturbed Model</th>
<th>Simulation 1</th>
<th>Simulation 2</th>
<th>Simulation 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2, a1</td>
<td>1.5708</td>
<td>1.5708</td>
<td>-1.2 x 10^-11</td>
<td>4 x 10^-10</td>
<td>3.6 x 10^-6</td>
</tr>
<tr>
<td>2, a1</td>
<td>1.5708</td>
<td>1.5708</td>
<td>8.3 x 10^-11</td>
<td>-1 x 10^-5</td>
<td>-7.2 x 10^-8</td>
</tr>
<tr>
<td>2, a2</td>
<td>0</td>
<td>0.00031275</td>
<td>1.8 x 10^-10</td>
<td>4.1 x 10^-6</td>
<td>5.1 x 10^-6</td>
</tr>
<tr>
<td>2, a3</td>
<td>0</td>
<td>-4.644 x 10^-6</td>
<td>4.2 x 10^-10</td>
<td>5.7 x 10^-6</td>
<td>6.5 x 10^-5</td>
</tr>
<tr>
<td>2, d1</td>
<td>0.28054</td>
<td>0.28046</td>
<td>9.5 x 10^-11</td>
<td>-0.0017</td>
<td>-2.1 x 10^-5</td>
</tr>
<tr>
<td>2, d2</td>
<td>0</td>
<td>7.701 x 10^-6</td>
<td>-9.8 x 10^-11</td>
<td>0.00014</td>
<td>2.1 x 10^-5</td>
</tr>
<tr>
<td>3, a1</td>
<td>1.5708</td>
<td>1.5708</td>
<td>-2.5 x 10^-6</td>
<td>0.00024</td>
<td>6.2 x 10^-5</td>
</tr>
<tr>
<td>3, a2</td>
<td>2.0944</td>
<td>2.094</td>
<td>-8.6 x 10^-9</td>
<td>4.8 x 10^-8</td>
<td>4.4 x 10^-5</td>
</tr>
<tr>
<td>3, a3</td>
<td>0</td>
<td>0.000257</td>
<td>-2.4 x 10^-8</td>
<td>-0.0018</td>
<td>7.7 x 10^-5</td>
</tr>
<tr>
<td>3, d1</td>
<td>0.28054</td>
<td>0.28069</td>
<td>-3.8 x 10^-10</td>
<td>0.00012</td>
<td>-4.4 x 10^-5</td>
</tr>
<tr>
<td>3, d2</td>
<td>0</td>
<td>-0.2891 x 10^-5</td>
<td>6.9 x 10^-9</td>
<td>3.9 x 10^-6</td>
<td>-3.9 x 10^-5</td>
</tr>
<tr>
<td>4, a1</td>
<td>1.5708</td>
<td>1.5707</td>
<td>-4.8 x 10^-8</td>
<td>-0.0019</td>
<td>-0.00013</td>
</tr>
<tr>
<td>4, a2</td>
<td>-2.0944</td>
<td>-2.094</td>
<td>-1.3 x 10^-6</td>
<td>-2.3 x 10^-5</td>
<td>3.2 x 10^-5</td>
</tr>
<tr>
<td>4, a3</td>
<td>0</td>
<td>0.0001152</td>
<td>-1 x 10^-9</td>
<td>-0.00018</td>
<td>-6.2 x 10^-5</td>
</tr>
<tr>
<td>4, d1</td>
<td>0.28054</td>
<td>0.28067</td>
<td>-8.8 x 10^-10</td>
<td>0.001</td>
<td>-2.8 x 10^-5</td>
</tr>
<tr>
<td>4, d2</td>
<td>0</td>
<td>-0.0002635</td>
<td>-3.3 x 10^-9</td>
<td>6.5 x 10^-5</td>
<td>1.7 x 10^-5</td>
</tr>
<tr>
<td>9, a1</td>
<td>0</td>
<td>0.00048966</td>
<td>-0.00049</td>
<td>-0.00049</td>
<td>-0.00049</td>
</tr>
<tr>
<td>9, a2</td>
<td>1.5708</td>
<td>1.5709</td>
<td>-0.00013</td>
<td>-0.00013</td>
<td>-0.00013</td>
</tr>
<tr>
<td>9, d1</td>
<td>0.6</td>
<td>0.59987</td>
<td>4.6 x 10^-9</td>
<td>1.7 x 10^-5</td>
<td>-1.5 x 10^-5</td>
</tr>
<tr>
<td>9, d2</td>
<td>0</td>
<td>-7.475 x 10^-5</td>
<td>7.5 x 10^-5</td>
<td>7.5 x 10^-5</td>
<td>1.9 x 10^-9</td>
</tr>
<tr>
<td>11, a1</td>
<td>1.5708</td>
<td>1.5707</td>
<td>-2.9 x 10^-6</td>
<td>0.00021</td>
<td>-2.2 x 10^-5</td>
</tr>
<tr>
<td>11, a2</td>
<td>2.0944</td>
<td>2.094</td>
<td>-1.1 x 10^-8</td>
<td>3.7 x 10^-5</td>
<td>-4.2 x 10^-5</td>
</tr>
<tr>
<td>11, d1</td>
<td>-0.28054</td>
<td>-0.28091</td>
<td>-5.4 x 10^-10</td>
<td>-3.3 x 10^-5</td>
<td>6.7 x 10^-5</td>
</tr>
<tr>
<td>11, d2</td>
<td>0</td>
<td>-0.00046711</td>
<td>-8 x 10^-9</td>
<td>0.00012</td>
<td>4.2 x 10^-5</td>
</tr>
<tr>
<td>12, a1</td>
<td>1.5708</td>
<td>1.5713</td>
<td>-5.3 x 10^-8</td>
<td>-0.00013</td>
<td>0.0011</td>
</tr>
<tr>
<td>12, a2</td>
<td>-2.0944</td>
<td>-2.094</td>
<td>-1.6 x 10^-8</td>
<td>6.7 x 10^-6</td>
<td>3.7 x 10^-5</td>
</tr>
<tr>
<td>12, d1</td>
<td>-0.28054</td>
<td>-0.28014</td>
<td>2.2 x 10^-9</td>
<td>-2.2 x 10^-5</td>
<td>3 x 10^-5</td>
</tr>
<tr>
<td>12, d2</td>
<td>0</td>
<td>-0.00041224</td>
<td>5 x 10^-9</td>
<td>5.4 x 10^-5</td>
<td>-5.3 x 10^-5</td>
</tr>
<tr>
<td>13, a1</td>
<td>0</td>
<td>0.0001265</td>
<td>-0.00013</td>
<td>-0.00013</td>
<td>-2.7 x 10^-8</td>
</tr>
<tr>
<td>13, a2</td>
<td>0</td>
<td>-0.00035271</td>
<td>1.8 x 10^-8</td>
<td>0.00012</td>
<td>8.3 x 10^-7</td>
</tr>
<tr>
<td>13, d1</td>
<td>0.5</td>
<td>0.59968</td>
<td>-2 x 10^-10</td>
<td>8.5 x 10^-5</td>
<td>7.8 x 10^-6</td>
</tr>
<tr>
<td>14, a1</td>
<td>0</td>
<td>0.00010595</td>
<td>-0.00011</td>
<td>-0.00011</td>
<td>-2.2 x 10^-8</td>
</tr>
<tr>
<td>14, a2</td>
<td>0</td>
<td>0.00035953</td>
<td>8 x 10^-8</td>
<td>-0.00017</td>
<td>-9.7 x 10^-6</td>
</tr>
<tr>
<td>14, d1</td>
<td>0.6</td>
<td>0.60005</td>
<td>-7.4 x 10^-9</td>
<td>4.4 x 10^-5</td>
<td>-5 x 10^-5</td>
</tr>
<tr>
<td>15, a1</td>
<td>0</td>
<td>-0.00033905</td>
<td>0.00034</td>
<td>0.00034</td>
<td>1.7 x 10^-11</td>
</tr>
<tr>
<td>15, a2</td>
<td>0</td>
<td>0.00046351</td>
<td>2.1 x 10^-8</td>
<td>-0.0013</td>
<td>0.00011</td>
</tr>
<tr>
<td>15, d1</td>
<td>-0.6</td>
<td>-0.59952</td>
<td>9.2 x 10^-10</td>
<td>-7.4 x 10^-5</td>
<td>-1.5 x 10^-5</td>
</tr>
<tr>
<td>16, a1</td>
<td>0</td>
<td>-3.1958 x 10^-5</td>
<td>3.2 x 10^-6</td>
<td>3.2 x 10^-5</td>
<td>6.2 x 10^-13</td>
</tr>
<tr>
<td>16, a2</td>
<td>0</td>
<td>-0.00040755</td>
<td>5.4 x 10^-8</td>
<td>6.1 x 10^-5</td>
<td>-4.7 x 10^-5</td>
</tr>
<tr>
<td>16, d2</td>
<td>-0.6</td>
<td>-0.60004</td>
<td>6.1 x 10^-9</td>
<td>-3.8 x 10^-5</td>
<td>4.2 x 10^-5</td>
</tr>
</tbody>
</table>

Table 4.5 Results of simulated calibrations on the Kiwibot. Column 1 identifies the model parameter (the leading digit represents the body number).
Figure 4.22  The measurement grid used for simulated Kiwibot calibration.
Figure 4.23  Positioning error, for first simulation, before calibration on the measurement grid. Errors are magnified by a factor of 25. The worst deviation is 3.18 mm.

Figure 4.24  Positioning error for first simulation after calibration. The worst deviation is 3.92 mm. Errors are magnified by a factor of 25.
Figure 4.25  Orientation error for first simulation before calibration. The lines represent the axis of rotation, the length of the line is proportional to the angle of rotation. The largest angular error is 0.535 deg.

Figure 4.26  Orientation error, for first simulation of the Kiwibot, after calibration. The largest angular error is 0.0519 deg.
Figure 4.27 Position error after implementation for second simulation of the Kiwibot. Errors are magnified by a factor of 25. The worst deviation is 3.92 mm.

Figure 4.28 Orientation error after implementation for second simulation of the Kiwibot. The largest angular error is 0.0518 deg.
4.3 SIMULATION

Figure 4.29 Position error after implementation for third simulation of the Kiwibot. Errors are magnified by a factor of 25. The worst deviation is 3.89 mm.

Figure 4.30 Orientation error, for first simulation of the Kiwibot, after calibration. The largest angular error is 0.0524 deg.
4.3.3 Delta Robot Simulation

The geometric model used for the Delta robot simulation is shown in Figures 4.31 and 4.32. Once again the complexity of the model prevents the use of Monte Carlo analysis. The nominal model definition matrix is shown in Table 4.6. As for the Kiwibot model, artificially constrained parameters reduce redundancy and decrease the complexity of the model. Frame 1 is fixed and identical to the base frame. Frame 6 already has 3 degrees of translational freedom relative to the base frame because frames 2 and 5 are free, therefore the freedoms of frame 6 are redundant and are constrained (similarly for frames 16, 19, 22, 25, and 28). Frames 7, 8 and 9 form a simulated ball joint and therefore do not need free parameters except for 1 which corresponds to the length of the parallel arm. Since frame 17 forms one half of a spherical closure pair, it only requires 3 degrees of freedom. It already has 2 degrees of freedom from the joint freedoms of joints 15 and 16 and therefore only requires 1 free parameter which corresponds to the length of the rod (similarly for joints 20, 23, 26, and 29).

Three simulations were performed in order to verify that the algorithm was working as expected. The first simulation involved using simulated full pose measurements with no noise as measurement data. The measurement points were arranged in a 3-dimensional grid identical to the grid used for the experimental measurements (cf. §4.4.2). It was found that if the first estimate of the Hessian matrix for the BFGS solver was too large, the solver would attempt to close badly distorted mechanisms. As the Hessian estimate was progressively refined, the performance of the algorithm improved, however the initial slow start can be avoided by choosing a better starting Hessian. It was found that \( H_0 = 1 \times 10^{-4} I \) was a good value to begin with. Occasionally the mechanism was so badly distorted that closure could not be achieved. This can be detected by testing the value of the cost function returned by connect.m. If the cost function was greater than \( 1 \times 10^{-10} \), the mechanism was considered to have failed to close. When this occurs, the model \( a_k \) is considered to be non-feasible and the value Inf is returned to the BFGS solver. This forces the solver to reduce its step length and try a model that is closer to the last successful one \( a_{k-1} \). While this violates the assumption of continuity in the cost function \( \rho \), it did not cause the algorithm to fail in any instance.

It was found that convergence was slow when the solver was near the solution. This was because the BFGS algorithm used would continue to select downward steps even when the step size was negligibly small. A related problem was that the update to the Hessian was being contaminated by small numerical errors during the small steps. The problems were solved by altering the termination criteria such that the algorithm would terminate if the step was smaller than a preset tolerance, or if the predicted decrease in the cost function (based on the local gradient) was negligible in relation to the value of the cost function.

Table 4.7 shows the nominal, perturbed, and calibrated parameter sets for compari-
The simulation results are shown as differences from the perturbed model. For the first simulation the solver was able to reduce the 2-norm of the residual from $0.449$ to $4.34 \times 10^{-6}$. The improvement in accuracy can be seen in Figures 4.33–4.36. The initial positioning errors are shown relative to the nominal grid. After calibration and implementation, the positioning errors are greatly reduced. Rotational parameters are shown in the same manner. The lines represent the axis of rotation, and the length of the line is proportional the angle of rotation about the axis. Note that since the Delta robot has only 3 controllable freedoms, the rotational errors cannot be corrected by adjustment of the actuators. Unlike the Kiwibot, there is virtually no control over the nacelle orientation.

The next simulation used simulated theodolite readings with no noise as input data. The results are shown in column 5 of Table 4.7. The result was further from the perturbed model than for simulation 1, probably because the theodolite readings have only 2 components and thus return less position information to the solver. The solver initially came to a rather poor solution with a residual norm of 0.081, however after trying several random starting points, a residual norm of 0.045 was achieved. Because the solver terminates when the step size becomes smaller than a preset minimum ($1 \times 10^{-10}$), the solution may be sub-optimal, particularly when the solver encounters the multidimensional equivalent of a steep sided valley with a relatively flat “valley floor”. There is little advantage in abandoning the minimum step size criterion because the solver can spend a very long time taking miniscule steps which do not make significant progress towards the true minimum. In such situations a new randomly selected starting point may converge to a superior solution. While issues of observability were not investigated in depth in this study, it is likely that some form of optimisation of the arrangement of measurement positions would increase the solvability of the problem (cf. §1.1.1). The pose errors after implementation are shown in Figures 4.37 and 4.38.

The final simulation was the same as the second simulation but with normally distributed random noise added to the simulated theodolite readings. The covariance of the measurement noise was based on the stated theodolite accuracy of ±20 seconds of arc for angular measurements. Assuming this represents a 90% confidence interval on Gaussian data, then the error variance would be

$$\sigma_m^2 = \left( \frac{20}{3600} \times \frac{\pi}{180} \times \frac{1}{2.71} \right)^2 = 1.28 \times 10^{-9},$$

$$\Sigma_m = I_n \otimes \begin{bmatrix} 1.28 \times 10^{-9} & 0 \\ 0 & 1.28 \times 10^{-9} \end{bmatrix}.$$ (4.6)

The actuator noise is based on the resolution of the step motors (400 steps per revolution)
with a reduction ratio of 80:1. An approximate variance was calculated using

\[
\sigma^2 = \left( \frac{2\pi}{400} \times \frac{1}{80} \times \frac{1}{2} \times \frac{1}{2.71} \right)^2
= 1.31 \times 10^{-9}
\]

so that

\[
\Sigma_\theta = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & 1.31 \times 10^{-9} & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 1.31 \times 10^{-9} & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & 1.31 \times 10^{-9} & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
\end{bmatrix}
\]  \hspace{1cm} (4.7)

The norm of the measurement residual was reduced from 0.449 to 0.089. As expected, the measurement residual is higher as a result of the measurement and actuator noise. Several starting points were tried before a reasonably low residual could be obtained. The identified value for parameter \(d_{1,26}\) was only 0.14 mm different from the perturbed model, while other parameters such as \(d_{1,5}\) were as much as 7.5 mm from the perturbed value. The position and orientation errors after implementation are shown in Figures 4.39 and 4.40. It is obvious from the results of the simulations that the model is more difficult to identify from theodolite measurements. Despite that, reasonable results were obtained by attempting a variety of starting points.
Figure 4.31 Arrangement of the kinematic model for simulation of the Delta robot. The mini-axes represent joint frames, the blue axes represent the axis of rotation for rotary joints, and the numerals represent the joint number. The model is shown in the "zero" position ($\theta_{i\ldots n} = 0$).

Figure 4.32 Kinematic model of the Delta robot with kinematic loops closed.
<table>
<thead>
<tr>
<th>Body</th>
<th>$\alpha_1$</th>
<th>$\alpha_2$</th>
<th>$\alpha_3$</th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>Predecessor</th>
<th>Closure</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>$\circ$ 0</td>
<td>$\circ$ 0</td>
<td>$\circ$ 0</td>
<td>$\circ$ 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2.</td>
<td>$\circ$ -1.5708</td>
<td>$\circ$ 0</td>
<td>$\bullet$  0</td>
<td>$\bullet$  0.2805</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3.</td>
<td>$\circ$ -1.5708</td>
<td>$\circ$ 2.0944</td>
<td>$\bullet$  0.2805</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4.</td>
<td>$\circ$ -1.5708</td>
<td>$\circ$ -2.0944</td>
<td>$\bullet$  0.2805</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5.</td>
<td>$\bullet$  0</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>0</td>
<td>$\bullet$  0.6</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>6.</td>
<td>$\circ$  1.5708</td>
<td>$\circ$ 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bullet$  0.15</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>7.</td>
<td>$\circ$  0</td>
<td>$\bullet$  0</td>
<td>$\bullet$  0.9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>8.</td>
<td>$\circ$  1.5708</td>
<td>$\circ$ 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>9.</td>
<td>$\circ$  0</td>
<td>$\circ$ -1.5708</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>10.</td>
<td>$\bullet$  0</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>$\bullet$  0.3</td>
<td>$\bullet$  0</td>
<td>$\bullet$  0</td>
<td>9</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>11.</td>
<td>$\bullet$  0</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>$\bullet$  0.3947</td>
<td>$\bullet$  -0.1641</td>
<td>$\bullet$  0</td>
<td>9</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>12.</td>
<td>$\bullet$  0</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>$\bullet$  0.2447</td>
<td>$\bullet$  -0.4239</td>
<td>$\bullet$  0</td>
<td>9</td>
<td>23</td>
<td>2</td>
</tr>
<tr>
<td>13.</td>
<td>$\bullet$  0</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>$\bullet$  0.0553</td>
<td>$\bullet$  -0.4239</td>
<td>$\bullet$  0</td>
<td>9</td>
<td>26</td>
<td>2</td>
</tr>
<tr>
<td>14.</td>
<td>$\bullet$  0</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>$\bullet$  -0.0947</td>
<td>$\bullet$  -0.1641</td>
<td>$\bullet$  0</td>
<td>9</td>
<td>29</td>
<td>2</td>
</tr>
<tr>
<td>15.</td>
<td>$\bullet$  0</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>0</td>
<td>$\bullet$  0.6</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>16.</td>
<td>$\circ$  1.5708</td>
<td>$\circ$ 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bullet$  -0.15</td>
<td>0</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>17.</td>
<td>$\circ$  0</td>
<td>$\circ$ 0</td>
<td>0</td>
<td>$\bullet$  0.9</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>18.</td>
<td>$\bullet$  0</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>$\bullet$  0.6</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>19.</td>
<td>$\circ$  1.5708</td>
<td>$\circ$ 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bullet$  0.15</td>
<td>0</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>20.</td>
<td>$\circ$  0</td>
<td>$\circ$ 0</td>
<td>0</td>
<td>$\bullet$  0.9</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>0</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>21.</td>
<td>$\bullet$  0</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>$\bullet$  0.6</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>22.</td>
<td>$\circ$  1.5708</td>
<td>$\circ$ 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bullet$  0.15</td>
<td>0</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>23.</td>
<td>$\circ$  0</td>
<td>$\circ$ 0</td>
<td>0</td>
<td>$\bullet$  0.9</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td>24.</td>
<td>$\bullet$  0</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>$\bullet$  0.6</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>25.</td>
<td>$\circ$  1.5708</td>
<td>$\circ$ 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bullet$  0.15</td>
<td>0</td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>26.</td>
<td>$\circ$  0</td>
<td>$\circ$ 0</td>
<td>0</td>
<td>$\bullet$  0.9</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>0</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>27.</td>
<td>$\bullet$  0</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>$\bullet$  0.6</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>28.</td>
<td>$\circ$  1.5708</td>
<td>$\circ$ 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bullet$  0.15</td>
<td>0</td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td>29.</td>
<td>$\circ$  0</td>
<td>$\circ$ 0</td>
<td>0</td>
<td>$\bullet$  0.9</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>30.</td>
<td>$\bullet$  0</td>
<td>$\bullet$  0</td>
<td>0</td>
<td>$\bullet$  0.15</td>
<td>$\bullet$  -0.196</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.6 Nominal model definition matrix for the Delta robot. Key: $\circ$ indicates free parameters in the model, $\bullet$ indicates artificially constrained parameters, unmarked parameters are automatically constrained.
### Table 4.7 Results of calibrations on the Delta robot

<table>
<thead>
<tr>
<th>Param.</th>
<th>Nominal Model</th>
<th>Perturbed Model</th>
<th>Simulation 1</th>
<th>Simulation 2</th>
<th>Simulation 3</th>
<th>Experimental Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,01</td>
<td>-1.5708</td>
<td>-1.5708</td>
<td>2.0 x 10^-6</td>
<td>-4.4 x 10^-8</td>
<td>-0.0061</td>
<td>0.00094</td>
</tr>
<tr>
<td>3,02</td>
<td>0.0000</td>
<td>0.0000</td>
<td>6.9 x 10^-5</td>
<td>0.00036</td>
<td>0.00037</td>
<td>0.00005</td>
</tr>
<tr>
<td>2,03</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.1 x 10^-7</td>
<td>0.00017</td>
<td>0.00035</td>
<td>0.00044</td>
</tr>
<tr>
<td>2,04</td>
<td>0.0000</td>
<td>0.0000</td>
<td>5.1 x 10^-5</td>
<td>3.1 x 10^-5</td>
<td>3.8 x 10^-5</td>
<td>-0.0011</td>
</tr>
<tr>
<td>3,05</td>
<td>2.9 x 10^-5</td>
<td>2.9 x 10^-5</td>
<td>0.00046</td>
<td>0.00009</td>
<td>0.00011</td>
<td>0.00000</td>
</tr>
<tr>
<td>2,06</td>
<td>-0.1 x 10^-5</td>
<td>-0.1 x 10^-5</td>
<td>1.7 x 10^-6</td>
<td>0.0012</td>
<td>0.00006</td>
<td>0.00002</td>
</tr>
<tr>
<td>3,07</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.1 x 10^-5</td>
<td>0.0017</td>
<td>0.00017</td>
<td>0.00018</td>
</tr>
<tr>
<td>2,08</td>
<td>-0.1 x 10^-5</td>
<td>-0.1 x 10^-5</td>
<td>4.0 x 10^-5</td>
<td>0.00014</td>
<td>0.00017</td>
<td>0.00014</td>
</tr>
<tr>
<td>3,09</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-4.6 x 10^-5</td>
<td>0.0027</td>
<td>0.00027</td>
<td>0.00026</td>
</tr>
<tr>
<td>3,10</td>
<td>0.0000</td>
<td>0.0000</td>
<td>1.6 x 10^-5</td>
<td>1.7 x 10^-5</td>
<td>1.8 x 10^-5</td>
<td>-0.0012</td>
</tr>
<tr>
<td>3,11</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.00000</td>
<td>0.00011</td>
<td>0.00011</td>
<td>0.00001</td>
</tr>
<tr>
<td>3,12</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.00000</td>
<td>0.00012</td>
<td>0.00012</td>
<td>0.00003</td>
</tr>
<tr>
<td>3,13</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.00000</td>
<td>0.00012</td>
<td>0.00012</td>
<td>0.00003</td>
</tr>
<tr>
<td>3,14</td>
<td>0.0000</td>
<td>0.0000</td>
<td>4.0 x 10^-6</td>
<td>0.00014</td>
<td>0.00014</td>
<td>0.00014</td>
</tr>
<tr>
<td>3,15</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.1 x 10^-5</td>
<td>0.00015</td>
<td>0.00015</td>
<td>0.00015</td>
</tr>
<tr>
<td>3,16</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.00000</td>
<td>0.00016</td>
<td>0.00016</td>
<td>0.00016</td>
</tr>
<tr>
<td>3,17</td>
<td>0.0190</td>
<td>0.0190</td>
<td>1.5 x 10^-6</td>
<td>0.0027</td>
<td>0.00037</td>
<td>0.00050</td>
</tr>
<tr>
<td>3,18</td>
<td>0.0190</td>
<td>0.0190</td>
<td>-1.5 x 10^-6</td>
<td>0.0039</td>
<td>0.00079</td>
<td>0.0043</td>
</tr>
<tr>
<td>3,19</td>
<td>0.0190</td>
<td>0.0190</td>
<td>-1.5 x 10^-6</td>
<td>0.0042</td>
<td>0.00084</td>
<td>0.0014</td>
</tr>
</tbody>
</table>

The corresponding to the introduced error d1,26 is also highlighted.
Figure 4.33  Positioning error before calibration on the measurement grid. The worst deviation is 25.7 mm. Deviations are shown magnified by a factor of 5.

Figure 4.34  Positioning error after calibration for simulation 1. The worst deviation is $2.23 \times 10^{-3}$ mm. Deviations are shown magnified by a factor of 5.
4.3 SIMULATION

Figure 4.35 Orientation error before calibration. The lines represent the axis of rotation, the length of the line is proportional to the angle of rotation. The largest angle of rotation is 5.162 deg.

Figure 4.36 Orientation error after calibration for simulation 1. The largest angle of rotation is 5.010 deg.
Figure 4.37  Positioning error after calibration for simulation 2 (cf. Figure 4.33). The worst deviation is 2.987 mm. Deviations are shown magnified by a factor of 5.

Figure 4.38  Orientation error after calibration for simulation 2 (cf. Figure 4.35). The largest angle of rotation is 4.997 deg.
4.3 SIMULATION

Figure 4.39 Positioning error after calibration for simulation 3 (cf. Figure 4.33). The worst deviation is 3.681 mm. Deviations are shown magnified by a factor of 5.

Figure 4.40 Orientation error after calibration for simulation 3 (cf. Figure 4.35). The largest angle of rotation is 4.994 deg.
CHAPTER 4  APPLICATION

4.4 Experimental Calibration

The results of §4.3.1 show that the magnitude of the residual errors after identification is related to the measurement and actuator noise levels and also to the number of measurements taken. Since the number of measurements is limited by practical considerations, there is a threshold value for the magnitude of a geometric error, under which identification is not possible. Parametric errors whose effect on the value of the cost function is of similar magnitude to the effect of noise are, therefore, not identifiable. Because the accuracy of the available measuring device was limited, it was decided to introduce an artificially exaggerated geometric error into the robot.

It was assumed that non-linearities due to the magnitude of the error would not affect the result. A substitute parallel arm was constructed that was 20 mm longer than the nominal length of 900 mm. Although an error of this size is very unlikely for this robot, the error is only 2.2% of the nominal length of the member. Most implementations of the Delta robot are considerably smaller than the Canterbury University Delta robot, thus the introduced error is not disproportionately large.

The fifth parallel arm (corresponding to frame 26 in the model) was replaced with the substitute arm. The identification and implementation processes were carried out on the data and the results are presented below.

4.4.1 Modelling

The model used for the experimental calibration is identical to the model used for the simulations.

4.4.2 Measurement

The measurements were carried out with a single theodolite with a laser range finder built in. The measurement points were arranged in a 3-dimensional cubic grid with sides 400 mm long. The points were spaced at 100 mm giving 125 measurement points in all. The theodolite was positioned such that the arms would not obscure the theodolite reflector which was fastened to the top of the nacelle plate. The stated accuracy of the theodolite was ±20 seconds of arc for angular measurements and ±1 mm for distance measurements. The theodolite reads azimuthal angle and bearing. Azimuthal angle is the angle between the vertical and the sight line of the theodolite. Bearing is the angle in the horizontal plane between a fixed reference point (a cross drawn on the wall in this experiment) and the sight line of the theodolite.

A reference position on the robot was obtained by attaching an accurately made 'tree' to a location hole in the base platform. The tree has 3 location holes for the theodolite reflector forming a triangle. Measurements of the azimuth and bearing at the three positions serve to locate the theodolite relative to a reference point on the
robot. The tree sights were taken several times during the course of the measurements in order to make certain that the robot was not changing position relative to the theodolite. During the first set of measurements it was found that the vibration of the motors was causing the robot to move across the floor. To solve this problem the robot legs were firmly bolted down.

A set of 3 back-sights drawn on the walls of the room served as fixed reference positions. These were used to verify that the theodolite had not moved during the measurement process. The azimuth and bearing of the home position were also checked repeatedly in order to detect any variation. No variation was detected.

Repeatability and drift were tested by re-measuring a selection of points after the measurement sequence was completed. It was found that the temperature of the robot had a detectable effect on the azimuth readings. This is most likely due to thermal expansion in the aluminium support legs. Much of the warming was due to heat transmitted from the motors over the 4 hours it took to take the measurements. The maximum discrepancy between hot and cold measurements was 40 seconds of arc. The theodolite was not sensitive enough to detect variability in the positioning of the robot when the robot was returned repeatedly to the same position from different start points. The smallest detectable variation at a range of 3.7 m is approximately 0.36 mm and the repeatability of the robot is likely to be better than this. With drift taken into account, the accuracy of the measurements is ±40 seconds of arc and ±2 mm for distance. The deviation of the end effector due to the error introduced into the mechanism is an order of magnitude larger thus it should be discernable from the measurements.

4.4.3 Identification

The identification process was carried out on the azimuthal angle and bearing measurements only because the range measurements were somewhat less accurate. The first task was to calculate the position of the theodolite relative to the base frame of the robot from measurements of the theodolite tree. An approximate result was calculated using trigonometry assuming the geometry of the tree to be perfectly accurate. The theodolite tree was accurately measured using a coordinate measuring machine and the approximate result was used as the starting point for a simple nonlinear optimisation which minimised the discrepancy between the measurements and the results of a mathematical model of the theodolite by adjusting the position and orientation of the tree. The theodolite was found to be situated at coordinates

\[ p_{\text{theod}} = \begin{bmatrix} -1.2197 \\ -3.4359 \\ -0.1611 \end{bmatrix} \]  (4.9)

relative to the base frame of the robot.

The identification solver was able to reduce the norm of the residual from \(2.70 \times 10^{-2}\)
to $1.50 \times 10^{-3}$. The resulting parameter set is shown in column 7 of Table 4.7. The residual norm after calibration is roughly twice the magnitude of the residual achieved in simulation 3 for the Delta robot. This indicates that the estimate of the measurement and actuator noise used in simulation 3 may be slightly low but not unrealistic. The effects of drift in the measurements due to thermal effects is the most likely cause of the increased residual.

4.4.4 Implementation

Implementation was performed on the resulting model and the anticipated positioning and orientation accuracy is shown in Figures 4.41–4.42. These figures were generated by assuming the true geometry of the robot to be identical to the perturbed model used in the simulations. It may be that geometric errors in the construction of the real robot exist and introduce their own bias to the measurements, and thus affect the solution. Ideally the measurement process should be repeated after implementation to check the improvement in accuracy, however this was not possible due to time constraints.

The Levenberg-Marquardt method was employed and convergence was usually achieved using approximately 10 function and gradient evaluations. The speed of the computations was unacceptably slow for the purpose of real time implementation in a robot controller because each function evaluation involved the computation of the direct geometric solution which is itself a nonlinear optimisation problem. An approximate approach was tried in which a line search was performed in the steepest descent direction. Only 1 gradient evaluation was required and a small number of direct geometric solutions—typically 3—were needed to find an acceptable minimum using a quadratic interpolation technique. The convergence tolerance on the direct geometric solution was also relaxed in order to decrease the number of iterations required. This reduced the computation time from 20 seconds to 6 seconds and the number of floating point operations was decreased from 17.7 MFlops to 4.5 MFlops. Obviously, the accuracy of the implementation suffered as a result of the simplified minimisation. The worst positional deviation was 13.4 mm after implementation by this method compared with 9.32 mm using the Levenberg-Marquardt method.
4.4 EXPERIMENTAL CALIBRATION

Figure 4.41 Anticipated positioning error after calibration assuming that introduced error on parallel arm 5 is the major source of geometric error (cf. Figure 4.35). The worst deviation is 9.322 mm. Deviations are shown magnified by a factor of 5.

Figure 4.42 Anticipated orientation error after calibration. The largest angle of rotation is 5.056 deg.
Chapter 5

Conclusion

5.1 Summary

This research has developed new and effective methods for geometric modelling of general parallel mechanisms of relatively high complexity, and for identification of geometric parameters of general parallel robots from pose measurements in the task space of the mechanism. Methods for implementing the resulting geometric model in a robot controller have also been investigated.

The geometric model has been applied successfully to a variety of mechanisms including the 3 degree of freedom Kiwibot spherical positioning device and the 3 degree of freedom Delta robot cartesian positioning device. The algorithm for joint closure was found to be reliable even for poor estimates of the starting configuration although the possibility of converging to an unexpected solution in an alternate workspace exists.

Parameter identification for calibration has been simulated for both the Kiwibot and Delta robot and the robustness and statistical properties of the method when applied to the 4 bar chain mechanism have been determined through Monte Carlo simulation. The calibration process was applied successfully to a real Delta robot with an introduced geometric error demonstrating that the method works in practice.

When only a subset of the model parameters of a robot are critical to its positioning accuracy, the calibration method provides for fixing the value of any non-critical parameter, thus removing it from the optimisation. Calibration on a reduced parameter set has been demonstrated by other authors to decrease the computational effort involved while still achieving significant accuracy enhancement (cf. §1.1.2).

The major difficulty with implementation is the speed of computation. Under the MATLAB environment it takes approximately 10 seconds to perform a direct geometric solution using the routine connect.m on the Delta robot model. The code was written for the purpose of investigating the workability of the method and while there is considerable scope for simplification and improvement, it is unlikely that the required speed can be obtained using MATLAB. By re-coding in a compiled language such as C, Fortran, or Pascal, the execution time may be decreased by a factor of about 30, but even
this is too slow for accurate control. Implementation on a floating point Digital Signal Processor (DSP) may decrease the computation time to the point where the method is practical for implementation. For the Kiwibot, a simplified direct geometric solution for a general geometry exists which involves a 3 degree of freedom nonlinear optimisation on the distances between the ball joint centres. This method is considerably faster to implement and may be applied for this robot.

5.2 Original Contributions

The major original contributions to knowledge in this field are:

- Development of a methodology for geometrically modelling parallel mechanisms that is capable of representing any parallel structure composed of rigid bodies and rotary, prismatic, or spherical joints. Unique features of the method are:
  - Modelling of closed loop mechanisms as open loop tree structures using the MCPC modelling convention.
  - Use of the Levenberg-Marquardt method to achieve loop closure.
  - Use of Euler parameters to represent spatial rotations.
  - Evaluation of a closed loop derivative matrix by application of constraints on the relative rate of change of closure joints in task space.

- Application of the geometric model to calibration enabling a systematic approach to calibration of general parallel mechanisms without the need to adapt the method for the particular mechanism under investigation. No other method has achieved this to date.

- Experimental verification of the calibration procedure on a real robot.

Other contributions are:

- A new method of analysing the degree of constraint and geometrical conditioning of closed loop mechanisms (cf. §2.5.2).

- A novel nominal geometric solution for the Delta robot (cf. §2.6.1).

- Exploration of practical problems in the solution of the identification and implementation problem (cf. §4.3–4.4).
5.3 Future Work

During the course of the project several problems that require further investigation or development were encountered. Among these were the following:

- The geometric modelling software is currently only capable of modelling mechanisms consisting of rotary and spherical joints. The theory for modelling of prismatic joints has been presented in this thesis but not implemented in the software. It would be worthwhile rewriting the software so that a variety of joint types can be modelled including 'higher pair' joints with complex joint parameter to joint displacement relationships.

- There is scope for refinement of most of the software. The code is designed to demonstrate that the methods will work without much consideration of efficiency and execution time. Coding in a compiled language such as C, Fortran, or Pascal would greatly decrease the solution time.

- The calibration problem is framed as a direct calibration problem (§1.1.2). There may be advantages in using the implicit calibration approach with a general geometric model for parallel mechanisms as opposed to the mechanism specific models used in the literature.

- Only 3 mechanisms have been investigated in this research. Many more mechanisms, notably the Stewart Platform, should be tried in order to test the applicability of the method.

- The model could be altered to account for first mode elastic deflections in members given the payload mass. Although the advantage in terms of accuracy would be insignificant for the robots studied in this project, such a model would be useful for calibration of lightweight robots with flexible members. The elastic properties of the mechanism's members could conceivably be estimated by the identification algorithm.
Appendix A

Relevant Mathematics

A.1 Euler Angles and Euler Parameters

A.1.1 Euler Angles

Any rotation of a body can be expressed as the result of 3 principal rotations. A principal rotation is a rotation about one of the 3 axes of a coordinate frame. Let the coordinate transformation matrices corresponding to the 1 2 and 3 axes respectively be written $C_1$, $C_2$, and $C_3$. Abbreviating $\sin(\alpha_i)$ as $s_i$ and $\cos(\alpha_i)$ as $c_i$, the principal rotation matrices are

$$x = -b \pm \frac{\sqrt{b^2 - 4ac}}{2a}$$  \hspace{1cm} (A.1)

$$C_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_1 & s_1 \\ 0 & -s_1 & c_1 \end{bmatrix}$$  \hspace{1cm} (A.2)

$$C_2 = \begin{bmatrix} c_2 & 0 & -s_2 \\ 0 & 1 & 0 \\ s_2 & 0 & c_2 \end{bmatrix}$$  \hspace{1cm} (A.3)

$$C_3 = \begin{bmatrix} c_3 & s_3 & 0 \\ -s_3 & c_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$  \hspace{1cm} (A.4)

The transformation matrix

$$R = C_1(\alpha_1)C_2(\alpha_2)C_3(\alpha_3)$$  \hspace{1cm} (A.5)

$$= \begin{bmatrix} c_2c_3 & c_2s_3 & -s_2 \\ s_1s_2c_3 - c_1s_3 & s_1s_2s_3 + c_1c_3 & s_1c_2 \\ c_1s_2c_3 + s_1s_3 & c_1s_2s_3 - s_1c_3 & c_1c_2 \end{bmatrix}$$  \hspace{1cm} (A.6)

represents a rotation about the 3 2 and 1 axes in sequence.
To find the Euler angles for a given rotation matrix $\mathbf{R}$, the following procedure is used: First, the particular sequence of rotations must be specified. For any Euler sequence, there will always be a single element of $\mathbf{R}$ that is a function of just one of the Euler angles. This element can be used to find one Euler angle. Next, there are 4 elements that are functions of the previously mentioned angle and one other angle. These can be used to find the second Euler angle. From there any element can be used to find the third Euler angle. For example, if the Euler sequence is 321 as in equation A.6 then the element in the first row and third column is $-\sin(\alpha_2)$ so

$$\alpha_2 = \arcsin(-r_{13}). \quad (A.7)$$

The other elements in the first row are functions of $\alpha_2$ and $\alpha_3$ so

$$\tan(\alpha_3) = \frac{r_{12}}{r_{11}} \quad (A.8)$$

$$\alpha_3 = \arctan \left( \frac{r_{12}}{r_{11}} \right). \quad (A.9)$$

The remaining Euler angle $\alpha_1$ may be calculated from any term but the simplest is to use

$$\tan(\alpha_1) = \frac{r_{23}}{r_{33}} \quad (A.10)$$

$$\alpha_1 = \arctan \left( \frac{r_{23}}{r_{33}} \right). \quad (A.11)$$

Note that if $\cos(\alpha_2) = 0$ then equations A.9 and A.11 cannot be used. This means that the Euler angles cannot be found from the rotation matrix. This is referred to as the Euler singularity and every Euler sequence has such a singularity.

### A.1.2 Euler Parameters

Any sequence of principal rotations can be represented as a single rotation of angle $\phi$ about an axis (described by unit vector $\mathbf{\hat{a}}$). Such a representation is always possible and non-singular in the forward and inverse direction. The Euler parameter representation is based on this idea. The Euler parameters

$$\Phi = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix} = \begin{bmatrix} \theta \sin \left( \frac{\phi}{2} \right) \\ \cos \left( \frac{\phi}{2} \right) \end{bmatrix}. \quad (A.12)$$

Note that

$$\Phi^T \Phi = 1. \quad (A.13)$$
Equation A.13 provides a constraint equation reducing the number of degrees of freedom in an Euler parameter set to 3.

The rotation matrix corresponding to a set of Euler parameters is

$$C = \left( \eta^2 - \varepsilon^T \varepsilon \right) I + 2 \varepsilon \varepsilon^T - 2 \eta \varepsilon \times.$$  \hspace{1cm} (A.14)

The Euler parameters corresponding to a rotation matrix $C$ are calculated as follows:

First, calculate $\eta$ using

$$\eta^2 = \frac{1}{4} \left( 1 + c_{11} + c_{22} + c_{33} \right).$$  \hspace{1cm} (A.15)

The sign chosen for $\eta$ is arbitrary and restricting $\eta$ to be positive makes a unique set of Euler parameters for a given rotation. If $\eta \neq 0$,

$$\varepsilon = \frac{1}{4\eta} \begin{bmatrix} c_{23} - c_{32} \\ c_{31} - c_{13} \\ c_{12} - c_{21} \end{bmatrix}$$  \hspace{1cm} (A.16)

where $c_{pq}$ is the element of $C$ on the $p$'th row and the $q$'th column. Situations where $\eta = 0$ correspond to rotations of $\pi, 3\pi, 5\pi, \ldots$, etc.

In practise, equations A.15 and A.16 are accurate as long as $\eta$ is greater than some critical value $\eta_c$. This is because $\varepsilon$ is calculated from the asymmetry of $C$ which tends to lose numerical significance for small $\eta$. For $\eta < \eta_c$ a more stable (but slower) method is used which relies on the property

$$C \varepsilon = \varepsilon,$$  \hspace{1cm} (A.17)

or

$$\left( C^T + C \right) \varepsilon = 2 \varepsilon$$  \hspace{1cm} (A.18)

which is faster to compute since $C^T + C$ has real eigenvalues. In general there will be 1 positive eigenvalue (except when $C$ is identically $I$). Its eigenvector will be a close estimate of $\hat{\varepsilon}$ and

$$\varepsilon = \sqrt{1 - \eta^2} \hat{\varepsilon}.$$  \hspace{1cm} (A.19)

When calculated by this method, $\varepsilon$ is accurate for all $\eta$. The sign of $\varepsilon$ may be lost during the eigenvalue computation. This may be recovered from the initial estimate of $\varepsilon$ given by equation A.16.

When $\eta < \eta_c$ but non-zero, a refined calculation of $\eta$ is used. If $\eta_0$ is the first estimate of $\eta$ from equation A.15 and $\varepsilon_0$ is obtained from equation A.16

$$\eta = \frac{|\varepsilon_0|}{4 \sqrt{1 - \eta_0^2}}.$$  \hspace{1cm} (A.20)
To show this, let
\[ u = \begin{bmatrix} c_{23} - c_{32} \\ c_{31} - c_{13} \\ c_{12} - c_{21} \end{bmatrix} \] (A.21)
and substituting into equation A.16
\[ \varepsilon = \frac{1}{4\eta} u. \] (A.22)
Taking the 2-norm of both sides
\[ |\varepsilon| = \frac{|u|}{4\eta} \] (A.23)
giving
\[ 4\eta \sqrt{1 - \eta^2} = |u| \] (A.24)
which may be solved for \( \eta \) by squaring both sides and solving the resulting quadratic in \( \eta^2 \). The alternative is to rearrange and solve equation A.20. In comparison, the numerical results of both methods are near identical and equation A.20 is faster to compute.

The critical value \( \eta_c \) depends on the floating point precision used. Figure A.1 was obtained by converting a set of Euler parameters \( \Phi_1 \) to a rotation matrix using equation A.14, then peforming the reverse conversion and storing the difference between the resulting set of Euler parameters \( \Phi_2 \) and the original. The computation was performed using standard double precision floating point arithmetic on a Pentium™ processor in the MATLAB environment. It can be seen that errors are negligible for \( \eta > 0.1 \) using equations A.15–A.16 but the eigenvector method performs far better for \( \eta < 0.1 \). At values \( \eta < 1 \times 10^{-8} \), \( \eta_5^2 \) is smaller than the floating point accuracy of the machine causing equation A.15 to lose significance.

If the coordinate frame rotates about an axis described by the vector \( \dot{\Omega} \) (§A.2) then the rate of change of the Euler parameters is given by
\[
\begin{bmatrix}
\frac{d\varepsilon}{ds} \\
\frac{d\eta}{ds} \\
-\frac{ds}{ds}
\end{bmatrix} = Z_{Eul}^{-1} \Omega
\] (A.25)
where
\[
Z_{Eul}^{-1} = \begin{bmatrix}
\frac{1}{2} (\varepsilon^X + \eta I_3) \\
-\frac{1}{2} \varepsilon^T
\end{bmatrix}.
\] (A.26)
If the Euler parameters change with respect to a parameter \( s \) then the angular rate of change vector \( \Omega \) is expressed as
\[
\Omega = 2 \left( \frac{\eta^2 I_3 - \eta \varepsilon^X + \varepsilon \varepsilon^T}{\eta} \right) \frac{d\varepsilon}{ds}
\]
A.2 Derivative of a Rotation Matrix

Key Result

The derivative of a rotation matrix $C_{0,i}$ with respect to an arbitrary parameter $s$ is

$$
\frac{dC_{0,i}}{ds} = \left( \sum_{j=1}^{i-1} C_{0,j-1,j} + C_{0,i-1,i} \right) \times C_{0,i}
$$

(A.28)

where

$$
\Omega_{j-1,i} = \begin{bmatrix} 1 & 0 & 0 \\ (C_i^T \hat{2}) & (C_i^T C_i^T \hat{3}) \end{bmatrix} \frac{da_i}{ds}
$$

$$
\Omega_{i-1,i} = \begin{bmatrix} 1 & 0 & 0 \\ (C_i^T \hat{2}) & 0_{3 \times 1} \end{bmatrix} \frac{da_i}{ds}
$$

$$
\frac{da_k}{ds} = \begin{bmatrix} \frac{d\alpha_{1k}}{ds} \\ \frac{d\alpha_{2k}}{ds} \\ \frac{d\alpha_{3k}}{ds} \end{bmatrix}^T
$$

(A.29)

and $\alpha_{1k}$, $\alpha_{2k}$, $\alpha_{3k}$ are the MCPC rotational parameters for some frame $k$.

The partial derivative of a principal rotation matrix with respect to its Euler angles
is a linear combination of its own elements. For example,

\[
\begin{align*}
C_i^T(\alpha_{1i}) &= \begin{bmatrix} 1 & 0 & 0 \\
0 & \cos(\alpha_{1i}) & -\sin(\alpha_{1i}) \\
0 & \sin(\alpha_{1i}) & \cos(\alpha_{1i}) \end{bmatrix}
\end{align*}
\]  
(A.30)

\[
\begin{align*}
\frac{\partial C_i^T(\alpha_{1i})}{\partial \alpha_{1i}} &= \begin{bmatrix} 0 & 0 & 0 \\
0 & -\sin(\alpha_{1i}) & -\cos(\alpha_{1i}) \\
0 & \cos(\alpha_{1i}) & -\sin(\alpha_{1i}) \end{bmatrix}
\end{align*}
\]  
(A.31)

Similarly,

\[
\begin{align*}
\frac{\partial C_i^T(\alpha_{2i})}{\partial \alpha_{2i}} &= 2^\times C_i^T
\end{align*}
\]  
(A.32)

\[
\begin{align*}
\frac{\partial C_i^T(\alpha_{3i})}{\partial \alpha_{3i}} &= 3^\times C_i^T.
\end{align*}
\]  
(A.33)

The rotation matrix \(C_{0,i}\) is defined in section 2.2 as

\[
C_{0,i} = \prod_{j=1}^{i-1} \left( V_j^T Q_j^T \right) V_i^T \quad \text{for } i > 1
\]  
(A.34)

where

\[
V_j = C_2(\alpha_{2j}) C_1(\alpha_{1j})
\]  
(A.35)

\[
Q_j = C_3(\alpha_{3j}).
\]  
(A.36)

Differentiating \(C_{0,i}\) with respect to an arbitrary parameter \(s\),

\[
\frac{dC_{0,i}}{ds} = \frac{d}{ds} \left( \prod_{j=1}^{i-1} \left( V_j^T Q_j^T \right) V_i^T + \prod_{j=1}^{i-1} \left( V_j^T Q_j^T \right) \frac{dV_i^T}{ds} \right)
\]

\[
= \sum_{j=1}^{i-1} \left( C_{0,j-1} \frac{dC_{j-1,i}}{ds} C_{j,i-1} V_i^T \right) + \left( \prod_{j=1}^{i-1} C_{j-1,i} \right) \frac{dV_i^T}{ds}
\]  
(A.37)

where

\[
C_{j-1,i} = V_j^T Q_j^T.
\]  
(A.39)
The derivative
\[
\frac{dC_{i-1,i}}{ds} = \frac{d}{ds} \left( C_1^T(\alpha_{1j})C_2^T(\alpha_{2j})C_3^T(\alpha_{3j}) \right)
\]
\[
= \frac{\partial C_{i-1,i}^T}{\partial \alpha_{1i}} d\alpha_{1i} \frac{ds}{d\alpha_{2i}} + \frac{\partial C_{i-1,i}^T}{\partial \alpha_{2i}} d\alpha_{2i} + \frac{\partial C_{i-1,i}^T}{\partial \alpha_{3i}} d\alpha_{3i} \frac{ds}{d\alpha_{3i}}
\]
\[
= i^\times C_1^T C_2^T C_3^T \frac{d\alpha_{1i}}{ds} + C_1^T \hat{2}^\times C_2^T C_3^T \frac{d\alpha_{2i}}{ds} + C_1^T C_2^T \hat{3}^\times C_3^T \frac{d\alpha_{3i}}{ds}
\]
\[
= \left( \frac{d\alpha_{1i}}{ds} i^\times + \frac{d\alpha_{2i}}{ds} (C_1^T \hat{2})^\times + \frac{d\alpha_{3i}}{ds} (C_1^T C_2^T \hat{3})^\times \right) C_{i-1,i}
\]
\[= \Omega_{i-1,i}^T C_{i-1,i}. \quad (A.40)\]

It is possible to treat \(\Omega_{i-1,i}\) as if it were a vector \(\tilde{\Omega}_{i-1,i}\) expressed in the \((i-1)\)th frame representing the rate of change of frame \(i\) relative to frame \(i-1\) due to the rate of change of \(\alpha_{1i}, \alpha_{2i}\) and \(\alpha_{3i}\). The quantity is analogous to angular velocity in velocity kinematics but in this context the term angular rate of change is more appropriate. In this notation, the superscript before the symbol represents the frame in which \(\Omega_{i-1,i}\) is expressed and the subscript indicates the frames involved. From equation A.40

\[
\Omega_{i-1,i} = \left[ \frac{d\alpha_{1i}}{ds} C_1^T \hat{2}^\times C_2^T \hat{3}^\times \right] \frac{d\alpha_{1i}}{ds}
\]
\[= Z (\alpha_{1i}, \alpha_{2i}) \frac{d\alpha_{1i}}{ds} \quad (A.41)\]

where
\[a_i = [\alpha_{1i} \alpha_{2i} \alpha_{3i}]^T. \quad (A.42)\]

Similarly to equations A.40–A.41, the derivative

\[
\frac{dV_i}{ds} = \frac{d}{ds} \left( C_1^T(\alpha_{1j})C_2^T(\alpha_{2j}) \right)
\]
\[
= \left( \frac{d\alpha_{1i}}{ds} i^\times + \frac{d\alpha_{2i}}{ds} (C_1^T \hat{2})^\times \right) V_i^T
\]
\[= \tilde{\Omega}_{i-1,i}^T V_i^T \quad (A.43)\]

where
\[\tilde{\Omega}_{i-1,i} = Z (\alpha_{1i}, \alpha_{2i}) T_z \frac{d\alpha_{1i}}{ds} \quad (A.44)\]

and
\[T_z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \quad (A.45)\]
From equations A.40 and A.43,

\[
\frac{dC_{0,i}}{ds} = \sum_{j=1}^{i-1} (C_{0,j-1} \bar{\Omega}_{j-1,j}^i C_{j-1,i-1} V_i^T) + C_{0,i-1} \bar{\Omega}_{i-1,i}^i V_i^T
\]

\[
= \sum_{j=1}^{i-1} (C_{0,j-1} \bar{\Omega}_{j-1,j}^i)^\times C_{0,i} + (C_{0,i-1} \bar{\Omega}_{i-1,i}^i)^\times C_{0,i}
\]

\[
= \left( \sum_{j=1}^{i-1} C_{0,j-1} \bar{\Omega}_{j-1,j}^i + C_{0,i-1} \bar{\Omega}_{i-1,i}^i \right)^\times C_{0,i} . \quad (A.46)
\]

### A.3 Singular Value Decomposition

It can be shown [Golub and Loan, 1989] that any matrix \( A \in \mathbb{R}^{m \times n} \) where \( m > n \) can be decomposed such that

\[
A = U S V^T \quad (A.47)
\]

where \( U \) is an \( m \times m \) orthonormal matrix and \( V \) is an \( n \times n \) orthonormal matrix. The matrix \( S \) has the following structure:

\[
S = \begin{bmatrix}
\sigma_1 & 0 & \cdots & \cdots & 0 \\
0 & \sigma_2 & & & \\
& & \ddots & & \\
& & & \sigma_r & \\
& & & & 0 \\
0 & \cdots & \cdots & \cdots & 0
\end{bmatrix} \quad (A.48)
\]

where \( \sigma_i \) is referred to as the \( i^{th} \) singular value of \( A \). If matrix \( A \) has rank \( r \) then there will be \( r \) nonzero singular values. The first \( r \) columns of \( V \) form an orthonormal basis for the range of \( A \) and the remaining \( n - r \) columns form an orthonormal basis for the nullspace of \( A \).

### A.4 QR Decomposition

Any matrix \( A \in \mathbb{R}^{m \times n} \) can be decomposed such that

\[
A = Q R E^T \quad (A.49)
\]
where \( Q \) is an \( m \times m \) orthonormal matrix and \( E \) is an \( n \times n \) permutation matrix. The \( R \) matrix has the following structure:

\[
R = \begin{bmatrix}
  r_{11} & r_{12} & \cdots & r_{1j} & \cdots & \cdots & r_{1n} \\
  0 & r_{22} & \cdots & r_{2j} & \cdots & \cdots \\
  \vdots & \vdots & \ddots & \vdots & \ddots & \ddots \\
  \vdots & \vdots & & 0 & \cdots & \cdots \\
  \vdots & \vdots & & \vdots & \ddots & \ddots \\
  \vdots & \vdots & & \vdots & \vdots & 0 \\
  0 & \cdots & \cdots & \cdots & \cdots & \cdots \\
\end{bmatrix}
\]  \hspace{1cm} (A.50)

The rank of \( R \) is \( j \) so there will be \( j \) non-zero elements along the main diagonal. The permutation matrix \( E \) is used to rearrange the columns of \( R \) so that the main diagonal elements are in descending order. The first \( j \) columns of \( Q \) form an orthonormal basis for the range of \( A \) and the remaining \( m - j \) columns of \( Q \) form an orthonormal basis for the nullspace of \( A \).

A.5 A Similarity Metric for Symmetric Positive Semidefinite Matrices

The Frobenius norm is defined as

\[
\|A\|_F = \left( \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}^2 \right)^{\frac{1}{2}}
\]  \hspace{1cm} (A.51)

and the difference between 2 matrices \( A \) and \( B \) is quantified by

\[
e(A; B) = \min \| A - Q^TBQ \|_F
\]  \hspace{1cm} (A.52)

where \( Q \) is some orthonormal matrix. Taking spectral decompositions of \( A \) and \( B \) such that

\[
A = V_A D_A V_A^T \\
B = V_B D_B V_B^T
\]  \hspace{1cm} (A.53)

where \( D_A \) and \( D_B \) are diagonal eigenvalue matrices and the columns of \( V_A \) and \( V_B \) are the corresponding eigenvectors. Equation (A.52) is minimised when

\[
Q = V_B V_A^T
\]  \hspace{1cm} (A.54)
assuming that the columns of $V_A$ and $V_B$ are similarly sorted.

The Frobenius norm is invariant under orthonormal transformations therefore

$$e(A; B) = \left\| V_A^T A V_A - V_A^T Q^T B Q V_A \right\|_F$$

$$= \left\| D_A - V_B^T B V_B \right\|_F$$

$$= \left\| D_A - D_B \right\|_F$$

$$= \left( \sum_{i=1}^{n} \left( \lambda_i^A - \lambda_i^B \right)^2 \right)^{\frac{1}{2}}. \quad (A.55)$$

The metric is zero when $A = B$ and positive otherwise. Normalising by the largest eigenvalue makes the metric invariant with the scale of the matrices so that $\bar{e}(A; B) = \bar{e}(kA; kB)$. The normalised metric is

$$\bar{e}(A; B) = \left( \sum_{i=1}^{n} \left( \frac{\lambda_i^A - \lambda_i^B}{\lambda_M} \right)^2 \right)^{\frac{1}{2}} \quad (A.56)$$

where

$$\lambda_M = \max \left( \lambda_1^A, \lambda_1^B \right). \quad (A.57)$$

In Table A.1, typical values of the normalised metric are given for a range of matrices. These were obtained experimentally by adding a scaled bias matrix $E$ to a test matrix $A = I_n$. The bias matrix is generated randomly and is at least positive semi-definite with norm $\|E\| = 1$. The difference metric varies with the value of the scaling factor $k$ and the dimension of the matrix $n$.

<table>
<thead>
<tr>
<th>Scale factor $k$</th>
<th>1</th>
<th>0.5</th>
<th>0.1</th>
<th>0.05</th>
<th>0.01</th>
<th>0.0001</th>
<th>$1 \times 10^{-6}$</th>
<th>$1 \times 10^{-8}$</th>
<th>$1 \times 10^{-10}$</th>
<th>$1 \times 10^{-12}$</th>
<th>$1 \times 10^{-14}$</th>
<th>$1 \times 10^{-16}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>0.52</td>
<td>0.1</td>
<td>0.05</td>
<td>0.01</td>
<td>0.0001</td>
<td>$4.9 \times 10^{-6}$</td>
<td>$4.9 \times 10^{-9}$</td>
<td>$4.9 \times 10^{-11}$</td>
<td>$4.9 \times 10^{-13}$</td>
<td>$4.9 \times 10^{-15}$</td>
<td>$4.9 \times 10^{-17}$</td>
</tr>
<tr>
<td>5</td>
<td>0.49</td>
<td>0.24</td>
<td>0.094</td>
<td>0.093</td>
<td>0.0049</td>
<td>0.0001</td>
<td>$2.9 \times 10^{-5}$</td>
<td>$2.9 \times 10^{-7}$</td>
<td>$2.9 \times 10^{-9}$</td>
<td>$2.9 \times 10^{-10}$</td>
<td>$2.9 \times 10^{-12}$</td>
<td>$2.9 \times 10^{-14}$</td>
</tr>
<tr>
<td>10</td>
<td>0.29</td>
<td>0.15</td>
<td>0.094</td>
<td>0.093</td>
<td>0.0049</td>
<td>0.0001</td>
<td>$1.9 \times 10^{-5}$</td>
<td>$1.9 \times 10^{-7}$</td>
<td>$1.9 \times 10^{-9}$</td>
<td>$1.9 \times 10^{-10}$</td>
<td>$1.9 \times 10^{-12}$</td>
<td>$1.9 \times 10^{-14}$</td>
</tr>
<tr>
<td>20</td>
<td>0.19</td>
<td>0.12</td>
<td>0.019</td>
<td>0.012</td>
<td>0.0049</td>
<td>0.0001</td>
<td>$1.2 \times 10^{-5}$</td>
<td>$1.2 \times 10^{-7}$</td>
<td>$1.2 \times 10^{-9}$</td>
<td>$1.2 \times 10^{-10}$</td>
<td>$1.2 \times 10^{-12}$</td>
<td>$1.2 \times 10^{-14}$</td>
</tr>
<tr>
<td>40</td>
<td>0.12</td>
<td>0.062</td>
<td>0.012</td>
<td>0.0062</td>
<td>0.0049</td>
<td>0.0001</td>
<td>$8.4 \times 10^{-6}$</td>
<td>$8.4 \times 10^{-8}$</td>
<td>$8.4 \times 10^{-10}$</td>
<td>$8.4 \times 10^{-12}$</td>
<td>$8.4 \times 10^{-14}$</td>
<td>$8.4 \times 10^{-16}$</td>
</tr>
<tr>
<td>80</td>
<td>0.084</td>
<td>0.042</td>
<td>0.0084</td>
<td>0.0042</td>
<td>0.0049</td>
<td>0.0001</td>
<td>$5.8 \times 10^{-6}$</td>
<td>$5.8 \times 10^{-8}$</td>
<td>$5.8 \times 10^{-10}$</td>
<td>$5.8 \times 10^{-12}$</td>
<td>$5.8 \times 10^{-14}$</td>
<td>$5.8 \times 10^{-16}$</td>
</tr>
<tr>
<td>160</td>
<td>0.058</td>
<td>0.029</td>
<td>0.0058</td>
<td>0.0042</td>
<td>0.0049</td>
<td>0.0001</td>
<td>$1 \times 10^{-6}$</td>
<td>$1 \times 10^{-8}$</td>
<td>$1 \times 10^{-10}$</td>
<td>$1 \times 10^{-12}$</td>
<td>$1 \times 10^{-14}$</td>
<td>$1 \times 10^{-16}$</td>
</tr>
</tbody>
</table>

Table A.1 Typical values of normalised difference metric $\bar{e}(A; A + kE)$. 
Appendix B

Matlab Software Suite for Calibration of Robots

This appendix contains a directory to the code developed for the Modelling, Visualisation, and Calibration package. An example of how to apply the routines is given in §B.2. The code may be found on the accompanying compact disc and the entire set of functions is printed in Lintott [2000]. The design of the Modelling, Visualisation, and Calibration package is discussed in §4.2.1. The code used to control the Parker Hannifin Digiplan IFX indexer (§4.2.3) may be found in Lintott and Jones [2000] or Jones [1997, Appendix F].

B.1 Directory of Files

1. The mcpc toolbox

   bfgs_idc_fn.m Function evaluator for bfgs.m (and variants). Calls cost function bfgsfundef.m and gradient function bfgsfundj.m.

   bfgsfundef.m Returns the calibration cost function.

   bfgsfundj.m Returns the calibration gradient (Jacobian, actually).

   bfgsimplement.m Wrapper function for implementation cost function implementfn.m and gradient function implementja.m.

   calbglob.m Sets up global variables for calib.m.

   calbglob2.m Sets up more global variables for calib.m.

   calib.m Performs calibration process. Wrapper for a variety of solvers.

   calibfn.m Evaluates the pose error at a single measurement.

   calibja.m Evaluates a section of the identification Jacobian corresponding to a single measurement.

   cljack.m Evaluates the closed loop Jacobian of a mechanism by a fast method.

   cljack_verbose.m Evaluates the closed loop Jacobian by the SVD method. Outputs information on the condition of the mechanism.

   connect.m Closes the loops of a tree mechanism. Uses 1m_min.m to minimise distances between closure joints.
connectfj.m Returns a cost function and gradient based on the distance between closure joints.

constrnt.m Works out which parameters to constrain in a geometric model for calibration and also integrates additional user-supplied constraints.

conv.eul.m Converts a rotation matrix into a set of Euler parameters using the method of §A.1.

dirjack.m Calculates the direct Jacobian of a tree mechanism with respect to all model parameters.

dirjack0.m Calculates the direct Jacobian of a tree mechanism with respect to the actuator parameters.

extchain.m Finds all links in the chain of bodies between the base frame of a mechanism and the specified frame.

implement.m Performs the implementation process.

implementfn.m Cost function for the implementation process.

implementja.m Gradient function for the implementation process.

gennoisyposes.m Generates a set of simulated full pose measurements given a list of measurement points in actuator space. Used to generate measurements for simulations. Adds actuator noise to the measurements.

genposes.m Generates a set of simulated full pose measurements given a list of measurement points in actuator space. Used to generate measurements for simulations.

mcpc2dim.m Converts a geometric model specified in mcpc format to dim format.

meas1theod.m Simulates a single theodolite measurement device.

measall.m Simulates a full pose measurement device.

measeuler.m Simulates an orientation measurement device.

measlaser.m Simulates a laser range finding measurement device.

2. The modelfit toolbox

bfgs.m Performs BFGS minimisation. Written by I. D. Coope.

bfgs_idc.m Performs BFGS minimisation. Slight modification of bfgs.m with extra termination conditions.

bfgs_show.m Performs BFGS minimisation. Modification of bfgs.m which graphically displays the topology encountered during the line search.

lm_min.m Performs Levenberg-Marquardt minimisation.
3. The rbk Toolbox

- clr.m Deletes all graphics objects generated by drawgeom.m.
- dirgeom.m Performs the direct geometric solution for a tree mechanism.
- drawgeom.m Draws the robot in a figure window.
- extrgeom.m Extracts information from the dim structure.

4. Miscellaneous

- crossmat.m Converts a 3 by 1 matrix $r$ into its cross matrix $r^\times$.
- msg.m A streamed an prioritised message display system allowing user control of the level of detail in messages from the software.
- @console An object for outputting streamed messages to the MATLAB command window.
- @stream A stream object. Messages written to a stream are output to an output object (such as a console).

B.2 Usage Examples

In the following sections, commands entered at the MATLAB command prompt are preceded with the `>` symbol. The examples are intended to demonstrate proper usage of the Modelling, Visualisation, and Calibration toolbox.

Installation

Copy the directory structure on the compact disc to a suitable local directory and set the MATLAB path to include the 3 top level directories mpc, modelfit, and rbk. Paths in MATLAB are set with the standard addpath.m command.

Rigid body Kinematics and Visualisation

First define a simple mechanism by creating a def matrix. A suitable matrix is stored in the file mdelta_1.mat.

```
load m4bar_1
```
APPENDIX B  MATLAB SOFTWARE SUITE FOR CALIBRATION OF ROBOTS

```matlab
>> def4

def4 =
Columns 1 through 7
0 0 0 0 0 0 0
0 0 0 1.0000 0 1.0000
0 0 1.0000 0 0 0
-1.5708 0 0 0 0 3.0000
1.5708 0 0 1.0000 0 4.0000
-1.5708 0 -1.0000 0 0 5.0000
Columns 8 through 9
0 0
0 0
0 0
0 0
0 0
2.0000 2.0000
```

The `def` matrix is a shorthand form of the more general `dim` format. To convert `def` to `dim`,

```matlab
>> dim=mcpc2dim(def4)

dim =
1.0000 0 0 0 0 0 0 0
0 1.0000 0 0 0 0 0 0
0 0 1.0000 0 0 0 0 0
1.0000 0 0 0 0 0 0 0
0 1.0000 0 0 0 0 0 0
0 0 1.0000 0 0 0 0 0
1.0000 0 0 0 1.0000 0 0 0
0 1.0000 0 1.0000 0 0 0 0
0 0 1.0000 0 0 0 0 0
1.0000 0 0 1.0000 0 0 0 0
0 1.0000 0 0 0 0 0 0
0 0 1.0000 0 0 0 0 0
1.0000 0 0 0 3.0000 0 0 0
0 0.0000 1.0000 0 0 0 0 0
0 -1.0000 0.0000 0 0 0 0 0
1.0000 0 0 0 4.0000 0 0 0
0 0.0000 -1.0000 0.0000 0 0 0 0
0 1.0000 0.0000 1.0000 0 0 0 0
1.0000 0 0 -1.0000 5.0000 0 0 0
0 0.0000 1.0000 0 2.0000 0 0 0
0 -1.0000 0.0000 0 2.0000 0 0 0
```

Next, define a configuration for the robot by setting the 6 actuator coordinates to arbitrary values and perform the direct geometric solution. The resulting configuration can
be viewed using `drawgeom.m`.

```matlab
» 0 = [0.1 0.3 -0.1 0.2 0.6 0.7]';
» cfg = dirgeom(0,dim)
```

```matlab
cfg =
  1.0000   0   0   0   0   0
   0  1.0000   0   0   0   0
   0   0  1.0000   0   0   0
  0.9950 -0.0998   0   0   0   0
  0.0998  0.9950   0   0   0   0
   0   0  1.0000   0   0   0
  0.9211 -0.3894   0 -0.0998  1.0000   0
  0.3894  0.9211   0  0.9950   0   0
   0   0  1.0000   0   0   0
  0.9950  0.0998   0  1.0000   0   0
 -0.0998  0.9950   0   0   0   0
   0   0  1.0000   0   0   0
  0.9752 -0.1977  0.0998  1.0000  3.0000   0
 -0.0978  0.0198  0.9950   0   0   0
 -0.1987 -0.9801  0.0000   0   0   0
  0.9037 -0.3799  0.1977  1.0998  4.0000   0
  0.3912  0.9201 -0.0198  0.9950   0   0
 -0.1743  0.0962  0.9801  0.0000   0   0
  0.5638 -0.7333 -0.3799  0.1962  5.0000   0
  0.3120 -0.2368  0.9201  0.6038  2.0000   0
 -0.7647 -0.6373  0.0952  0.1743  2.0000   0
```

```matlab
» drawgeom(cfg);
```

The mechanism should now be displayed in a figure window as in Figure B.1. Next the mechanism's loops will be closed with `connect.m`. The actuator (joint 1) is set to an angle of 0.15 rad.

```matlab
» [0c chi]=connect([0.15 1],0,def4)
```

```matlab
0c =
  0.1500
  0.3000
  0.1500
 -0.0000
 -0.1500
   0
chi =
  1.0647e-024
```

The returned parameter `chi` gives the cost function after closure as an indication of how successful the closure algorithm was. Next, display the robot in a different colour for
comparison.

```matlab
cfg = dirgeom(Dc,dim);
opt=drawgeom;
opt.LinkColour='g';
drawgeom(cfg);
```

The closed robot is displayed along with the result of the previous call to drawgeom as in Figure B.2. The graphics objects created by drawgeom may be cleared with

```matlab
clr
```

Calibration

The following is an example calibration of a Delta robot using measurements from a single theodolite. For calibration, a set of measurements are required along with a suitable geometric model. Such a set of data is stored in `mdelta.mat`.

```matlab
load mdelta
```

Each of the arguments that are to be sent to `calib.m` are included

- **Meas** The set of experimental or simulated measurements.
- **Frame** The frame (or frames) from which the measurements were taken.
- **Testpts** The nominal actuator settings which determine the test points.
- **Act** The joints in the model that are considered to be actuators (including the type 3 joints, which are treated as actuators during calibration).
- **Def** A set joint settings which serve as a starting estimate for the connect algorithm. The matrix corresponds column-for-column with Testpts.
- **Measdev** The name of the file that models the measurement device.
- **Def** The geometry definition matrix.
- **AdCn** A matrix of additional parameter constraints. Each row represents a constraint on an additional parameter: For example, to constrain the \( \alpha_2 \) parameter of joint 5, add the row \([5 2 0 0]\) to the matrix. The two zeros are there in anticipation of an addition to the code to allow bilateral constraints. By substituting zeros, the parameter is fixed to its starting value.
- **Opt** A set of optional parameters stored in a structure variable.
Figure B.1 Graphical output from `drawgeom.m`. An open 4 bar linkage.

Figure B.2 Graphical output from `drawgeom.m`. A closed 4 bar linkage whose free joint angles were determined by `connect.m`. 
First, extract the optional parameter set by calling \texttt{calib.m} with no parameters.

\begin{verbatim}
>> Opt = calib

Opt =
  options: [1x18 double]
  closure: 1.0000e-10
  jacobian: 'lsfunj'
  hessian: []
  distortion: 0.0100
  dummyrun: 0
  method: 'bfgs_idc'
CALIB_LAST_Y: []
\end{verbatim}

\begin{verbatim}
>> Opt.hessian = eye(74)*0.00001;
\end{verbatim}

The options are explained in the help text of \texttt{calib.m} which may be viewed by typing

\begin{verbatim}
>> help calib
\end{verbatim}

The identification process is started by typing

\begin{verbatim}
>> [g c r] = calib(Meas,Frame,Testpts,Act,Oo,Measdev,Def,AdCn,Opt)
\end{verbatim}

Initialising

Calibrating...

<table>
<thead>
<tr>
<th>Step</th>
<th>Value</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 BFGS</td>
<td>f=0.00199237</td>
<td>step=1</td>
</tr>
<tr>
<td>2 BFGS</td>
<td>f=0.00167726</td>
<td>step=1</td>
</tr>
<tr>
<td>3 BFGS</td>
<td>f=0.001666146</td>
<td>step=1</td>
</tr>
</tbody>
</table>

The identification process may take several hours.

To activate the quadratic bias function, (cf. §3.3) the global variable \texttt{CALIB_EPS} must be given a positive value and the centre (lowest point) of the function must be defined. The following code to iteratively decrease the bias function may be written into a script file and executed.

\begin{verbatim}
CALIB_EPS = 1e-4
CALIB_EPS CENTRE = Def
finished = 0;
g = Def;
while ~finished,
    g_old = g;
    [g c r J H] = calib(Meas,Frame,Testpts,Act,Oo,Measdev,g,AdCn,Opt);
\end{verbatim}
Opt.hessian = H;
CALIB_EPS=CALIB_EPS/10;
disp(sprintf('Calibration changed g by \%g,norm(g-g_old)))
if norm(g-g_old)<1e-4
    CALIB_EPS=CALIB_EPS/100
end
if CALIB_EPS==0, finished = 1; end
if norm(g-g_old)<1e-10,
    CALIB_EPS = 0;
end
save('result_intermediate','g','c','r','J','H','CALIB_EPS');
end
save('result','g','c','r','J','H');

Implementation

Given an identified model, the actuator coordinates that will achieve a desired pose can be calculated with implement.m.

```matlab
» load calib_result
» load mdelta
» opt=implement;
» opt.method = 'lm_min';
» opt.format = 'xyz';
» p=[-0.2 -0.2 -1.2];
» O=implement(p,Frame,Act,Co2,g,opt)
```

```matlab
O =
  1.0681
  0.64595
  0.96205
```

Finally, a check on the accuracy of the result using the perturbed model that was used to generate the measurements defreal. It is important to note that while calib.m automatically constrains the type 3 joints, connect.m does not. The type 3 joints are constrained manually in the following:

```matlab
» t3=find(g(:,9)==3);
» [0s chi]=connect([0 Act; zeros(size(t3)) t3],0o2,defreal);
» cfg=dirgeom(0s,mcp2dim(defreal));
» [Fr R]=extrgeom(30, cfg);
```
APPENDIX B  MATLAB SOFTWARE SUITE FOR CALIBRATION OF ROBOTS

```matlab
» ID=[R; conv_eul(Fr,[0 0 1 0]')];

ID =
    -0.20007
    -0.20061
    -1.1998
    -0.028595
    -0.00036438
    0.69072
    0.72256
```
References


REFERENCES


