# THESIS REPORT

## Ph.D.

A Three Degree of Freedom Parallel Manipulator with Only Translational Degrees of Freedom

*by R. E. Stamper*
*Advisor: L-W. Tsai*

Ph.D. 97-4




*Sponsored by*
*the National Science Foundation*
*Engineering Research Center Program,*
*the University of Maryland,*
*Harvard University,*
*and Industry*

ABSTRACT

Title of Dissertation: A THREE DEGREE OF FREEDOM PARALLEL MANIPULATOR WITH ONLY TRANSLATIONAL DEGREES OF FREEDOM

Richard Eugene Stamper, Doctor of Philosophy, 1997

Dissertation directed by: Professor Lung-Wen Tsai
Department of Mechanical Engineering
and Institute for Systems Research

In this dissertation, a novel parallel manipulator is investigated. The manipulator has three degrees of freedom and the moving platform is constrained to only translational motion. The main advantages of this parallel manipulator are that all of the actuators can be attached directly to the base, closed-form solutions are available for the forward kinematics, the moving platform maintains the same orientation throughout the entire workspace, and it can be constructed with only revolute joints.

Closed-form solutions for both the forward and inverse kinematics problems are presented. It is shown that the inverse kinematics problem has up to four real solutions, and the forward kinematics problem has up to 16 real solutions. The Jacobian matrix for the manipulator is also developed, and used to identify

singular poses of the manipulator, where the manipulator instantaneously gains or loses a degree of freedom.

The manipulator workspace volume as a function of link lengths and leg orientation is determined using the Monte Carlo method. A procedure for characterizing the quality of the workspace is also developed. Using these results, optimization studies for maximum workspace volume and for well-conditioned workspace volume are conducted. The objective function for the well-conditioned optimization study is defined as the integration of the reciprocal of the condition number of the Jacobian matrix over the workspace volume, and named the global condition index.

Three different models are developed for the manipulator dynamics, with numerical simulations presented for all three models. The first model is based upon the application of the Newton-Euler equations of motion used in conjunction with the Jacobian matrix to map the inertial and gravitational loadings of the moving platform to the actuators. The second model was developed to give a more complete characterization of the dynamics, and is based upon the Lagrangian multiplier approach. The third model neglects the highly coupled nature of the manipulator and models each input link individually. This model is developed for use with single-input single-output type controllers.

A prototype was fabricated to demonstrate this manipulator. Three controllers are developed and tested on the prototype, where each type of control tested relied on different characterizations of the manipulator dynamics. The three controllers are a PID controller, a computed torque controller, and an iterative learning controller.

The research presented in this dissertation establishes this parallel manipu-

lator as a viable robotic device for three degree of freedom manipulation. The manipulator offers the advantages associated with other parallel manipulators, such as light weight construction; while avoiding some of the traditional disadvantages of parallel manipulators such as the extensive use of spherical joints and coupling of the platform orientation and position.

# A THREE DEGREE OF FREEDOM PARALLEL MANIPULATOR WITH ONLY TRANSLATIONAL DEGREES OF FREEDOM

by

Richard Eugene Stamper

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland at College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1997

Advisory Committee:

Professor Lung-Wen Tsai, Chairman/Advisor
Professor P. Krishnaprasad
Associate Professor Y. Wang
Associate Professor G. Zhang
Assistant Professor G. Walsh

DEDICATION

To Anne

# ACKNOWLEDGEMENTS

I owe a debt of gratitude to Dr. Lung-Wen Tsai. He has been an excellent advisor, providing sound technical guidance and offering an outstanding example of professional conduct. I am truly lucky to have him as an advisor. The guidance of Dr. Gregory Walsh has also been proven to be valuable during the course of this research. I also want to thank Dr. Krishnaprasad and his graduate students in the Intelligent Servo Systems Lab. Numerous impromptu discussions with the people associated with his lab have been useful and have enriched my graduate school experience. I especially want to thank George Kantor who assembled the AC100 controller prototyping system. His efforts made my work much easier. Thanks should also be extended to Drs. Wang and Zhang for their effort and advice. Furthermore, I want to thank Dave Artigliere and Remi Taulemesse for their help with developing the prototype manipulator.

The work presented in this dissertation was supported in part by the National Science Foundation under Grant No. NSF EEC 94-02384. Thanks should also be extended to the Department of Mechanical Engineering and the Institute for Systems Research, particularly to Drs. Anand and Marcus for supporting the fabrication of the prototype manipulator.

TABLE OF CONTENTS

**List of Tables** **vii**

**List of Figures** **viii**

**Nomenclature** **xi**

**1 Introduction** **1**
1.1 Background . . . . . 1
1.2 Prior Work . . . . . 2
1.3 Outline . . . . . 6
1.4 Contributions . . . . . 8

**2 Description of Manipulator** **9**
2.1 Introduction . . . . . 9
2.2 Manipulator Structure . . . . . 9
2.3 Manipulator Mobility . . . . . 13
2.4 Special Case Manipulator Description . . . . . 14

**3 Manipulator Kinematics** **15**
3.1 Introduction . . . . . 15
3.2 Inverse Kinematics . . . . . 16
3.2.1 Derivation of Inverse Kinematic Solutions . . . . . 17
3.2.2 Inverse Kinematic Solution for Special Case Manipulator . 18
3.3 Forward Kinematics . . . . . 20
3.3.1 Derivation of Forward Kinematic Solutions . . . . . 21
3.3.2 Forward Kinematic Solution for Special Case Manipulator 26
3.3.3 Numerical Example . . . . . 28
3.4 Summary . . . . . 33

**4 Jacobian and Singularity Analysis** **35**
4.1 Introduction . . . 35
4.2 Derivation of Jacobian Matrix . . . 39
4.3 Inverse Kinematic Singularities . . . 43
4.4 Forward Kinematic Singularities . . . 44
4.5 Summary . . . 49

**5 Workspace Analysis and Optimization** **51**
5.1 Introduction . . . 51
5.2 Determination of Workspace Volume . . . 53
5.3 Optimization for Total Workspace Volume . . . 55
5.4 Determination of Workspace Condition . . . 56
5.5 Optimization for Well Conditioned Workspace Volume . . . 57
5.6 Summary . . . 64

**6 Analysis of Manipulator Dynamics** **65**
6.1 Introduction . . . 65
6.2 Direct Application of Newton-Euler Equations of Motion for Simplified Manipulator . . . 67
6.3 Lagrange Based Dynamic Analysis . . . 71
6.4 Single Link Dynamic Model . . . 76
6.5 Numerical Simulations . . . 79
6.6 Summary . . . 88

**7 Prototype Design and Accuracy** **91**
7.1 Introduction . . . 91
7.2 Description of Prototype Design . . . 93
7.3 Accuracy of Prototype . . . 98
7.4 Summary . . . 108

**8 Controller Design and Validation** **109**
8.1 Introduction . . . 109
8.2 PID Control . . . 111
8.2.1 Closed-Loop Model . . . 112
8.2.2 Gain Selection . . . 114
8.2.3 Integrator Windup . . . 118
8.2.4 PID Control Experimental Results . . . 119
8.3 Computed Torque Control . . . 125
8.3.1 Computed Torque Control Design . . . 125
8.3.2 Computed Torque Control Experimental Results . . . 127
8.4 Iterative Learning Control . . . 132
8.4.1 Iterative Learning Control Design . . . 133

8.4.2 Iterative Learning Control Experimental Results . . . . . 135
8.5 Summary . . . . . 139

**9 Summary and Future Research** **144**
9.1 Summary . . . . . 144
9.2 Future Research . . . . . 146

**Appendix A Constants for Forward Kinematics** **149**

**Appendix B Evaluation of Partial Derivatives of Constraint Functions for Langrange Based Dynamics** **155**

**Appendix C Evaluation of Partial Derivatives of Lagrange Function for Langrange Based Dynamics** **162**

**Appendix D Link Drawings** **164**

**Appendix E Complete Single Leg PID Block Diagram** **168**

**Appendix F Trajectory Generator Flowcharts** **169**

**Appendix G Trajectory Generation Code** **172**

**References** **185**

LIST OF TABLES


3.1 Manipulator parameters and input angles for numerical example. . 29

3.2 Solutions of $t_{31}$ for the forward kinematics numerical example. . . 31

3.3 Position of the moving platform for all the solutions of the numerical example for a general case manipulator. . . . . 32

3.4 Position of the moving platform for all the solutions of the numerical example for a special case manipulator. . . . . 33

5.1 Procedure to determine manipulator workspace volume using the Monte Carlo method. . . . . 54

5.2 Procedure to estimate global condition index of the manipulator. . 58

6.1 Manipulator design parameters for dynamic numerical simulations. 80

7.1 Position of moving platform as determined with measurement jig for poses used to measure geometric accuracy. . . . . 102

7.2 Measured input link angular displacement for each test pose. . . . 103

7.3 Moving platform position calculated from input link angular displacement and the associated position error. . . . . 104

7.4 Moving platform position error along x, y, and z coordinate axes. 106

7.5 Moving platform position errors calculated with shifted origin. . . 107

LIST OF FIGURES

1.1 A version of the Stewart platform. . . . . 3
1.2 A special form of the Stewart platform with concentric spherical joints. . . . . 4
1.3 The Delta Platform. . . . . 5

2.1 Schematic of the three-DOF manipulator. . . . . 10
2.2 Depiction of the joint angles and link lengths for leg $i$. . . . . 12

4.1 Description of leg $i$ used for derivation of the Jacobian. . . . . 40
4.2 Example of forward kinematic singular configuration where all the upper arm linkages are in the plane of the moving platform. . . . 47
4.3 Example of a forward kinematic singular configuration where two upper arm linkages are parallel. . . . . 48

5.1 Workspace of manipulator designed for maximum workspace volume. . . . . 56
5.2 Workspace of manipulator designed for maximum global condition index. . . . . 59
5.3 Reciprocal of the condition number at the $z$=.5 plane for the total workspace optimized manipulator. . . . . 60

5.4 Reciprocal of the condition number at the $z = .5$ plane for the global condition index optimized manipulator. . . . . 61
5.5 Workspace of a manipulator designed for maximum total workspace volume with an additional constraint that requires the legs to be separated by 120 degrees. . . . . 62
5.6 Reciprocal of the condition number at the $z$=.5 plane for a manipulator with 120 degree leg separation that is optimized for total workspace volume. . . . . 63

6.1 Schematic of the $i^{th}$ leg, showing the orientation of the manipulator and the direction of the applied actuator torque for the dynamic analysis. . . . . 68
6.2 Single link dynamics model for the $i^{th}$ leg. . . . . 77
6.3 Angular displacement of the input links to follow the trajectory used for numerical simulations of the dynamics models. . . . . 81
6.4 Velocity of moving platform along the example trajectory. . . . . 82
6.5 Actuator torques for the simplified dynamics model using Newton-Euler equations of motion and the Jacobian matrix. . . . . 84
6.6 Actuator torques for the example trajectory using the Lagrangian dynamics model. . . . . 86
6.7 Actuator torques for the example trajectory using the single link dynamics model. . . . . 87

7.1 Photograph of manipulator prototype. . . . . 92
7.2 Block diagram of AC-100 controller prototyping system. . . . . 94
7.3 Example of controller block diagram using SystemBuild. . . . . 97

7.4 Jig setup to measure position of moving platform. . . . . 100
7.5 Prototype position measurement technique. . . . . 101

8.1 PID controller block diagram for each input link. . . . . 113
8.2 Step response for closed-loop system with optimized gains and initial guesses for the gains. . . . . 116
8.3 Actuator torques for the closed-loop step response with optimized gains and initial guesses for the gains. . . . . 117
8.4 PID controller block with changes made to correct integrator windup. . . . . 119
8.5 Input-link joint angle error obtained from the PID controller. . . . 121
8.6 Moving platform position error as estimated from the input link joint errors obtained from the PID controller. . . . . 122
8.7 Condition number of the Jacobian matrix for the prototype manipulator as it follows the sample trajectory. . . . . 124
8.8 Diagram of computed torque approach. . . . . 127
8.9 Input-link joint errors obtained from the computed torque controller. 130
8.10 Moving platform position error as estimated from the input link joint errors obtained from the computed torque controller. . . . . 131
8.11 Input-link joint errors obtained from the iterative learning controller after 100 trials. . . . . 136
8.12 Moving platform position error as estimated from the input link joint errors obtained from the iterative learning controller. . . . . 137
8.13 Leg 1 input link joint errors after selected number of trials. . . . . 141
8.14 Leg 2 input link joint errors after selected number of trials. . . . . 142
8.15 Leg 3 input link joint errors after selected number of trials. . . . . 143

NOMENCLATURE

| | |
|---|---|
| $a$ | length of input link |
| $b$ | length of four-bar linkage |
| $c$ | distance from center of moving platform to joint with each leg |
| $d$ | length of link connecting four-bar linkage to moving platform |
| $e$ | length of link connecting four-bar linkage to input link |
| $i$ | index counter |
| $j$ | index counter |
| $k_n$ | constant used for kinematics solution |
| $m_a$ | mass of one input link |
| $m_b$ | mass of one connecting rod in upper arm linkage assembly |
| $m_c$ | mass of the moving platform |
| $m_d$ | mass of the link connecting the input link to the upper arm |
| $m_e$ | mass of the link connecting the upper arm to the moving platform |
| $p_x$ | the X component of the position of P in the XYZ frame |
| $p_y$ | the Y component of the position of P in the XYZ frame |
| $p_z$ | the Z component of the position of P in the XYZ frame |
| $p_{ui}$ | the U component of the position of P in the UVW frame of leg $i$ |
| $p_{vi}$ | the V component of the position of P in the UVW frame of leg $i$ |
| $p_{wi}$ | the W component of the position of P in the UVW frame of leg $i$ |
| $r$ | distance from center of fixed platform to joint with each leg |
| $t_{ji}$ | the tangent of the half-angle for joint $j$ on leg $i$ |
| $A_i$ | joint location between the input link and the fixed base |

$B_i$ joint location between the input link and link $BC$ for leg $i$

$C_i$ joint location between link $BC$ and link $CD$ for leg $i$

$D_i$ joint location between link $CD$ and link $DE$ for leg $i$

$E_i$ joint location between link $DE$ and the moving platform for leg $i$

$\mathbf{I}$ identity matrix

$I_m$ mass moment of inertia of the motor rotor

$\mathbf{J}$ Jacobian matrix

$\mathbf{J}_I$ Jacobian matrix associated with the inverse kinematics

$\mathbf{J}_F$ Jacobian matrix associated with the forward kinematics

$L$ Lagrange function, T-V

$P$ the center of the moving platform

$O$ the center of the fixed platform

$T$ total kinetic energy of the manipulator

$T_{ai}$ kinetic energy of the input link of the $i^{th}$ leg

$T_{bi}$ kinetic energy of one connecting rod of the $i^{th}$ leg

$T_c$ kinetic energy of the moving platform

$V$ total potential energy of the manipulator

$V_{ai}$ potential energy of the input link of the $i^{th}$ leg

$V_{bi}$ potential energy of one connecting rod of the $i^{th}$ leg

$V_c$ potential energy of the moving platform

$V_{p,x}$ velocity of P on the moving platform in the $X$ direction of the XYZ coordinate frame

$V_{p,y}$ velocity of P on the moving platform in the $Y$ direction of the XYZ coordinate frame

| | |
|---|---|
| $V_{p,z}$ | velocity of P on the moving platform in the $Z$ direction of the XYZ coordinate frame |
| $W$ | workspace volume of the manipulator |
| $\bar{a}_P$ | acceleration of moving platform |
| $\bar{f}$ | forces applied to moving platform |
| $\bar{n}_{CD,i}$ | unit vector in the direction of the upper arm linkage |
| $\bar{p}$ | a vector from $O$ on the fixed platform to $P$ on the moving platform |
| $\tilde{p}_i$ | a vector from $A_i$ to $P$ in the $UVW$ coordinate frame |
| $\alpha_i$ | arbitrary constant |
| $\eta$ | global condition index for manipulator workspace |
| $\theta_{ji}$ | the angular joint displacement of joint $j$ on leg $i$ |
| $\lambda$ | condition number of the Jacobian matrix |
| $\tau_i$ | actuator torque for $i^{th}$ leg |
| $\phi_i$ | the angular displacement of each leg relative to the X axis |
| $\omega_{ni}$ | the angular velocity of the $n^{th}$ link of the $i^{th}$ leg |
| $\Gamma$ | matrix of learning parameters |

# Chapter 1

# Introduction

## 1.1 Background

Machines that manipulate their surroundings have accounted for many significant changes in the way people live their daily lives. The exploration of the parallel manipulator presented in this dissertation is an effort to expand our understanding of the options available for the manipulation of our surroundings.

Parallel manipulators are robotic devices that differ from the more traditional serial robotic manipulators by virtue of their kinematic structure. Parallel manipulators are composed of multiple closed kinematic loops. Typically, these kinematic loops are formed by two or more kinematic chains that connect a moving platform to a base, where one joint in the chain is actuated and the other joints are passive. This kinematic structure allows parallel manipulators to be driven by actuators positioned on or near the base of the manipulator. In contrast, serial manipulators do not have closed kinematic loops and are usually actuated at each joint along the serial linkage. Accordingly, the actuators that are located at each joint along the serial linkage can account for a significant

portion of the loading experienced by the manipulator. Whereas the links of a parallel manipulator generally need not carry the load of the actuators. This allows the parallel manipulator links to be made lighter than the links of an analogous serial manipulator. Hence, parallel manipulators can enjoy the potential benefits associated with light weight construction such as high-speed operation and improved load to weight ratios. Some parallel manipulators have also been shown to display outstanding rigidity as a result of the multiple closed kinematic loops (Tahmasebi and Tsai, 1995). The most significant drawback of parallel manipulators is that they have smaller workspaces than serial manipulators of similar size.

## 1.2 Prior Work

Probably the most famous parallel manipulator is the Stewart platform (see Fig. 1.1). It was originally designed by Stewart (1965) as a flight simulator, and versions of it are still commonly used for that purpose. Since then, the Stewart platform has also been used for other applications such as milling machines (Aronson, 1996), pointing devices (Gosselin and Hamel, 1994), and an underground excavation device (Arai, 1991).

The Stewart platform has been studied extensively (Hunt, 1983; Fichter, 1986; Griffis and Duffy, 1989; Innocenti and Parenti-Castelli, 1993; and Nanua et al., 1990). Generally, the Stewart platform has six limbs, where each limb is connected to both the base and the moving platform by spherical joints located at each end of the limb. Actuation of the platform is typically achieved by changing the lengths of the limbs. While, these six-limbed manipulators offer

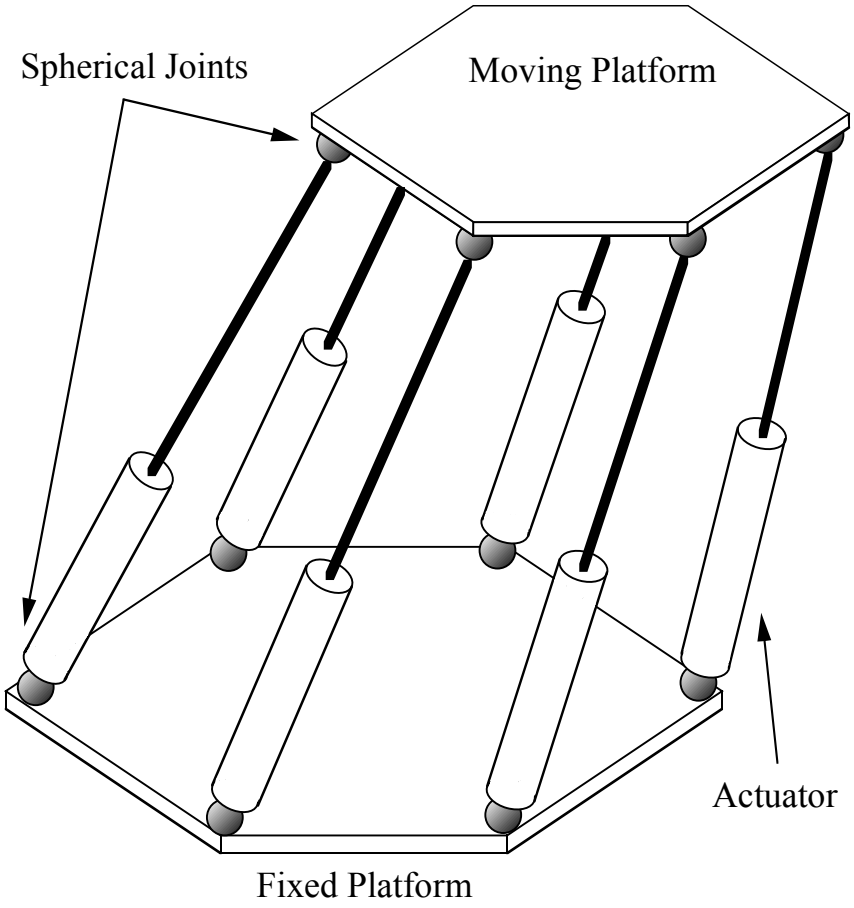



Figure 1.1: A version of the Stewart platform.

good rigidity, simple inverse kinematics, and high payload capacity, they suffer the following disadvantages.

1. Their direct kinematics are difficult to solve.

2. Position and orientation of the moving platform are coupled.

3. Precise spherical joints are difficult to manufacture at low cost.

Moreover, the closed-form direct kinematic solutions that have been reported in the literature have generally required special forms of the Stewart Platform

(Nanua et al., 1990; Griffis and Duffy, 1989; Lin et al., 1994). Most commonly, these special forms require concentric spherical joints (see Fig. 1.2). In these special forms, pairs of spherical joints may present design and manufacturing problems.

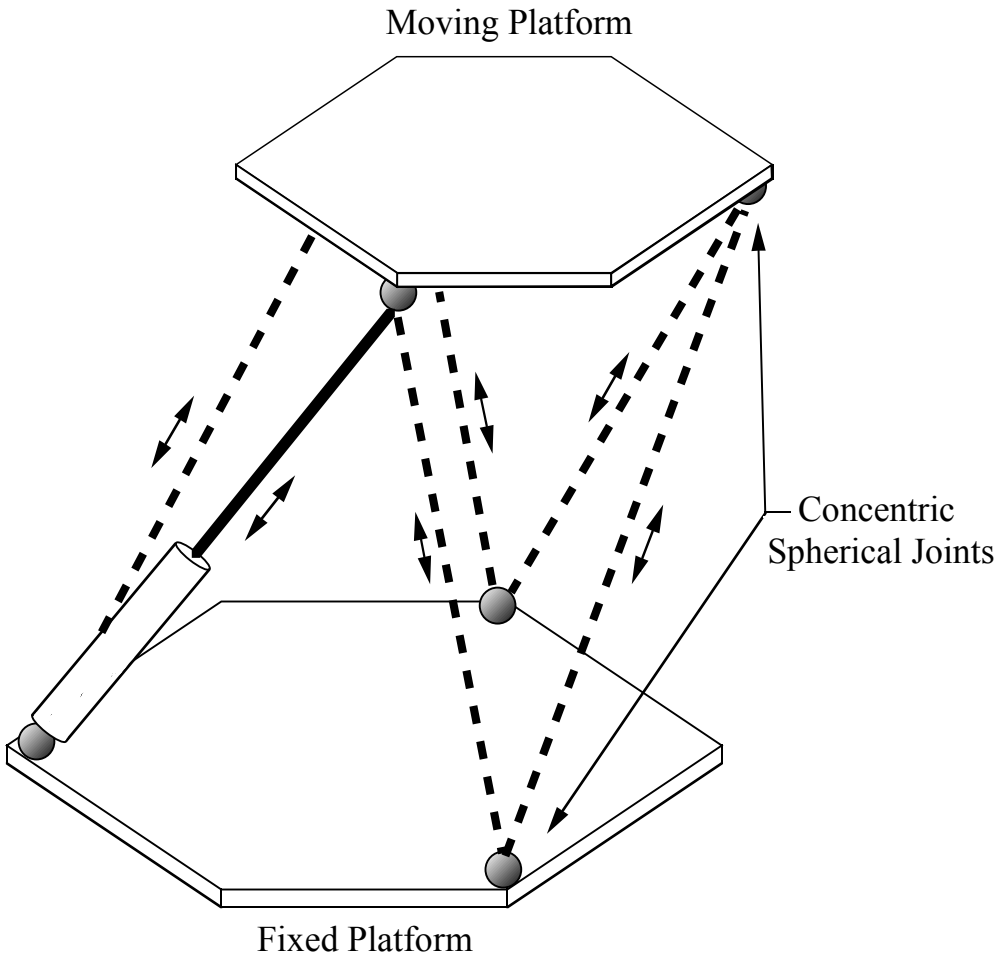



Figure 1.2: A special form of the Stewart platform with concentric spherical joints.

A three degree of freedom parallel manipulator that does not suffer from the first two of the listed disadvantages was designed by Clavel (1988) and others at the Swiss Federal Institute of Technology. The manipulator, called the DELTA

robot, has only translational degrees of freedom, and is shown in Fig. 1.3. Closed-form solutions for both the inverse and forward kinematics have been developed for the DELTA robot (Pierrot et al., 1990). Additionally, the position and orientation of the moving platform are uncoupled in the DELTA design. However, the DELTA robot construction does employ spherical joints.

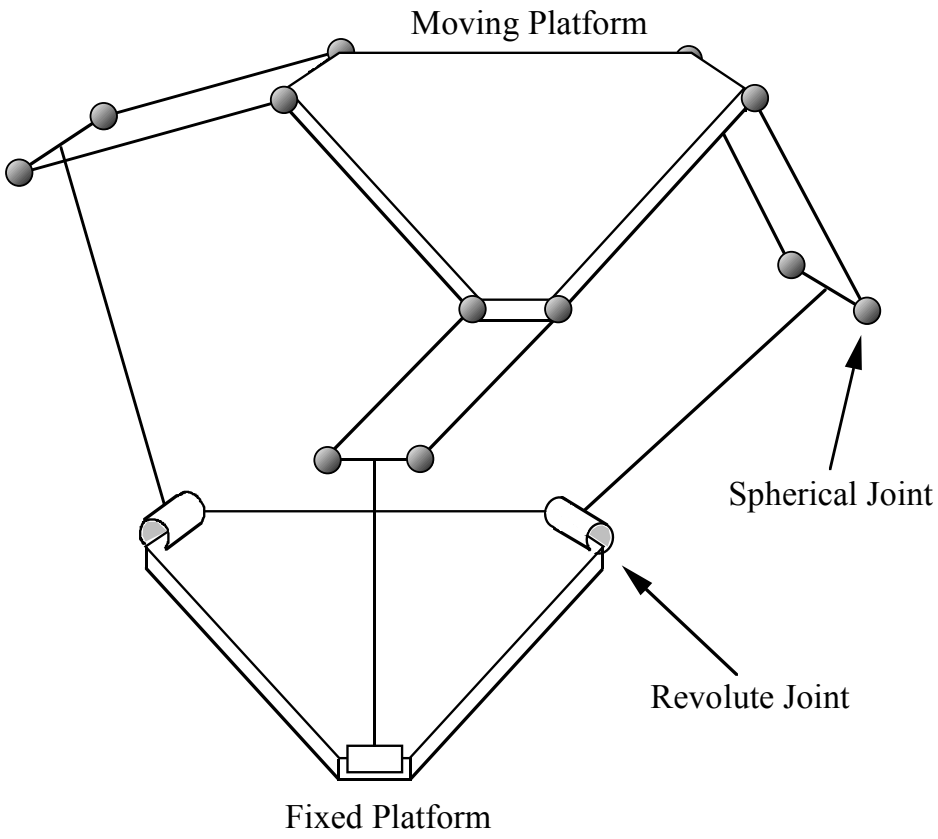



Figure 1.3: The Delta Platform.

A new parallel manipulator that eliminated the need for the spherical joints was invented by Tsai (1997). This manipulator uses only revolute joints to constrain the moving platform to three translational degrees of freedom. Developing

an understanding of this manipulator is the aim of the research presented in this dissertation.

## 1.3 Outline

A description of the manipulator is presented in Chapter 2, where the manipulator structure and mobility is discussed, and a special case of the manipulator is also described. Following the manipulator description, the key kinematic relationships are developed in Chapters 3 and 4. First the inverse kinematic problem is addressed, where given the position of the moving platform the goal is to find the sets of input joint values that allow the moving platform to achieve that position. It is shown that the inverse kinematics problem has up to four real solutions. Next, the forward kinematic problem is considered, where the goal is to determine the possible moving platform positions that result from a given set of input joint values. Unlike serial manipulators, the forward kinematics problem is more difficult than the inverse kinematics problem for parallel manipulators. In this case, the forward kinematics problem is reduced to a $32^{nd}$ degree polynomial in a single variable, the half-angle tangent of an unknown joint angle, using the diayltic elimination method. Of the 32 solutions suggested by the polynomial, 16 are found to be extraneous, leaving 16 possible solutions for the forward kinematic problem. The kinematic analysis is continued in Chapter 4 with the development of the Jacobian matrix that provides a transformation from the velocity of the moving platform in cartesian space to the actuated joint velocities in joint space. The Jacobian is then used to search for conditions that result in singular manipulator configurations where the mobility of the manipulator

instantaneously changes.

In Chapter 5, the workspace of the manipulator is considered. Workspace volume as a function of the manipulator parameters is determined using the Monte Carlo method. A procedure for characterizing the quality of the workspace is also developed. This characterization of the workspace quality is based upon the condition number of the Jacobian matrix, which provides some insight to the magnification of input joint angle uncertainty onto the position error of the moving platform. Using these results, optimization studies for maximum workspace volume and for well-conditioned workspace volume are conducted.

Models for the dynamics of the manipulator are presented in Chapter 6. Three different models are developed, with numerical simulations presented for all three models. The first model is based upon the application of the Newton-Euler equations of motion used in conjunction with the Jacobian matrix to map the inertial and gravitational loadings of the moving platform to the actuators. This model is developed for use with a computed torque controller since it does a good job of capturing the character of the dynamics while being computationally efficient. The second model was developed to give a more complete characterization of the dynamics, and is based upon the Lagrangian multiplier approach. The third model neglects the highly coupled nature of the manipulator and models each input link individually. This model is developed for use with single-input single-output type controllers.

A prototype was fabricated to demonstrate this manipulator. A description of the prototype and the accuracy of the prototype are provided in Chapter 7. Three controllers were developed for the manipulator and tested on the prototype. A PID controller, a computed torque controller, and an iterative learning

controller were all applied to the prototype, and the results are presented in Chapter 8. Finally, a summary and suggestions for future research are provided in Chapter 9.

## 1.4 Contributions

The aim of this dissertation is to expand the understanding of this new parallel manipulator. The contributions to this understanding are summarized by the following list:

1. Creation of a new class of parallel manipulators with three translational degrees of freedom.

2. Derivation of closed-form solutions for both the inverse and forward kinematics problems for this new class of parallel manipulators.

3. Development of methods to determine the workspace volume for the new manipulator.

4. Design guidelines to achieve maximum workspace volume or maximum well-conditioned workspace volume.

5. Development of models for the manipulator dynamics that are suitable for various purposes.

6. Prototype demonstration of this type of manipulator.

# Chapter 2

# Description of Manipulator

## 2.1 Introduction

In this chapter, the parallel manipulator that is the focus of this research is described. The configuration of the links and the joints that define this manipulator are presented, along with the associated nomenclature. The mobility of the manipulator is also considered.

## 2.2 Manipulator Structure

A schematic of the manipulator being considered is shown in Fig. 2.1, where the stationary platform is labeled 0 and the moving platform is labeled 16. Three identical limbs connect the moving platform to the stationary platform. Each limb consists of an input link and an upper arm. The input links are labeled 1, 2, and 3. Each upper arm is a planar four-bar parallelogram: links 4, 7, 10, and 13 for the first limb; 5, 8, 11, and 14 for the second limb; and 6, 9, 12, and 15 for the third limb. All of the links and platforms are considered rigid bodies.

For each limb, the upper arm, input links, and the two platforms are con-

Figure 2.1: Schematic of the three-DOF manipulator.

nected by three parallel revolute joints at axes $A_i$, $B_i$, and $E_i$ as shown in Fig. 2.1. The axes of these revolute joints are perpendicular to the axes of the four-bar parallelogram for each limb. There is also a small offset between the upper arm assembly and the axes of the revolute joints at $B_i$ and $E_i$. These offsets are a function of the geometry of links 4, 5, 6, 13, 14, and 15. The axes of $A_1$, $A_2$, and $A_3$ lie on a plane attached to the fixed platform. Similarly, the axes of $E_1$, $E_2$, and $E_3$ lie on a plane attached to the moving platform.

A reference frame (XYZ) is attached to the fixed base at point O, located at the center of the fixed platform. The x and y axes lie in the same plane as defined by the axes of $A_1$, $A_2$, and $A_3$. The angle $\phi_i$ for the $i^{th}$ leg as shown in Fig. 2.1 defines the angular orientation of the leg relative to the XYZ frame on the fixed platform. Another coordinate system ($U_iV_iW_i$) is attached to the fixed base at $A_i$ for each leg, such that the $u_i$-axis is perpendicular to the axis of rotation of the joint at $A_i$ and at an angle $\phi_i$ from the x-axis, while being in the plane of the fixed platform. The $v_i$-axis is along the joint axis of $A_i$.

The $i^{th}$ leg of the manipulator is shown in Fig. 2.2. The vector $\bar{p}$ is the position vector of point P in the (XYZ) coordinate frame, where P is attached at the center of the moving platform. The angle $\theta_{1i}$ is measured from $\bar{u}$ to $\overline{AB}$. The angle $\theta_{2i}$ is defined from the $\bar{u}$ direction to $\overline{BC}$. The angle $\theta_{3i}$ is defined by the angle from the $\bar{v}$ direction to $\overline{CD}$. The link lengths are also shown in Fig. 2.2.

For the purposes of this research, $\theta_{11}, \theta_{12}$, and $\theta_{13}$ are considered the actuated joints. Other combinations of actuated joints are also possible, but actuating $\theta_{11}, \theta_{12}$, and $\theta_{13}$ offers the advantage of attaching each of the actuators to ground.

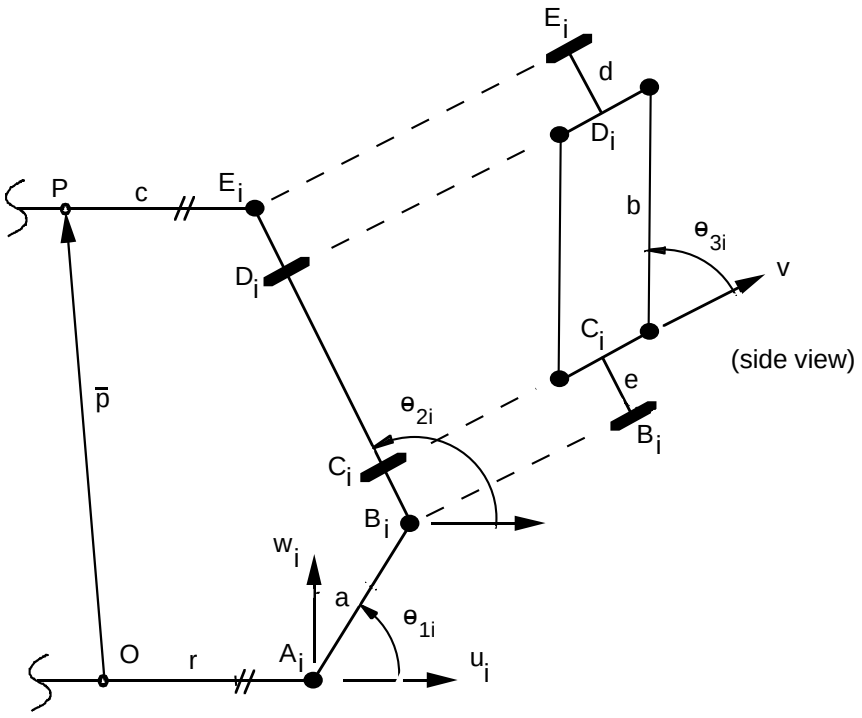



Figure 2.2: Depiction of the joint angles and link lengths for leg $i$.

## 2.3 Manipulator Mobility

Considering the manipulator mobility, let $F$ be the degrees of freedom, $n$ the number of links, $j$ the number of joints, $f_i$ the degrees of freedom associated with the $i^{th}$ joint, and $\lambda = 6$, the motion parameter. Then, the degrees of freedom of a mechanism is generally governed by the following mobility equation:

$$F = \lambda(n - j - 1) + \sum_i f_i \tag{2.1}$$

For the manipulator shown in Fig. 2.1, $n = 17, j = 21$, and $f_i = 1$ for $i = 1, 2, ..., 21$. Applying Eq. (2.1) to the manipulator produces: $F = 6(17 - 21 - 1) + 21 = -9$. Hence, the manipulator is an overconstrained mechanism. However, due to the arrangement of the links and joints, many of the constraints imposed by the joints are redundant and the resulting mechanism does have three translational degrees of freedom. To understand this, first observe that the three revolute joints at $A_i$, $B_i$, and $E_i$ for the $i^{th}$ leg have parallel axes as shown in Fig. 2.2. Since these axes always remain parallel, and the 4-bar parallelogram does not effect the orientation of the moving platform, the $i^{th}$ leg provides two constraints on the rotation of the moving platform about the z and $u_i$ axes. Hence, the combination of any two limbs constrains rotation about the x, y, and z axes. Accordingly, the moving platform remains in the same angular orientation with respect to the fixed platform from the constraints of any two legs. This leaves the mechanism with three translational degrees of freedom and constrains any rotation of the moving platform.

## 2.4 Special Case Manipulator Description

A special case of the manipulator is formed when $d = e = 0$. That occurs when the links that attach the four-bar linkage to the moving platform and to the input links have zero length such that the axes of the revolute joints at $B$ and $C$ intersect and the axes of the revolute joints at $D$ and $E$ intersect. This special case results in a manipulator with a less complex kinematic structure, and is similar to a manipulator devised by Clavel (1988) that employed spherical joints to obtain the desired kinematic structure.

# Chapter 3

# Manipulator Kinematics

## 3.1 Introduction

Manipulator kinematics deals with the study of the manipulator motion as constrained by the geometry of the links. The kinematic analysis is done without regard to the forces or torques that cause or result from the motion. Typically, the study of manipulator kinematics is divided into two parts, inverse kinematics and forward (or direct) kinematics. The inverse kinematics problem involves mapping a known position of the output link of the manipulator to a set of input joint variables that will achieve that position. The forward kinematic problem involves the mapping from a known set of input joint variables to a position of the moving platform that results from those given inputs. Generally, as the number of closed kinematic loops in the manipulator increases, the difficulty of solving the forward kinematic relationships increases while the difficulty of solving the inverse kinematic relationships decreases. As an example, the forward kinematics problem for a traditional 6 degree of freedom serial link manipulator is relatively simple, while the inverse kinematic problem is difficult (e.g.

Raghavan and Roth, 1993). In contrast, the inverse kinematics of the 6 degree of freedom Stewart platform is relatively simple, but the forward kinematics is difficult (e.g. Zhang and Song, 1994).

For this manipulator, the inverse kinematics problem is solved algebraically and shows that there are four solutions for each leg for the general case manipulator, and two solutions for the special case manipulator, where $d = e = 0$ (see Fig. 2.2). The forward kinematics problem is solved with the application of a dialytic elimination method and shows that there are 16 solutions for a given set of input joint angles for the general case manipulator. It's also shown algebraically that there are two solutions to the forward kinematics problem for the special case manipulator.

## 3.2 Inverse Kinematics

The objective of the inverse kinematics solution is to define a mapping from the position of the moving platform in a cartesian space to the set of joint angles that achieve that position. For this analysis, the position of the moving platform is considered known, and is given by the position vector $\bar{p}$, which defines the location of $P$ at the center of the moving platform in the $XYZ$ coordinate frame. The inverse kinematics analysis produces a set of three joint angles for each leg ($\theta_{1i}, \theta_{2i}$, and $\theta_{3i}$ for the $i^{th}$ leg) that define the possible postures for each leg for the given position of the moving platform.

### 3.2.1 Derivation of Inverse Kinematic Solutions

The following transformation expresses the position of $P$ in the (UVW) coordinate frame attached at point $A$ for leg $i$:

$$\begin{bmatrix} p_{ui} \\ p_{vi} \\ p_{wi} \end{bmatrix} = \begin{bmatrix} \cos(\phi_i) & \sin(\phi_i) & 0 \\ -\sin(\phi_i) & \cos(\phi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} + \begin{bmatrix} -r \\ 0 \\ 0 \end{bmatrix}. \tag{3.1}$$

Expressions for $p_{ui}, p_{vi}$, and $p_{wi}$ are given by:

$$p_{ui} = a\cos(\theta_{1i}) - c + [d + e + b\sin(\theta_{3i})]\cos(\theta_{2i}), \tag{3.2}$$

$$p_{vi} = b\cos(\theta_{3i}), \tag{3.3}$$

$$p_{wi} = a\sin(\theta_{1i}) + [d + e + b\sin(\theta_{3i})]\sin(\theta_{2i}). \tag{3.4}$$

Two solutions are immediately found for $\theta_{3i}$ from Eq. (3.3):

$$\theta_{3i} = \pm \arccos\left(\frac{p_{vi}}{b}\right) \tag{3.5}$$

With $\theta_{3i}$ known, an equation with $\theta_{1i}$ as the only unknown is generated by isolating the $\theta_{2i}$ terms in Eqs. (3.2) and (3.4) and then summing the squares of those two equations so that $\theta_{2i}$ is eliminated with the application of the Pythagorean relationship:

$$\begin{aligned}(p_{ui} + c)^2 + p_{wi}^2 + a^2 - 2a(p_{ui} + c)\cos(\theta_{1i}) - 2ap_{wi}\sin(\theta_{1i}) \\ = (d + e)^2 + 2(d + e)b\sin(\theta_{3i}) + b^2\sin(\theta_{3i})^2.\end{aligned} \tag{3.6}$$

To transform Eq. (3.6) into a polynomial expression, a half-angle tangent is defined as:

$$t_{1i} = \tan\left(\frac{\theta_{1i}}{2}\right), \tag{3.7}$$

producing the following relationships:

$$\sin(\theta_{1i}) = \frac{2t_{1i}}{1+t_{1i}^2} \quad \text{and} \quad \cos(\theta_{1i}) = \frac{1-t_{1i}^2}{1+t_{1i}^2} \quad . \tag{3.8}$$

The half-angle substitution is applied to Eq. (3.6), and simplified to produce:

$$l_{2i}t_{1i}^2 + l_{1i}t_{1i} + l_{0i} = 0, \tag{3.9}$$

where:

$$\begin{aligned}
l_{0i} &= p_{wi}^2 + p_{ui}^2 + 2cp_{ui} - 2ap_{ui} + a^2 + c^2 - d^2 - e^2 \\
&\quad -b^2\sin(\theta_{3i})^2 - 2be\sin(\theta_{3i}) - 2bd\sin(\theta_{3i}) - 2de - 2ac, \\
l_{1i} &= -4ap_{wi}, \\
l_{2i} &= p_{wi}^2 + p_{ui}^2 + 2cp_{ui} + 2ap_{ui} + a^2 + c^2 - d^2 - e^2 \\
&\quad -b^2\sin(\theta_{3i})^2 - 2be\sin(\theta_{3i}) - 2bd\sin(\theta_{3i}) - 2de + 2ac.
\end{aligned}$$

Equation (3.9) can be solved for $t_{1i}$, producing two possible values for $\theta_{1i}$ for each of the two solutions found for $\theta_{3i}$. With $\theta_{1i}$ and $\theta_{3i}$ known, $\theta_{2i}$ is found by back-substitution into Eqs. (3.2) and (3.4). Hence, for a given position of the moving platform, there are four possible configurations for each leg.

### 3.2.2 Inverse Kinematic Solution for Special Case Manipulator

For the special case manipulator, where $d = e = 0$, so that the joint axes at $B$ and $C$ intersect and the joint axes at $D$ and $E$ intersect, the four solutions generated by Eqs. (3.9) and (3.5) represent just two postures. That is to say that only two postures are possible for each leg for a given position of the moving

platform. This happens because the two solutions for each $\theta_{3i}$ result in the same posture.

To understand why each $\theta_{3i}$ results in only one posture, first label the two solutions of (3.5) as $\theta_{3i}^{(1)}$ and $\theta_{3i}^{(2)}$. From Eq. (3.5), it can be observed that $\theta_{3i}^{(1)}$ and $\theta_{3i}^{(2)}$ are symmetrical about the revolute axis of joint $B$. Since $\theta_{3i}^{(1)}$ and $\theta_{3i}^{(2)}$ are symmetrical about the revolute axis of joint $B$ the following relationship exists between the two solutions:

$$\sin(\theta_{3i}^{(1)}) = -\sin(\theta_{3i}^{(2)}). \tag{3.10}$$

Accordingly, the solutions of Eq. (3.9) for $\theta_{1i}$ are independent of which solution of $\theta_{3i}$ is used since all the terms with $\theta_{3i}$ are of the form $\sin(\theta_{3i})^2$ when $d = e = 0$. If the two solutions of $\theta_{2i}$ associated with each solution of $\theta_{3i}$ are labeled $\theta_{2i}^{(1)}$ and $\theta_{2i}^{(2)}$, it's possible to show that $\theta_{2i}^{(1)}$ and $\theta_{2i}^{(2)}$ are separated by $\pi$. This can be observed by solving Eqs. (3.2) and (3.4) for $\cos(\theta_{2i})$ and $\sin(\theta_{2i})$ as follows:

$$\cos(\theta_{2i}) = \frac{p_{ui} + c - a\cos(\theta_{1i})}{b\sin(\theta_{3i})}, \tag{3.11}$$

$$\sin(\theta_{2i}) = \frac{p_{wi} - a\sin(\theta_{1i})}{b\sin(\theta_{3i})}. \tag{3.12}$$

From Eqs. (3.11), (3.12), and (3.10) it can be observed that $\sin(\theta_{2i}^{(1)}) = -\sin(\theta_{2i}^{(2)})$ and that $\cos(\theta_{2i}^{(1)}) = -\cos(\theta_{2i}^{(2)})$. As a result, a single value trigonometric function for the arctangent can be applied to show that $\theta_{2i}^{(1)} = \theta_{2i}^{(2)} \pm \pi$.

So, either value of $\theta_{3i}$ chosen as the solution for Eq. (3.5) will result in the same posture. This is true since, if the leg posture is calculated for a chosen value of $\theta_{3i}$, the posture generated from the solution not chosen for $\theta_{3i}$ will simply result in the reflection of the four bar linkage about the joint axis of $B$, and then a rotation about that axis by $\pi$ radians, producing the same leg posture.

## 3.3 Forward Kinematics

The objective of the forward kinematics solution is to define a mapping from the known set of the actuated joint angles to the unknown position of the moving platform. For this manipulator the joint angles that are considered known are the angles formed by the input links and the base of the manipulator, $\theta_{11}$, $\theta_{12}$, and $\theta_{13}$. The unknown position of the moving platform is described by the position vector $\bar{p}$, which defines the location of $P$ at the center of the moving platform in the $XYZ$ coordinate frame.

Unlike serial manipulators, the forward kinematics problem is more difficult to solve for parallel manipulators than the inverse kinematics problem. Often the forward kinematics problem for parallel manipulators is reduced to solving a large system of polynomial equations. Three common approaches to solving these systems of polynomial equations are outlined by Raghavan and Roth (1995). The three approaches are the Dialytic Elimination, Polynomial Continuation, and Grobner bases.

The approach applied in this case is Sylvester's Dialytic Elimination procedure (Salmon 1964). This procedure is used to eliminate one or more unknowns from a system of equations. This method was used by Husain and Waldron (1994) to solve the inverse and direct kinematics problems of a three-limbed parallel platform with two actuated joints and four passive joints associated with each limb. This procedure has also been applied to the forward kinematics problems of the Stewart platform. Lin, Griffis, and Duffy (1992) used it to solve the kinematics for the 4-4 Stewart platform. Innocenti and Parenti-Castelli (1993) applied the procedure to reduce the direct kinematics problem of the 5-5 Stewart platform to a $40^{th}$ degree polynomial in a single unknown. Lazard and

Merlet (1994) used it to show that a three-legged version of the Stewart platform has twelve different forward kinematic solutions.

For this analysis, the loop closure equations are written for each leg, and are then algebraically reduced to two $16^{th}$ degree equations in two unknowns, the half-angle tangent of $\theta_{31}$ and $\theta_{32}$. The dialytic elimination method is then used to eliminate one of the unknowns from these two equations leaving a single $32^{nd}$ degree equation in a single unknown. Of the 32 solutions generated by this method, 16 are extraneous.

### 3.3.1 Derivation of Forward Kinematic Solutions

First, relationships are written for the position of P in the UVW coordinate frame for the $i^{th}$ leg, resulting in the following expressions for $p_{ui}, p_{vi}$, and $p_{wi}$:

$$p_{ui} = a\cos(\theta_{1i}) - c + [d + e + b\sin(\theta_{3i})]\cos(\theta_{2i}), \tag{3.13}$$

$$p_{vi} = b\cos(\theta_{3i}), \tag{3.14}$$

$$p_{wi} = a\sin(\theta_{1i}) + [d + e + b\sin(\theta_{3i})]\sin(\theta_{2i}). \tag{3.15}$$

To create relationships for the position of P in the XYZ coordinate frame for each leg, Eqs. (3.13), (3.14),and (3.15) are then substituted into Eq. (3.16), the transformation between the two coordinate systems.

$$\begin{bmatrix} p_{ui} \\ p_{vi} \\ p_{wi} \end{bmatrix} = \begin{bmatrix} \cos(\phi_i) & \sin(\phi_i) & 0 \\ -\sin(\phi_i) & \cos(\phi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} + \begin{bmatrix} -r \\ 0 \\ 0 \end{bmatrix} \tag{3.16}$$

This results in a system of 9 equations in 9 unknowns ($p_x, p_y, p_z, \theta_{21}, \theta_{22}, \theta_{23}, \theta_{31}, \theta_{32}$, and $\theta_{33}$):

$$p_x \;\cos(\phi_i) + p_y \sin(\phi_i) - a\cos(\theta_{1i})$$
$$-r + c - [d + e + b\sin(\theta_{3i})]\cos(\theta_{2i}) = 0, \tag{3.17}$$

$$p_y \;\cos(\phi_i) - p_x \sin(\phi_i) - b\cos(\theta_{3i}) = 0, \tag{3.18}$$

$$p_z \;-a\sin(\theta_{1i}) - [d + e + b\sin(\theta_{3i})]\sin(\theta_{2i}) = 0, \tag{3.19}$$

for $i = 1, 2$, and 3. The solution of this set of equations, represents the solution of the forward kinematics problem. To determine the solution, Eqs. (3.17), (3.18), and (3.19) are manipulated to produce two equations in the tangent of the half-angle of $\theta_{31}$ and $\theta_{32}$. These two equations are then solved using the dailytic elimination method. The algebra required to achieve this solution is what follows.

Letting $\phi_1 = 0$, an expression for $p_y$ is found by rewriting Eq. (3.18) for $i = 1$:

$$p_y = b\cos(\theta_{31}). \tag{3.20}$$

An expression for $p_x$ is developed by substituting (3.20) into Eq. (3.18) for $i = 2$:

$$p_x = \frac{b}{\sin(\phi_2)}\left[\cos(\phi_2)\cos(\theta_{31}) - \cos(\theta_{32})\right]. \tag{3.21}$$

An expression without $\theta_{2i}$ is generated by isolating the $\theta_{2i}$ terms in Eqs. (3.17) and (3.19) and then summing the squares of those two equations along with the square of Eq. (3.18) so that $\theta_{2i}$ is eliminated with the application of the Pythagorean relationship:

$$
\begin{aligned}
& p_x^2 + p_y^2 + p_z^2 - 2a\sin(\theta_{1i})p_z \\
& \quad +2\left[c - r - a\cos(\theta_{1i})\right]\left[p_x\cos(\phi_i) + p_y\sin(\phi_i)\right] \\
& \quad +a^2 + (r-c)^2 - 2a(c-r)\cos(\theta_{1i}) \\
& \quad -b^2 - (d+e)^2 - 2b(d+e)\sin(\theta_{3i}) = 0,
\end{aligned} \tag{3.22}
$$

for $i = 1, 2$, and 3.

An equation that is linear in $p_x,\ p_y,\ p_z,\ \sin(\theta_{31})$, and $\sin(\theta_{32})$ is generated by subtracting Eq. (3.22) for $i = 1$ from Eq. (3.22) for $i = 2$:

$$
k_1 p_x + k_2 p_y + k_3 p_z + k_4\sin(\theta_{31}) + k_5\sin(\theta 32) + k_6 = 0, \tag{3.23}
$$

where the constants are defined in the appendix. Similarly, an equation that is linear in $p_x,\ p_y,\ p_z,\ \sin(\theta_{31})$, and $\sin(\theta_{33})$ is generated by subtracting Eq. (3.22) for $i = 1$ from Eq. (3.22) for $i = 3$:

$$
k_7 p_x + k_8 p_y + k_9 p_z + k_{10}\sin(\theta_{31}) + k_{11}\sin(\theta 33) + k_{12} = 0. \tag{3.24}
$$

An expression for $p_z$ is generated by substituting Eqs. (3.20) and (3.21) into Eq. (3.23), and solving for $p_z$:

$$
p_z = k_{13}\sin(\theta_{31}) + k_{14}\cos(\theta_{31}) + k_{15}\sin(\theta_{32}) + k_{16}\cos(\theta_{32}) + k_{17}. \tag{3.25}
$$

Substituting the expressions for $p_x,\ p_y$, and $p_z$ into Eq. (3.24) produces an equation that is linear in $\sin(\theta_{31}),\ \cos(\theta_{31}),\ \sin(\theta_{32}),\ \cos(\theta_{32})$, and $\sin(\theta_{33})$:

$$
\begin{aligned}
& k_{18}\ \sin(\theta_{31}) + k_{19}\cos(\theta_{31}) + k_{20}\sin(\theta_{32}) \\
& \quad +k_{21}\cos(\theta_{32}) + k_{22}\sin(\theta_{33}) + k_{23} = 0.
\end{aligned} \tag{3.26}
$$

Substituting $p_x$ and $p_y$ into Eq. (3.18) for $i = 3$ generates another equation in only $\theta_{31}$, $\theta_{32}$, and $\theta_{33}$:

$$\sin(\phi_2 - \phi_3)\cos(\theta_{31}) + \sin(\phi_3)\cos(\theta_{32}) - \sin(\phi_2)\cos(\theta_{33}) = 0. \tag{3.27}$$

A third equation in $\theta_{31}$ and $\theta_{32}$ is created by substituting the expressions for $p_x$, $p_y$, and $p_z$ into Eq. (3.22) for $i = 1$:

$$\begin{aligned}
&k_{24}\cos^2(\theta_{31}) + k_{25}\sin^2(\theta_{31}) + k_{26}\cos^2(\theta_{32}) + k_{27}\sin^2(\theta_{32}) \\
&+k_{28}\cos(\theta_{31})\sin(\theta_{31}) + k_{29}\cos(\theta_{31})\cos(\theta_{32}) + k_{30}\cos(\theta_{31})\sin(\theta_{32}) \\
&+k_{31}\sin(\theta_{31})\cos(\theta_{32}) + k_{32}\sin(\theta_{31})\sin(\theta_{32}) + k_{33}\cos(\theta_{32})\sin(\theta_{32}) \\
&+k_{34}\cos(\theta_{31}) + k_{35}\sin(\theta_{31}) + k_{36}\cos(\theta_{32}) + k_{37}\sin(\theta_{32}) + k_{38} = 0.
\end{aligned} \tag{3.28}$$

This leaves three equations (3.26), (3.27), and (3.28) in three unknowns ($\theta_{31}$, $\theta_{32}$, and $\theta_{33}$). Solving Eqs. (3.26) and (3.27) for $\sin(\theta_{33})$ and $\cos(\theta_{33})$ respectively, and then substituting these expressions into the Pythagorean relationship, $\sin^2(\theta_{33}) + \cos^2(\theta_{33}) = 1$, yields:

$$\begin{aligned}
&k_{39}\cos^2(\theta_{31}) + k_{40}\sin^2(\theta_{31}) + k_{41}\cos^2(\theta_{32}) + k_{42}\sin^2(\theta_{32}) \\
&+k_{43}\cos(\theta_{31})\sin(\theta_{31}) + k_{44}\cos(\theta_{31})\cos(\theta_{32}) + k_{45}\cos(\theta_{31})\sin(\theta_{32}) \\
&+k_{46}\sin(\theta_{31})\cos(\theta_{32}) + k_{47}\sin(\theta_{31})\sin(\theta_{32}) + k_{48}\cos(\theta_{32})\sin(\theta_{32}) \\
&+k_{49}\cos(\theta_{31}) + k_{50}\sin(\theta_{31}) + k_{51}\cos(\theta_{32}) + k_{52}\sin(\theta_{32}) + k_{53} = 0.
\end{aligned} \tag{3.29}$$

Considering Eqs. (3.28) and (3.29) along with the relationship $\sin(\theta_{3i})^2 + \cos(\theta_{3i})^2 = 1$ as functions of four independent variables, $\sin(\theta_{31})$, $\cos(\theta_{31})$, $\sin(\theta_{32})$, and $\cos(\theta_{32})$, the system of equations can be considered as four $2^{\text{nd}}$ degree polynomials in four unknowns. The total degree of the system is 16. Therefore, there are at most 16 solutions.

Equations (3.28) and (3.29) are transformed into polynomials by applying the half-angle tangent relationships (3.8) and multiplying by $[(1+t_{31}^2)^2(1+t_{32}^2)^2]$ to clear the denominators, which is only valid if $(1+t_{31}^2) \neq 0$ and $(1+t_{32}^2) \neq 0$. Hence, any solution with values of $t_{31} = \pm i$ or $t_{32} = \pm i$ must be discarded from the final solution. The polynomials are expressed with $t_{31}$ suppressed as shown:

$$g_1 t_{32}^4 + g_2 t_{32}^3 + g_3 t_{32}^2 + g_4 t_{32} + g_5 = 0, \tag{3.30}$$

$$g_6 t_{32}^4 + g_7 t_{32}^3 + g_8 t_{32}^2 + g_9 t_{32} + g_{10} = 0, \tag{3.31}$$

where:

$$\begin{aligned}
g_1 &= k_{54}t_{31}^4 + k_{55}t_{31}^3 + k_{56}t_{31}^2 + k_{57}t_{31} + k_{58}, \\
g_2 &= k_{59}t_{31}^4 + k_{60}t_{31}^3 + k_{61}t_{31}^2 + k_{62}t_{31} + k_{63}, \\
g_3 &= k_{64}t_{31}^4 + k_{65}t_{31}^3 + k_{66}t_{31}^2 + k_{67}t_{31} + k_{68}, \\
g_4 &= k_{69}t_{31}^4 + k_{70}t_{31}^3 + k_{71}t_{31}^2 + k_{72}t_{31} + k_{73}, \\
g_5 &= k_{74}t_{31}^4 + k_{75}t_{31}^3 + k_{76}t_{31}^2 + k_{77}t_{31} + k_{78}, \\
g_6 &= k_{79}t_{31}^4 + k_{80}t_{31}^3 + k_{81}t_{31}^2 + k_{82}t_{31} + k_{83}, \\
g_7 &= k_{84}t_{31}^4 + k_{85}t_{31}^3 + k_{86}t_{31}^2 + k_{87}t_{31} + k_{88}, \\
g_8 &= k_{89}t_{31}^4 + k_{90}t_{31}^3 + k_{91}t_{31}^2 + k_{92}t_{31} + k_{93}, \\
g_9 &= k_{94}t_{31}^4 + k_{95}t_{31}^3 + k_{96}t_{31}^2 + k_{97}t_{31} + k_{98}, \\
g_{10} &= k_{99}t_{31}^4 + k_{100}t_{31}^3 + k_{101}t_{31}^2 + k_{102}t_{31} + k_{103}.
\end{aligned}$$

The dialytic elimination method is applied to Eqs. (3.30) and (3.31), pro-

ducing the following matrix equation:

$$\begin{bmatrix} g_5 & g_4 & g_3 & g_2 & g_1 & 0 & 0 & 0 \\ g_{10} & g_9 & g_8 & g_7 & g_6 & 0 & 0 & 0 \\ 0 & g_5 & g_4 & g_3 & g_2 & g_1 & 0 & 0 \\ 0 & g_{10} & g_9 & g_8 & g_7 & g_6 & 0 & 0 \\ 0 & 0 & g_5 & g_4 & g_3 & g_2 & g_1 & 0 \\ 0 & 0 & g_{10} & g_9 & g_8 & g_7 & g_6 & 0 \\ 0 & 0 & 0 & g_5 & g_4 & g_3 & g_2 & g_1 \\ 0 & 0 & 0 & g_{10} & g_9 & g_8 & g_7 & g_6 \end{bmatrix} \begin{bmatrix} 1 \\ t_{32} \\ t_{32}^2 \\ t_{32}^3 \\ t_{32}^4 \\ t_{32}^5 \\ t_{32}^6 \\ t_{32}^7 \end{bmatrix} = 0. \tag{3.32}$$

For a nontrivial solution to exist for Eq. (3.32), the determinant of the square matrix must equal 0. This produces a $32^{nd}$ degree polynomial in $t_{31}$. This equation can be solved for $t_{31}$, and then the values of $p_x$, $p_y$, and $p_z$ are determined by back-substitution. Of the 32 solutions generated by this method, 16 are extraneous which can be shown by checking the 32 solutions against Eqs. (3.17), (3.18), and (3.19). These extraneous solutions occur when $(t_{31}^2 + 1) = 0$.

### 3.3.2 Forward Kinematic Solution for Special Case Manipulator

The solution of the forward kinematics problem for the special case manipulator, where $d = e = 0$, is less complex than the solution for the general case manipulator. The solution can be reduced to a quadratic equation in a single unknown without the use of the dialytic elimination method.

First rewrite Eq. (3.22) with $d = e = 0$:

$$p_x^2 + p_y^2 + p_z^2 + 2\left[c - r - a\cos(\theta_{1i})\right]\left[p_x\cos(\phi_i) + p_y\sin(\phi_i)\right]$$
$$-2a\sin(\theta_{1i})p_z + a^2 + (r-c)^2 - 2a(c-r)\cos(\theta_{1i}) - b^2 = 0, \quad (3.33)$$

for $i = 1, 2$, and 3. This leaves a system of three equations in three unknowns, $p_x, p_y$, and $p_z$. Geometrically, each of these three equations describe a sphere with a radius of $b$, and with a center displaced from the joint $B_i$ by a distance of $(r - c)$, the size difference of the platforms. The solution of this set of equations represents the intersection of those three spheres and also the solution of the forward kinematics problem for the special case manipulator.

The plane that contains the circle of intersection created by the spheres of leg 1 and leg $j$, where $j = 2$ and 3, is found by subtracting Eq. (3.33) for $i = 1$ from Eq. (3.33) for $i = j$:

$$l_{1j}p_x + l_{2j}p_y + l_{3j}p_z + l_{4j} = 0, \quad (3.34)$$

where:

$$l_{1j} = 2\cos(\phi_j)\left[a\cos(\theta_{1j}) + r - c\right] - 2\cos(\phi_1)\left[a\cos(\theta_{11}) + r - c\right],$$
$$l_{2j} = 2\sin(\phi_j)\left[a\cos(\theta_{1j}) + r - c\right] - 2\sin(\phi_1)\left[a\cos(\theta_{11}) + r - c\right],$$
$$l_{3j} = 2a\sin(\theta_{1j}) - 2a\sin(\theta_{11}),$$
$$l_{4j} = \left[a\cos(\theta_{11}) + r - c\right]^2 + a^2\sin^2(\theta_{11}) - \left[a\cos(\theta_{1j}) + r - c\right]^2 - a^2\sin^2(\theta_{1j}).$$

Equation (3.34) for $j = 2$ and 3 provides a system of equations that is linearly independent as long as the centers of the spheres are not colinear, which is unlikely to be realized in practical embodiments of the manipulator. This system of equations defines a line in $\Re^3$ that must contain point P if there is a real

solution. The intersection of this line with any of the spheres described by Eq. (3.33) solves the forward kinematics problem. In this case, solving Eq. (3.34), where $j = 2$ and 3, for $p_y$ and $p_z$ in terms of $p_x$ and then substituting the resulting expressions into Eq. (3.33) for $i = 1$, yields:

$$k_{104}p_x^2 + k_{105}p_x + k_{106} = 0, \tag{3.35}$$

where the constants are defined in Appendix A. The values for $p_y$ and $p_z$ that correspond to $p_x$ are found by back substitution into Eq. (3.34).

### 3.3.3 Numerical Example

As an example of the forward kinematics solution for a general case manipulator, let the manipulator parameters and input angles be as given in Table 3.1.

Table 3.1: Manipulator parameters and input angles for numerical example.

| |
|---|
| $a = 4$ |
| $b = 5.8$ |
| $c = 5$ |
| $d = e = 0.1$ |
| $r = 5$ |
| $\phi_1 = 0$ deg |
| $\phi_2 = 120$ deg |
| $\phi_3 = 240$ deg |
| $\theta_{11} = 10$ deg |
| $\theta_{12} = 45$ deg |
| $\theta_{13} = 35$ deg |

The $32^{nd}$ degree polynomial that results from the determinant of the square matrix in Eq. (3.32) is:

$$
\begin{aligned}
& t_{31}^{32} + 0.0012t_{31}^{31} - 0.7849t_{31}^{30} - 0.0059t_{31}^{29} - 8.7107t_{31}^{28} \\
& -0.0189t_{31}^{27} + 5.6986t_{31}^{26} + 0.0370t_{31}^{25} + 33.2810t_{31}^{24} \\
& +0.1023t_{31}^{23} - 17.5210t_{31}^{22} - 0.0855t_{31}^{21} - 72.6129t_{31}^{20} \\
& -0.2698t_{31}^{19} + 29.5200t_{31}^{18} + 0.0761t_{31}^{17} + 98.6375t_{31}^{16} \\
& +0.3915t_{31}^{15} - 29.3813t_{31}^{14} + 0.0146t_{31}^{13} - 85.1775t_{31}^{12} \\
& -0.3188t_{31}^{11} + 17.2424t_{31}^{10} - 0.0761t_{31}^{9} + 45.5498t_{31}^{8} \\
& +0.1366t_{31}^{7} - 5.5129t_{31}^{6} + 0.0511t_{31}^{5} - 13.7638t_{31}^{4} \\
& -0.0239t_{31}^{3} + 0.7391t_{31}^{2} - 0.0112t_{31} + 1.7965 = 0.
\end{aligned}
$$

The roots of this polynomial are given in Table 3.2.

There are 16 real roots. So, for the given input angles there are 16 possible poses for this manipulator. The 16 extraneous solutions are determined by checking the solutions against the condition that $t_{31} \neq \pm i$, as imposed during the formulation of Eqs. (3.30) and (3.31).

As an example of a real solution, consider the root $t_{31} = 1.218$, so that $\theta_{31} = 101.2$ deg. The angle $\theta_{32}$ is found by back-substituting $t_{31}$ into Eqs. (3.30) and (3.31), yielding $t_{32} = 1.221$ and in turn $\theta_{32} = 101.4$ deg. With $\theta_{31}$ and $\theta_{32}$ known, $p_x$ and $p_y$ can be solved for directly from Eqs. (3.20) and (3.21). In this example, $p_x = 1.971$ and $p_y = -1.131$. Equation (3.22) can then be used to solve for $p_z = 6.245$, completing the forward kinematics solution. The positions of the moving platform for all the solutions of the numerical example were computed in a similar manner, and are given in Table 3.3.

A numerical example of the special case manipulator is also presented. The special case manipulator parameters are selected to be similar to the example presented for the general case manipulator example, where the same parameters are used as shown in Table 3.1 except that the lengths of $d$ and $e$ are included in the upper connecting rod length, $b$. So, $d$ and $e$ both change from a length of 0.1 to 0, and the connecting rod length changes from 5.8 to 6.0, leaving the combined length of each leg the same, as well as maintaining the same proportions between the upper arm assembly and the input link for the two examples. Using these parameters, Eq. (3.35) is solved for the two possible values of $p_x$, and then the associated values for $p_y$ and $p_z$ are found by back substitution into Eq. (3.34). The results are displayed in Table 3.4. It can be observed from Tables 3.3 and 3.4 that the 16 solutions for the general case manipulator tend to cluster around the one of the two solutions for the similar special case manipulator.

Table 3.2: Solutions of $t_{31}$ for the forward kinematics numerical example.

| Solution No. | $t_{31}$ |
|---|---|
| 1 | -1.213 |
| 2 | -1.135 |
| 3 | -1.125 |
| 4 | -1.052 |
| 5 | -1.027 |
| 6 | -0.955 |
| 7 | -0.951 |
| 8 | -0.884 |
| 9-24 | $0.000 \pm 1.000\, i$ (multiplicity 8) |
| 25 | 0.881 |
| 26 | 0.947 |
| 27 | 0.951 |
| 28 | 1.022 |
| 29 | 1.054 |
| 30 | 1.128 |
| 31 | 1.138 |
| 32 | 1.218 |

Table 3.3: Position of the moving platform for all the solutions of the numerical example for a general case manipulator.

| Solution No. | $p_x$ | $p_y$ | $p_z$ |
|---|---|---|---|
| 1 | 2.281 | -1.106 | 5.931 |
| 2 | 2.502 | -0.729 | 6.059 |
| 3 | 2.058 | -0.678 | 5.927 |
| 4 | 2.282 | -0.294 | 6.036 |
| 5 | -0.643 | -0.155 | -2.520 |
| 6 | -0.791 | 0.266 | -2.292 |
| 7 | -0.508 | 0.293 | -2.697 |
| 8 | -0.649 | 0.710 | -2.439 |
| 9-24 | Extraneous | Extraneous | Extraneous |
| 25 | -1.090 | 0.730 | -2.492 |
| 26 | -0.956 | 0.318 | -2.760 |
| 27 | -1.229 | 0.290 | -2.338 |
| 28 | -1.088 | -0.126 | -2.577 |
| 29 | 1.967 | -0.306 | 6.353 |
| 30 | 1.738 | -0.696 | 6.231 |
| 31 | 2.197 | -0.748 | 6.385 |
| 32 | 1.971 | -1.131 | 6.245 |

Table 3.4: Position of the moving platform for all the solutions of the numerical example for a special case manipulator.

| Solution No. | $p_x$ | $p_y$ | $p_z$ |
|---|---|---|---|
| 1 | -0.955 | 0.319 | -2.762 |
| 2 | 2.210 | -0.739 | 6.392 |

## 3.4 Summary

The inverse kinematics problem for this manipulator was defined as a mapping from a given position of the moving platform to the set of joint angles that allows the platform to achieve that position. The solution was presented for both the general case manipulator and the special case manipulator, where axes of the joints at B and C intersect such that $d = e = 0$. The solution for the general case manipulator resulted in four solutions for each leg for a given position of the moving platform, where for each real solution there is a unique posture for the leg. The solution for the special case manipulator also resulted in four solutions. However, the four solutions only produce at most two unique physical posture.

The forward kinematics problem for this manipulator was defined as a mapping from a given set of input joint angles, ($\theta_{11}, \theta_{12}$, and $\theta_{13}$), to the position of the moving platform. The solution was presented for both the general case manipulator and the special case manipulator, where axes of the joints at B and C intersect such that $d = e = 0$.

The solution for the general case manipulator was reduced to a $32^{nd}$ degree polynomial in the half-angle tangent of a single unknown joint angle. This was

accomplished by the application of the diayltic elimination method. Of the 32 solutions suggested by the polynomial, 16 are found to be extraneous, leaving 16 possible solutions for the forward kinematics problem.

The solution for the special case manipulator was reduced to a quadratic polynomial in a single variable. Accordingly, there are only two solutions to the forward kinematics problem for the special case manipulator.

# Chapter 4

# Jacobian and Singularity Analysis

## 4.1 Introduction

For parallel manipulators, the Jacobian matrix provides a transformation from the velocity of the end-effector in cartesian space to the actuated joint velocities (e.g. Gosselin and Angeles, 1988; Wang and Gosselin, 1996) as shown in Eq. (4.1):

$$\dot{\mathbf{q}} = \mathbf{J}\dot{\mathbf{x}}, \tag{4.1}$$

where $\dot{\mathbf{q}}$ is an $m$-dimensional vector that represents a set of actuated joint rates, $\dot{\mathbf{x}}$ is an $n$-dimensional output velocity vector of the end-effector, and $\mathbf{J}$ is the $n \times m$ Jacobian matrix. For the manipulator being considered for this research, the Jacobian matrix is a square matrix since the three actuated joints map onto the three output coordinates of the moving platform. However, it is possible that $m \neq n$. As an example, a redundant manipulator can have more than six actuated joints, while the end-effector will at most have six degrees of freedom, so that $m > n$.

This definition of the Jacobian matrix is slightly different from what is tradi-

tionally defined for serial manipulators, where the Jacobian provides a transformation from the joint velocities to the end-effector velocity. This change of the Jacobian definition for parallel manipulators follows naturally from the dualities between parallel and serial manipulators (Waldron and Hunt, 1988), and is done as a matter of convenience.

As an extension of this definition, Gosselin and Angeles (1990) presented a two-part Jacobian. It assumes that the relationship between the input coordinates, $\mathbf{q}$, and the output coordinates, $\mathbf{x}$, can be written in the following form:

$$\mathbf{F}(\mathbf{q}, \mathbf{x}) = 0, \tag{4.2}$$

where $\mathbf{F}$ is an $n$ dimensional implicit function of $\mathbf{x}$ and $\mathbf{q}$. Differentiating Eq. (4.2) with respect to time results in the following relationship:

$$\mathbf{J}_F \dot{\mathbf{x}} + \mathbf{J}_I \dot{\mathbf{q}} = 0, \tag{4.3}$$

where

$$\mathbf{J}_F = \frac{\partial \mathbf{F}}{\partial \mathbf{x}} \quad , \qquad \mathbf{J}_I = \frac{\partial \mathbf{F}}{\partial \mathbf{q}} \quad , \tag{4.4}$$

and where $\mathbf{J}_F$ is an $n \times n$ Jacobian matrix and $\mathbf{J}_I$ is an $n \times m$ Jacobian matrix. Both $\mathbf{J}_F$ and $\mathbf{J}_I$ are configuration dependent, i.e. $\mathbf{J}_F = \mathbf{J}_F(\mathbf{x}, \mathbf{q})$ and $\mathbf{J}_I = \mathbf{J}_I(\mathbf{x}, \mathbf{q})$. The advantage of the two-part Jacobian is that it allows the identification of various types of singularities.

Independent of which form it takes, the Jacobian provides many useful insights to the performance of a manipulator. The Jacobian matrix is often used for trajectory generation purposes since for a given desired end-effector velocity, it's possible to map that velocity back to the joint space. Jacobian analysis is

also used to determine the singular positions of a manipulator, since when a manipulator is at a singular position the Jacobian matrix is also singular (Gosselin and Angeles, 1990; Gosselin and Sefrioui, 1992; Wang and Gosselin, 1996). Similarly, the Jacobian is used to describe the workspace boundaries of a manipulator (Oblak and Kohli, 1988). Furthermore, the Jacobian is useful for characterizing the stiffness of a manipulator, which provides some insight to the mechanical advantage the end-effector has relative to the actuated joints (Gosselin, 1990a; Tahmasebi and Tsai, 1995). The condition number of the Jacobian matrix has also been used as a performance index for optimizing manipulator design since it provides a measure of the amplification of the error between the actuated joints and the position of the end-effector (Gosselin and Angeles, 1988, 1989, 1991). A recent novel application of the Jacobian has been the analysis and synthesis of under-actuated force generating mechanisms (Gosselin, 1996).

A manipulator singularity describes a manipulator posture that results in an instantaneous change in the mobility of the manipulator. These are undesirable postures, since the control of the manipulator in these postures becomes problematic or the manipulator is at the limit of the workspace. Accordingly, it's important to understand the conditions that result in a manipulator singularity. The identification of those conditions for this manipulator is the focus of this chapter.

Using the classification scheme presented by Gosselin and Angeles (1990), singularities for parallel manipulators can be categorized into three types. The first type occurs when different branches of the inverse kinematics problem converge. This type of singularity results in a loss of mobility, and occurs at the boundary of a manipulator workspace. The second type of singularity occurs

when different branches of the forward kinematics problem converge. This type of singularity results in additional degrees of freedom at the end-effector. So, for a manipulator in this singular posture, the end-effector has one or more degrees of freedom even when the actuated joints are locked. This also means that there are some forces or torques which can be applied to the end-effector that cannot be resisted by the actuators. The third type of of singularity occurs when the manipulator is in a posture that produces a singularity of the first type and a singularity of the second type simultaneously.

Various approaches have been used to determine the singularities of parallel manipulators, but most involve a Jacobian based analysis, since the Jacobian matrix degenerates when the manipulator is in a singular position (Ma and Angeles, 1991, Shi and Fenton, 1992). The approach presented by Gosselin and Angeles (1990) divides the Jacobian into two parts to aid in the classification of the singularities. This approach was also applied to the determination of the singularity loci of a spherical three degree of freedom platform (Gosselin and Serfrioui, 1992) and a spatial four-degree-of-freedom parallel platform (Wang and Gosselin, 1996). A singularity classification system that is not based upon the traditional Jacobian matrix was presented by Zlatanov et. al. (1995), which further refined the classification of manipulator singularities into 6 categories. This classification system was based upon the degenerate cases of a system of linear equations that map both the passive and actuated joint velocities to the output velocities of the manipulator. Merlet (1989) used Grassman geometry to identify the singular conditions for parallel manipulators. Mouly and Merlet (1992) demonstrated this technique for a special case of the Stewart platform with pairs of concentric spherical joints on the moving platform.

In this chapter, for the analysis of the three degree of freedom parallel manipulator, $\mathbf{q}$ is a vector of the actuated joint variables and $\mathbf{x}$ is the position vector of the moving platform:

$$\mathbf{q} = \begin{bmatrix} \theta_{11} \\ \theta_{12} \\ \theta_{13} \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}.$$

The Jacobian matrices are derived by differiniating the loop closure equation for each leg of the manipulator and then solving the resulting system of equations so that it takes the following form:

$$\mathbf{J_I} \begin{bmatrix} \dot{\theta_{11}} \\ \dot{\theta_{12}} \\ \dot{\theta_{13}} \end{bmatrix} = \mathbf{J_F} \begin{bmatrix} V_{p,x} \\ V_{p,y} \\ V_{p,z} \end{bmatrix},$$

where $V_{p,x}, V_{p,y}$, and $V_{p,z}$ are the $X, Y$, and $Z$ components of the velocity of point $P$ on the moving platform in the world coordinate frame (see Fig. 2.1), and where $\mathbf{J_I}$ and $\mathbf{J_F}$ are $3 \times 3$ Jacobian matrices. The Jacobian matrices are in terms of the manipulator joint angles, and are accordingly dependent upon the position and posture of the manipulator. The singularities of the manipulator are then found by examining the conditions that result in singular Jacobian matrices.

## 4.2 Derivation of Jacobian Matrix

Referring to Fig. 4.1 that shows the $i^{th}$ leg of the manipulator with links of the leg numbered 1 through 4, the first step in deriving the Jacobian matrices is to

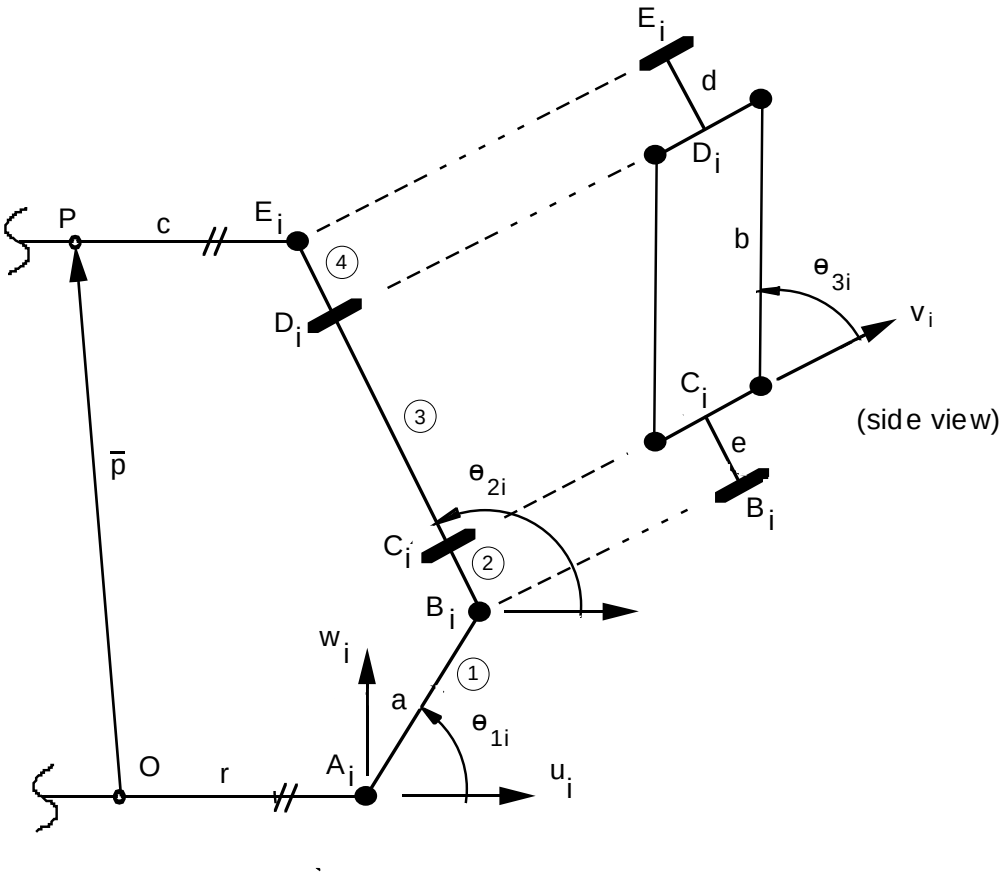



Figure 4.1: Description of leg $i$ used for derivation of the Jacobian.

write a loop closure equation for the $i^{th}$ leg in the UVW frame:

$$\overline{OA_i} + \overline{AB_i} = \overline{OP_i} + \overline{PE_i} + \overline{ED_i} + \overline{DC_i} + \overline{CB_i}. \tag{4.5}$$

Differiniating Eq. (4.5) with respects to time and expressing the resulting equation in the $UVW$ coordinate frame provides:

$$\overline{\omega}_{1i} \times \overline{AB_i} = \overline{V}_{P,uvw} + \overline{\omega}_{3i} \times \overline{DC_i} + \overline{\omega}_{2i} \times \left(\overline{ED_i} + \overline{CB_i}\right), \tag{4.6}$$

where $\omega_{ni}$ is the angular velocity of the $n^{th}$ link of the $i^{th}$ leg in the $UVW$ coordinate frame, and is given by:

$$\overline{\omega}_{1i} = \begin{bmatrix} 0 \\ -\dot{\theta}_{1i} \\ 0 \end{bmatrix}, \quad \overline{\omega}_{2i} = \begin{bmatrix} 0 \\ -\dot{\theta}_{2i} \\ 0 \end{bmatrix}, \quad \text{and} \quad \overline{\omega}_{3i} = \begin{bmatrix} \dot{\theta}_{3i}\sin(\theta_{2i}) \\ -\dot{\theta}_{2i} \\ -\dot{\theta}_{3i}\cos(\theta_{2i}) \end{bmatrix}.$$

Rewriting Eq. (4.6) in terms of the link parameters and joint angles results in:

$$\begin{bmatrix} -a\dot{\theta}_{1i}\sin(\theta_{1i}) \\ 0 \\ a\dot{\theta}_{1i}\cos(\theta_{1i}) \end{bmatrix} = \begin{bmatrix} V_{p,u} - b\dot{\theta}_{3i}\cos(\theta_{2i})\cos(\theta_{3i}) + \dot{\theta}_{2i}\sin(\theta_{2i})\left[d + e + b\sin(\theta_{3i})\right] \\ V_{p,v} + b\dot{\theta}_{3i}\sin(\theta_{3i}) \\ V_{p,w} - b\dot{\theta}_{3i}\sin(\theta_{2i})\cos(\theta_{3i}) - \dot{\theta}_{2i}\cos(\theta_{2i})\left[d + e + b\sin(\theta_{3i})\right] \end{bmatrix}. \tag{4.7}$$

The Jacobian matrices are determined by considering Eq. (4.7) as a system of three equations in three unknowns, ($\dot{\theta}_{1i}, \dot{\theta}_{2i}$, and $\dot{\theta}_{3i}$) and then solving for a relationship in just $\dot{\theta}_{1i}$ for the $i^{th}$ leg. This is accomplished by first solving the $2^{nd}$ row of Eq. (4.7) for $\dot{\theta}_{3i}$:

$$\dot{\theta}_{3i} = -\frac{V_{p,v}}{b\sin(\theta_{3i})}. \tag{4.8}$$

This expression for $\dot{\theta}_{3i}$ is substituted into the $1^{st}$ and $2^{nd}$ rows of Eq. (4.7), leaving two equations in two unknowns, ($\dot{\theta}_{1i}$ and $\dot{\theta}_{2i}$). These two equations are combined so as to eliminate $\dot{\theta}_{2i}$, and simplified to produce Eq. (4.9):

$$\begin{aligned} a\dot{\theta}_{1i}\sin(\theta_{2i} - \theta_{1i})\sin(\theta_{3i}) = {} & V_{p,u}\cos(\theta_{2i})\sin(\theta_{3i}) \\ & + V_{p,v}\cos(\theta_{3i}) + V_{p,w}\sin(\theta_{2i})\sin(\theta_{3i}). \end{aligned} \tag{4.9}$$

Equation (4.9) is transformed from the UVW coordinate frame for leg $i$ to the $XYZ$ frame of the manipulator using the following relationship:

$$\begin{bmatrix} V_{p,ui} \\ V_{p,vi} \\ V_{p,wi} \end{bmatrix} = \begin{bmatrix} \cos(\phi_i) & \sin(\phi_i) & 0 \\ -\sin(\phi_i) & \cos(\phi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_{p,x} \\ V_{p,y} \\ V_{p,z} \end{bmatrix}. \tag{4.10}$$

This transformation is repeated for each leg, and the resulting equations are rearranged so as to produce the Jacobian matrix:

$$\mathbf{J_I} \begin{bmatrix} \dot{\theta}_{11} \\ \dot{\theta}_{12} \\ \dot{\theta}_{13} \end{bmatrix} = \mathbf{J_F} \begin{bmatrix} V_{p,x} \\ V_{p,y} \\ V_{p,z} \end{bmatrix}, \tag{4.11}$$

where:

$$\mathbf{J}_F = \begin{bmatrix} j_{F_{11}} & j_{F_{12}} & j_{F_{13}} \\ j_{F_{21}} & j_{F_{22}} & j_{F_{23}} \\ j_{F_{31}} & j_{F_{32}} & j_{F_{33}} \end{bmatrix}, \quad \mathbf{J}_I = \mathrm{diag}(j_{I_1}, j_{I_2}, j_{I_3}), \tag{4.12}$$

and

$$\begin{aligned} j_{F_{i1}} &= \cos(\theta_{2i})\sin(\theta_{3i})\cos(\phi_i) - \cos(\theta_{3i})\sin(\phi_i), \\ j_{F_{i2}} &= \cos(\theta_{3i})\cos(\phi_i) + \cos(\theta_{2i})\sin(\theta_{3i})\sin(\phi_i), \\ j_{F_{i3}} &= \sin(\theta_{2i})\sin(\theta_{3i}), \\ j_{I_i} &= a\sin(\theta_{2i} - \theta_{1i})\sin(\theta_{3i}) \end{aligned}$$

for $i = 1, 2$, and 3.

Sometimes the Jacobian matrix for a parallel manipulator is not dimensionally uniform, and this creates some problems when using the Jacobian matrix for other types of analysis such examining the stiffness or condition of the workspace

of a manipulator. However, in the case of this manipulator, if the Jacobian matrix is considered as in the form given by Eq. (4.1), where the Jacobian matrix can be calculated by

$$\mathbf{J} = \mathbf{J_I}^{-1}\mathbf{J_F}, \tag{4.13}$$

then the Jacobian matrix is dimensionally uniform with each element of $\mathbf{J}$ having the dimension of $(\text{length})^{-1}$. This occurs because the manipulator has only translational degrees of freedom at the moving platform and all the actuated joints are revolute joints. Hence, all the elements of the output vector, $\mathbf{x}$, for this manipulator have the dimension of length and all of the elements of the input vector, $\mathbf{q}$, have the dimension of radian, resulting in a Jacobian matrix that is dimensionally uniform.

## 4.3 Inverse Kinematic Singularities

Using the Jacobian matrices as given in Eq. (4.11), the inverse kinematic singularities occur when the following condition is satisfied:

$$\det(\mathbf{J}_I) = 0. \tag{4.14}$$

When Eq. (4.14) is satisfied the manipulator loses one or more degrees of freedom. If $\mathbf{J}_I$ is singular and the dimension of the null-space of $\mathbf{J}_I$ is nonzero, then there exist some nonzero $\dot{\mathbf{q}}$ vectors that produce $\dot{\mathbf{x}}$ vectors that are zero in some directions. Hence, there are moving platform velocities that are not achievable. This occurs when the manipulator is at a workspace boundary where different solutions to the inverse kinematics solution converge.

This type of singularity is found by setting the diagonal elements of $\mathbf{J}_I$ equal to zero. This leads to the following conditions for an inverse kinematic singularity:

$$(\theta_{1i} - \theta_{2i}) = 0 \text{ or } \pi \tag{4.15}$$

or

$$\theta_{3i} = 0 \text{ or } \pi, \tag{4.16}$$

for $i$=1 or 2 or 3. So, whenever any of these conditions are satisfied for any leg, the manipulator is in an inverse kinematic singular posture, and is at the limit of the manipulator workspace. Physically, this condition occurs when the upper arm linkage and the input link of a leg are in the same plane or when all the links of the four bar linkage that comprise the upper arm for a leg are co-linear.

## 4.4 Forward Kinematic Singularities

When a forward kinematic singularity occurs, the moving platform of the manipulator gains one or more degrees of freedom, so that the moving platform still has some mobility even when all of the actuators are locked. Forward kinematic singularities occur when the following condition is satisfied:

$$\det(\mathbf{J}_F) = 0. \tag{4.17}$$

To understand why this is the case, examine Eq. (4.11). If the dimension of the nullspace for $\mathbf{J}_F$ is non-zero, there are some moving platform velocities, $\dot{\mathbf{x}}$ that are possible even when $\dot{\theta}$ is zero. Hence the moving platform is still mobile while the actuated joints are locked. Moreover, this shows that there are forces that

can be applied to the moving platform that are not resisted by the actuators when the platform is in this type of singular posture.

Accordingly, Eq. (4.17) provides the necessary and sufficient conditions for a forward kinematic singularity. Now, we consider some manipulator postures that satisfy this condition.

The first set of postures that satisfies Eq. (4.17) is inferred by imposing linear dependence on the columns of $\mathbf{J}_F$, such that

$$\alpha_1 \begin{bmatrix} j_{F_{11}} \\ j_{F_{21}} \\ j_{F_{31}} \end{bmatrix} + \alpha_2 \begin{bmatrix} j_{F_{12}} \\ j_{F_{22}} \\ j_{F_{32}} \end{bmatrix} + \alpha_3 \begin{bmatrix} j_{F_{13}} \\ j_{F_{23}} \\ j_{F_{33}} \end{bmatrix} = 0, \tag{4.18}$$

for some real values of $\alpha_1, \alpha_2$, and $\alpha_3$, where not all $\alpha$'s are zero. By examination, one set of conditions that satisfies Eq. (4.18) is found when $j_{F_{13}} = j_{F_{23}} = j_{F_{33}} = 0$. This satisfies Eq. (4.18), since any value of $\alpha_3$ could be chosen, while letting $\alpha_1 = \alpha_2 = 0$. In terms of manipulator variables, this condition is rewritten as:

$$\sin(\theta_{21})\sin(\theta_{31}) = \sin(\theta_{22})\sin(\theta_{32}) = \sin(\theta_{23})\sin(\theta_{33}) = 0. \tag{4.19}$$

Equation (4.19) shows that the manipulator is in a forward kinematic singular position whenever all legs are in a posture such that:

$$\theta_{2i} = 0 \text{ or } \pi$$

or

$$\theta_{3i} = 0 \text{ or } \pi.$$

for all $i = 1$ and 2 and 3.

The physical interpretation of this condition is that the manipulator displays this type of forward kinematic singular position any time the four-bar linkages

of all three legs are in the same plane as the moving platform. In this manipulator configuration, the manipulator actuators cannot resist a force applied to the moving platform in the z-direction. The front and top views of a manipulator in this type of singular configuration are shown in Fig. 4.2. In this example the upper arm assemblies of all three legs are in the same plane as the moving platform, resulting in a forward kinematic singularity. This example configuration is only possible when $r + a \geq b + c + d + e$.

A second set of postures that satisfies Eq. (4.17) is found when any two of the upper arm linkages are parallel. To show this, a unit vector in the direction of the upper arm, $\overline{CD}$, for the $i^{th}$ leg is defined as $\bar{n}_{CD,i}$, where:

$$\bar{n}_{CD,i} = \begin{bmatrix} \cos(\phi_i)\sin(\theta_{3,i})\cos(\theta_{2,i}) - \sin(\phi_i)\cos(\theta_{3,i}) \\ \sin(\phi_i)\sin(\theta_{3,i})\cos(\theta_{2,i}) + \cos(\phi_i)\cos(\theta_{3,i}) \\ \sin(\theta_{3,i})\sin(\theta_{2,i}) \end{bmatrix} \tag{4.20}$$

If any two upper arm linkages are parallel, then

$$\bar{n}_{CD,i} = \pm\bar{n}_{CD,i+1}. \tag{4.21}$$

Substituting the relationships that result from Eq. (4.21) into Eq. (4.12) reveals that two of the rows of $\mathbf{J}_F$ are multiples of each other when two of the upper arm linkages are parallel. Hence the matrix is singular and the manipulator is in a singular position of the forward kinematic type when any two of the upper arm linkages are parallel to each other. As an example, the front and top views of a manipulator in this type of singular configuration are shown in Fig. 4.3. In this example, the manipulator cannot resist any force applied in the plane of the moving platform. Note that in this example all three upper arm linkages are parallel, however it is only necessary that two of the upper arm linkages be parallel to be in a singular position of this type.

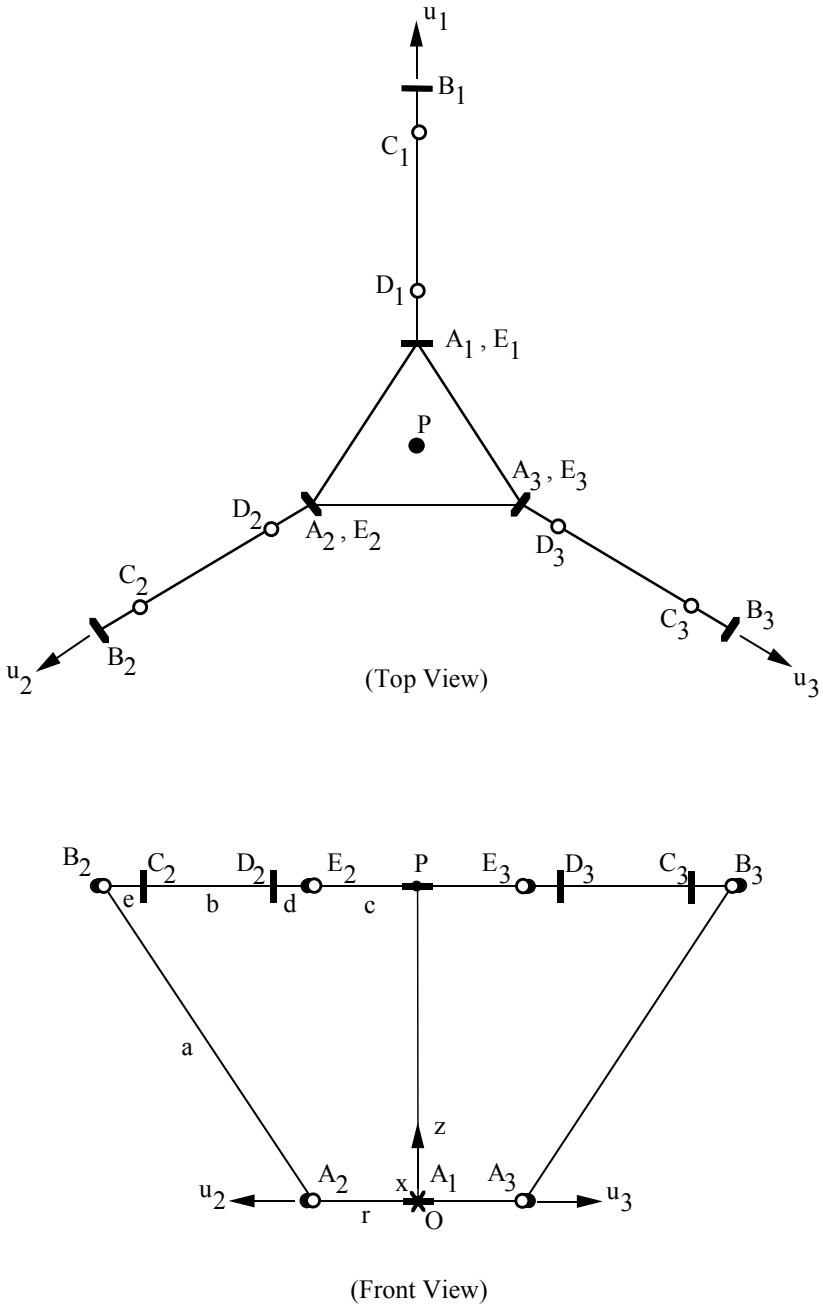



Figure 4.2: Example of forward kinematic singular configuration where all the upper arm linkages are in the plane of the moving platform.

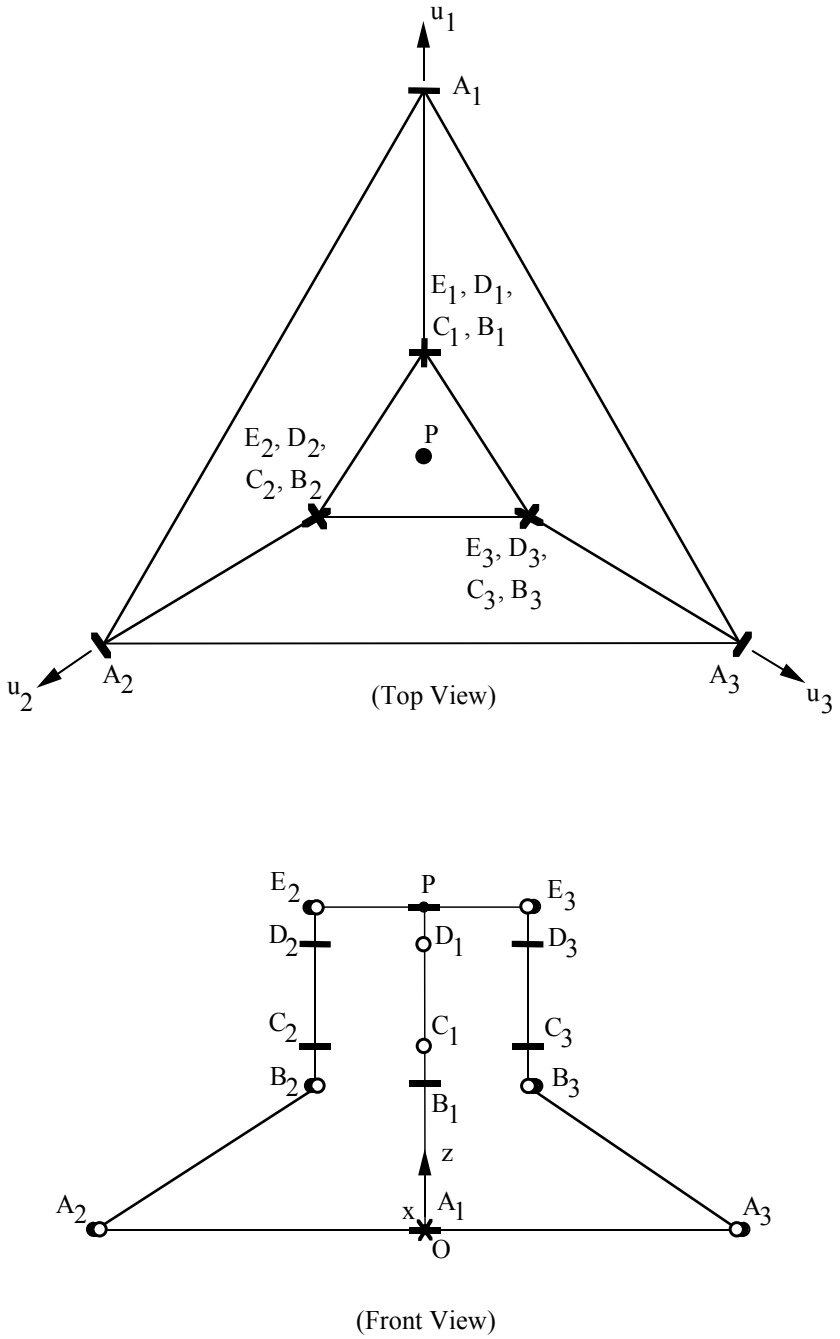



Figure 4.3: Example of a forward kinematic singular configuration where two upper arm linkages are parallel.

## 4.5 Summary

The Jacobian matrix is derived for the general case manipulator. The derivation of the Jacobian matrix for the special case manipulator, where $d = e = 0$, is not treated separately since the derivation for the special case manipulator results in the identical Jacobian matrix as given for the general case manipulator. Furthermore, the Jacobian matrix given by Eq. (4.13) for this manipulator is dimensionally uniform. Accordingly, there is no need to establish a set of generalized coordinates for the manipulator to create a dimensionally uniform Jacobian matrix.

Several singular positions of the manipulator are found by examining the conditions that cause either of the Jacobian matrices as given by Eq. (4.11) to be singular. These singular positions were classified into inverse kinematic singular positions and forward kinematic singular positions. The inverse kinematic singular positions occur at the boundary of the manipulator workspace and result in an instantaneous reduction of the mobility of the manipulator. The forward kinematic singular positions occur at the convergence of different branches of the forward kinematic solution and result in additional mobility at the moving platform even when all the input links are locked in place. This search for singular conditions is not exhaustive, but does account for all the singular conditions observed during the course of this research.

Two conditions were found that result in inverse kinematic singular positions. The first is when the input link and the upper arm linkage for any leg is in the same plane. The second condition occurs when all the links of the four bar linkage that comprise the upper arm for a leg are co-linear.

Two conditions were also found that result in forward kinematic singular

positions. The first occurs when all the upper arm linkages are in the same plane as the moving platform. The second occurs when any two of the upper arm linkages are parallel to each other.

# Chapter 5

# Workspace Analysis and Optimization

## 5.1 Introduction

Parallel manipulators offer several advantages relative to serial manipulators, but generally these gains are realized at the cost of manipulator workspace. Consequently, it's incumbent upon the engineer to design the manipulator with the optimization of workspace in mind. This chapter addresses workspace oriented manipulator design by considering two optimization studies. First, the optimization of total manipulator workspace volume is considered without regard to the condition of the workspace. Second,the optimization of a manipulator to obtained a well-conditioned workspace is considered.

Optimization of manipulator workspace volume is dependent upon a means of determining the workspace of a parallel manipulator for a given set of design variables. Oblak and Kohi (1988) described manipulator workspace from the standpoint of a Jacobian analysis and D-surfaces, where one or more of the joints achieve a limit position. A geometric based algorithm to generate a graphical representation of the workspace for a six degree of freedom parallel plat-

form in a given orientation was generated by Gosselin et al (1992) and Gosselin (1990b). The workspace of the DELTA4 robot, a 4 degree of freedom parallel manipulator, was presented by Clavel (1988), where the DELTA4 workspace was described by the intersection of geometric primitives that are defined by various robot design parameters. A similar approach was applied to the DELTA4 by Sternheim (1988), where the workspace described by the intersection of the geometric primitives was visualized using solid modeling software. Rastegar and Perel (1988) and Alciatore and Ng (1994) applied the Monte Carlo method to determine workspace boundaries. For this research, a numerical value for the workspace volume is computed using the Monte Carlo method.

A parallel manipulator designed for maximum workspace volume may not however be the optimal design for practical applications. It's possible that a parallel manipulator that is optimized for total workspace will result in a manipulator with undesirable kinematic characteristics such as poor dexterity or manipulability. One measure of these characteristics used by Salisbury and Craig (1982) and Angeles and Lopez-Cajun (1988) is based upon the condition number of the manipulator Jacobian matrix, where the Jacobian matrix maps the actuated joint velocities to the velocity of the moving platform in cartesian space. Merlet (1996) explored the optimal design of a Gough platform with an objective function that considered the total workspace volume and the positioning error of the moving platform at several discrete locations within the workspace so as to characterize the worst positioning error within the workspace. A summary of other manipulator dexterity characterizations is presented by Klein and Blaho (1987).

The condition of the manipulator in a local sense was considered by Gosselin

and Angeles when they examined the optimization of a spherical three degree of freedom parallel manipulator (1989) and a planar three degree of freedom parallel manipulator (1988). The criteria they used to evaluate the manipulator designs were symmetry, global workspace, and the condition of the Jacobian of the manipulator at a home position. Global performance indices, that consider the dexterity of the manipulator over the entire workspace, were developed by Park and Brockett (1994) and Gosselin and Angeles (1991). The global performance index developed by Gosselin and Angeles is based upon the integration of the reciprocal of the condition number over the entire workspace. A similar approach is taken for this research.

## 5.2 Determination of Workspace Volume

The manipulator workspace volume can be calculated using various procedures. One obvious approach is to integrate a differential volume over the entire workspace. However, the complexity of establishing the limits of integration for a general case manipulator with this approach compels the use of a different numerical technique. For this research the manipulator workspace volume, $W$, is numerically approximated using the Monte Carlo method, as outlined in Table 5.1. This procedure produces a numeric value that is used for the optimization of the total workspace of the manipulator.

Table 5.1: Procedure to determine manipulator workspace volume using the Monte Carlo method.

| | |
|---|---|
| Step 1: | A hemisphere with a radius equal to that of the total leg length of the manipulator, $a + b + d + e$, is defined that encases the entire possible workspace of the manipulator. |
| Step 2: | A large number of points, $n_{total}$, are randomly selected within the hemisphere. |
| Step 3: | Each point is tested to determine if it falls within the manipulator workspace. This is accomplished by solving the inverse kinematics problem for each leg as described by Eqs. (3.5) and (3.9). If all the joints angles are real, then the point is within the workspace. |
| Step 4: | The number of points that fall within the workspace, $n_{in}$, is tallied. |
| Step 5: | The workspace volume is estimated by multiplying the volume of the hemisphere by the ratio of points that fall within the workspace to the total number of points selected: $$W \approx \pi(a + b + d + e)^3 \frac{2n_{in}}{3n_{total}}.$$ |

## 5.3 Optimization for Total Workspace Volume

The objective of the total workspace optimization is to determine the values of the manipulator design variables that result in the largest total manipulator workspace. The design variables considered are:

- the leg link lengths, $a, b$, and $(d + e)$;
- the relative size of the platforms, $(c - r)$;
- the relative angular position of the legs, $\phi_2$ and $\phi_3$, where leg 1 is assumed to align with the $X$ axis so that $\phi_1 = 0$.

Note that the sum of the two offsets, $d + e$, is treated as a single design variable as shown in Eqs. (3.5) and (3.9). Similarly the relative size of the platforms, $(c - r)$, is considered as a single design variable.

In order to bound the solution and to ensure a practical realization, the objective function is subject to the following constraints:

- the total leg length is not to exceed one, $a + b + d + e \leq 1$;
- each leg must have an angular separation of at least $5^\circ$ from each of the other legs;
- all link lengths must be positive.

Given this problem formulation, the optimization is computed using the Matlab optimization toolbox and produced the following results: $a = .4$, $b = .6$, $d = e = 0$, $(c - r) = 0$, $\phi_2 = 5^\circ$, and $\phi_3 = 355^\circ$. Both the angular leg separation constraint and the total leg length constraints are active. A plot of the manipulator workspace with these design variables is shown in Fig. 5.1. Note that the

optimization routine drove the values of $\phi_2$ and $\phi_3$ to the edge of the allowable design variable space. That is, the angles $\phi$ were driven so that the legs are each separated by 5°. This is because the volume of the manipulator workspace is the intersection of three torii, which reaches a maximum when they align.

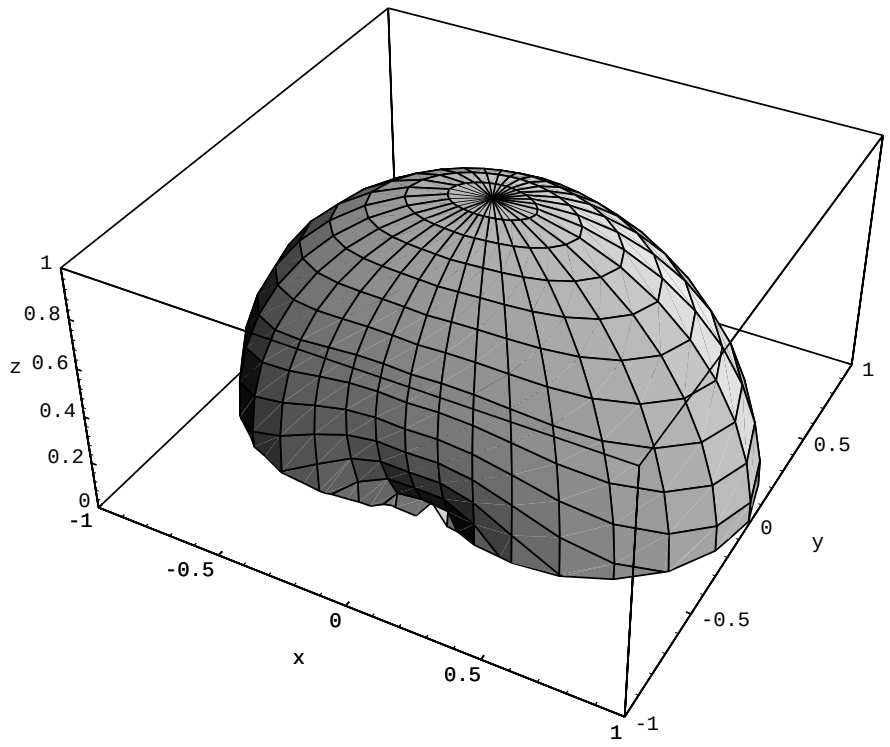



Figure 5.1: Workspace of manipulator designed for maximum workspace volume.

## 5.4 Determination of Workspace Condition

A global condition index, $\eta$, that considers the condition number of the Jacobian over the entire workspace is defined for the manipulator as:

$$\eta = \int_W \frac{1}{\lambda}\, dW, \tag{5.1}$$

where $\lambda$ is the condition number of the Jacobian at a given position in the workspace and $W$ is the manipulator workspace.

The condition number of the Jacobian matrix, $J$, is defined as:

$$\lambda = \| J \| \| J^{-1} \|, \tag{5.2}$$

where $\| \cdot \|$ is the 2 norm of the matrix.

As with the calculation of the workspace volume, the complexity of generating a closed-form solution for the global condition index as given by Eq. (5.1), compels the use of a numerical solution technique. Accordingly, the Monte Carlo method is employed as outlined in Table 5.2.

## 5.5 Optimization for Well Conditioned Workspace Volume

The objective of the well-conditioned workspace optimization is to determine the values of the manipulator design variables that result in the best global condition index. The same set of design variables that were used during the total workspace optimization is also used for the well conditioned workspace optimization. The objective function is also subject to the same constraints as were used during the total workspace optimization. The well-conditioned workspace optimization is computed using the Matlab optimization toolbox and produced the following results: $a = .44$, $b = .56$, $d = e = 0$, $(c - r) = 0$, $\phi_2 = 120°$, and $\phi_3 = 240°$ when 200,000 points were used for the Monte Carlo method. The only active constraint is the total leg length constraint. A plot of the manipulator workspace with these design variables is shown in Fig. 5.2.

Table 5.2: Procedure to estimate global condition index of the manipulator.

| | |
|---|---|
| Steps 1-3: | Same as steps 1-3 for workspace estimation in Table 5.1. |
| Step 4: | The condition index sum, $S$, which is the sum of the reciprocal of the condition number of each point that falls within the workspace, is calculated by $S = \sum_i \frac{1}{\lambda_i}$, for the $i$ points that fall within the workspace. |
| Step 5: | The global condition index, $\eta$, is determined by multiplying the volume of the hemisphere and the condition index sum, and then dividing by the total number of points selected, i.e.: $$\eta = \frac{2\pi(a+b+d+e)^3 S}{3n_{total}}$$ |

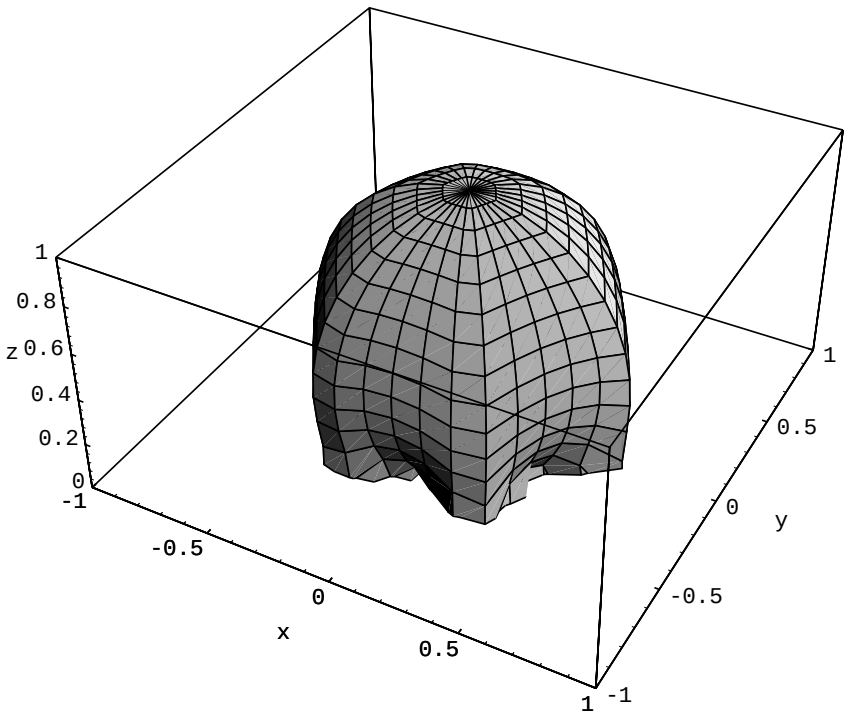



Figure 5.2: Workspace of manipulator designed for maximum global condition index.

The impact of the design on the condition of the workspace is significant, and can be seen in Figs. 5.3 and 5.4, where the condition number is plotted across a plane of the workspace at $z$=.5 for both the manipulator optimized for total workspace and the manipulator optimized for well conditioned workspace. Figure 5.3 shows that the workspace is ill conditioned for the manipulator that

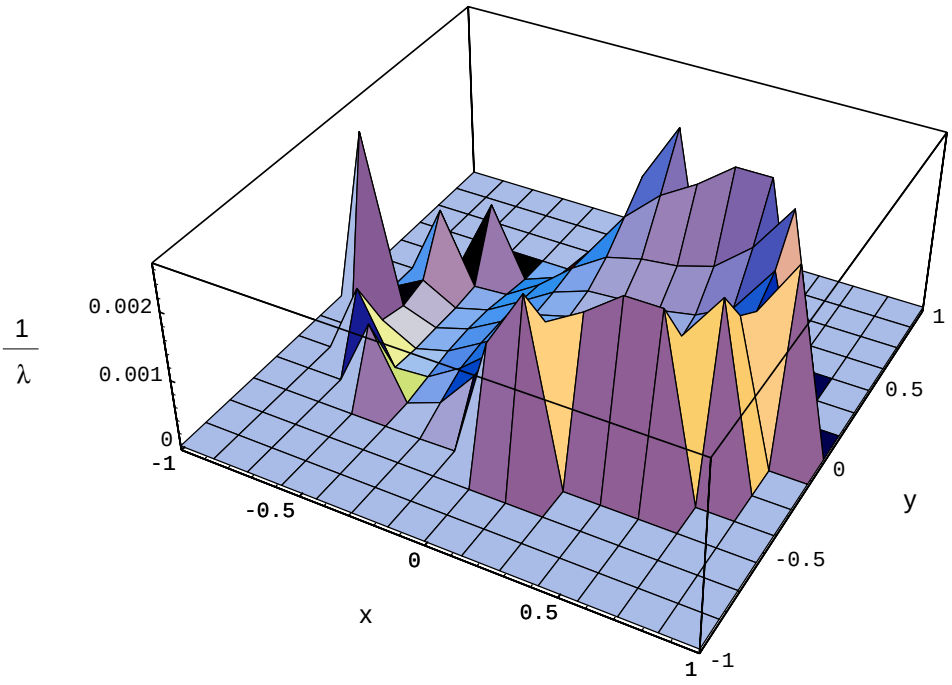



Figure 5.3: Reciprocal of the condition number at the $z$=.5 plane for the total workspace optimized manipulator.

is optimized for total workspace volume with a maximum condition index, $\frac{1}{\lambda}$ of $\sim 0.002$ in the $z$=.5 plane. This results in a poor manipulator since positioning errors at the actuator are significantly magnified at the end effector. Whereas Fig. 5.4 shows a much better conditioned workspace with a maximum condition condition index of $\sim 0.6$ in the $z$=.5 plane, suggesting that this is the better design when considering the positioning performance of the manipulator.

It's also interesting to note that when the position of the legs about the platform is constrained to be symmetrical, so that $\phi_2 = 120°$ and $\phi_3 = 240°$, the results obtained from the total workspace volume optimization still differ significantly from the results obtained from the global condition index optimization. The link lengths produced are $a = .32,\ b = .68,\ (d + e) = 0,$ and $(c - r) = 0$ when the manipulator is optimized for total workspace with legs that are constrained to be symmetrically located about the platform. The workspace of a manipulator with these design parameters is illustrated in Fig. 5.5, and the reciprocal of the condition number across a plane of the workspace at $z = .5$ for such a manipulator is shown in Fig. 5.6. The maximum condition index is approximately equal to 0.3 as compared to 0.6 obtained from the well-conditioned workspace optimization.

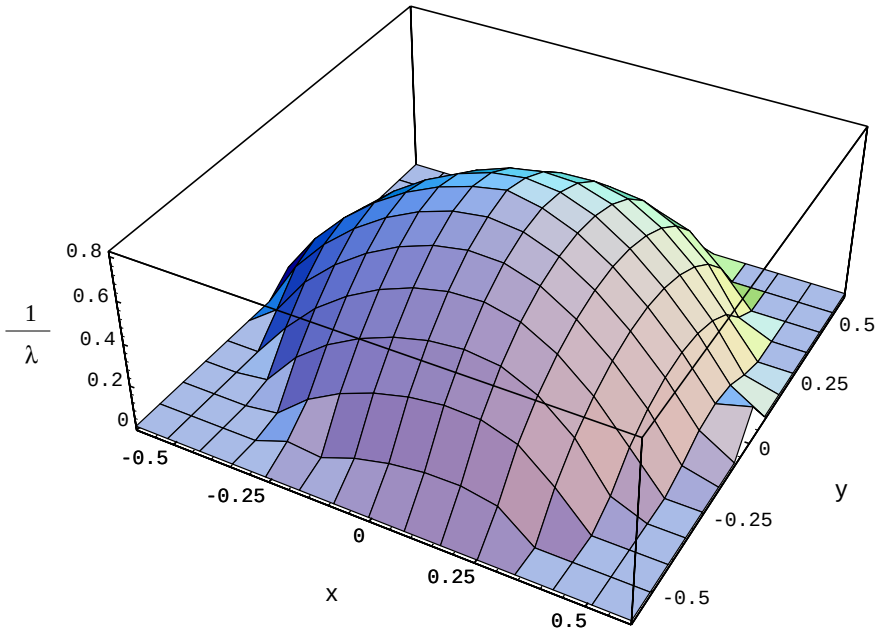



Figure 5.4: Reciprocal of the condition number at the $z$=.5 plane for the global condition index optimized manipulator.

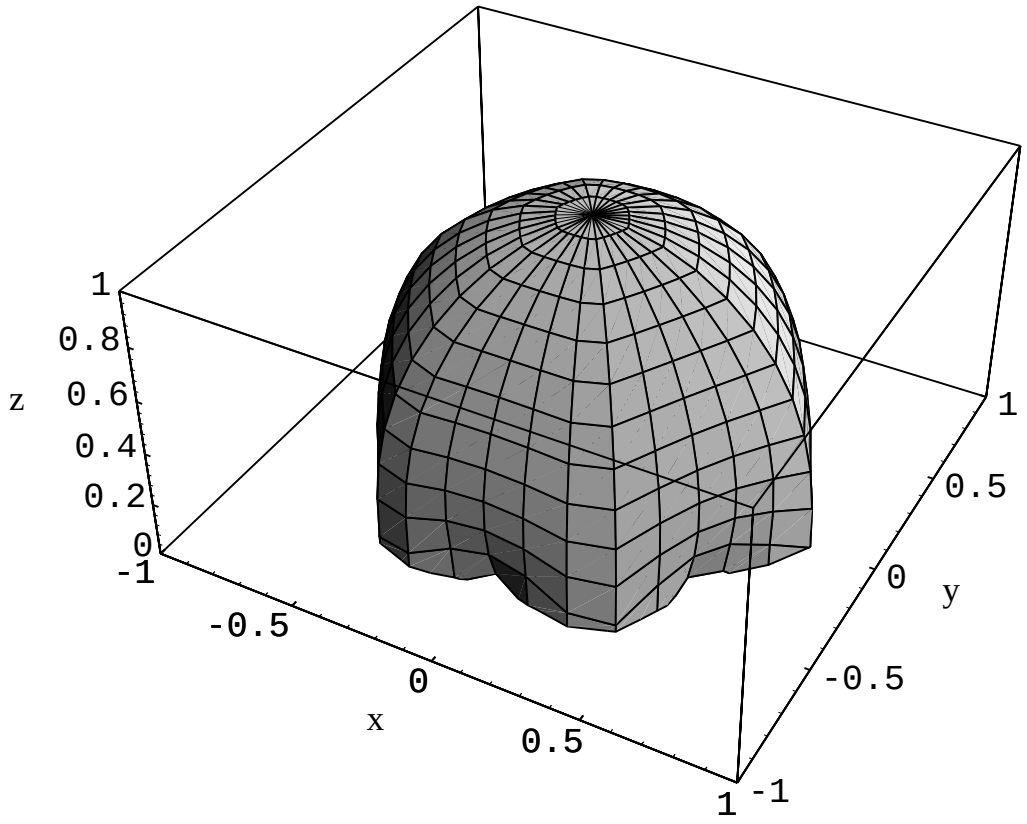



Figure 5.5: Workspace of a manipulator designed for maximum total workspace volume with an additional constraint that requires the legs to be separated by 120 degrees.

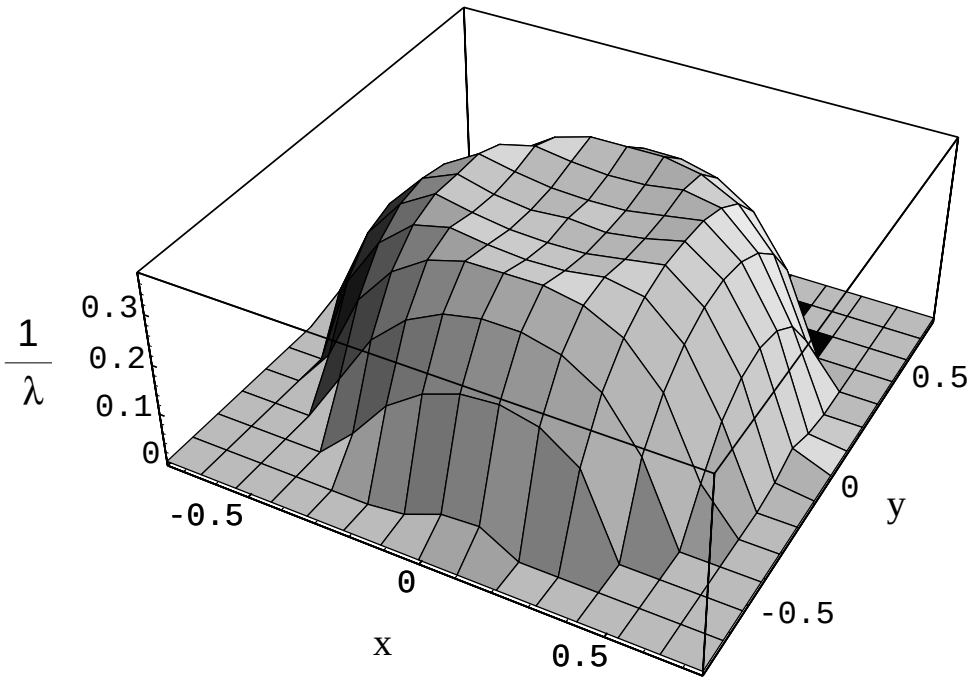



Figure 5.6: Reciprocal of the condition number at the $z = .5$ plane for a manipulator with 120 degree leg separation that is optimized for total workspace volume.

## 5.6 Summary

Two procedures based upon the Monte Carlo method are presented to determine the volume and global condition index for the manipulator workspace. The global condition index provides a characterization of the quality of manipulator workspace and is a function of the condition number of the Jacobian matrix. These procedures are then applied to the optimization of the manipulator design parameters for both total workspace and global conditioning index.

The optimization of the total workspace shows that the lower leg of the manipulator should comprise 40% of the total leg length and the upper arm parallelogram should should comprise the remaining 60% of the total leg length, while the links that provide the offset between joints B and C and also between D and E should have a zero length. Furthermore, the legs should also be at the smallest angular offset possible.

The optimization of the manipulator design for the global condition index results in a manipulator where the lower leg comprises 44% of the total leg length and the upper arm comprises 56% of the total leg length, while each leg has an angular separation of 120°. The global condition index is a function that considers the condition number of the manipulator Jacobian over the entire workspace.

These results show that a parallel manipulator of this type that is designed to optimized total workspace volume is significantly different from one that is optimized for a well conditioned workspace. Moreover, these results show that a manipulator of this type that is designed to maximize total workspace volume will result in an ill conditioned workspace.

# Chapter 6

# Analysis of Manipulator Dynamics

## 6.1 Introduction

The dynamics of the manipulator are considered in this chapter, where for a given trajectory of the moving platform it is desired to determine the torques applied at the input links by the actuators that will achieve that trajectory. An understanding of the manipulator dynamics is important from several different perspectives. First, it is necessary to properly size the actuators and other manipulator components. Without a model of the manipulator dynamics, it becomes difficult to predict the actuator torque requirements and in turn equally difficult to properly select the actuators. Second, a dynamics model is useful for developing a control scheme. With an understanding of the manipulator dynamics, it is possible to design a controller with better performance characteristics than would typically be found using heuristic methods after the manipulator has been constructed. Moreover, some control schemes such as the computed torque controller rely directly on the dynamics model to predict the desired actuator torque to be used in a feedforward manner.

Several approaches have been used to characterize the dynamics of parallel manipulators. The most common approaches are based upon Lagrange formulations (Miller and Clavel, 1992), the application of Hamilton's Principle (Miller, 1995), and the direct application of the Newton's equations of motion (Guglielmetti and Longchamp, 1994).

For this research, three approaches are developed with each approach disregarding friction at the passive joints. The first approach is based upon a simplified dynamics model and the direct application of Newton-Euler equations of motion. This approach assumes that the inertial and gravitational loading of the moving platform and a portion of the upper arm assembly can be mapped back to the actuated joint using the Jacobian matrix. The advantage of this approach is that it is less computationally intensive than the Lagrangian models, while still capturing the character of the manipulator dynamics. This is desirable if the dynamics model is to be used as the basis for a computed torque control scheme.

The second approach is based upon Lagrangian multipliers (Griffiths, 1985). This approach is selected to give a more complete characterization of the manipulator dynamics, with the most significant assumptions being that the mass of each long connecting rod of the upper arms is evenly divided and concentrated at the two joints on the rods and that the short connecting links between $BC$ and $DE$ are massless.

The third approach is developed for use with a proportional-integral-derivative (PID) type control. Accordingly, an uncoupled equation of motion is developed for each individual input link and linearized about a given operating condition.

The assumptions made for each of these approaches are discussed in more

detail during the development of the models. Results of the three different dynamics models are compared using a numerical example.

## 6.2 Direct Application of Newton-Euler Equations of Motion for Simplified Manipulator

This model of the manipulator dynamics assumes that the fixed base of the manipulator is mounted above the workspace where it operates such that gravity is acting in the positive z-direction as shown in Fig. 6.1. This dynamics model also assumes that the mass of each connecting rod, $m_b$, in the upper arm assembly is evenly divided between, and concentrated at, joints $B_i$ and $E_i$. This assumption can be made without significantly compromising the accuracy of the model since the concentrated mass model of the connecting rods does capture some of the dynamics of the rods. Moreover, the connecting rods should not play a dominant role in the manipulator dynamics because the connecting rods can be made quite light relative to the rest of the manipulator since these connecting rods don't need to carry any of the load associated with the mass of the actuators.

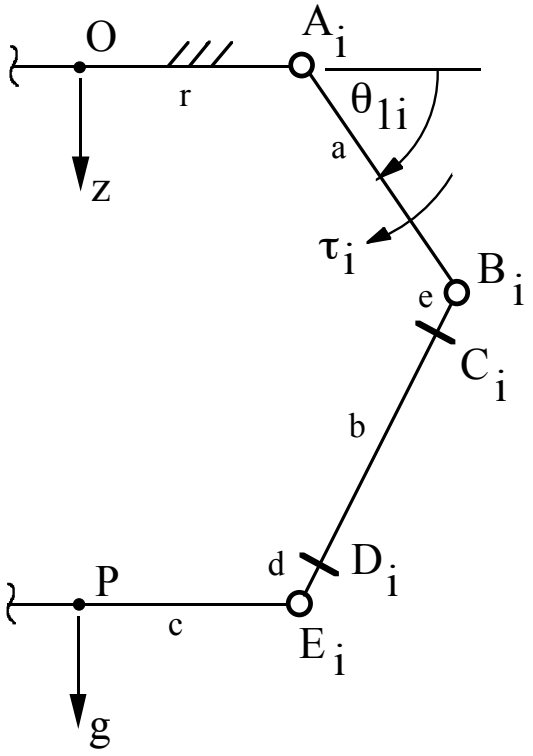



Figure 6.1: Schematic of the $i^{th}$ leg, showing the orientation of the manipulator and the direction of the applied actuator torque for the dynamic analysis.

Once these assumptions are made, the equation of motion is written by summing the moments about the actuated joint for the $i^{th}$ leg:

$$\sum M_{A_i} = I_A \ddot{\theta}_{1i} + c_d \, \dot{\theta}_{1i} + \tau^*_{mp,i}, \tag{6.1}$$

for $i = 1, 2$, and 3, and where $\sum M_{A_i}$ is the sum of the moments applied at joint $A_i$; $I_A$ is the mass moment of inertia of the input link, the motor rotor, and one-half of the two connecting-rod masses, $m_b$, that is considered to be concentrated at $B_i$; $\tau^*_{MP,i}$ is the $i^{th}$ element of $\bar{\tau}^*_{mp}$ which is an array of the interial loads at joint $A_i$ due to the acceleration of the moving platform and the other half of the connecting rod masses that are considered concentrated at $E_i$; and $c_d$ is the

viscous damping coefficient of the actuator. Note that there are two connecting rods in each parallelogram each with mass $m_b$. Expressions for $I_A$ and $\bar{\tau}^*_{mp}$ are given by Eqs. (6.2) and (6.3):

$$I_A = I_m + \frac{1}{3} m_a a^2 + m_b a^2, \tag{6.2}$$

$$\bar{\tau}^*_{mp} = (\mathbf{J}^T)^{-1} \left(3\, m_b + m_c\right) \bar{a}_P, \tag{6.3}$$

where:

| | |
|---|---|
| $I_m$ | is the mass moment of inertia of the motor rotor, |
| $m_a$ | is the mass of the input link, |
| $m_b$ | is the mass of each rod in link $CD$, |
| $m_c$ | is the mass of the moving platform and payload, |
| $a$ | is the link length of the input link, and |
| $\mathbf{J}$ | is the Jacobian matrix as given in Eq. (4.1), and |
| $\bar{a}_P$ | is the acceleration of the moving platform. |

An expression for the resultant moment at joint $A_i$, due to the motor torques and the gravitational force, for all three legs is given by:

$$\begin{bmatrix} \sum M_{A_1} \\ \sum M_{A_2} \\ \sum M_{A_3} \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} + \frac{1}{2}\, a\, m_a\, g \begin{bmatrix} \cos(\theta_{11}) \\ \cos(\theta_{12}) \\ \cos(\theta_{13}) \end{bmatrix} + a\, m_b\, g \begin{bmatrix} \cos(\theta_{11}) \\ \cos(\theta_{12}) \\ \cos(\theta_{13}) \end{bmatrix} + (\mathbf{J}^T)^{-1}\, m \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}, \tag{6.4}$$

where $\tau_i$ is the torque applied by the actuator for the $i^{th}$ leg, and $m = 3\ m_b + m_c$.

Substituting Eqs. (6.4) and (6.3) into Eq. (6.1) as written for three legs, and solving for the actuator torques yields:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = -a\ g\ (\frac{1}{2}m_a + m_b) \begin{bmatrix} \cos(\theta_{11}) \\ \cos(\theta_{12}) \\ \cos(\theta_{13}) \end{bmatrix} - (\mathbf{J}^T)^{-1}\ m \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + c_d \begin{bmatrix} \dot{\theta}_{11} \\ \dot{\theta}_{12} \\ \dot{\theta}_{13} \end{bmatrix} + I_A \begin{bmatrix} \ddot{\theta}_{11} \\ \ddot{\theta}_{12} \\ \ddot{\theta}_{13} \end{bmatrix} + (\mathbf{J}^T)^{-1}\ m\ \bar{a}_P. \tag{6.5}$$

An expression for the acceleration of the moving platform, $\bar{a}_P$, in terms of the manipulator joint angles is given by Eq. (6.6). This is found by differentiating the Jacobian relationship for the manipulator as given by Eq. (4.1).

$$\bar{a}_P = \mathbf{J}^{-1} \begin{bmatrix} \ddot{\theta}_{11} \\ \ddot{\theta}_{12} \\ \ddot{\theta}_{13} \end{bmatrix} + \frac{d}{dt}(\mathbf{J}^{-1}) \begin{bmatrix} \dot{\theta}_{11} \\ \dot{\theta}_{12} \\ \dot{\theta}_{13} \end{bmatrix}. \tag{6.6}$$

Substituting Eq. (6.6) into Eq. (6.5) and grouping the terms results in:

$$\bar{\tau} = \mathbf{M}(\mathbf{q})\ \ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\ \dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}), \tag{6.7}$$

where:

$$\mathbf{q} = \begin{bmatrix} \theta_{11} \\ \theta_{12} \\ \theta_{13} \end{bmatrix}, \qquad \bar{\tau} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix},$$

$$\mathbf{M}(\mathbf{q}) = I_A\ \mathbf{I} + (\mathbf{J}^T)^{-1}\ m\ \mathbf{J}^{-1}, \tag{6.8}$$

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = c_d \, \mathbf{I} + (\mathbf{J}^T)^{-1} \, m \, \frac{d}{dt}(\mathbf{J}^{-1}), \tag{6.9}$$

$$\mathbf{G}(\mathbf{q}) = -a \, g \, (\frac{1}{2} m_a + m_b) \begin{bmatrix} \cos(\theta_{11}) \\ \cos(\theta_{12}) \\ \cos(\theta_{13}) \end{bmatrix} - (\mathbf{J}^T)^{-1} \, m \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}, \tag{6.10}$$

and $\mathbf{I}$ is the identity matrix. (6.11)

This approach offers the advantage of being less computationally intensive than the Lagrangian approach, while still capturing most of the dynamic characteristics of the manipulator. These characteristics makes it an attractive choice for the dynamics model to be employed in a computed torque control scheme. A numerical example of this method is presented later in this chapter and is compared to the other methods developed for the manipulator dynamics.

## 6.3 Lagrange Based Dynamic Analysis

A more common approach used to characterize manipulator dynamics is the Lagrangian approach, which is presented here as a comparison to the application of Newton's equation of motion. The assumptions made for the Lagrangian dynamics model are the same assumptions made for the model based on the application of Newton's equation of motion except for a few subtle differences. Once again the mass of each connecting rod in the upper arm assembly, $m_b$, is considered as divided and concentrated at two points. However, for the Lagrangain model the masses are concentrated at joints $C_i$ and $D_i$, rather than $B_i$ and $E_i$ as was done in the previous dynamic model, offering a slightly better approximation of the

manipulator dynamics. Also, in the Lagrangian model the damping at the actuator is disregarded since the Lagrangian model does not readily accommodate viscous damping as is assumed for the actuators.

For this analysis, Lagrange multipliers are used with nine generalized coordinates. The analysis could be done using just three generalized coordinates, and no Lagrange multipliers, however the expression for the Lagrange function would be cumbersome to use due to the complex kinematics of the manipulator. In general, the Lagrange multiplier approach involves solving the following system of equations:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_j}\right) - \frac{\partial L}{\partial q_j} = Q_j + \sum_{i=1}^{k} \lambda_i \mathbf{A}_{ij}, \tag{6.12}$$

for $j = 1$ to $n$, where:

| | |
|---|---|
| $i$ | is the constraint index, |
| $j$ | is the generalized coordinate index, |
| $k$ | is the number of of constraint functions, |
| $n$ | is the number of generalized coordinates, |
| $L$ | is the Lagrange function, where $L = T - V$, |
| $T$ | is the total kinetic energy of the manipulator, |
| $V$ | is the total potential energy of the manipulator, |
| $q_j$ | is the $j^{th}$ generalized coordinate, |
| $\lambda_i$ | is the Lagrange multiplier, |
| $f_i$ | is a constraint equation, |
| $Q_j$ | is a generalized external force, and |
| $\mathbf{A}_{ij} = \frac{\partial f_i}{\partial q_j}$. | |

For this manipulator, the generalized coordinates are chosen to be: $p_x, p_y, p_z$, $\theta_{21}, \theta_{22}, \theta_{23}, \theta_{11}, \theta_{12}$, and $\theta_{13}$. As a result, Eq. (6.12) represents a system of nine

equations in nine variables, where the nine variables are $\lambda_i$ for $i = 1$ to 6, and the three actuator torques, $Q_j$ for $j = 7, 8$, and 9. The external generalized forces, $Q_j$ for $j = 1$ to 6, are zero since there is no externally applied force at the moving platform and the joints at $B_i$ are passive.

This formulation requires six constraint equations, $f_i$ for $i = 1$ to 6, that are written in terms of the generalized coordinates. The first three constraint equations are drawn from the fact that the distance between joints $C$ and $D$ will always be equal to the length of the upper arm connecting rod, $b$:

$$f_i = (D_{xi} - C_{xi})^2 + (D_{yi} - C_{yi})^2 + (D_{zi} - C_{zi})^2 - b^2 = 0, \tag{6.13}$$

where:

$$\begin{aligned}
C_{xi} &= \cos(\phi_i)\left[r + a\cos(\theta_{1i}) + e\cos(\theta_{2i})\right], \\
C_{yi} &= \sin(\phi_i)\left[r + a\cos(\theta_{1i}) + e\cos(\theta_{2i})\right], \\
C_{zi} &= a\sin(\theta_{1i}) + e\sin(\theta_{2i}), \\
D_{xi} &= p_x + \cos(\phi_i)\left[c - d\cos(\theta_{2i})\right], \\
D_{yi} &= p_y + \sin(\phi_i)\left[c - d\cos(\theta_{2i})\right], \\
D_{zi} &= p_z - d\sin(\theta_{2i}),
\end{aligned}$$

for $i = 1, 2$, and 3.

The second set of constraint equations are drawn from the relationship that the upper arm linkage is at an angle of $\theta_{2i}$ from the XY plane:

$$f_i = E_{wi} - B_{wi} + (B_{ui} - E_{ui})\tan(\theta_{2i}), \tag{6.14}$$

where:

$$B_{ui} = a\cos(\theta_{1i}),$$

$$
\begin{aligned}
B_{wi} &= a\sin(\theta_{1i}), \\
E_{ui} &= p_x\cos(\phi_i) + p_y\sin(\phi_i) + c - r, \\
E_{wi} &= p_z,
\end{aligned}
$$

for $i = 4, 5$, and 6.

Solving the system of equations given by Eq. (6.12) is made easier by grouping the first set of six equations together so that they form a system of six equations in six unknowns. The unknowns in this system are the Lagrange multipliers:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{p_x}}\right) - \frac{\partial L}{\partial p_x} = \sum_{i=1}^{6} \lambda_i \mathbf{A}_{i1}, \tag{6.15}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{p_y}}\right) - \frac{\partial L}{\partial p_y} = \sum_{i=1}^{6} \lambda_i \mathbf{A}_{i2}, \tag{6.16}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{p_z}}\right) - \frac{\partial L}{\partial p_z} = \sum_{i=1}^{6} \lambda_i \mathbf{A}_{i3}, \tag{6.17}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta_{21}}}\right) - \frac{\partial L}{\partial \theta_{21}} = \sum_{i=1}^{6} \lambda_i \mathbf{A}_{i4}, \tag{6.18}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta_{22}}}\right) - \frac{\partial L}{\partial \theta_{22}} = \sum_{i=1}^{6} \lambda_i \mathbf{A}_{i5}, \tag{6.19}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta_{23}}}\right) - \frac{\partial L}{\partial \theta_{23}} = \sum_{i=1}^{6} \lambda_i \mathbf{A}_{i6}. \tag{6.20}$$

Having found the Lagrange multipliers, the actuator torques can be determined directly from the three remaining equations:

$$\tau_1 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta_{11}}}\right) - \frac{\partial L}{\partial \theta_{11}} - \sum_{i=1}^{6} \lambda_i \mathbf{A}_{i7}, \tag{6.21}$$

$$\tau_2 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta_{12}}}\right) - \frac{\partial L}{\partial \theta_{12}} - \sum_{i=1}^{6} \lambda_i \mathbf{A}_{i8}, \tag{6.22}$$

$$\tau_3 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta_{13}}}\right) - \frac{\partial L}{\partial \theta_{13}} - \sum_{i=1}^{6} \lambda_i \mathbf{A}_{i9}, \tag{6.23}$$

where $\tau_i$ is the actuator torque for the $i^{th}$ leg.

To apply Eqs. (6.15)-(6.23), an expression must be developed for the Lagrange function, $L$. The Lagrange function is defined as the difference between the system kinetic energy, T, and its potential energy, V:

$$L = T - V. \tag{6.24}$$

The total potential energy of the manipulator was calculated relative to the plane of the stationary platform of the manipulator, and was found to be:

$$V = V_c + \sum_{i=1}^{3}(V_{bi} + V_{ai}), \tag{6.25}$$

where $V_c$ is the potential energy of the moving platform, $V_{bi}$ is the potential energy of the connecting rods of leg $i$, and $V_{ai}$ is the potential energy of the input link on leg $i$:

$$V_c = -m_c\, g\, p_z, \tag{6.26}$$

$$V_{bi} = -m_b\, g\, [p_z + a\sin(\theta_{1i}) + (e-d)\sin(\theta_{2i})]\,, \tag{6.27}$$

$$V_{ai} = -\frac{1}{2} m_a\, g\, a\sin(\theta_{1i}). \tag{6.28}$$

The total kinetic energy, $T$, for this manipulator is given by:

$$T = T_c + \sum_{i=1}^{3}(T_{bi} + T_{ai}), \tag{6.29}$$

where $T_c$ is the kinetic energy of the moving platform, $T_{bi}$ is the kinetic energy of the connecting rods of leg $i$, and $T_{ai}$ is the kinetic energy of the input link and the rotor on leg $i$:

$$T_c = \frac{1}{2} m_c(\dot{p_x}^2 + \dot{p_y}^2 + \dot{p_z}^2), \tag{6.30}$$

$$
\begin{aligned}
T_{bi} &= \frac{1}{2}m_b(\dot{p_x}^2 + \dot{p_y}^2 + \dot{p_z}^2) + dm_b\dot{\theta_{2i}}\dot{p_y}\sin(\phi_i)\sin(\theta_{2i}) \\
&\quad + dm_b\dot{\theta_{2i}}\dot{p_x}\cos(\phi_i)\sin(\theta_{2i}) + \frac{1}{2}a^2m_b\dot{\theta_{1i}}^2 \\
&\quad + \frac{1}{2}m_b\dot{\theta_{2i}}^2(d^2+e^2) + aem_b\dot{\theta_{1i}}\dot{\theta}_{2i}\cos(\theta_{1i}-\theta_{2i}) \\
&\quad - dm_b\dot{\theta_{2i}}\dot{p_z}\cos(\theta_{2i}) \qquad (6.31) \\
T_{ai} &= \frac{1}{6}m_a a^2\dot{\theta_{1i}}^2 + \frac{1}{2}I_m\dot{\theta_{1i}}^2 \qquad (6.32)
\end{aligned}
$$

Solving Eqs. (6.15)-(6.23) for the actuator torques for a given trajectory also requires the evaluation of the **A** matrix, where $a_{ij} = \frac{\partial f_i}{\partial q_j}$. The evaluation of the elements of **A** is provided in Appendix B. The evaluation of the required partial derivatives of the Lagrange function are included in Appendix C.

## 6.4 Single Link Dynamic Model

A model based on a single link approximation of the manipulator is also developed. This model results in an equation of motion that is a linearly uncoupled approximation of the manipulator dynamics. This model is developed for use with a PID control model that is developed in another chapter.

A schematic of the single link model is shown in Fig. 6.2, where $c_d$ is the viscous damping of the motor, $g$ is the acceleration due to gravity, $m_a$ is the mass of the input link, $I_m$ is the actuator rotor inertia, $\theta_{1i}$ is the angular displacement of the input link from the base of the manipulator, and $\tau_i$ is the applied torque for the $i^{th}$ leg. It assumes that the mass of the moving platform, $m_c$, is evenly dived between the three legs and concentrated at the end of the input link, and that the mass of each connecting rod, $m_b$ can be considered as concentrated at

the end of the input link, so that $m_{eq} = 2\, m_b + \frac{1}{3}\, m_c$. This model also assumes that each link can be modeled separately, so that it neglects the influence of the motion of the rest of the manipulator on the input link that is being modeled. Both of these assumptions are substantial, but are made to provide a dynamic model that will allow the development of a PID controller for the manipulator.

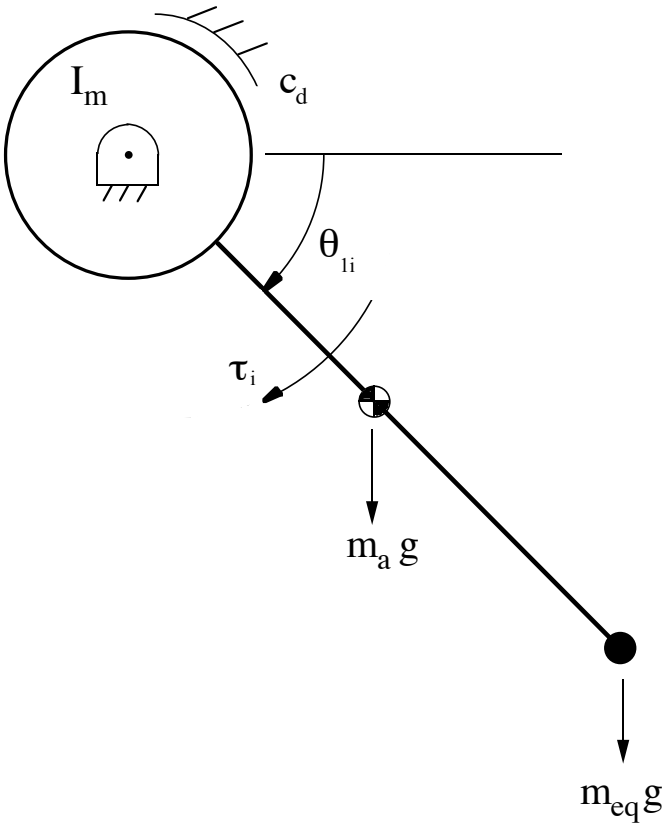



Figure 6.2: Single link dynamics model for the $i^{th}$ leg.

The equation of motion for the simplified link model for the $i^{th}$ leg is found by summing the torques about the revolute joint and applying Euler's equation

of motion:

$$\tau_i = \alpha\ \ddot{\theta}_{1i} + c_d\ \dot{\theta}_{1i} - \beta\ \cos(\theta_{1i}), \tag{6.33}$$

where:

$$\alpha = I_m + \frac{1}{3}\ m_a\ a^2 + m_{eq}\ a^2,$$
$$\beta = a\ g\left(\frac{1}{2}\ m_a + m_{eq}\right).$$

Equation (6.33) is linearized about an operating point of $\theta_{1i,o} = \frac{\pi}{4}$ and $\tau_{i,o} = -\beta\cos(\frac{\pi}{4})$. This leads to Eq. (6.34) in terms $\hat{\theta}_{1i}$ and $\hat{\tau}_i$, where $\hat{\theta}_{1i}$ is the differential displacement of the link relative to the operating point of $\theta_{1i,o} = \frac{\pi}{4}$, and $\hat{\tau}_i$ is the differential torque applied to the link relative to the operating point of $\tau_{i,o} = -\beta\cos(\frac{\pi}{4})$. Both $\hat{\theta}_{1i}$ and $\hat{\tau}_i$ are considered to be small for the linearization.

$$\hat{\tau}_i = \alpha\ \ddot{\hat{\theta}}_{1i} + c_d\ \dot{\hat{\theta}}_{1i} + \beta\sin(\frac{\pi}{4})\ \hat{\theta}_{1i}\ , \tag{6.34}$$

where:

$$\hat{\theta}_{1i} = \theta_{1i} - \frac{\pi}{4}\ , \text{and}$$
$$\hat{\tau}_i = \tau_i + \beta\cos(\frac{\pi}{4})\ .$$

Equation (6.34) provides a simplified linear uncoupled dynamic model of the manipulator that is useful for developing single input single output (SISO) type controllers, like a PID controller. This model does supply some insights to the dynamic behavior of the manipulator, however the model's ability to fully capture the dynamics of the manipulator is marginal.

## 6.5 Numerical Simulations

Numerical simulations are performed to demonstrate these three approaches and to provide some insights to the dynamics of the manipulator as it follows a given trajectory. To accomplish this, a trajectory for the moving platform and a set of manipulator design parameters are assumed. The trajectory is selected to represent a typical motion that might be used during a practical application of the manipulator. Given the trajectory and the manipulator design parameters, a set of actuator torques is computed using the dynamics model based upon the the application of Newton-Euler equations of motion. Next, using the same trajectory and manipulator design parameters a second set of actuator torques is calculated using the Lagrangian model. Finally, the actuator torques are computed using the single link dynamics model.

The manipulator design parameters that are used for the simulations are given in Table 6.1. The trajectory used for these simulations has the moving platform following three straight line segments between four points, $Q_1, Q_2, Q_3$, and $Q_4$. The first segment has the moving platform moving from $Q_1$, located at (0,0,400) in the XYZ coordinate system, to $Q_2$, located at (0,0,350), in 0.4 seconds, where the length dimensions are given in millimeters. The second segment goes from $Q_2$ to $Q_3$ at (50,50,350) in 0.8 seconds. The third and final segment goes from $Q_3$ to $Q_4$ at (50,50,400) in 0.8 seconds. For each segment of the trajectory the moving platform starts and finishes with zero velocity and accelerates or decelerates at 245.2 cm/s$^2$. The angular displacement required of each input link to follow the example trajectory is shown in Fig. 6.3. The velocity of the moving platform as it follows the example trajectory is shown in Fig. 6.4.

Table 6.1: Manipulator design parameters for dynamic numerical simulations.

| |
|---|
| $a = 203.2$ mm |
| $b = 254.0$ mm |
| $c = r = 127.0$ mm |
| $d = e = 15.9$ mm |
| $\phi_1 = 0$ deg |
| $\phi_2 = 120$ deg |
| $\phi_3 = 240$ deg |
| $m_a = 0.184$ kg |
| $m_b = 0.085$ kg |
| $m_c = 0.413$ kg |
| $I_m = 0.00434$ N $\cdot$ m $\cdot$ s$^2$ |
| $c_d = 0.0027$ N $\cdot$ m $\cdot$ s |

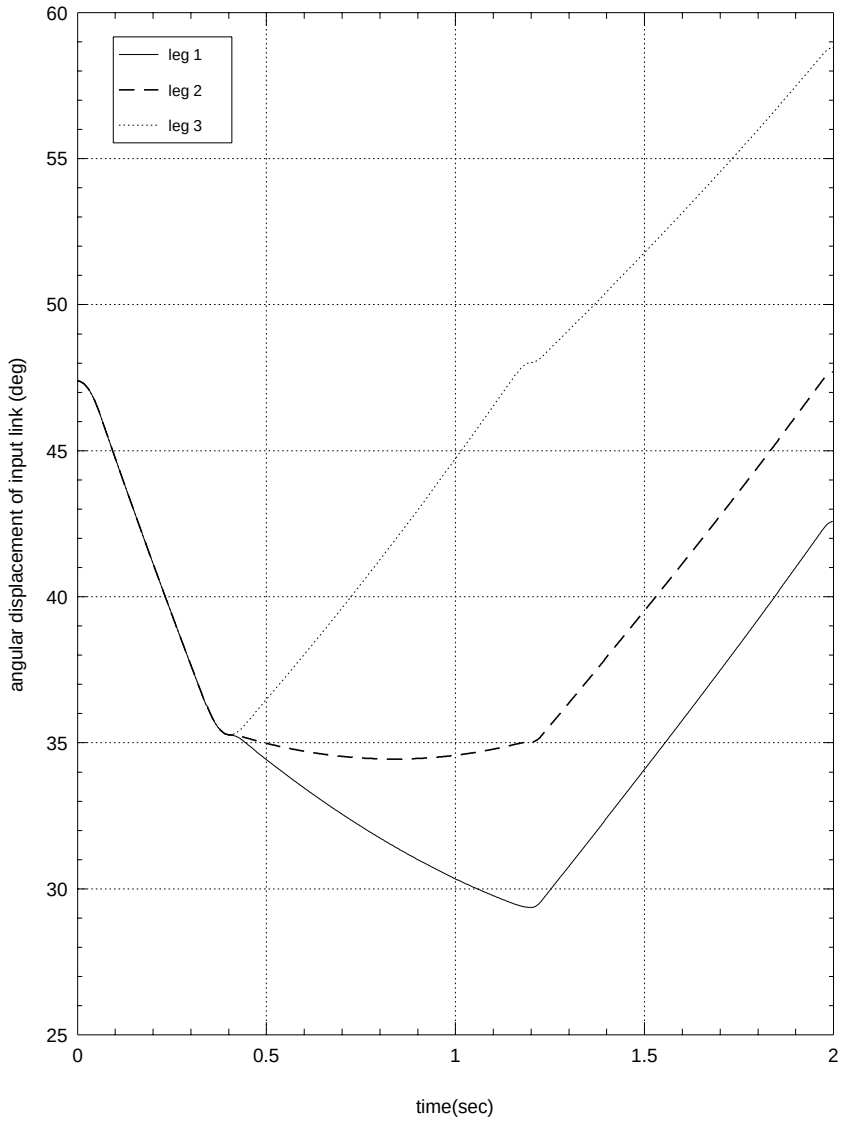



Figure 6.3: Angular displacement of the input links to follow the trajectory used for numerical simulations of the dynamics models.

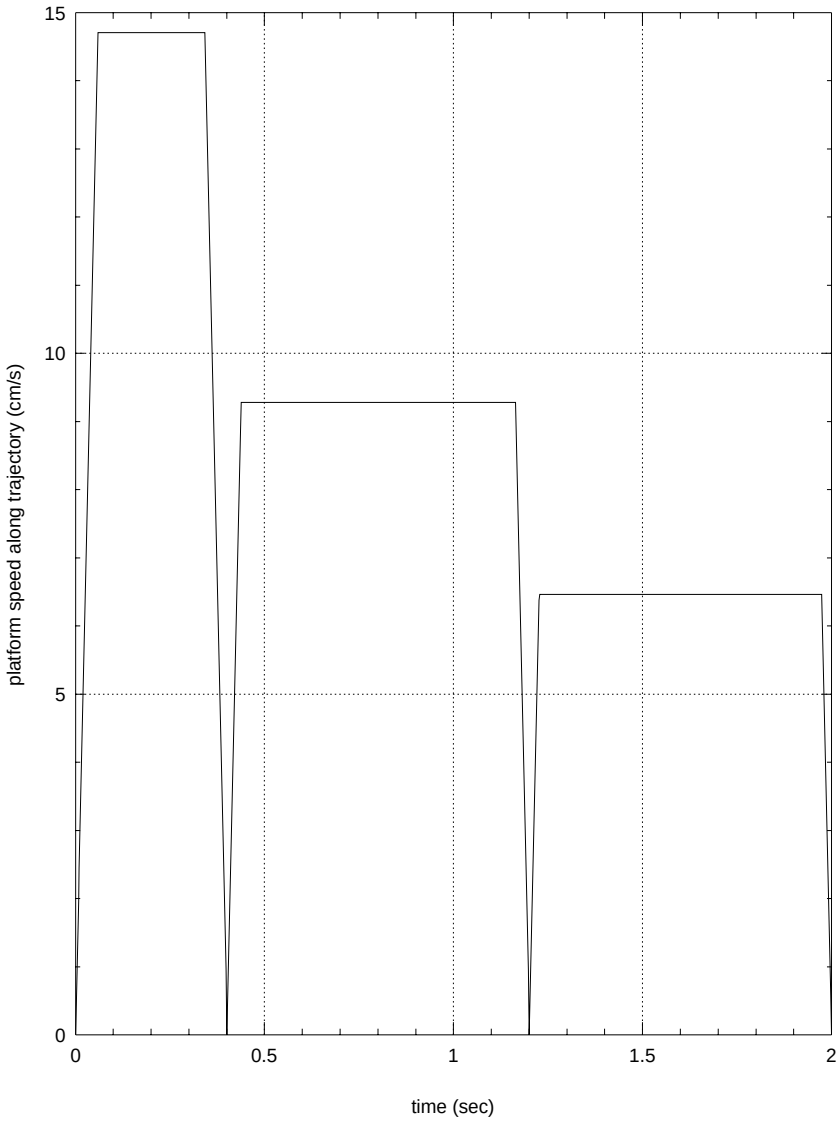



Figure 6.4: Velocity of moving platform along the example trajectory.

Given this sample trajectory and set of manipulator design parameters, the actuator torques are calculated using the direct application of Newton-Euler equations of motion. The results of this method is shown in Fig. 6.5. From this figure it can be seen that during the first segment of the trajectory, from $t = 0$ seconds to $t = 0.4$ seconds, all the legs undergo the same motion and the load from the moving platform is evenly divided between the three legs, causing each actuator to produce the identical torque during this segment. This segment, and each of the other segments, of the trajectory is divided into three regions. The first region is the acceleration region during which the actuators supply an initial torque to accelerate the moving platform until it reaches the second region of the segment, the constant velocity region. For the first segment of this trajectory, once the constant velocity region is reached there is a step reduction in the magnitude of the actuator torque to allow the moving platform to travel at a constant velocity. During this constant velocity region, the magnitude of the torque gradually increases. This is due to the changing configuration of the manipulator, and in turn the changing Jacobian, and not dynamic loading. Finally, during the deceleration region of the first trajectory segment, the magnitude of the actuator torque once again takes a step-wise decrease to allow the gravitational force to slow and finally stop the manipulator at the second point along the trajectory, $Q_2$. The two remaining trajectory segments follow a similar pattern.

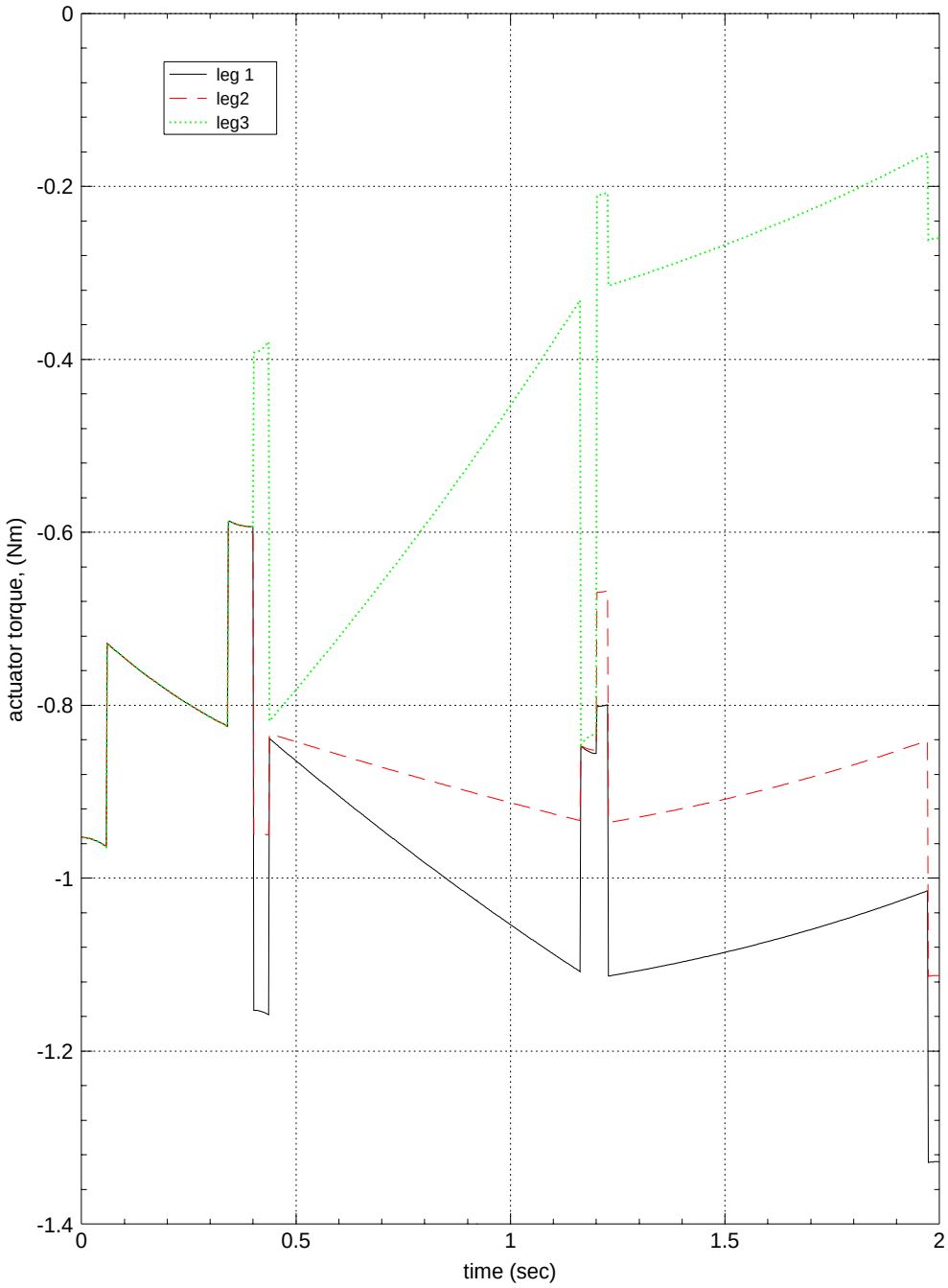



Figure 6.5: Actuator torques for the simplified dynamics model using Newton-Euler equations of motion and the Jacobian matrix.

The actuator torques for the example trajectory were also calculated using the Lagrange multiplier approach and the single link dynamic model. The results from these two simulations are shown in Figs. 6.6 and 6.7.

The torques calculated using the Lagrangian model display very close agreement with the torques calculated using the direct application of Newton-Euler equations of motion. The small differences between the two results are due to lack of damping in the Lagrangian model and the different assumptions regarding where the mass of the connecting rods are considered concentrated. This close agreement helps validate the somewhat unusual approach of mapping the inertial loads of the moving platform back to the input link as was done in the model based upon the direct application of Newton's equation of motion. It also provides some confidence for using this model as the basis of a computed torque control scheme.

The actuator torques calculated using the single link dynamic model, as shown in Fig. 6.7, does not display very good agreement with the other two more complete dynamic models. This poor agreement highlights the highly coupled nature of the dynamics of this manipulator. The single link model completely disregards the influence of the rest of the manipulator on the dynamics of any one leg. Accordingly, this model does capture some of the character of the dynamics of the manipulator, but also misses a significant portion of it. However, this model does offer the advantage of being linear and uncoupled, which is a significant advantage when developing single-input single-output type controllers.

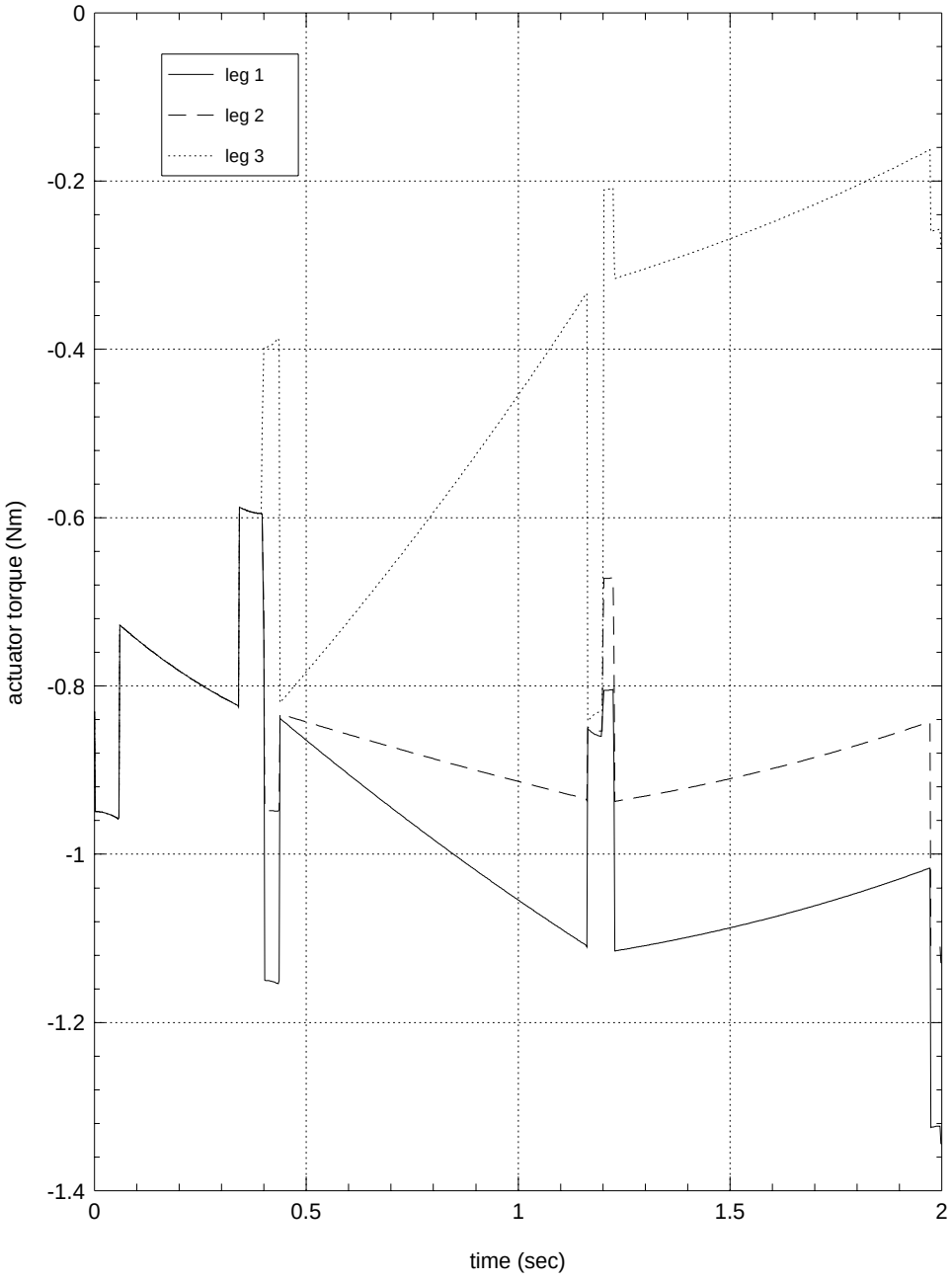



Figure 6.6: Actuator torques for the example trajectory using the Lagrangian dynamics model.

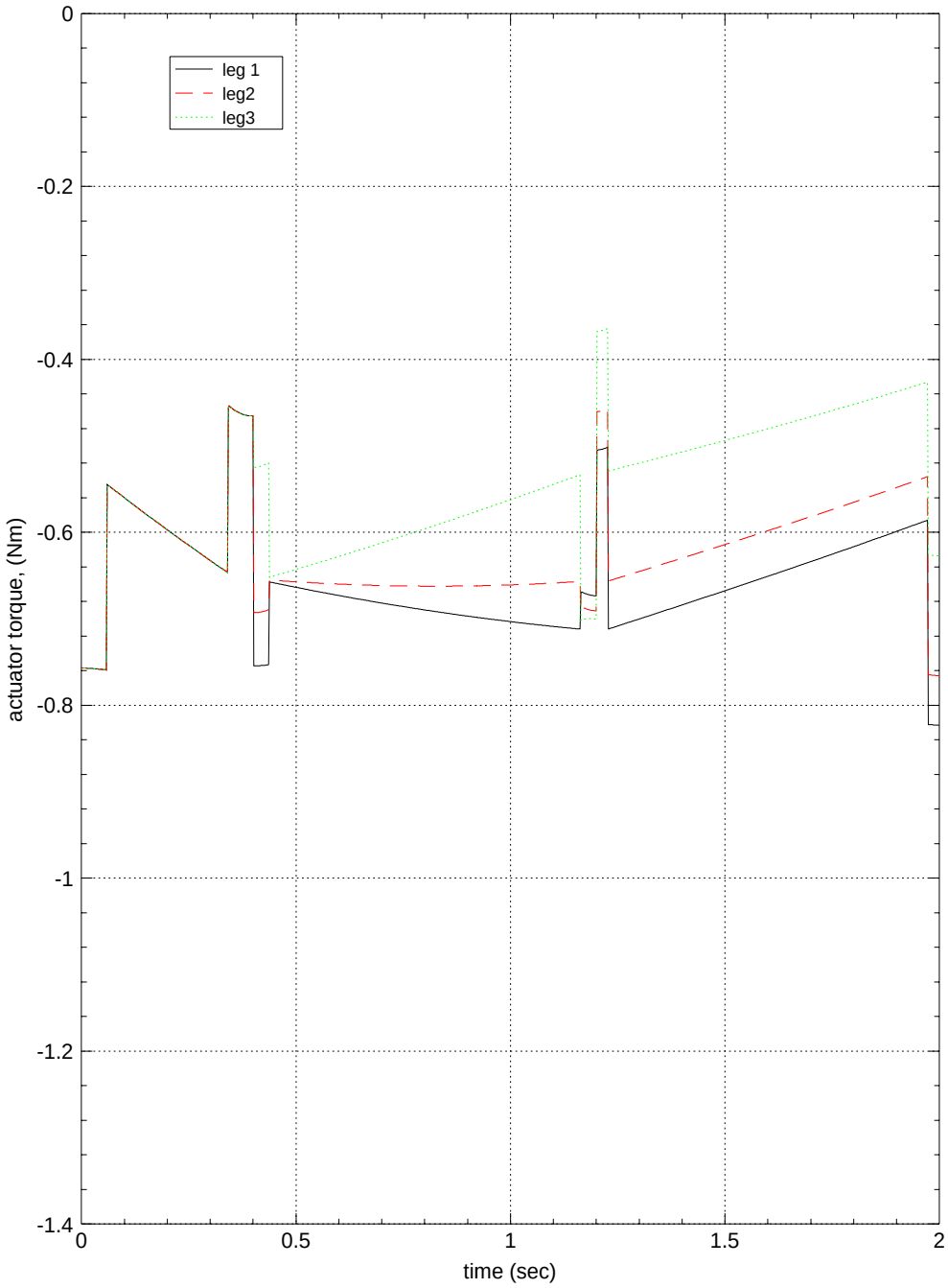



Figure 6.7: Actuator torques for the example trajectory using the single link dynamics model.

As expected, the time required to calculate the actuator torques for the given example trajectory was significantly different for the Lagrange based dynamics model and the dynamics model based on the direct application of Newton's equation of motion. The Lagrange based calculation of the actuator torques took an average of 1.19 seconds per time step where the simplified model based upon Newton's equation of motion took an average of 0.45 seconds per time step. Both methods were written using Xmath, a mathematical software package, and ran on a Sparc station IPX. The code that was written to calculate the actuator torques was not optimized in any way to minimize the runtime. However, both programs used the same programming structure. Clearly, the speed of either method can be improved with better programming and faster hardware. However, the runtime results for these programs are provided to give some sense of the relative computation time required by the two approaches, since the main advantage of the dynamics model based on the direct application of Newton's equation of motion is that it is less computationally intensive.

## 6.6 Summary

Three approaches are presented to model the dynamics of the manipulator. The first is based upon the direct application of Newton's equation of motion and a simplified dynamics model of the manipulator. It assumes that the mass of the connecting rods of the upper arm assembly are evenly divided between, and concentrated at the joints where the upper arm assembly is attached to the input link and the moving platform. The actuator torques are then calculated by applying Newton's equation of motion to the input link and approximating

the inertial and gravitational torques at the actuator due to the moving platform by mapping the net force acting on the moving platform back to the actuators with the inverse of the transpose of the Jacobian matrix. This approach is developed in an attempt to create a dynamic model that compares favorably with the ability of the Lagrangian approach to characterize the dynamics of the manipulator while being less computationally intensive, making it attractive as the inverse dynamic model for a computed torque control scheme. The second approach is a Lagrangian based method. The Lagrangian function is developed for the manipulator assuming that the mass of each long connecting link, $CD$, on the upper arm assembly could be considered as distributed equally as two point masses at either end of the link. Once the Lagrange function was determined, Lagrangian multipliers were employed to develop a system of nine equations in nine unknowns that yield the actuator torques for a given trajectory. The third approach is based on a single link model for each leg of the manipulator. This model assumes that the mass of the moving platform is evenly divided between the three legs and concentrated at the end of the input link, and that the mass of each connecting rod, $m_b$ can be considered as concentrated at the end of the input link. The single link model also assumes that each leg can be modeled separately, so that it neglects the influence of the motion of the rest of the manipulator on the input link that is being modeled. The model is also linearized about an operating point of the manipulator. These assumptions result in a dynamic model that does provide some insight to the dynamics behavior of the manipulator, but does a poor job of fully capturing the dynamics of the manipulator. However, this model does have the advantage of providing a linearly uncoupled approximation of the dynamics which is useful for the development of a PID controller for the

manipulator.

A numerical example is presented to demonstrate the results of the three methods for a sample trajectory and manipulator design parameters. In general there is an excellent agreement between the Lagrangian model and the model based on the Newton's equation of motion. The numerical example is also used to demonstrate that the dynamic model based on Newton's equation of motion requires less time per evaluation to calculate the actuator torques than does the Lagrangian based approach.

# Chapter 7

# Prototype Design and Accuracy

## 7.1 Introduction

A prototype of the manipulator was designed and constructed to provide a proof of concept for this parallel manipulator and to provide a visually appealing demonstration model to build more interest in this research. The prototype is fully actuated and computer controlled. A user interface has also been developed that allows the user to construct desired moving platform trajectories from a combination of straight line and arc trajectories. A photograph of the manipulator is shown in Fig. 7.1. Note that the fixed base of the prototype manipulator is mounted above the workspace where it operates such that the force of gravity pulls the moving platform away from the fixed platform.

In this chapter, the design of the prototype manipulator hardware, a description of the selected components, and the system used to implement the controller are presented. The accuracy of the manipulator is also considered.

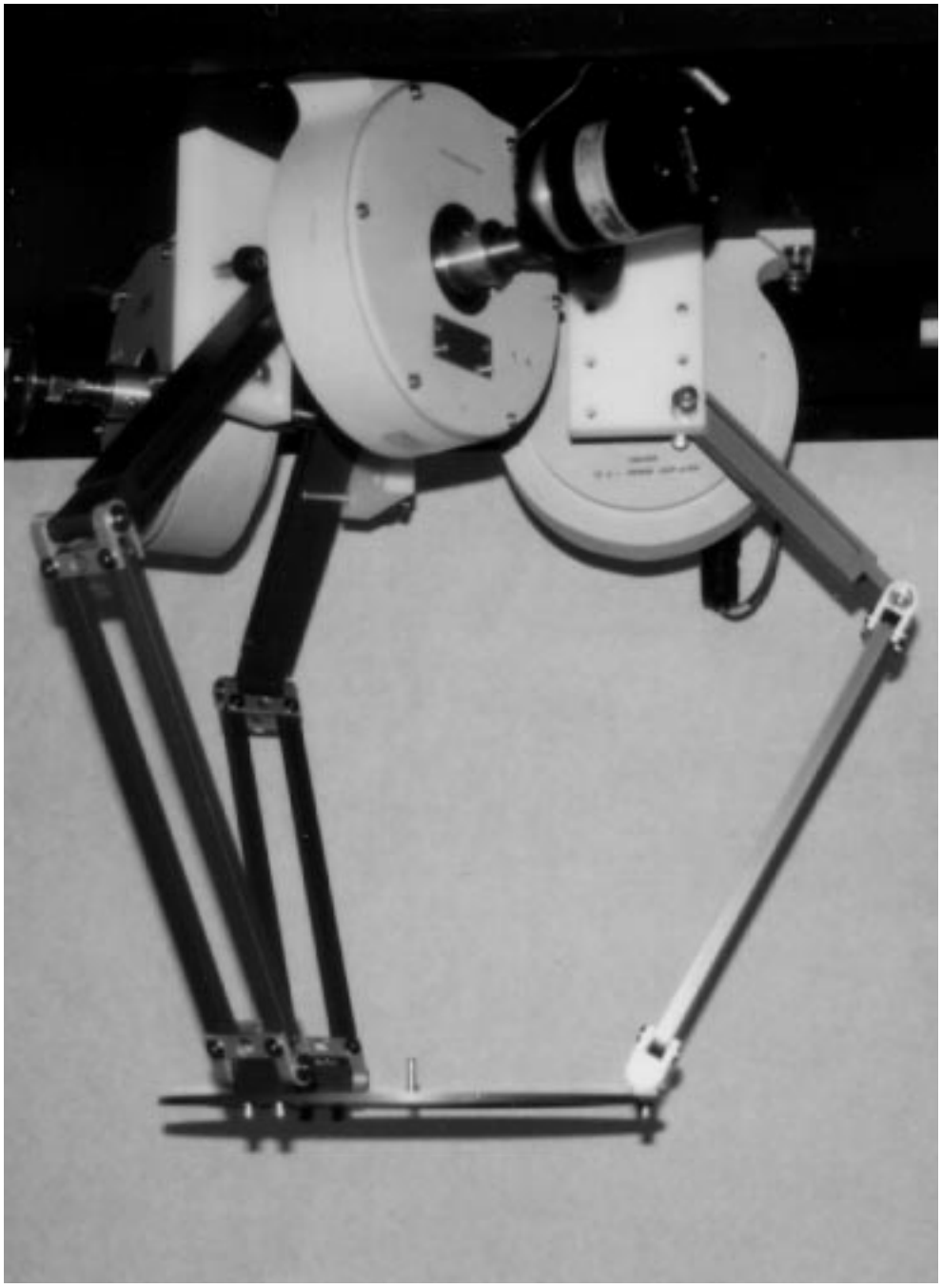


Figure 7.1: Photograph of manipulator prototype.

## 7.2 Description of Prototype Design

The prototype manipulator as shown in Fig. 7.1 is actuated by three permanent magnet brushed DC motors (Inland T-5152B DC Torque Motor) that are directly coupled to the input links of the manipulator. A direct drive configuration was chosen to eliminate the positioning error that would result from the backlash that is typical of a gear train. The position of each input link is measured with an incremental encoder (Heidenhain #ROD 426.001). Each encoder has a resolution of 5000 counts/revolution, and is geared to the motor shaft with an anti-backlash gear set that has a 2.5:1 gear ratio. Hence, each incremental encoder count corresponds to 0.0288 degrees of displacement of the input link. The links of the manipulator are fabricated from aluminum alloy (6061-T6). The nominal design parameters and measured link properties for the prototype manipulator are the same as were used for the dynamic simulations, and are listed in Table 6.1. The choice for the ratio of the link lengths and the angular orientation of the legs was guided by the results of Chapter 5 to provide a well conditioned workspace. The drawings of the links are provided in Appendix D.

Control of the prototype manipulator is achieved using a controller prototyping station assembled by Kantor (1995). The controller prototyping station is based on Integrated Systems Inc.'s AC-100 system. A block diagram describing the AC-100 system is shown in Fig. 7.2.

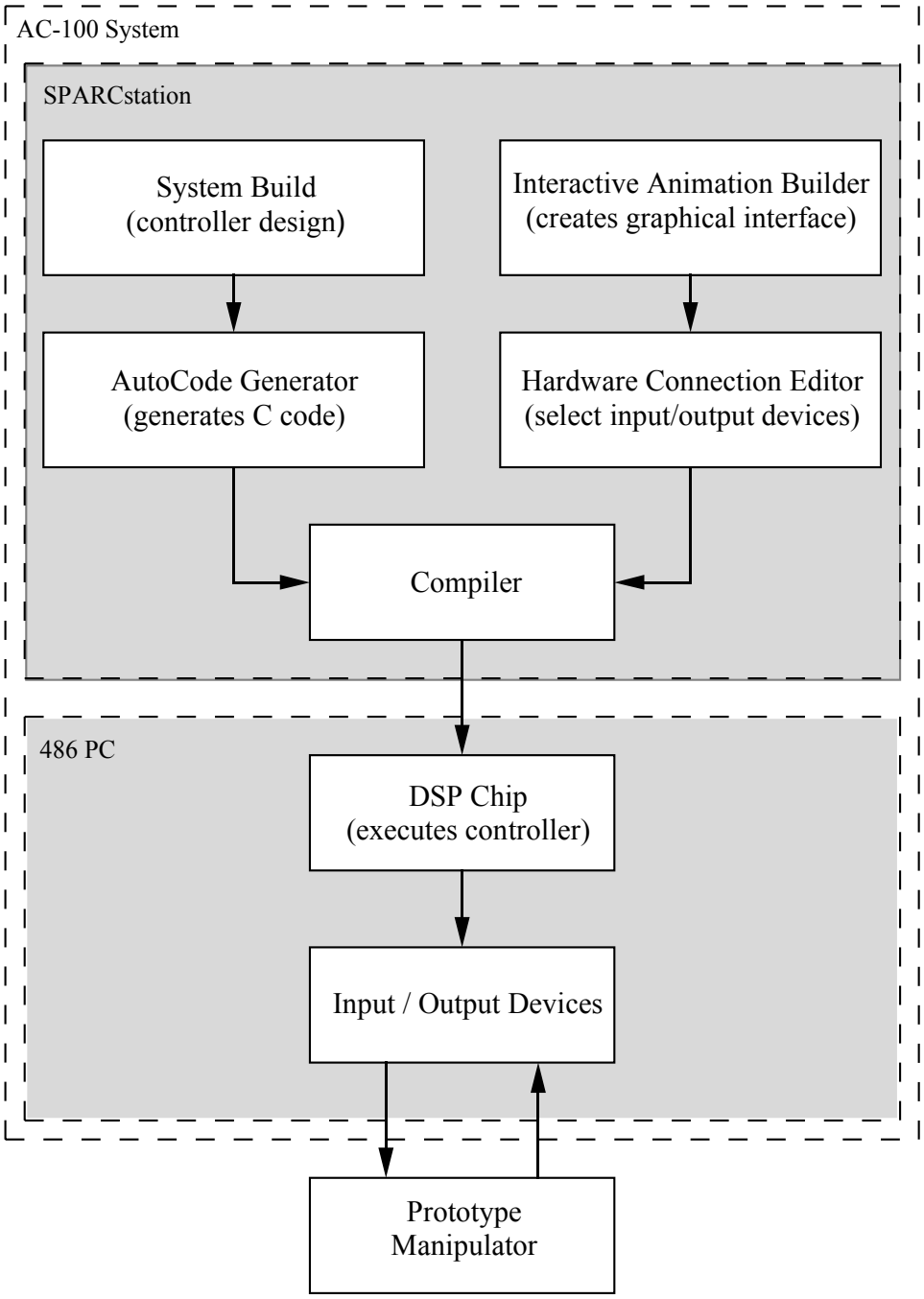



Figure 7.2: Block diagram of AC-100 controller prototyping system.

The AC-100 software resides on two computers, a SPARCstation IPX and a 486 PC equipped with a digital signal processing (DSP) chip. The AC-100 uses Integrated Systems Inc.'s MatrixX family of software products to allow the user to design the controller, specify the input-output connections, design graphical instrumentation panels, and generate C code that is used by a digital signal processing (DSP) chip for the real-time control of the manipulator. The controller is designed in block diagram form using the SystemBuild feature of the MatrixX software. The graphical instrumentation panel allows commands to be sent to the controller, and also provides a means of monitoring the system outputs during the operation of the controller. Once the controller block diagram is defined, the input-output connections are specified, and the C code is generated; the code is downloaded to the PC where it is compiled and executed to control the prototype manipulator.

A block diagram for an example controller developed for a single leg of the prototype manipulator using the SystemBuild software is shown in Fig. 7.3. The example controller has 1 input, the number of counts from the encoder, and 2 outputs, the position error and the control signal supplied to the amplifier that drives the actuator. The example controller converts the number of counts of the encoder into the angular position of the input link in radians in the "Leg1 Actual Position (Rad)" block. The position error of the input link is determined by comparing the measured position of the input link to the desired position of the input link as supplied by the "Leg1 Desired Position (Rad)" block. The error signal is supplied to a control block, which in this example is a PID controller. The various controllers that are used for the prototype manipulator are discussed in more detail in the next chapter. The output signal from the "Leg1 PID

Controller" block is supplied to the power amplifier that drives the motor for leg 1. This example is the basis for the prototype manipulator controllers that were developed, and is provided to illustrate how the controller is developed using the AC-100 system. The complete controllers for the prototype manipulator include a similar block diagram for each leg, and also has provisions for initializing the encoders, and implementing the unique features of the various types of controllers used for the prototype manipulator. An example of a complete block diagram used for a PID controller for a single leg is given in Appendix E.

The desired position of the input link as supplied by the "Leg1 Desired Position (Rad)" block in Fig. 7.3 is calculated by a trajectory generator that was written by a summer research assistant, Remi Taulemesse. The trajectory generator allows the user to define a trajectory for the moving platform that is composed of a series of segments, with each segment being a line or circular arc in end-effector space. The trajectory of the moving platform is mapped into joint space using the inverse kinematics, and is supplied to the "Leg1 Desired Position (Rad)" block. Each segment of a trajectory is divided into a constant acceleration, constant velocity, and constant de-acceleration part to achieve the desired motion in a prescribed amount of time. The trajectory generation programs were written using Xmath, a mathematical programming language for the MatrixX family of software products. The flow charts for the trajectory generator are given in Appendix F, and the program code listing is provided in Appendix G.

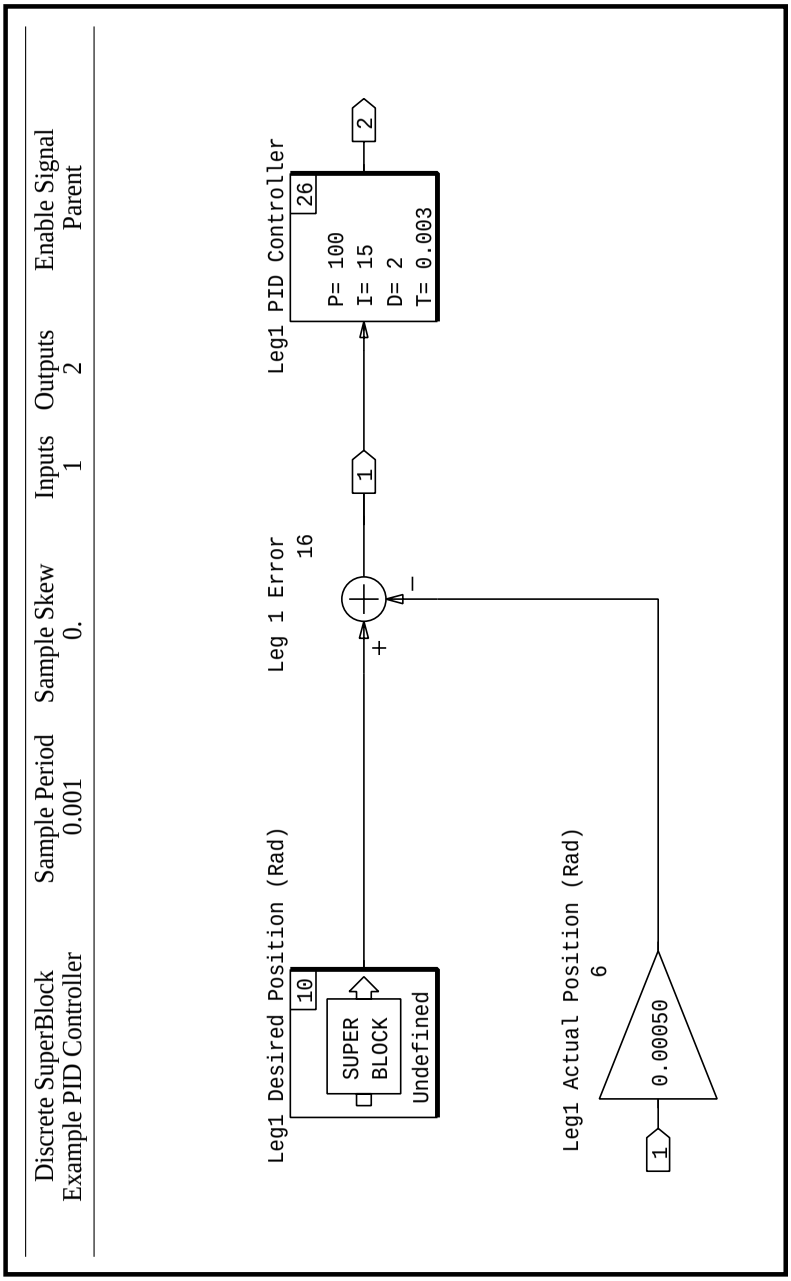



Figure 7.3: Example of controller block diagram using SystemBuild.

## 7.3 Accuracy of Prototype

There are several sources of error between the mathematical model of a manipulator and the practical embodiment of a manipulator that impact its accuracy. Whitney, Lozinski, and Rourke (1986) divided these into two sources: geometric and non-geometric errors. The geometric errors are a function of errors in the physical parameters that relate the various link axes to one another. This is primarily driven by fabrication errors (e.g. link lengths, and alignment of the bearing surfaces). Non-geometric errors are driven by other factors such as controller performance or compliance in the links and joints due to loading. This chapter only considers the accuracy of the manipulator as related to geometric errors and the resolution of the position sensing devices for the input links.

To gain some insight to the geometric positioning accuracy of the prototype manipulator, it was manually placed into several different discrete poses. For each pose, the geometric positioning error was determined by comparing the measured position of the moving platform with the position of the moving platform calculated from the input joint angles as measured by the encoders. To measure the actual position of the moving platform, a jig was fabricated that constrains the moving platform at a known position relative to the stationary platform (see Fig. 7.4).

The location of the jig is fixed to the stationary platform with two locating pins that are mounted on the prototype base. One pin is located at the origin of the manipulator coordinate system and the other is located along the x-axis, so that the position of the jig relative to the manipulator coordinate system is known. A series of holes were machined in the plate at the end of the jig. These holes are used to constrain the manipulator to the discrete poses that

are used to characterize the accuracy of the manipulator. The location of the holes relative to the locating pins and the diameter of the holes were measured with a coordinate measuring machine with a reported accuracy of $\pm 0.005$ mm. The position of the moving platform is constrained by seating a precision ball of known diameter in one of the holes machined into the jig, and also in the hole located at the center of the moving platform (see Fig. 7.5). So, by knowing the position and diameter of the hole on the jig, the diameter of the ball, and the diameter of the hole at the center of the moving platform, it is possible to determine the position of the moving platform. The resulting measurement from this procedure is considered the actual measured position of the manipulator for the discrete poses, and is compared to the position calculated from the encoder measured input link angles to provide a geometric positioning error.

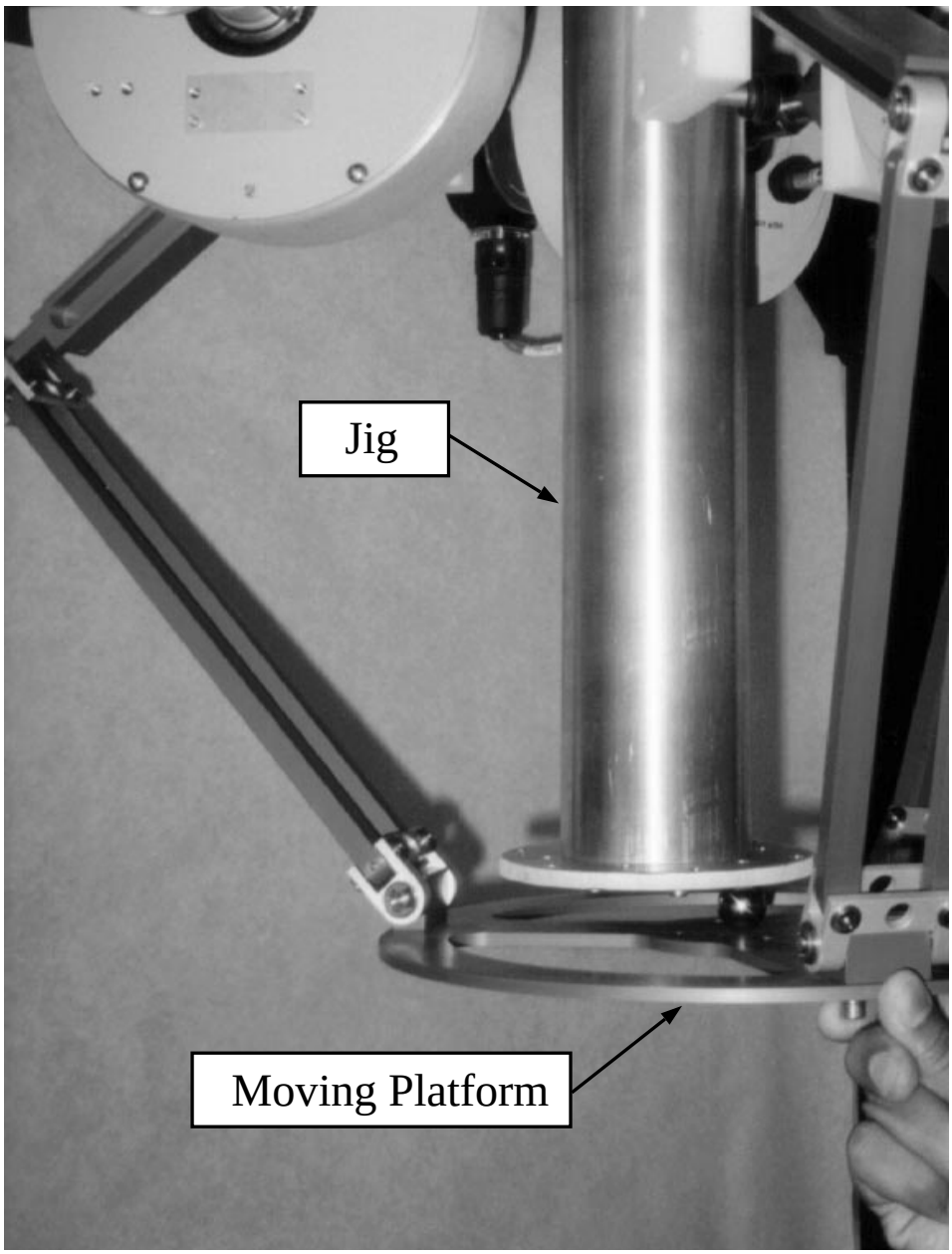



Figure 7.4: Jig setup to measure position of moving platform.

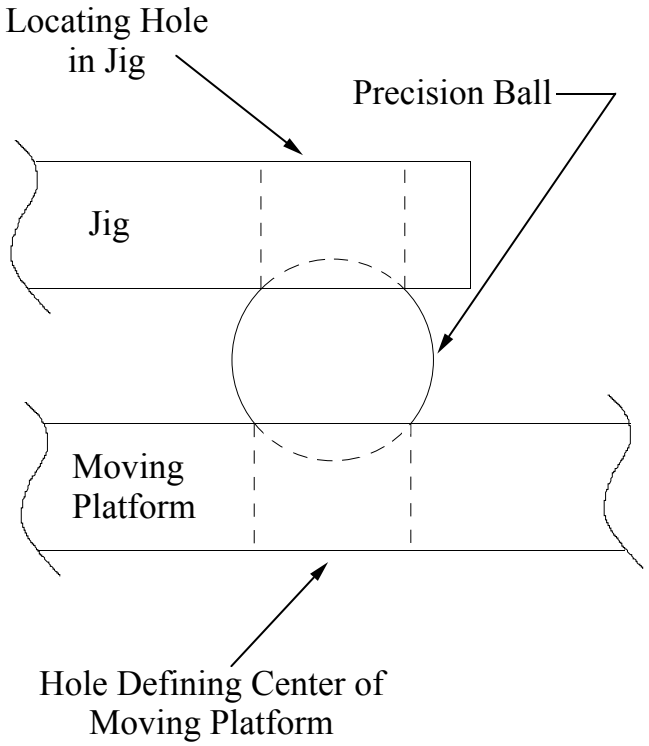



Figure 7.5: Prototype position measurement technique.

Ten different poses were tested with the prototype manipulator using the jig to constrain the moving platform. The coordinates of the moving platform as determined with the jig for each pose are given in Table 7.1.

Table 7.1: Position of moving platform as determined with measurement jig for poses used to measure geometric accuracy.

| Pose No. | $p_x$ (mm) | $p_y$ (mm) | $p_z$ (mm) |
|---|---|---|---|
| 1 | 0.686 | 44.625 | 329.433 |
| 2 | -29.263 | 48.283 | 329.486 |
| 3 | -38.400 | 20.932 | 329.260 |
| 4 | 39.180 | 22.578 | 329.209 |
| 5 | 57.183 | 2.080 | 329.016 |
| 6 | -37.059 | -22.868 | 328.874 |
| 7 | 39.467 | -20.991 | 328.828 |
| 8 | 2.065 | -44.031 | 328.658 |
| 9 | -25.862 | -49.675 | 328.633 |
| 10 | 30.825 | -47.536 | 328.620 |

The angular displacements of the three input links were measured for each pose using the encoders, and are shown in Table 7.2. Using these values, the position of the moving platform was calculated for each pose. The coordinates of the calculated position of the moving platform are listed under $p_{x,\text{calc}}$, $p_{y,\text{calc}}$, and $p_{z,\text{calc}}$ in Table 7.3. The moving platform position errors were calculated for each pose and are also shown in Table 7.3. The positioning errors were found by calculating the distance between the moving platform as measured by the jig and as calculated from the the angular displacement of the input links using the

forward kinematic relationships.

Table 7.2: Measured input link angular displacement for each test pose.

| Pose No. | $\theta_{11}$ (deg) | $\theta_{12}$ (deg) | $\theta_{13}$ (deg) |
|---|---|---|---|
| 1 | 31.5100 | 25.2888 | 39.2092 |
| 2 | 37.1548 | 22.4952 | 38.0284 |
| 3 | 37.9036 | 25.4616 | 32.5852 |
| 4 | 24.5116 | 32.3160 | 39.1228 |
| 5 | 21.6892 | 37.4424 | 38.4604 |
| 6 | 37.7020 | 32.4024 | 25.7884 |
| 7 | 24.3964 | 38.3928 | 32.9884 |
| 8 | 31.1932 | 38.5368 | 25.8460 |
| 9 | 36.5788 | 37.7592 | 22.9372 |
| 10 | 26.7004 | 41.8200 | 28.6108 |

These data suggest that the geometric accuracy of the prototype manipulator is approximately 5mm in this region of the workspace. In other words, the moving platform position calculated using the angular displacement of the input links as measured by the encoders is within 5mm of the actual position of the moving platform. This is poor accuracy compared to the accuracy that is theoretically possible. Given that the encoders have the ability to resolve the angular displacement of the input links to $\pm 0.0288$ degrees and that there are no deviations from the nominal design dimensions, a geometric positioning accuracy of 0.13 mm should be possible for the prototype in the first pose.

It is possible to significantly improve the accuracy of the prototype manipulator through calibration. Calibration involves a parameter estimation procedure,

Table 7.3: Moving platform position calculated from input link angular displacement and the associated position error.

| Pose No. | $p_{x,\text{calc}}$ (mm) | $p_{y,\text{calc}}$ (mm) | $p_{z,\text{calc}}$ (mm) | error (mm) |
|---|---|---|---|---|
| 1 | 4.295 | 46.954 | 331.211 | 4.649 |
| 2 | -25.529 | 50.579 | 331.587 | 4.861 |
| 3 | -34.902 | 22.888 | 332.069 | 4.894 |
| 4 | 42.563 | 24.160 | 330.759 | 4.044 |
| 5 | 60.643 | 3.724 | 330.459 | 4.093 |
| 6 | -33.848 | -21.285 | 332.273 | 4.937 |
| 7 | 42.748 | -19.185 | 330.888 | 4.274 |
| 8 | 5.042 | -42.824 | 331.300 | 4.159 |
| 9 | -22.992 | -48.372 | 331.709 | 4.404 |
| 10 | 33.683 | -46.419 | 330.817 | 3.774 |

where the parameters of the kinematic model (e.g. link lengths and leg orientations) are varied so as to minimize the errors that result from using the kinematic model. Ideally, the new estimates for the parameters are more representative of the true values than the original uncalibrated values for the parameters, resulting in a kinematic model that is more faithful to the actual manipulator. However, in this case a rigorous calibration isn't possible without a more general kinematic model since the current model doesn't capture the influence of many key individual parameters. For example, the kinematic model assumes that all the legs are identical and that u-axes (see Fig. 2.1) for the three legs intersect at a single point. So, if the prototype positioning errors are caused by a single link

being too long or a misalignment of the axis of an input link such that the u-axis does not pass through the origin of the coordinate system, the kinematic model would not be able to reflect that in the calculation of the position of the moving platform.

However it may be possible to make some improvements in the prototype manipulator accuracy without the general kinematics model. For example, the data in Tables 7.1 and 7.3 suggest that there might be some misalignment between the origin of the manipulator coordinate frame as defined by the locating pin on the base, and the origin of the coordinate frame as defined by the location and orientation of the input links. This can be observed from the consistent offset in the measured position error in the x and y directions (see Table 7.4). This type of misalignment could have easily occurred when the motors were mounted to the base of the manipulator.

If it's assumed that the there is a misalignment equal to the average offset, and the origin of the coordinate frame as defined by the pin is shifted by that amount in the x and y directions, then the errors are reduced as shown in Table 7.5.

Even though the accuracy is improved by approximately 1.5 mm by shifting the origin, there is still significant potential for improvement from a rigorous calibration procedure. The calibration of this type of manipulator and the development of the more general kinematic model required to accomplish the more rigorous calibration are both areas that merit future research.

Table 7.4: Moving platform position error along x, y, and z coordinate axes.

| Pose No. | x-axis error (mm) | y-axis error (mm) | z-axis error (mm) |
|---|---|---|---|
| 1 | -3.609 | -2.329 | -1.778 |
| 2 | -3.734 | -2.296 | -2.101 |
| 3 | -3.498 | -1.956 | -2.809 |
| 4 | -3.383 | -1.582 | -1.550 |
| 5 | -3.460 | -1.644 | -1.443 |
| 6 | -3.211 | -1.583 | -3.399 |
| 7 | -3.281 | -1.806 | -2.060 |
| 8 | -2.977 | -1.207 | -2.642 |
| 9 | -2.870 | -1.303 | -3.076 |
| 10 | -2.858 | -1.117 | -2.197 |
| Avg. | -3.288 | -1.682 | -2.306 |

Table 7.5: Moving platform position errors calculated with shifted origin.

| Pose No. | adjusted error (mm) |
|---|---|
| 1 | 1.919 |
| 2 | 2.234 |
| 3 | 2.830 |
| 4 | 1.556 |
| 5 | 1.454 |
| 6 | 3.401 |
| 7 | 2.064 |
| 8 | 2.702 |
| 9 | 3.127 |
| 10 | 2.309 |

## 7.4 Summary

The design of the prototype manipulator that was built to support the research is described. The actuation of the prototype is achieved by three permanent magnet DC motors that are directly coupled to the input links. Sensing of the input link angular displacement for each leg is accomplished by an incremental encoder geared to each motor so as to provide a resolution of 0.0288 degrees of input link displacement per encoder count. Drawings of the key links are provided in Appendix D.

The supporting systems required to operate the manipulator under computer control are also described. These include the controller prototyping system assembled by Kantor (1995) and the trajectory generator written by Remi Taulemesse.

Finally, the accuracy of the manipulator is considered. A method for measuring the position of the moving platform using a measurement jig is described, and is used to estimate the geometric positioning accuracy of the prototype. Using this approach, the positioning accuracy of the prototype was estimated to be approximately 3.5 mm over the tested region of the workspace. This demonstrated accuracy is an order of magnitude worse than what is theoretically possible. Accordingly, the need for better calibration of the prototype, and a more general kinematic model to support the calibration is discussed.

# Chapter 8

# Controller Design and Validation

## 8.1 Introduction

Three different types of controllers are developed and tested for the prototype manipulator to explore the advantages, disadvantages, and performance of each type as applied to this parallel manipulator. Each of these comes from three different broad categories of controllers and has significantly different requirements regarding the prior knowledge of the manipulator dynamics and the time required to develop and implement.

The first type of controller is based upon the classical single-input single-output proportional-integral-derivative (PID) type controller. This approach views each actuator of the manipulator independently, and essentially ignores the highly coupled, non-linear nature of the manipulator. Implementation of the PID controller is simple and in general provides reasonable performance. This is particularly true in the situation where link actuation is provided by a motor with a large gear ratio existing between the motor and the driven link so that the manipulator dynamics is dominated by the inertia of the motor rotor. However,

for the prototype manipulator there is no gear train between the actuator and the input link. So, the prototype manipulator should highlight the deficiencies of this type of control.

The second controller is a computed torque type, where a model of the inverse dynamics of the manipulator is used to estimate the actuation required for the manipulator to achieve a desired trajectory. Since this type of controller takes into account the non-linear and coupled nature of the manipulator, the potential performance of this type of controller should be quite good. However, the difficulty with this type of controller is that it requires a good model of the manipulator inverse dynamics and good estimates of the model parameters, making this controller more complex and difficult to implement than the classic servo type controller.

The third type is an iterative learning controller. This type of controller is based upon the repeated trials of a manipulator attempting to follow a given desired trajectory. After each trial a learning rule is applied to the measured trajectory tracking errors to produce a new actuation schedule that reduces the trajectory tracking errors for the next trial. The advantage of this type of controller is that the controller still accounts for the complex dynamics of most manipulators while not requiring an accurate model of the manipulator dynamics. A significant disadvantage of this type of controller is the time required for the necessary trials that allows the controller to converge to acceptably small trajectory tracking errors for each new trajectory.

This chapter presents a section for each type of controller developed for the prototype manipulator, with each section providing a brief description of the controller, a discussion of the design of the controller, and finally the experimen-

tal results of the controller as applied to the prototype manipulator as it follows a sample trajectory.

## 8.2 PID Control

The PID controller is a single-input, single-output (SISO) controller that produces a control signal that is a sum of three terms. The first term is proportional (P) to the positioning error, the second term is proportional to the integral (I) of the error, and the third is proportional to the derivative (D) of the error. The PID controller is the most common type of control algorithm used in engineering practice, with one survey in 1991 suggesting that 90% of controllers in Japan were of the PID type (Yamamoto and Hashimoto, 1991). PID controllers are used because the application of the PID controller is relatively straight forward, while providing reasonable results for a wide range of applications.

Since the PID controller is a SISO controller, and the manipulator has three degrees of freedom, three separate PID control loops are used to control the prototype manipulator, with each one controlling the position of an input link. The desired angular positions of the input links are determined by mapping the desired moving platform trajectory, as given in Cartesian space, into joint space by using the inverse kinematics relationships.

The key to obtaining the best possible performance from a PID controller is the selection of the gains that determine the influence of the three control terms on the control signal. Many gain tuning strategies have been developed. Generally, these tuning strategies are based upon the open-loop system step responses, open-loop frequency responses, optimization techniques, or other analytical ap-

proaches. Several of these gain tuning strategies, as well as a general discussion of PID controllers, are presented by Åström and Hägglund (1995).

The gains used to control the prototype manipulator were selected using an optimization technique. This was done by developing a simple model for a single input link that was used to simulate the closed-loop response of the input link to a step input at the desired position. This simulation was then used with an optimization routine to find a set of gains that minimized an objective function that considered the position error of the input link and the actuating torques. The application of this approach and the results are presented in the following sections.

### 8.2.1 Closed-Loop Model

The block diagram of the closed-loop model for each input link of the manipulator is shown in Fig. 8.1, where $G_{\mathrm{C}}$ represents the dynamics of the controller, $G_{\mathrm{amp,motor}}$ represents the transfer function of the power amplifier and motor, $G_{\mathrm{link}}$ represents the dynamics of the single-link dynamics model of the manipulator developed in section 6.4, $\tau$ is the torque applied to the input link, and $e$ is the error between the desired input link position, $\theta_{1,\mathrm{d}}$, and the actual position of the input link, $\theta_{1,\mathrm{act}}$. To determine the response of this closed-loop system, it's necessary to model each of these blocks.

The controller block, $G_{\mathrm{C}}$, can be represented in the Laplace domain by the following equation for a PID controller,

$$G_{\mathrm{C}}(s) = k_p + k_d\, s + \frac{k_i}{s}\,, \tag{8.1}$$

where $k_p$, $k_d$, and $k_i$ are the controller gains. The selection of these gains deter-

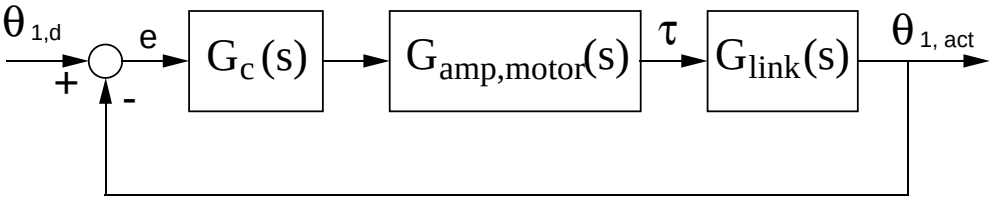



Figure 8.1: PID controller block diagram for each input link.

mine the performance characteristics of the controller.

The block representing the amplifier and motor dynamics, $G_{\text{amp,motor}}$, is modeled as a simple gain since the amplifiers are configured for current control, and the torque supplied by the motor is directly proportional to the current supplied by the amplifier. This is a reasonable approach since the response of motor torque to the control signal is much faster than the response of the manipulator input link to the torque from the motor. Any non-linearities of the motor are also neglected. The value for the gain for $G_{\text{amp,motor}}$ was determined experimentally to be 0.99 $\text{N} \cdot \text{m/volt}$ for the prototype manipulator.

To determine a relationship for $G_{\text{link}}$, the single-link dynamic model for is assumed as shown in Fig. 6.2, and modeled by Eq. (6.34). Taking the Laplace transformation of Eq. (6.34) provides the following relationship for $G_{\text{link}}$:

$$\frac{\hat{\theta}_{1i}(s)}{\hat{\tau}_i(s)} = \frac{1}{\alpha s^2 + c_d s + \beta \sin(\frac{\pi}{4})} \,, \tag{8.2}$$

where $\hat{\theta}_{1i}$ and $\hat{\tau}_i$ represent the differential angular displacement of the link and the differential torque applied to the link about the operating point of $\theta_{1i,o} = \frac{\pi}{4}$ and $\tau_{i,o} = -\beta \cos(\frac{\pi}{4})$ where $\theta_{1i}$ and $\tau_i$ are shown in Fig. 6.2.

The numeric values for the coefficients for Eq. (8.2) are estimated for the prototype manipulator from measurements made of the links and information

obtained from the motor data sheets. These estimates produced the following values:

$$\begin{aligned} \alpha &= 0.0196 \text{ N} \cdot \text{m} \cdot \text{s}^2 \\ \beta &= 0.797 \text{ N} \cdot \text{m} \\ c_d &= 0.0027 \text{ N} \cdot \text{m} \cdot \text{s} \end{aligned}$$

Now that a model of the closed-loop system for each input link of the prototype manipulator is defined, it's necessary to select the gains that achieve the desired performance. The selection of these gains is discussed in the next section.

### 8.2.2 Gain Selection

The gains for the controller are selected using a numerical optimization approach. Using the closed-loop system shown in the Fig. 8.1, the step response of the system is simulated using the Xmath software package. The results of that simulation are used to minimize an objective function, where the design variables are the gains, $k_p$, $k_d$, and $k_i$. The objective of the optimization is to minimize the following function:

$$f(e, \hat{\tau}, t) = q \int_0^t e(t)^2 \, dt + p \int_0^t \hat{\tau}(t)^2 \, dt, \tag{8.3}$$

subject to the constraint of no overshoot in position, where $e(t)$ is the position error of the input link, $\hat{\tau}(t)$ is the differential torque applied to the input link, and $p$ and $q$ are scalar quantities that are selected to scale the two terms of the objective functions. For this controller design, $q$ and $p$ are selected to approximate an even weighting of the two control terms by setting them equal to the

reciprocal of the square of the largest value expected during normal operation:

$$q = \frac{1}{e_{max}^2} = 3280 \text{ (rad)}^{-2} \tag{8.4}$$

$$p = \frac{1}{\hat{\tau}_{max}^2} = .0046 \text{ (N} \cdot \text{m)}^{-2} \tag{8.5}$$

The no overshoot constraint is common in robotic applications, and is applied in order to minimize the chance of unintended collisions with obstacles that are outside the planned trajectory, and also to attempt to keep the end-effector from approaching undesirable locations in the workspace, such as singular locations.

The optimization procedure resulted in gain values of $k_p = 63.5$, $k_i = 15.0$, and $k_d = 1.6$. The initial guess for the set of gains was $k_p = 100$, $k_i = 10$, and $k_d = 1.0$. In Fig. 8.2, which shows the step responses for each of these set of gains, a significant improvement in the step response performance of the input link can be observed. The settling time of the arm is reduced and the overshoot associated with the initial set of gains is eliminated. The optimized gains also require less torque from the actuators as can be seen in Fig. 8.3, where the simulated actuator torques for the initial set of gains and the optimized gains are plotted for the step response.

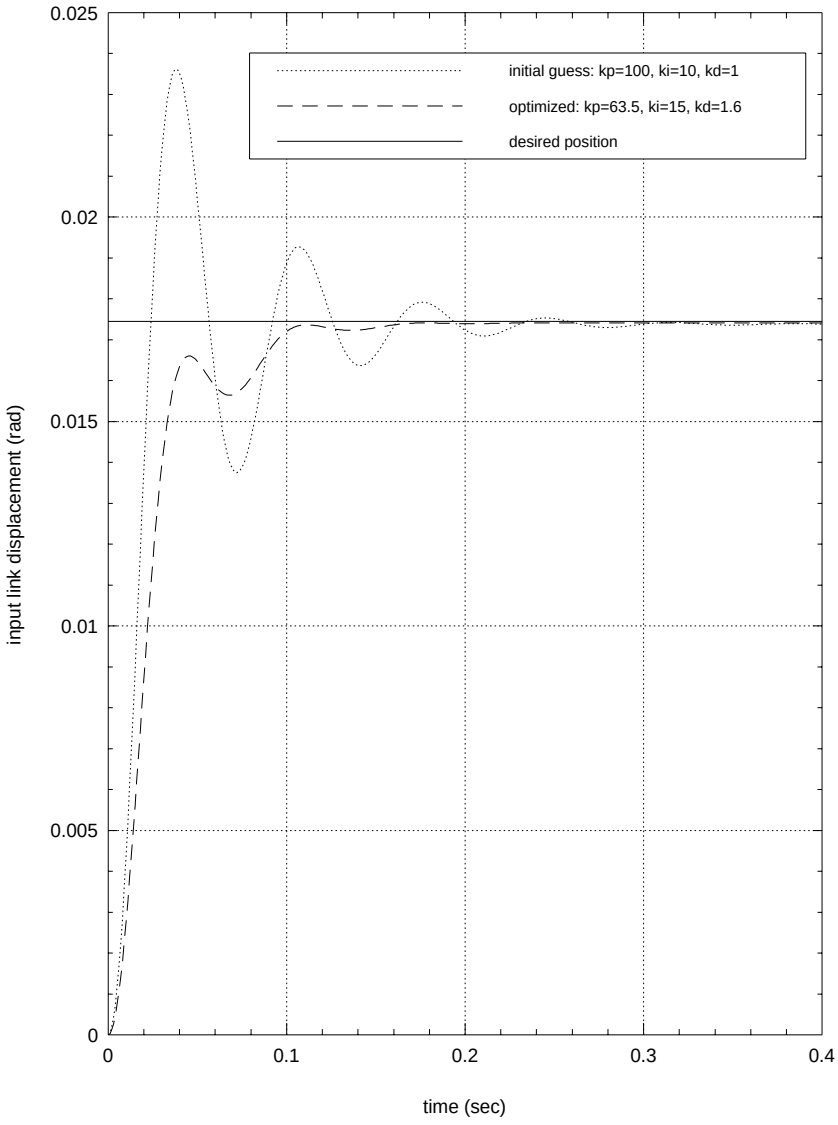



Figure 8.2: Step response for closed-loop system with optimized gains and initial guesses for the gains.

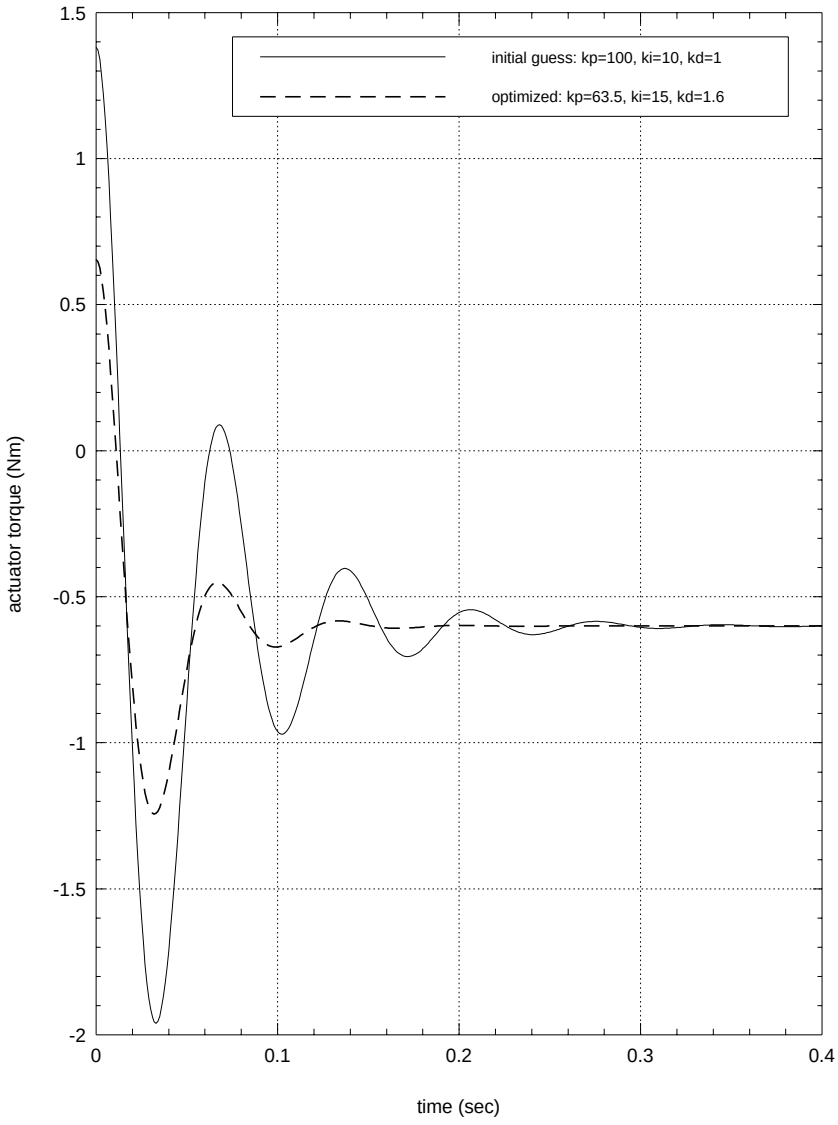



Figure 8.3: Actuator torques for the closed-loop step response with optimized gains and initial guesses for the gains.

### 8.2.3 Integrator Windup

Systems that use PID controllers are susceptible to a problem called integrator windup. This can occur when the actuator reaches a limit of the actuation that it can supply, such that the signal from the controller stops having an impact on the system actuation. When this happens, the integrating term of the PID controller continues to integrate the error while the system actuation remains constant. This integrated error term can become large, and once the system returns to a state where the the actuator is no longer saturated there can be a substantial delay before the integrating term of the controller returns to normal operation. During this delay, while the integrating term is reseting, the performance of the controlled system will most likely be poor, usually resulting in excessive overshoot.

In an effort to mitigate the problem associated with integrator windup, a back-calculation approach is applied as proposed by Fertik and Ross (1967) and described in more detail by Åström and Rundqwist (1989). The implementation of the back-calculation approach requires a change in the controller block shown in Fig. 8.1 and represented by Eq. (8.1) to the form shown in Fig. 8.4.

The back-calculation approach adds a block that models actuator saturation, and creates another feedback signal based upon this model. This feedback signal, $e_s$, is zero when the actuator model predicts that the actuator is not saturated. When the actuator is saturated, $e_s$ reduces the influence of the integrating term of the controller. Selection of $T_t$ determines how quickly the integrating term is reduced. A value suggested for $T_t$ by Åström and Hägglund (1995) is:

$$T_t = \sqrt{\frac{k_d}{k_i}} \tag{8.6}$$

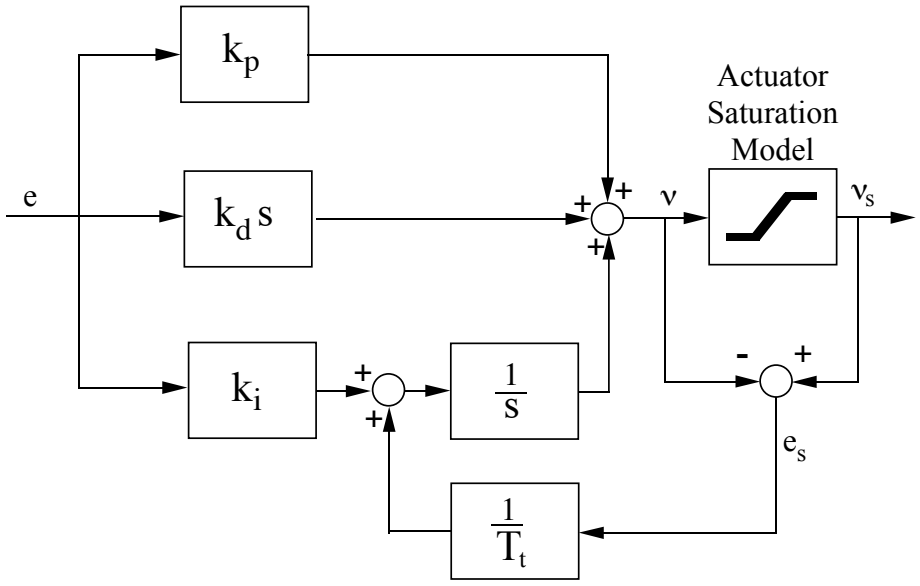



Figure 8.4: PID controller block with changes made to correct integrator windup.

This value of $T_t$ results in a control system where the response of the anti-windup loop has a shorter time constant than the integrating term and a longer time constant than the derivative term of the controller. This offers a compromise between quick relief from integrator windup and sensitivity to very short actuator saturations that may result from the derivative term reacting to sudden changes of error. For the PID controller used for the prototype manipulator, Eq. (8.6) was applied to find $T_t = 0.33$. The performance of the prototype manipulator using this control scheme is discussed in the next section.

### 8.2.4 PID Control Experimental Results

The performance of the prototype manipulator and the associated PID control system is demonstrated by providing a sample trajectory to the controller, and then measuring the positioning errors of the input links as it attempted to follow that trajectory. The sample trajectory selected is the same trajectory as was

used for the numerical example used to illustrate the various dynamics models. The sample trajectory has the platform moving in a series of three straight lines starting from rest at a position of (0,0,400). The first segment of the trajectory has the platform moving from (0,0,400) to (0,0,350) in 0.4 seconds. The remainder of the trajectory has the platform moving from (0,0,350) to (50,50,350) in 0.8 seconds, and finally on to (50,50,400) in 0.8 seconds, where are the lengths are given in millimeters. During each segment of trajectory, of the trajectory the moving platform starts and finishes with zero velocity and accelerates or decelerates at 245.2 cm/sec$^2$. The angular displacement required of each input link to follow the example trajectory is shown in Fig. 6.3. The velocity of the moving platform as it follows the example trajectory is shown in Fig. 6.4.

The joint errors at the input link for the manipulator as it attempted to follow the sample trajectory are shown in Fig. 8.5.

From the input link errors shown in Fig. 8.5, it's possible to gain some insight to the positioning error of the moving platform. In this case, the position error of the moving platform is not directly measured, but the position error due to the joint errors of the input links can be estimated using the Jacobian and is shown in Fig. 8.6.

The position error of the moving platform degrades as it follows the sample trajectory. There are two significant causes for this decline in performance as the platform follows the sample trajectory. The first is that as the manipulator follows this sample trajectory, the single link dynamic model used to design the controller deviates further from the operating conditions used to model the system, and the interaction between the legs that are not accounted for in the model become more important.

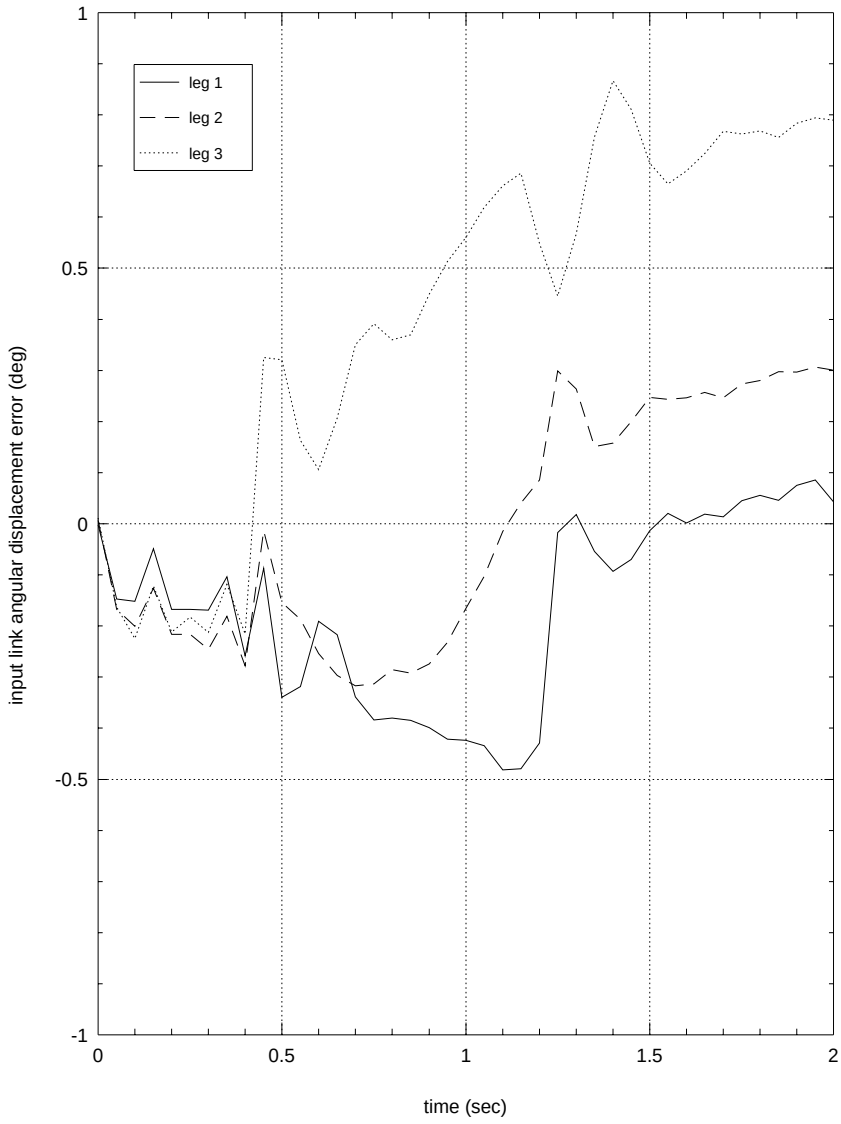



Figure 8.5: Input-link joint angle error obtained from the PID controller.

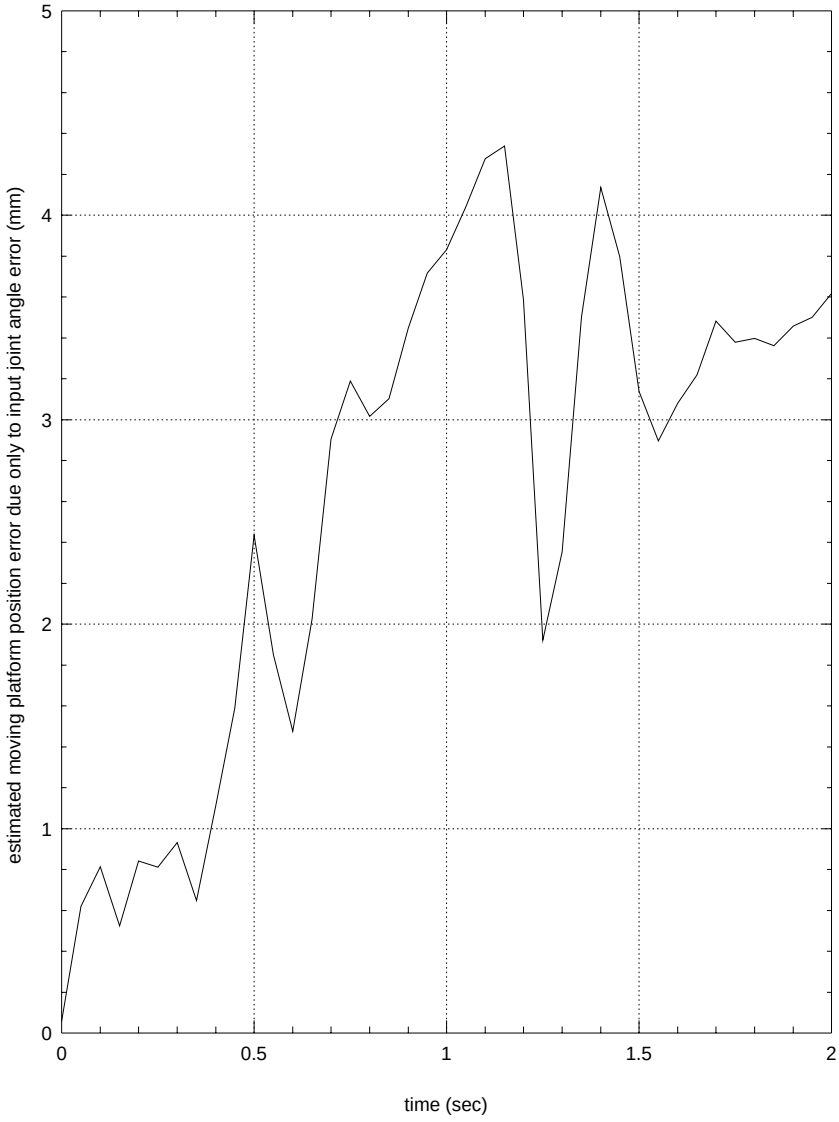



Figure 8.6: Moving platform position error as estimated from the input link joint errors obtained from the PID controller.

Accordingly, the performance of the controller for each input link degrades as the deviation from the assumed dynamic model increases. The second reason, which also compounds the errors from the prior reason, is associated with the changing configuration of the manipulator as it follows the sample trajectory. The condition number of the Jacobian matrix for the manipulator changes as the manipulator follows the sample trajectory as shown in Fig. 8.7. It can be observed in Fig. 8.7 that from $t = 0.4$ until $t = 2.0$, the condition number of the Jacobian matrix steadily increases. Accordingly, the amplification of the input link errors onto the moving platform error during this period of time also increases. Moreover, the nature of the force transmission between the moving platform and the actuators change such that disturbances at the moving platform are also magnified at the actuators.

There are several approaches available to improve the trajectory tracking performance of the prototype. One approach is to develop a control scheme that employees some form of external metrology to directly measure the position of the moving platform, and then control the actuators versus that signal. This should help mitigate the problems associated with the mapping of joint errors onto the position errors by the Jacobian matrix. Other approaches include the application of control schemes based on more complete dynamic models of the manipulator, such as a computed torque controller. The results of applying this type of controller to the manipulator is discussed in the next section.

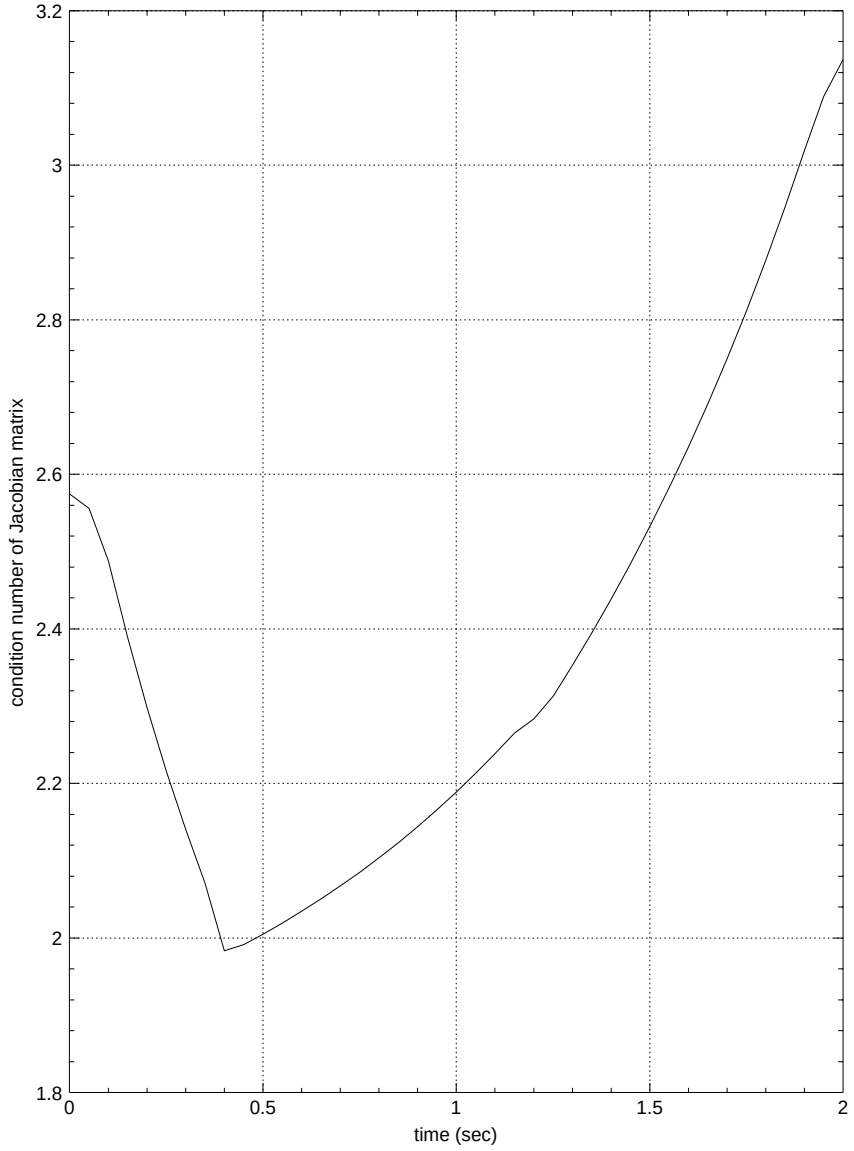



Figure 8.7: Condition number of the Jacobian matrix for the prototype manipulator as it follows the sample trajectory.

## 8.3 Computed Torque Control

A second type of controller is developed for the prototype manipulator to examine if it is possible to improve the performance of the trajectory tracking of the prototype manipulator by utilizing a more complete understanding of the manipulator dynamics in the controller design. This second controller employs a computed torque control approach, and it uses a model of the manipulator dynamics to estimate the actuator torques that will result in the desired trajectory. An outer control loop is applied to the system to correct any trajectory tracking errors that result from the computed torques. The application of the computed actuator torques minimizes the non-linearities of the closed loop system, and should improve the performance of the controller. The disadvantage of this approach is that it requires a reasonably accurate and computationally efficient model of the inverse dynamics of the manipulator to function as a real time controller.

### 8.3.1 Computed Torque Control Design

The dynamics model developed in chapter 6 using the direct application of Newton-Euler equations of motion given by Eq. (6.7) is used as the inverse dynamics model for the computed torque controller:

$$\bar{\tau} = \mathbf{M}(\mathbf{q})\ \ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\ \dot{\mathbf{q}} + \mathbf{G}(\mathbf{q})\ , \tag{8.7}$$

where:

$$\mathbf{q} = \begin{bmatrix} \theta_{11} \\ \theta_{12} \\ \theta_{13} \end{bmatrix}, \qquad \bar{\tau} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix},$$

$\mathbf{M}$ is a $3 \times 3$ symmetric matrix capturing the mass information of the manipulator, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is a $3 \times 3$ matrix containing the centrifugal, coriolis, and viscous friction terms, and $\mathbf{G}(\mathbf{q})$ is a $3 \times 1$ array capturing the gravitational terms of the dynamics.

For the implementation of the controller, Eq. (8.7) is rewritten as:

$$\bar{\tau} = \mathbf{M}(\mathbf{q})\ \ddot{\mathbf{q}} + \mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}), \tag{8.8}$$

where $\mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\ \dot{\mathbf{q}} + \mathbf{G}(\mathbf{q})$.

Using the computed torque approach with a proportional-derivative (PD) outer control loop, the applied actuator torques are calculated at each time step using the following computed torque law as described by Lewis et al. (1993):

$$\bar{\tau}_{cp} = \mathbf{M}\left[\ddot{\mathbf{q}}_d + \mathbf{K}_d\ \dot{\mathbf{e}} + \mathbf{K}_p\ \mathbf{e}\right] + \mathbf{N}\ , \tag{8.9}$$

where $\bar{\tau}_{cp}$ is the computed torque applied to input links, $\mathbf{K}_d$ is the diagonal matrix of the the derivative gains, $\mathbf{K}_p$ is the diagonal matrix of the proportional gains, and $\mathbf{e}$ is the array of the position errors of the input links, $\mathbf{e} = \mathbf{q}_d - \mathbf{q}$. A diagram of the computed torque scheme is shown in Fig. 8.8.

To show that the computed torque control scheme linearizes the controlled system, the torques computed by Eq. (8.9) are substituted into Eq. (8.8), yielding:

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{M}\ddot{\mathbf{q}}_d + \mathbf{M}\left[\mathbf{K}_d\ \dot{\mathbf{e}} + \mathbf{K}_p\ \mathbf{e}\right]\ . \tag{8.10}$$

Figure 8.8: Diagram of computed torque approach.

Pre-multiplying each term of Eq. (8.10) by $\mathbf{M}^{-1}$, and substituting the relationship, $\ddot{\mathbf{q}}_d - \ddot{\mathbf{q}} = \ddot{\mathbf{e}}$, provides the following linear relationship for the error:

$$\ddot{\mathbf{e}} + \mathbf{K}_d\, \dot{\mathbf{e}} + \mathbf{K}_p\, \mathbf{e} = 0. \tag{8.11}$$

This relationship can be used to select the gains to give the desired nature of the closed-loop error response since the solution of Eq. (8.11) provides a second order damped system with a natural frequency of $\omega_n$, and a damping ratio of $\zeta$, where:

$$\omega_n = \sqrt{k_p}\,, \qquad \zeta = \frac{k_d}{2\,\sqrt{k_p}}\,, \tag{8.12}$$

and $k_p$ and $k_d$ are the diagonal elements of $\mathbf{K}_p$ and $\mathbf{K}_d$.

### 8.3.2 Computed Torque Control Experimental Results

The trajectory tracking performance of the manipulator as controlled with the computed torque controller was demonstrated using the same sample trajectory as was used for the PID controller, and described in section 8.2.4. The values

for the gain matrices, $\mathbf{K}_p$ and $\mathbf{K}_d$, were determined experimentally by initially setting the gains to maintain the following relationship:

$$k_d = 2\sqrt{k_p}, \tag{8.13}$$

such that the error response is critically damped. These gains were adjusted using a trial and error process until a desirable system response was achieved, at gain values of $k_p = 1750$ and $k_d = 10$. The system response was also improved by increasing the gain of the motor amplifiers by approximately 20%. The results obtained using these values are shown in Figs. 8.9 and 8.10, where Fig. 8.9 shows the position error of each of the input links as the manipulator attempts to follow the desired trajectory and Fig. 8.10 shows the absolute position error of the moving platform as estimated by transforming the joint angle errors into position error using the inverse of the Jacobian matrix. During this example the $\mathbf{M}$ and $\mathbf{N}$ matrices were approximated beforehand by evaluating the elements of the matrices at the desired positions and velocities of the input links at each time step as opposed to the actual positions and velocities at each time step. For the example trajectory, this approach works well since there is little deviation from the desired trajectory. However, in general another approach should be employed to evaluate $\mathbf{M}$ and $\mathbf{N}$, since the difference between the desired trajectory and actual position may not always be small. It is computationally prohibitive to completely solve the system dynamics during each time step for the real time controller since it requires solving the complete forward kinematics problem at each time step. Accordingly, a look-up table approach is a reasonable alternative for real-time computation of the elements of $\mathbf{M}$ and $\mathbf{N}$.

Overall, Figs. 8.9 and 8.10 show improved tracking performance for the computed torque controller as compared to the PID controller performance as shown

in Figs. 8.5 and 8.6 for the sample trajectory. The most striking difference between the computed torque and PID controller performances besides the overall reduction in error is the decrease in the spread of the joint errors during the trajectory. This is a result of the computed torques canceling the non-linear components of the controlled system. This cancelation is not complete since the error does not asymptotically approach zero as predicted by Eq. (8.11). However, it can be observed that the frequency of the oscillation, 6.7 Hz, of the error predicted from Eq. (8.12) is approximately the same frequency of the error oscillation displayed by the prototype while tracking the trajectory.

It also appears from Figs. 8.9 and 8.10 that the computed torque controller could benefit from additional damping. Furthermore, Eq. (8.13) suggests that the derivative gain should be increased to achieve the desired critical damping. However, when the derivative gain was increased for the actual controller, a vibration was introduced to the prototype. It's not clear what caused these vibrations, but it is suspected to be related to the discrete nature of the position encoders and the difficulties of amplifying the differentiated encoder signal to obtain the angular velocity of the input link in the feedback loop.

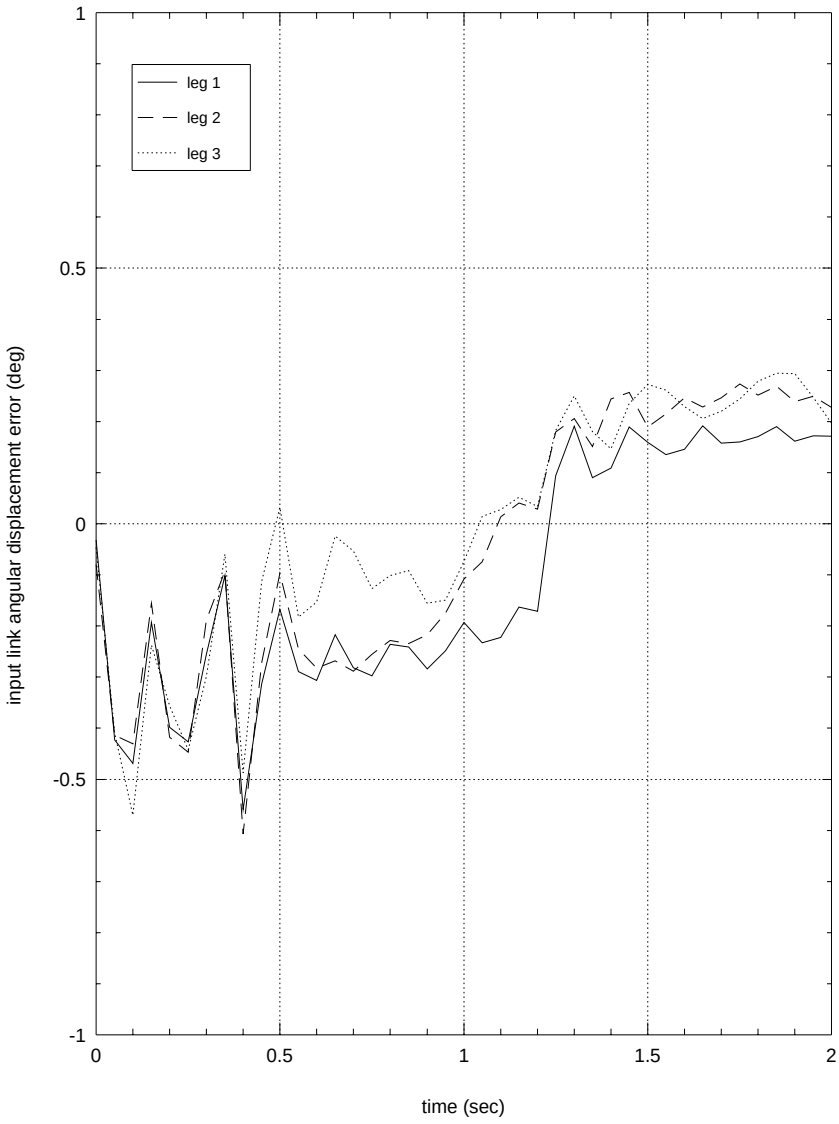



Figure 8.9: Input-link joint errors obtained from the computed torque controller.

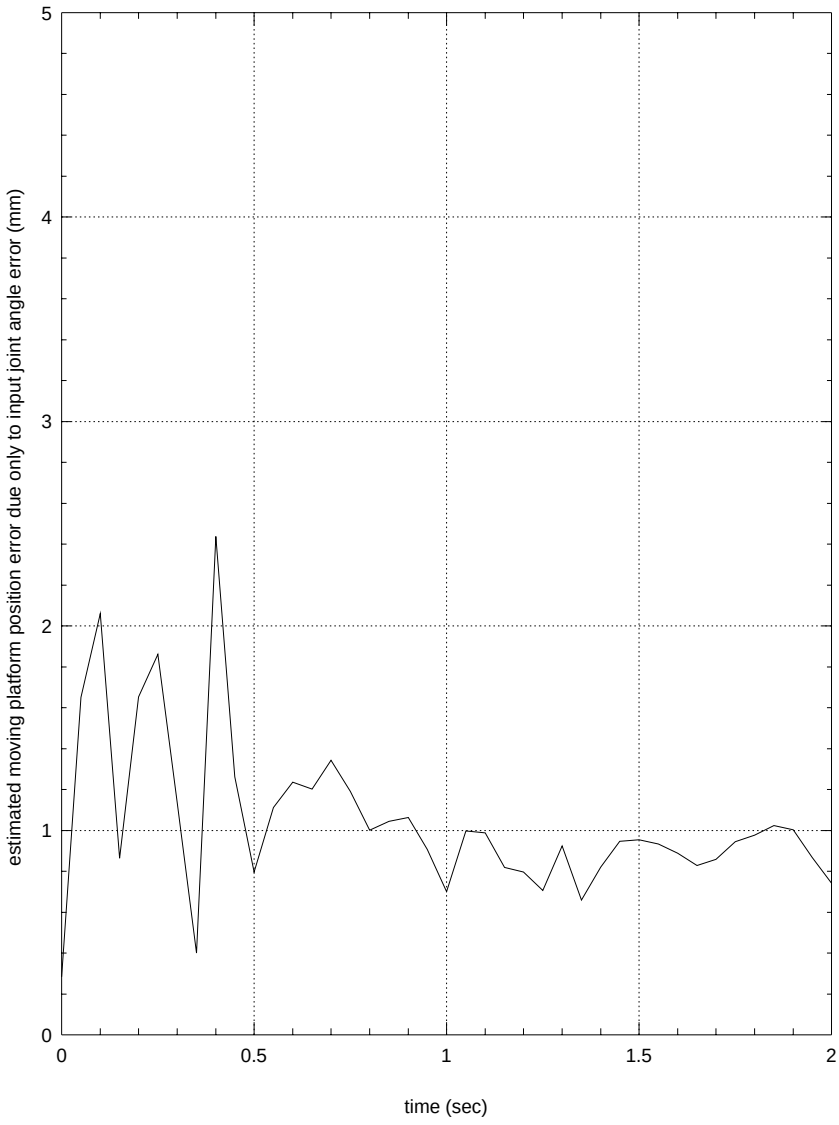



Figure 8.10: Moving platform position error as estimated from the input link joint errors obtained from the computed torque controller.

## 8.4 Iterative Learning Control

The third and final controller developed for the manipulator is an iterative learning controller. Like the computed torque controller, the iterative learning controller takes into account the dynamics of the manipulator to improve the trajectory tracking performance of the manipulator. However, unlike the computed torque controller the manipulator dynamics information does not come from an explicit mathematical model of the manipulator dynamics. Instead, the dynamics information for a given desired trajectory is learned from repeated attempts to follow that trajectory. Accordingly, the learning controller requires multiple trials of a manipulator attempting to follow a desired trajectory. The trajectory tracking errors are recorded for each trial and are used as an input to a learning rule to modify the actuation of the manipulator in the next trial so as to improve the trajectory tracking performance. This iterative improvement process continues until the trajectory tracking errors are acceptably small. Learning based controllers have been used to control computer disk drives (Tomizuka et al., 1989), and have been shown to be a feasible controller for manipulators with flexible joints (Wang, 1995).

The main advantages of the iterative learning control approach are that a rigorous model of the manipulator dynamics and accurate values for the model parameters are not required to develop the controller, and that it is relatively simple to implement. Furthermore, this approach considers the complete manipulator dynamics, including phenomenon that are difficult to model, such as backlash and friction. However, there are several disadvantages of this approach relative to the other controllers presented in this chapter. First, the system being controlled may need to accommodate a large number of trials to converge to an

acceptable solution. Second, each new trajectory requires another set of trials for the manipulator to converge to the new trajectory. Third, once the trajectory has been learned, disturbances are not rejected by the controller since it behaves as an open loop controller while it tracks the trajectory unless an additional closed loop controller is added to augment the iterative learning controller.

### 8.4.1 Iterative Learning Control Design

Some of the initial work on iterative learning controls was presented by Arimoto et al. (1984) where they proposed an iterative "betterment process" for both linear and non-linear time invariant systems. The non-linear system considered by Arimoto et al. (1984), assumes the following form:

$$\dot{x}(t) = f(x(t), t) + \mathbf{B}\, u(t)\ , \tag{8.14}$$

$$y(t) = \mathbf{C}\, x(t)\ , \tag{8.15}$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^r$, $y(t) \in \mathbb{R}^r$, $\mathbf{B} \in \mathbb{R}^{n \times r}$, $\mathbf{C} \in \mathbb{R}^{r \times n}$, for $t \in [0, T]$ and function $f$:$\mathbb{R}^n \rightarrow \mathbb{R}^n$.

For control of the prototype manipulator, the output is considered to be the joint angles of the input links:

$$y(t) = \begin{bmatrix} \theta_{11}(t) \\ \theta_{12}(t) \\ \theta_{13}(t) \end{bmatrix}\ .$$

The desired outputs, $y_d(t)$, are found by mapping the desired trajectory of the moving platform into joint space. The system inputs, $u(t)$, are the torques applied to the input link by the actuators, while $\mathbf{B}$ and the function $f$ are considered unknown.

The "betterment process" proposed by Arimoto et al. (1984) to control this system is given by the following learning rule:

$$u_{i+1}(t) = u_i(t) + \Gamma\, \dot{e}_i(t)\ , \tag{8.16}$$

where $u_i(t)$ is the input of the system for the $i^{th}$ trial, $e_i(t) = y_d(t) - y(t)$ is the output tracking error for the $i^{th}$ trial, and $\Gamma \in \mathbb{R}^{r\times r}$ is a matrix of learning parameters. Arimoto et al. (1984) showed that this learning rule caused $\dot{e}(t)$ to converge to zero for non-linear time invariant systems under the following conditions:

C1: the function $f(\cdot,\cdot)$ satisfies a Lipschitz continuity condition so that there is a $\alpha < \infty$ such that for all $t \in [0,T]$,
$\|f(x_1,t) - f(x_2,t)\| \ \leq\ \alpha\, \|x_1 - x_2\|$ ,

C2: $\|\mathbf{I} - \mathbf{CB}\Gamma\|_\infty < 1$ for all $t \in [0,T]$,

C3: $u_0(t)$ and $y_d(t)$ are continuously differentiable on $[0,T]$,

C4: $y_d(0) = \mathbf{C}x(0)$ ,

where $\mathbf{I}$ is the identity matrix. Hence, the learning rule given by Eq. (8.16) can be applied iteratively to determine the system inputs that will achieve a desired output.

The key to implementing the iterative learning rule described in Eq.(8.16), aside from starting at the desired initial conditions and satisfying the continuity conditions, is the selection of the elements of $\Gamma$ so that condition 2 is satisfied. In this case, a diagonal $\Gamma$ matrix was assumed with equal values for each diagonal element. Improved convergence of the learning controller may be possible by properly selecting a full matrix, but that was not explored. The values for the diagonal elements were determined in a trial and error fashion during experiments with the actual manipulator, with reasonable performance being obtained

at $\Gamma(i,i) = 0.1$ N $\cdot$ m $\cdot$ s for $i = 1, 2$, and 3. This $\Gamma$ matrix was used to demonstrate the learning controller with the prototype manipulator. The results are presented in the next section.

### 8.4.2 Iterative Learning Control Experimental Results

The trajectory tracking performance of the manipulator as controlled with the iterative learning controller is demonstrated using the same sample trajectory as was used for both the PID and computer torque controllers, and as described in section 8.2.4.

The results obtained using the iterative learning controller as described in the previous section after 100 trials are shown in Figs. 8.11 and 8.12, where Fig. 8.11 shows the position error of each of the input links as the manipulator attempts to follow the desired trajectory and Fig. 8.12 shows the absolute position error of the moving platform as estimated by transforming the joint angle errors into position error using the inverse of the Jacobian matrix. Figures 8.11 and 8.12 demonstrate trajectory tracking performance after 100 trials that was on par with, or slightly better than was obtained with the computed torque controller. Even better performance should be attainable with more iterations as the error continues to converge toward zero.

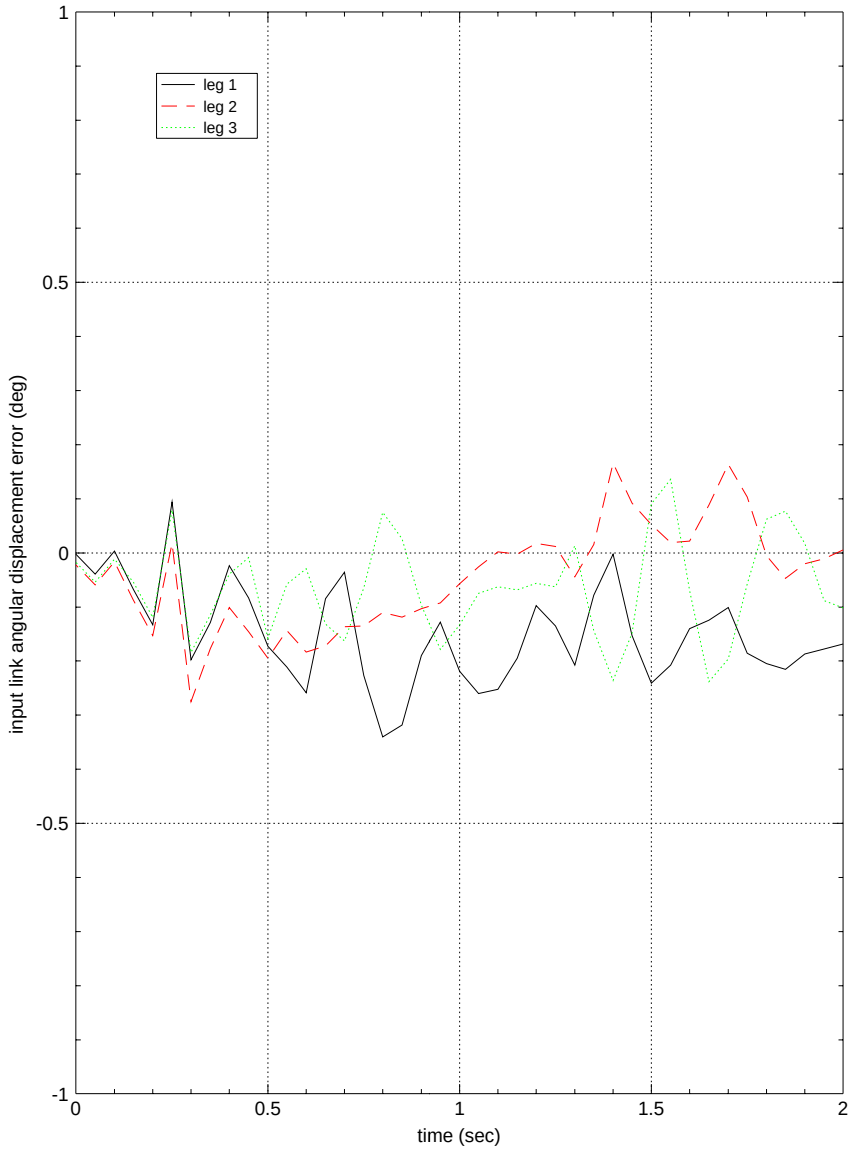



Figure 8.11: Input-link joint errors obtained from the iterative learning controller after 100 trials.

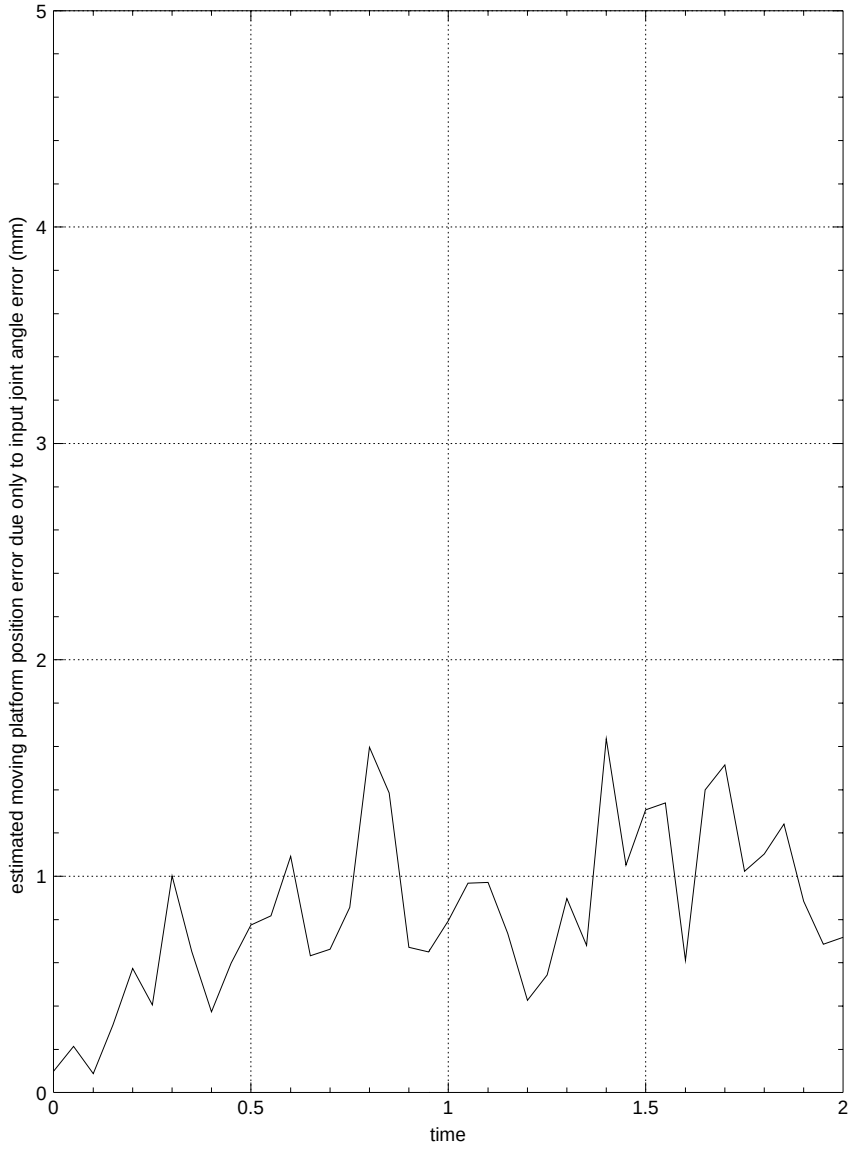



Figure 8.12: Moving platform position error as estimated from the input link joint errors obtained from the iterative learning controller.

Some insights to the convergence of the trajectory tracking error are gained by observing Figs. 8.13-8.15, where the tracking error for the three legs are plotted for the initial trial, the $50^{th}$ trial, and the $100^{th}$ trial. The actuator torques for the initial trial were obtained using a PID controller with a large integral gain. It can be observed in Figs. 8.13 -8.15 that the error converges most rapidly during the very early portions of the trajectory, up to approximately 0.2 seconds. From an intuitive point, this is reasonable since during a trial the controller is in an open loop configuration and any errors in the early portions of the trajectory will propagate and accumulate throughout the rest of the trajectory. This is also reasonable from a mathematical point of view since Arimoto et al. (1984) showed convergence to the desired trajectory by proving that there exist positive constants $\lambda$ and $\rho$, where $0 \leq \rho < 1$ such that:

$$\|\dot{e}_{i+1}\|_\lambda \leq \rho \|\dot{e}_i\|_\lambda \ , \tag{8.17}$$

when conditions C1-C4 are satisfied, where the $\lambda$-norm is defined by:

$$\|e(\cdot)\|_\lambda = \sup_{0 \leq t \leq T} \{e^{-\lambda t} \|e(t)\|_\infty\} \tag{8.18}$$

so that it discounts the errors as time increases.

The main difficulty experienced with implementing the iterative learning controller was the large number of trials required to obtain acceptable performance due to the slow convergence of the error. Arimoto et al. (1984) demonstrated that it is possible to improve the rate of convergence by increasing the magnitude of the elements of $\Gamma$ given condition 2 is satisfied. However, while meaningful increases in the magnitude of the elements of $\Gamma$ for control of the prototype manipulator did improve the rate of convergence in the early portions of the trajectory, it also led to unacceptably large errors as time increased during the

trajectory. There are some methods that could be applied to combat this problem that were not addressed as part of this work, but would be interesting to pursue. For example, it may be possible to learn several segments of a trajectory in a piecewise fashion and then combine them to form a complete trajectory. An improvement should also be achievable by using the results of the learning rule as a feedforward signal to a closed-loop controller as suggested by Hauser (1987).

## 8.5 Summary

Three separate controllers were developed and demonstrated using the prototype parallel manipulator, with the first being a PID type controller, the second being a computed torque controller, and the third being a iterative learning controller. The computed torque and iterative learning controllers both demonstrated comparable improved trajectory tracking performance relative to the PID controller. The performance improvements demonstrated by the computed torque and iterative learning controllers stem from their ability to account for the nonlinear and coupled nature of the manipulator dynamics.

The computed torque controller is the most complex controller of the three to implement since it requires a good model of the manipulator inverse dynamics. It's suspected that the performance of the computed torque controller could be improved for the prototype manipulator by incorporating a better model of the motor friction. The current model only considers viscous friction at the actuator. Accordingly, the inverse dynamics model, and in turn the computed torque controller performance, could be improved by including the influence of kinetic and static friction. This is especially relevant during trajectory segments

were the motor operates at slow speed or experiences torque reversals.

The iterative learning controller is relatively easy to implement, but it is encumbered by the large number or trails required for the trajectory tracking error to converge to an acceptable range. It may be possible to improve the convergence of the trajectory tracking error by incorporating the results of the iterative learning controller as a feedforward term in a closed loop controller or perhaps by learning the trajectory in a piecewise fashion.

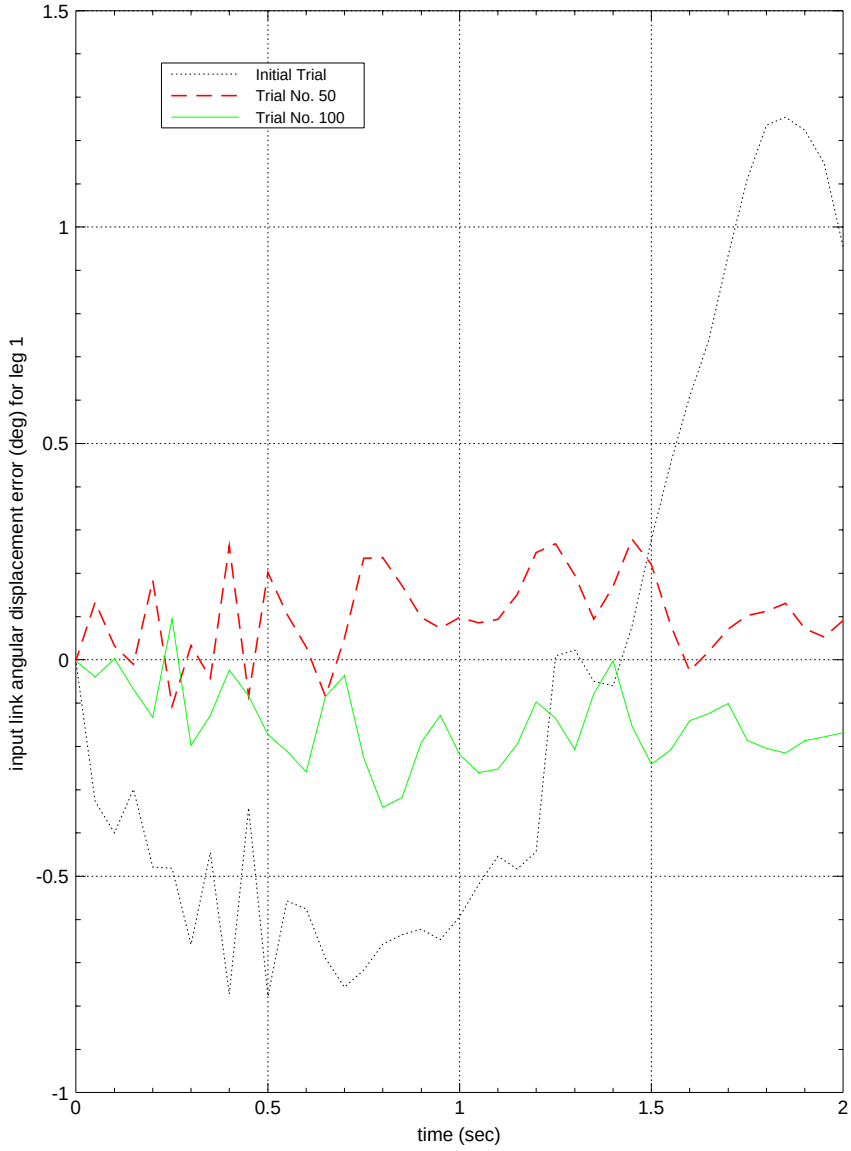



Figure 8.13: Leg 1 input link joint errors after selected number of trials.

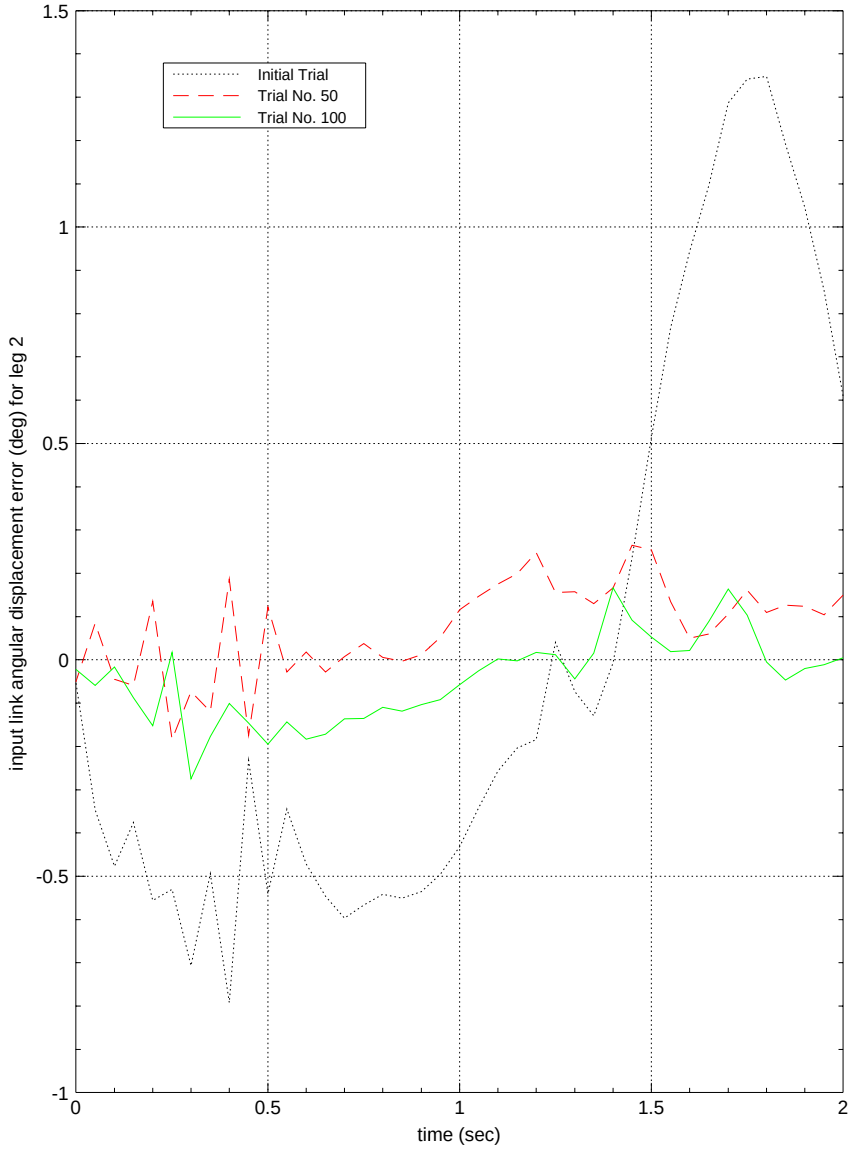



Figure 8.14: Leg 2 input link joint errors after selected number of trials.

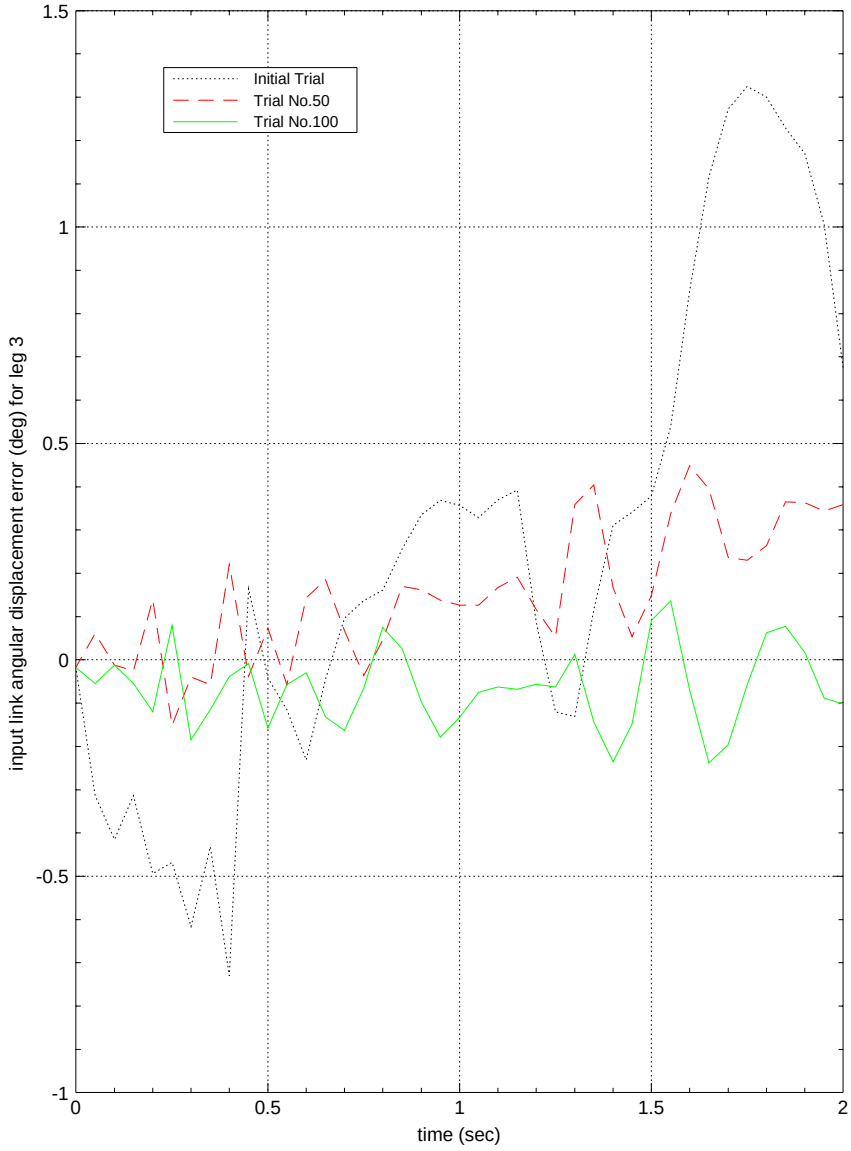



Figure 8.15: Leg 3 input link joint errors after selected number of trials.

# Chapter 9

# Summary and Future Research

## 9.1 Summary

In this dissertation, a novel parallel manipulator is investigated. The manipulator has three degree of freedoms and the moving platform is constrained to only translational motion. The main advantages of this parallel manipulator are that all of the actuators can be attached directly to the base, closed-form solutions are available for the forward and inverse kinematics, the moving platform maintains the same orientation throughout the entire workspace, and it can be constructed with only revolute joints. The most significant disadvantage is the limited workspace relative to a serial manipulator of comparable size. A description of the manipulator joint and link configuration is presented in Chapter 2, along with a discussion of the manipulator mobility.

Closed-form solutions for both the forward and inverse kinematics problems are given in Chapter 3. It is shown that the inverse kinematics problem has up to four real solutions, and the forward kinematics problem has up to 16 real solutions. The kinematics of the manipulator are explored further in Chapter

4, where the Jacobian matrix for the manipulator is developed. The Jacobian matrix maps the velocity of the moving platform in cartesian space to the input joint velocities in joint space, and is used in Chapter 4 to search for singular poses of the manipulator, where the manipulator instantaneously gains or loses a degree of freedom.

In Chapter 5, the workspace of the manipulator is considered. Workspace volume as a function of the manipulator parameters is determined using the Monte Carlo method. A procedure for characterizing the quality of the workspace is also developed. Using these results, optimization studies for maximum workspace volume and for well-conditioned workspace volume are conducted. An objective function for the well-conditioned optimization study is defined as the integration of the reciprocal of the condition number of the Jacobian matrix over the workspace volume, and named the global condition index. The results of the optimization study show that a manipulator that is optimized for well-conditioned workspace has legs that are evenly separated by 120°, and a moving platform and a fixed platform of the same size. Also, each input link is 44% of the total length of each leg, and each upper connecting arm is 56% of the total length of each leg.

Models for the dynamics of the manipulator are presented in Chapter 6. Three different models are developed, with numerical simulations presented for all three models. The first model is based upon the application of the Newton-Euler equations of motion used in conjunction with the Jacobian matrix to map the inertial and gravitational loadings of the moving platform to the actuators. This model is developed for use with a computed torque controller since it does a good job of capturing the character of the dynamics while being computa-

tionally efficient. The second model was developed to give a more complete characterization of the dynamics, and is based upon the Lagrangian multiplier approach. The third model neglects the highly coupled nature of the manipulator and models each input link individually. This model is developed for use with single-input single-output type controllers.

A prototype was fabricated to demonstrate this manipulator. A description of the prototype and the accuracy of the prototype are provided in Chapter 7. Three controllers are developed for the manipulator and tested on the prototype. A PID controller, a computed torque controller, and an iterative learning controller are all applied to the prototype, and the results are given in Chapter 8.

The research presented in this dissertation establishes this parallel manipulator as a viable robotic device for three degree of freedom manipulation. The manipulator offers the advantages associated with other parallel manipulators, such as light weight construction; while avoiding some of the traditional disadvantages of parallel manipulators such as the extensive use of spherical joints and coupling of the platform orientation and position.

## 9.2 Future Research

One of the disappointments of this research was the positioning accuracy demonstrated by the prototype manipulator. It should be possible to significantly improve upon the geometric positioning accuracy with a rigorous calibration of the prototype. This rigorous calibration requires a more general kinematic model that does not assume that each leg is identical, and also allows a more general

location and orientation of the joint between the input link and the fixed base. Accordingly, development of a general kinematic model is an area that deserves future research. Along with development of the more general kinematics model, an effort should be made to eliminate the extraneous solutions generated by the forward kinematics solution.

Another area of research that would benefit this manipulator is the investigation of alternative sensing and actuation schemes. It's not difficult to imagine applications where sensing and actuation schemes that are different from the one used in the prototype would provide better performance. For instance, if accurate positioning of the moving platform is of primary importance (e.g. integrated circuit lead bonding), then direct sensing of the moving platform position should provide better performance than inferring the moving platform position from the angular displacement of the input links. Hence, an exploration of the sensing and actuation configurations available to this manipulator and their relative merits would be beneficial.

It would also be interesting to examine this manipulator as a compliant mechanism. One of the distinguishing characteristics of this manipulator is that it can be constructed using only revolute joints with non-intersecting axes, which can easily be approximated in compliant mechanisms. So, potentially this manipulator could be molded in a single piece using the injection molding process, providing an inexpensive means of producing a spatial mechanism that constrains motions to three translational degrees of freedom. This type of compliant mechanism would also lend itself to the application as an underactuated or passive mechanism, where one or more of the actuated joints are replaced with elastic elements. Gosselin (1996) examined underactuated planar parallel manipulators

that are used for force generation as opposed to motion generation. It would be interesting to extend Gosselin's work to spatial mechanisms using the manipulator studied in this dissertation.

# Appendix A Constants for Forward Kinematics

$$
\begin{aligned}
k_1 &= 2\left[c - r - a\cos(\theta_{11})\right] - 2\cos(\phi_2)\left[c - r - a\cos(\theta_{12})\right] \\
k_2 &= 2\sin(\phi_2)\left[r - c + a\cos(\theta_{12})\right] \\
k_3 &= 2a\left[\sin(\theta_{12}) - \sin(\theta_{11})\right] \\
k_4 &= -2b(d+e) \\
k_5 &= 2b(d+e) \\
k_6 &= 2a(r-c)\left[\cos(\theta_{11}) - \cos(\theta_{12})\right] \\
k_7 &= 2\left[c - r - a\cos(\theta_{11})\right] - 2\cos(\phi_3)\left[c - r - a\cos(\theta_{13})\right] \\
k_8 &= 2\sin(\phi_3)\left[r - c + a\cos(\theta_{13})\right] \\
k_9 &= 2a\left[\sin(\theta_{13}) - \sin(\theta_{11})\right] \\
k_{10} &= -2b(d+e) \\
k_{11} &= 2b(d+e) \\
k_{12} &= 2a(r-c)\left[\cos(\theta_{11}) - \cos(\theta_{13})\right] \\
k_{13} &= -\frac{k_4}{k_3} \\
k_{14} &= -\frac{k_2 b\sin(\phi_2) + k_1 b\cos(\phi_2)}{k_3\sin(\phi_2)} \\
k_{15} &= -\frac{k_5}{k_3} \\
k_{16} &= \frac{k_1 b}{k_3\sin(\phi_2)} \\
k_{17} &= -\frac{k_6}{k_3} \\
k_{18} &= k_{10} + k_9 k_{13} \\
k_{19} &= k_8 b + \frac{k_7 b\cos(\phi_2)}{\sin(\phi_2)} + k_9 k_{14}
\end{aligned}
$$

$$
\begin{aligned}
k_{20} &= k_9 k_{15} \\
k_{21} &= k_9 k_{16} - \frac{k_7 b}{\sin(\phi_2)} \\
k_{22} &= k_5 \\
k_{23} &= k_{12} + k_9 k_{17} \\
k_{24} &= b^2 + k_{14}^2 \sin^2(\phi_2) \\
k_{25} &= k_{13}^2 \sin^2(\phi_2) \\
k_{26} &= b^2 + k_{16}^2 \sin^2(\phi_2) \\
k_{27} &= k_{15}^2 \sin^2(\phi 2) \\
k_{28} &= 2k_{13}k_{14} \sin^2(\phi_2) \\
k_{29} &= 2k_{14}k_{16} \sin^2(\phi_2) - 2b^2 \cos(\phi_2) \\
k_{30} &= 2k_{14}k_{15} \sin^2(\phi_2) \\
k_{31} &= 2k_{13}k_{16} \sin^2(\phi_2) \\
k_{32} &= 2k_{13}k_{15} \sin^2(\phi_2) \\
k_{33} &= 2k_{15}k_{16} \sin^2(\phi_2) \\
k_{34} &= 2bc \cos(\phi_2) \sin(\phi_2) - 2ab \cos(\phi_2) \sin(\phi_2) \cos(\theta_{11}) - 2br \cos(\phi_2) \sin(\phi_2) \\
&\quad + 2k_{14}k_{17} \sin^2(\phi_2) - 2ak_{14} \sin^2(\phi_2) \sin(\theta_{11}) \\
k_{35} &= -2bd \sin^2(\phi_2) - 2be \sin^2(\phi_2) + 2k_{13}k_{17} \sin^2(\phi_2) - 2ak_{13} \sin^2(\phi_2) \sin(\theta_{11}) \\
k_{36} &= -2bc \sin(\phi_2) + 2ab \cos(\theta_{11}) \sin(\phi_2) + 2br \sin(\phi_2) + 2k_{16}k_{17} \sin^2(\phi_2) \\
&\quad - 2ak_{16} \sin^2(\phi_2) \sin(\theta_{11}) \\
k_{37} &= 2k_{15}k_{17} \sin^2(\phi_2) - 2ak_{15} \sin^2(\phi_2) \sin(\theta_{11}) \\
k_{38} &= \sin^2(\phi_2)(a^2 - b^2 + c^2 - 2ac \cos(\theta_{11}) - d^2 - 2de - e^2 + k_{17}^2 - 2cr \\
&\quad + 2ar \cos(\theta_{11}) + r^2 - 2ak_{17} \sin(\theta_{11}))
\end{aligned}
$$

$$
\begin{aligned}
k_{39} &= k_{19}^2 \sin^2(\phi_2) + k_{22}^2 \sin^2(\phi_2 - \phi_3) \\
k_{40} &= k_{18}^2 \sin^2(\phi_2) \\
k_{41} &= k_{21}^2 \sin^2(\phi_2) + k_{22}^2 \sin^2(\phi_3) \\
k_{42} &= k_{20}^2 \sin^2(\phi_2) \\
k_{43} &= 2k_{18}k_{19} \sin^2(\phi_2) \\
k_{44} &= 2k_{19}k_{21} \sin^2(\phi_2) + 2k_{22}^2 \sin(\phi_2 - \phi_3) \sin(\phi_3) \\
k_{45} &= 2k_{19}k_{20} \sin^2(\phi_2) \\
k_{46} &= 2k_{18}k_{21} \sin^2(\phi_2) \\
k_{47} &= 2k_{18}k_{20} \sin^2(\phi_2) \\
k_{48} &= 2k_{20}k_{21} \sin^2(\phi_2) \\
k_{49} &= 2k_{19}k_{23} \sin^2(\phi_2) \\
k_{50} &= 2k_{18}k_{23} \sin^2(\phi_2) \\
k_{51} &= 2k_{21}k_{23} \sin^2(\phi_2) \\
k_{52} &= 2k_{20}k_{23} \sin^2(\phi_2) \\
k_{53} &= \sin^2(\phi_2) \left[ k_{23}^2 - k_{22}^2 \right] \\
k_{54} &= k_{24} + k_{26} + k_{29} - k_{34} - k_{36} + k_{38} \\
k_{55} &= -2k_{28} - 2k_{31} + 2k_{35} \\
k_{56} &= -2k_{24} + 4k_{25} + 2k_{26} - 2k_{36} + 2k_{38} \\
k_{57} &= 2k_{28} - 2k_{31} + 2k_{35} \\
k_{58} &= k_{24} + k_{26} - k_{29} + k_{34} - k_{36} + k_{38} \\
k_{59} &= -2k_{30} - 2k_{33} + 2k_{37} \\
k_{60} &= 4k_{32}
\end{aligned}
$$

$$
\begin{aligned}
k_{61} &= -4k_{33} + 4k_{37} \\
k_{62} &= 4k_{32} \\
k_{63} &= 2k_{30} - 2k_{33} + 2k_{37} \\
k_{64} &= 2k_{24} - 2k_{26} + 4k_{27} - 2k_{34} + 2k_{38} \\
k_{65} &= -4k_{28} + 4k_{35} \\
k_{66} &= -4k_{24} + 8k_{25} - 4k_{26} + 8k_{27} + 4k_{38} \\
k_{67} &= 4k_{28} + 4k_{35} \\
k_{68} &= 2k_{24} - 2k_{26} + 4k_{27} + 2k_{34} + 2k_{38} \\
k_{69} &= -2k_{30} + 2k_{33} + 2k_{37} \\
k_{70} &= 4k_{32} \\
k_{71} &= 4k_{33} + 4k_{37} \\
k_{72} &= 4k_{32} \\
k_{73} &= 2k_{30} + 2k_{33} + 2k_{37} \\
k_{74} &= k_{24} + k_{26} - k_{29} - k_{34} + k_{36} + k_{38} \\
k_{75} &= -2k_{28} + 2k_{31} + 2k_{35} \\
k_{76} &= -2k_{24} + 4k_{25} + 2k_{26} + 2k_{36} + 2k_{38} \\
k_{77} &= 2k_{28} + 2k_{31} + 2k_{35} \\
k_{78} &= k_{24} + k_{26} + k_{29} + k_{34} + k_{36} + k_{38} \\
k_{79} &= k_{39} + k_{41} + k_{44} - k_{49} - k_{51} + k_{53} \\
k_{80} &= -2k_{43} - 2k_{46} + 2k_{50} \\
k_{81} &= -2k_{39} + 4k_{40} + 2k_{41} - 2k_{51} + 2k_{53} \\
k_{82} &= 2k_{43} - 2k_{46} + 2k_{50}
\end{aligned}
$$

$$
\begin{aligned}
k_{83} &= k_{39} + k_{41} - k_{44} + k_{49} - k_{51} + k_{53} \\
k_{84} &= -2k_{45} - 2k_{48} + 2k_{52} \\
k_{85} &= 4k_{47} \\
k_{86} &= -4k_{48} + 4k_{52} \\
k_{87} &= 4k_{47} \\
k_{88} &= 2k_{45} - 2k_{48} + 2k_{52} \\
k_{89} &= 2k_{39} - 2k_{41} + 4k_{42} - 2k_{49} + 2k_{53} \\
k_{90} &= -4k_{43} + 4k_{50} \\
k_{91} &= -4k_{39} + 8k_{40} - 4k_{41} + 8k_{42} + 4k_{53} \\
k_{92} &= 4k_{43} + 4k_{50} \\
k_{93} &= 2k_{39} - 2k_{41} + 4k_{42} + 2k_{49} + 2k_{53} \\
k_{94} &= -2k_{45} + 2k_{48} + 2k_{52} \\
k_{95} &= 4k_{47} \\
k_{96} &= 4k_{48} + 4k_{52} \\
k_{97} &= 4k_{47} \\
k_{98} &= 2k_{45} + 2k_{48} + 2k_{52} \\
k_{99} &= k_{39} + k_{41} - k_{44} - k_{49} + k_{51} + k_{53} \\
k_{100} &= -2k_{43} + 2k_{46} + 2k_{50} \\
k_{101} &= -2k_{39} + 4k_{40} + 2k_{41} + 2k_{51} + 2k_{53} \\
k_{102} &= 2k_{43} + 2k_{46} + 2k_{50} \\
k_{103} &= k_{39} + k_{41} + k_{44} + k_{49} + k_{51} + k_{53} \\
k_{104} &= 1 + \frac{k_{108}^2}{k_{109}^2} + \frac{k_{111}^2}{k_{112}^2}
\end{aligned}
$$

$$
\begin{aligned}
k_{105} &= \frac{2k_{107}k_{108}}{k_{109}^2} + \frac{2k_{110}k_{111}}{k_{112}^2} - 2k_{113}\cos(\phi_1) - \frac{2k_{113}k_{108}}{k_{109}}\sin(\phi_1) - \frac{2ak_{111}}{k_{112}}\sin(\theta_{11}) \\
k_{106} &= k_{113}^2 - b^2 + \frac{k_{107}^2}{k_{109}^2} + \frac{k_{110}^2}{k_{112}^2} + a^2\sin^2(\theta_{11}) - \frac{2k_{107}k_{113}}{k_{109}}\sin(\phi_1) - \frac{2ak_{110}}{k_{112}}\sin(\theta_{11}) \\
k_{107} &= k_{118}k_{121} - k_{119}k_{120} \\
k_{108} &= k_{115}k_{118} - k_{114}k_{119} \\
k_{109} &= k_{116}k_{119} - k_{117}k_{118} \\
k_{110} &= k_{117}k_{120} - k_{116}k_{121} \\
k_{111} &= k_{114}k_{117} - k_{115}k_{116} \\
k_{112} &= k_{116}k_{119} - k_{117}k_{118} \\
k_{113} &= a\cos(\theta_{11}) + r - c \\
k_{114} &= 2\cos(\phi_2)\left[a\cos(\theta_{12}) + r - c\right] - 2\cos(\phi_1)\left[a\cos(\theta_{11}) + r - c\right] \\
k_{115} &= 2\cos(\phi_3)\left[a\cos(\theta_{13}) + r - c\right] - 2\cos(\phi_1)\left[a\cos(\theta_{11}) + r - c\right] \\
k_{116} &= 2\sin(\phi_2)\left[a\cos(\theta_{12}) + r - c\right] - 2\sin(\phi_1)\left[a\cos(\theta_{11}) + r - c\right] \\
k_{117} &= 2\sin(\phi_3)\left[a\cos(\theta_{13}) + r - c\right] - 2\sin(\phi_1)\left[a\cos(\theta_{11}) + r - c\right] \\
k_{118} &= 2a\sin(\theta_{12}) - 2a\sin(\theta_{11}) \\
k_{119} &= 2a\sin(\theta_{13}) - 2a\sin(\theta_{11}) \\
k_{120} &= \left[a\cos(\theta_{11}) + r - c\right]^2 + a^2\sin^2(\theta_{11}) - \left[a\cos(\theta_{12}) + r - c\right]^2 - a^2\sin^2(\theta_{12}) \\
k_{121} &= \left[a\cos(\theta_{11}) + r - c\right]^2 + a^2\sin^2(\theta_{11}) - \left[a\cos(\theta_{13}) + r - c\right]^2 - a^2\sin^2(\theta_{13})
\end{aligned}
$$

# Appendix B
# Evaluation of Partial Derivatives of Constraint Functions for Langrange Based Dynanics

For $i$ =1, 2, and 3,

and $j = 1$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial p_x} \\ &= 2p_x + 2c\cos(\phi_i) - 2r\cos(\phi_i) - 2a\cos(\phi_i)\cos(\theta_{1i}) \\ &\quad -2d\cos(\phi_i)\cos(\theta_{2i}) - 2e\cos(\phi_i)\cos(\theta_{2i}) \end{aligned}$$

For $i$ =4, 5, and 6,

and $j = 1$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial p_x} \\ &= -\cos(\phi_i)\tan(\theta_{2i}) \end{aligned}$$

For $i$ =1, 2, and 3,

and $j = 2$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial p_y} \\ &= 2p_y + 2c\sin(\phi_i) - 2r\sin(\phi_i) - 2a\sin(\phi_i)\cos(\theta_{1i}) \\ &\quad -2d\sin(\phi_i)\cos(\theta_{2i}) - 2e\sin(\phi_i)\cos(\theta_{2i}) \end{aligned}$$

For $i$ =4, 5, and 6,

and $j = 2$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial p_y} \\ &= -\sin(\phi_i)\tan(\theta_{2i}) \end{aligned}$$

For $i$ =1, 2, and 3,

and $j = 3$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial p_z} \\ &= 2p_z - 2a\sin(\theta_{1i}) - 2d\sin(\theta_{2i}) - 2e\sin(\theta_{2i}) \end{aligned}$$

For $i$ =4, 5, and 6,

and $j = 3$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial p_z} \\ &= 1 \end{aligned}$$

For $i$ =1, and $j = 4$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial \theta_{21}} \\ &= [-p_x\sin(\phi_1 - \theta_{21}) + 2a\sin(\theta_{11} - \theta_{21}) + 2c\sin(\theta_{21}) \\ &\quad -2r\sin(\theta_{21}) + p_x\sin(\phi_1 + \theta_{21}) + p_y\cos(\phi_1 - \theta_{21}) \\ &\quad -2p_z\cos(\theta_{21}) - p_y\cos(\phi_1 + \theta_{21})]\,(d + e) \end{aligned}$$

For $i$ =2, 3, 5, and 6

and $j = 4$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial \theta_{21}} \\ &= 0 \end{aligned}$$

For $i$ =4 and $j = 4$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial \theta_{21}} \\ &= \frac{1}{\cos(\theta_{21})^2} \left[ r - c - p_x \cos(\phi_1) + a \cos(\theta_{11}) - p_y \sin(\phi_1) \right] \end{aligned}$$

For $i$ =1, 3, 4, and 6

and $j = 5$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial \theta_{22}} \\ &= 0 \end{aligned}$$

For $i$ =2 and $j = 5$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial \theta_{22}} \\ &= [-p_x \sin(\phi_2 - \theta_{22}) + 2a \sin(\theta_{12} - \theta_{22}) + 2c \sin(\theta_{22}) \\ &\quad -2r \sin(\theta_{22}) + p_x \sin(\phi_2 + \theta_{22}) + p_y \cos(\phi_2 - \theta_{22}) \\ &\quad -2p_z \cos(\theta_{22}) - p_y \cos(\phi_2 + \theta_{22})] \, (d + e) \end{aligned}$$

For $i$ =5 and $j = 5$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial \theta_{22}} \\ &= \frac{1}{\cos(\theta_{22})^2} \left[ r - c - p_x \cos(\phi_2) + a \cos(\theta_{12}) - p_y \sin(\phi_2) \right] \end{aligned}$$

For $i$ =1, 2, 4, and 5,

and $j = 6$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial \theta_{23}} \\ &= 0 \end{aligned}$$

For $i$ =3 and $j = 6$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial \theta_{23}} \\ &= [-p_x \sin(\phi_3 - \theta_{23}) + 2a \sin(\theta_{13} - \theta_{23}) + 2c \sin(\theta_{23}) \\ &\quad -2r \sin(\theta_{23}) + p_x \sin(\phi_3 + \theta_{23}) + p_y \cos(\phi_3 - \theta_{23}) \\ &\quad -2p_z \cos(\theta_{23}) - p_y \cos(\phi_3 + \theta_{23})] \, (d + e) \end{aligned}$$

For $i$ =6 and $j = 6$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial \theta_{23}} \\ &= \frac{1}{\cos(\theta_{23})^2} \left[ r - c - p_x \cos(\phi_3) + a \cos(\theta_{13}) - p_y \sin(\phi_3) \right] \end{aligned}$$

For $i$ =1 and $j = 7$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial \theta_{11}} \\ &= 2a\,[p_x \cos(\phi_1)\sin(\theta_{11}) + p_y \sin(\phi_1)\sin(\theta_{11}) + (c-r)\sin(\theta_{11}) \\ &\quad -(c+e)\sin(\theta_{11}-\theta_{21}) - p_z\cos(\theta_{11})] \end{aligned}$$

For $i$ =2, 3, 5, and 7

and $j = 7$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial \theta_{11}} \\ &= 0 \end{aligned}$$

For $i$ =4 and $j = 7$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial \theta_{11}} \\ &= -a\,[\cos(\theta_{11}) + \tan(\theta_{21})\sin(\theta_{11})] \end{aligned}$$

For $i$ =1, 3, 4, and 6,

and $j = 8$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial \theta_{12}} \\ &= 0 \end{aligned}$$

For $i$ =2 and $j = 8$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial \theta_{12}} \\ &= 2a\left[p_x \cos(\phi_2)\sin(\theta_{12}) + p_y \sin(\phi_2)\sin(\theta_{12}) + (c-r)\sin(\theta_{12})\right. \\ &\quad \left. -(c+e)\sin(\theta_{12}-\theta_{22}) - p_z\cos(\theta_{12})\right] \end{aligned}$$

For $i$ =5 and $j = 8$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial \theta_{12}} \\ &= -a\left[\cos(\theta_{12}) + \tan(\theta_{22})\sin(\theta_{12})\right] \end{aligned}$$

For $i$ =1, 2, 4, and 5,

and $j = 9$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial \theta_{13}} \\ &= 0 \end{aligned}$$

For $i$ =3 and $j = 9$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial \theta_{13}} \\ &= 2a\left[p_x \cos(\phi_3)\sin(\theta_{13}) + p_y \sin(\phi_3)\sin(\theta_{13}) + (c-r)\sin(\theta_{13})\right. \\ &\quad \left. -(c+e)\sin(\theta_{13}-\theta_{23}) - p_z\cos(\theta_{13})\right] \end{aligned}$$

For $i$ =6 and $j = 9$:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial q_j} = \frac{\partial f_i}{\partial \theta_{13}} \\ &= -a\left[\cos(\theta_{13}) + \tan(\theta_{23})\sin(\theta_{13})\right] \end{aligned}$$

# Appendix C
# Evaluation of Partial Derivatives of Langrange Function for Langrange Based Dynanics

For $j$ =1:

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) &= (3m_b + m_c)\,\ddot{x} + dm_b \sum_{i=1}^{3}\left(\ddot{\theta}_{2i}\cos\phi_i \sin\theta_{2i} + \dot{\theta}_{2i}^2 \cos\phi_i \cos\theta_{2i}\right) \\ \frac{\partial L}{\partial x} &= 0 \end{aligned}$$

For $j$=2:

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{y}}\right) &= (3m_b + m_c)\,\ddot{y} + dm_b \sum_{i=1}^{3}\left(\ddot{\theta}_{2i}\sin\phi_i \sin\theta_{2i} + \dot{\theta}_{2i}^2 \sin\phi_i \cos\theta_{2i}\right) \\ \frac{\partial L}{\partial y} &= 0 \end{aligned}$$

For $j$=3:

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{z}}\right) &= (3m_b + m_c)\,\ddot{z} - dm_b \sum_{i=1}^{3}\left(\ddot{\theta}_{2i}\cos\theta_{2i} - \dot{\theta}_{2i}^2 \sin\theta_{2i}\right) \\ \frac{\partial L}{\partial z} &= -g\,(3m_b + m_c) \end{aligned}$$

For $j$=4, 5, and 6:

$$\begin{aligned}\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_{2n}}\right) &= d^2 m_b \ddot{\theta}_{2n} + e^2 m_b \ddot{\theta}_{2n} + aem_b \ddot{\theta}_{1n} \cos\left(\theta_{1n} - \theta_{2n}\right) \\ &\quad -aem_b \dot{\theta}_{1n}\left(\dot{\theta}_{1n} - \dot{\theta}_{2n}\right) \sin\left(\theta_{1n} - \theta_{2n}\right) - dm_b \ddot{z} \cos\theta_{2n} \\ &\quad +dm_b \dot{z}\dot{\theta}_{2n} \sin\theta_{2n} + dm_b \ddot{x} \cos\phi_n \sin\theta_{2n} + dm_b \dot{x}\dot{\theta}_{2n} \cos\phi_n \cos\theta_{2n} \\ &\quad +dm_b \ddot{y} \sin\phi_n \sin\theta_{2n} + dm_b \dot{y}\dot{\theta}_{2n} \sin\phi_n \cos\theta_{2n}\end{aligned}$$

$$\begin{aligned}\frac{\partial L}{\partial \theta_{2n}} &= (d-e) g m_b \cos\theta_{2n} + dm_b \dot{\theta}_{2n} \dot{x} \cos\phi_n \cos\theta_{2n} \\ &\quad +dm_b \dot{\theta}_{2n} \dot{y} \sin\phi_n \cos\theta_{2n} + aem_b \dot{\theta}_{1n}\dot{\theta}_{2n} \sin(\theta_{1n} - \theta_{2n}) \\ &\quad +dm_b \dot{\theta}_{2n} \dot{z} \sin\theta_{2n}\end{aligned}$$

for $n$ =1, 2, and 3.

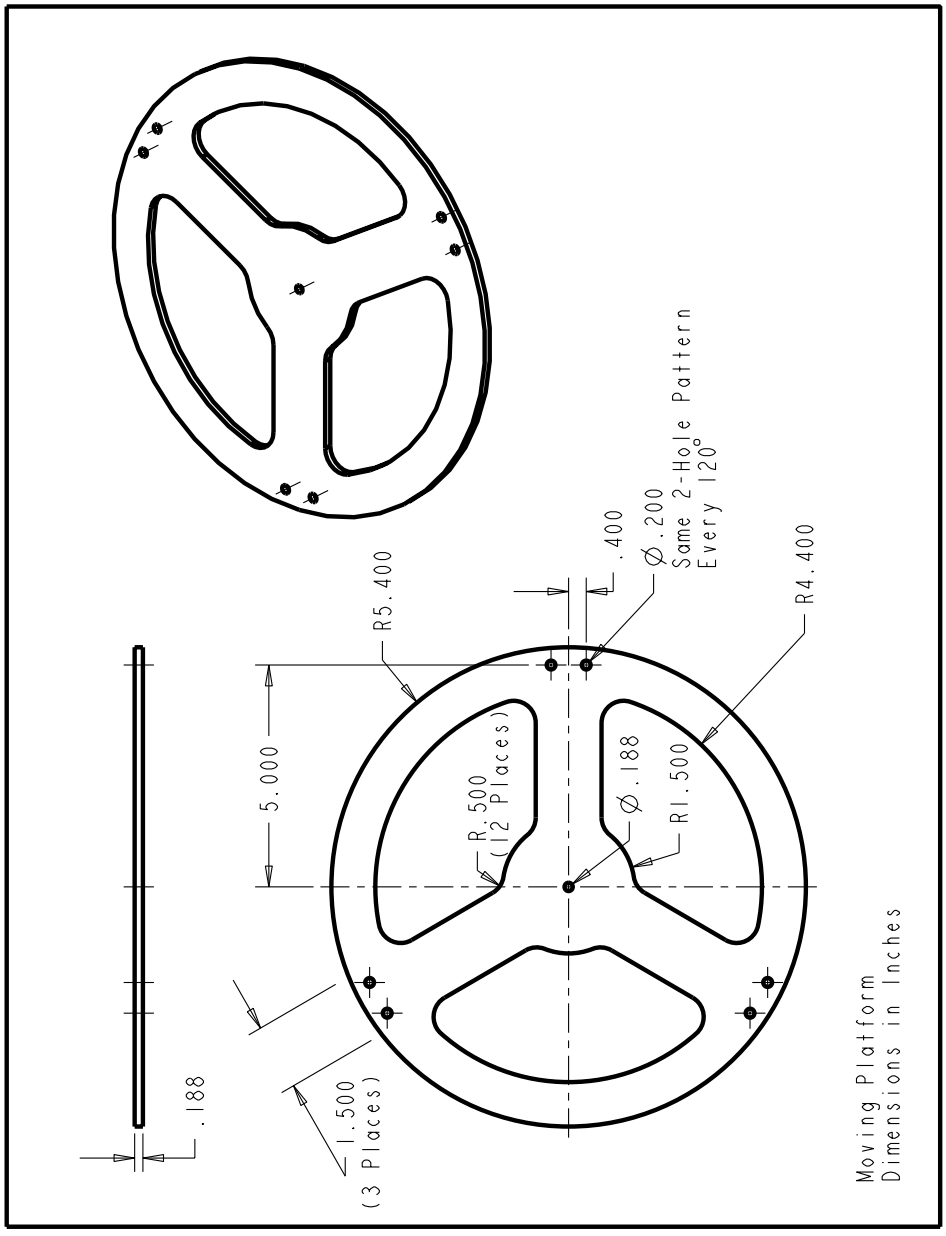
For $j$=7, 8, and 9:

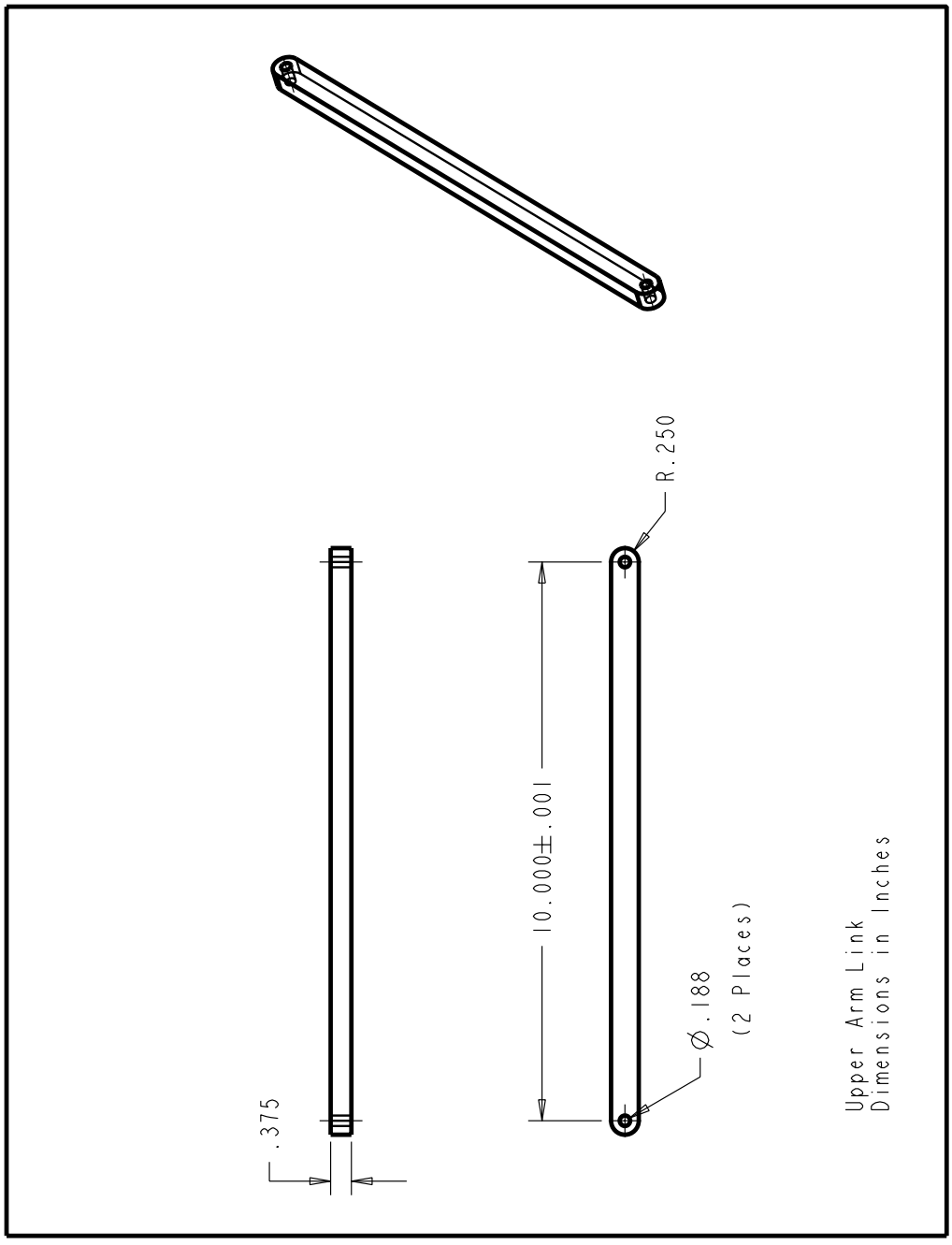
$$\begin{aligned}\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_{1n}}\right) &= (I_m + \frac{1}{3}a^2 m_a + a^2 m_b)\ddot{\theta}_{1n} + aem_b \ddot{\theta}_{2n} \cos(\theta_{1n} - \theta_{2n}) \\ &\quad -aem_b \dot{\theta}_{2n}(\dot{\theta}_{1n} - \dot{\theta}_{2n}) \sin(\theta_{1n} - \theta_{2n})\end{aligned}$$

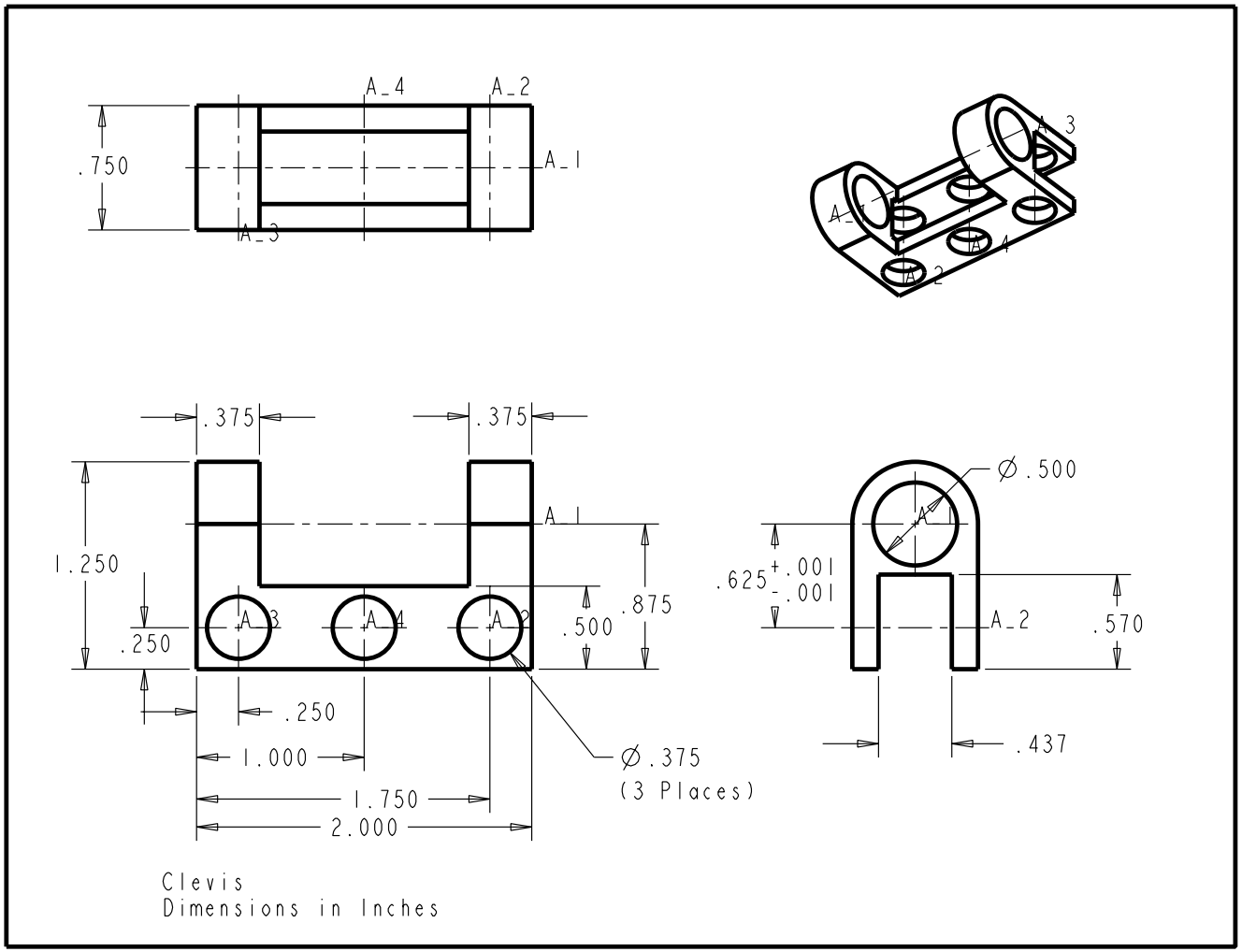
$$\frac{\partial L}{\partial \theta_{1n}} = -\frac{1}{2} agm_a \cos\theta_{1n} - agm_b \cos\theta_{1n} - aem_b \dot{\theta}_{1n}\dot{\theta}_{2n} \sin(\theta_{1n} - \theta_{2n})$$

for $n$ =1, 2, and 3.

# Appendix D
# Link Drawings

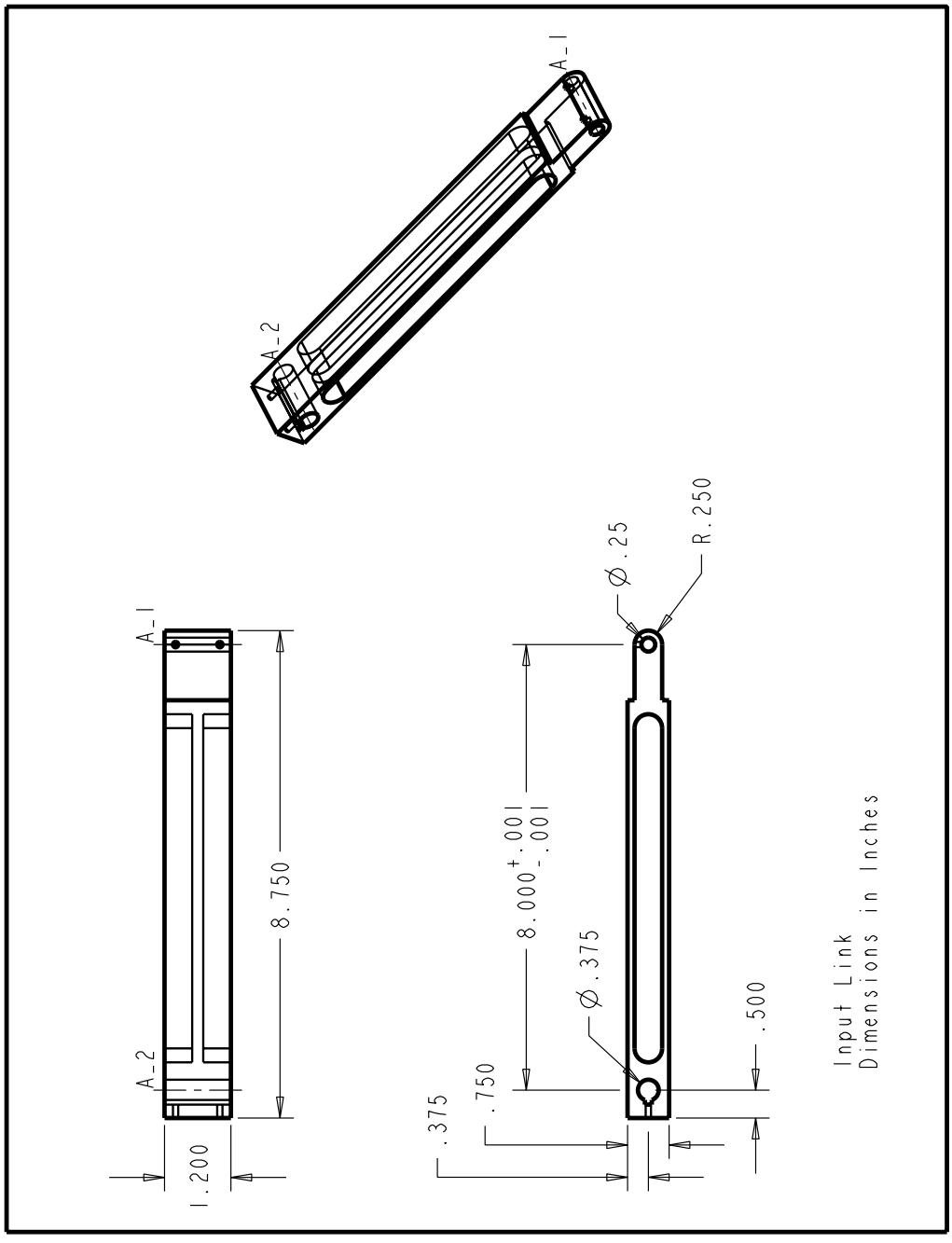



R.250
10.000±.001
Ø.188
(2 Places)
.375
Upper Arm Link
Dimensions in Inches



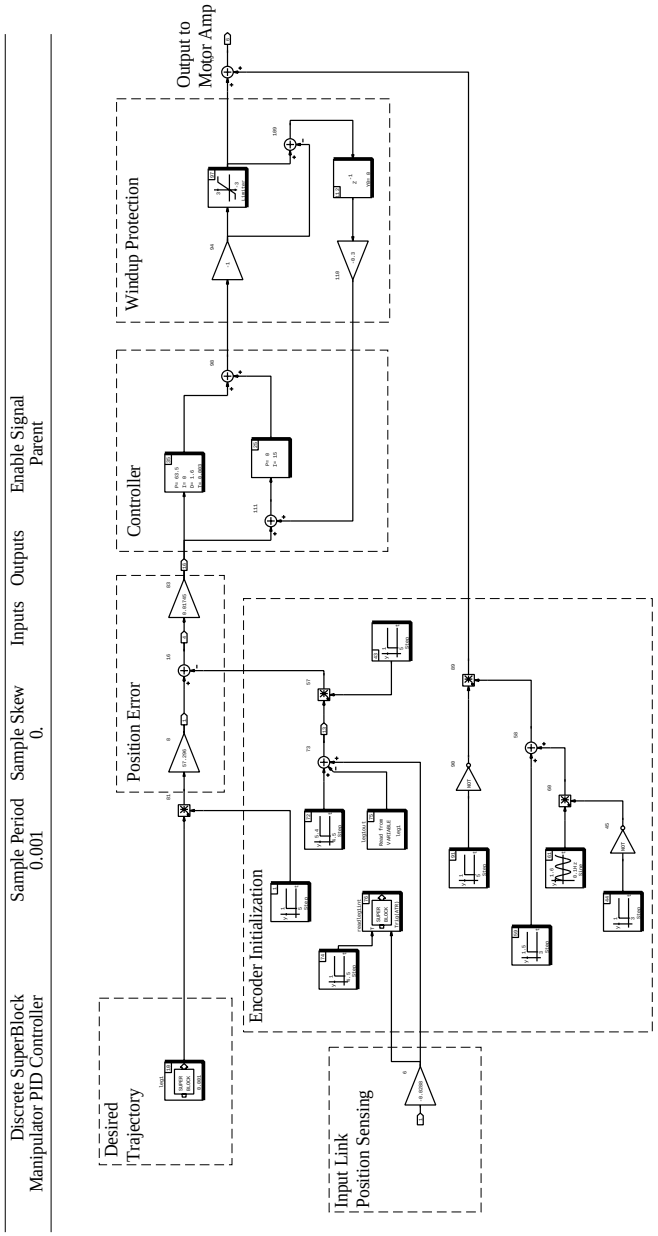

Clevis
Dimensions in Inches

SCALE 1.500

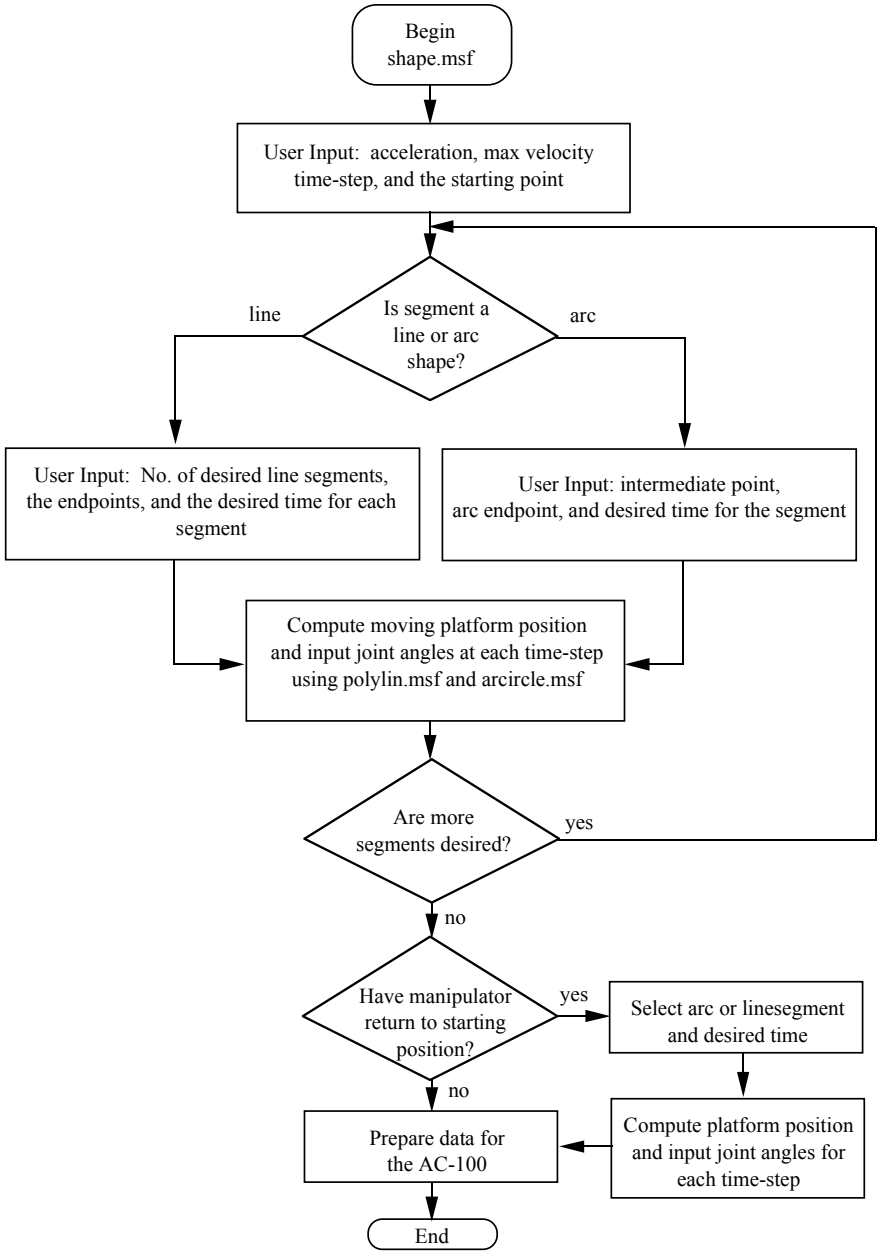

A_1
A_2
A_1
A_2
8.750
1.200
Ø.25
R.250
8.000 +.001 -.001
Ø.375
.750
.375
.500
Input Link
Dimensions in Inches
SCALE 0.500 SCALE 0.500

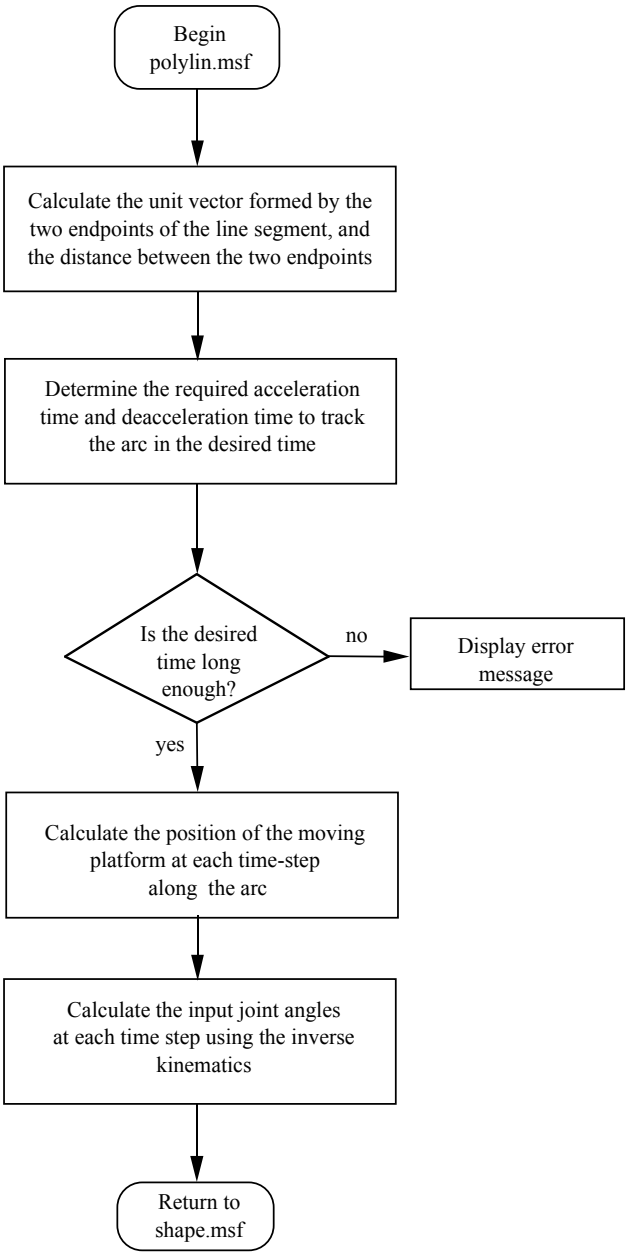
# Appendix E

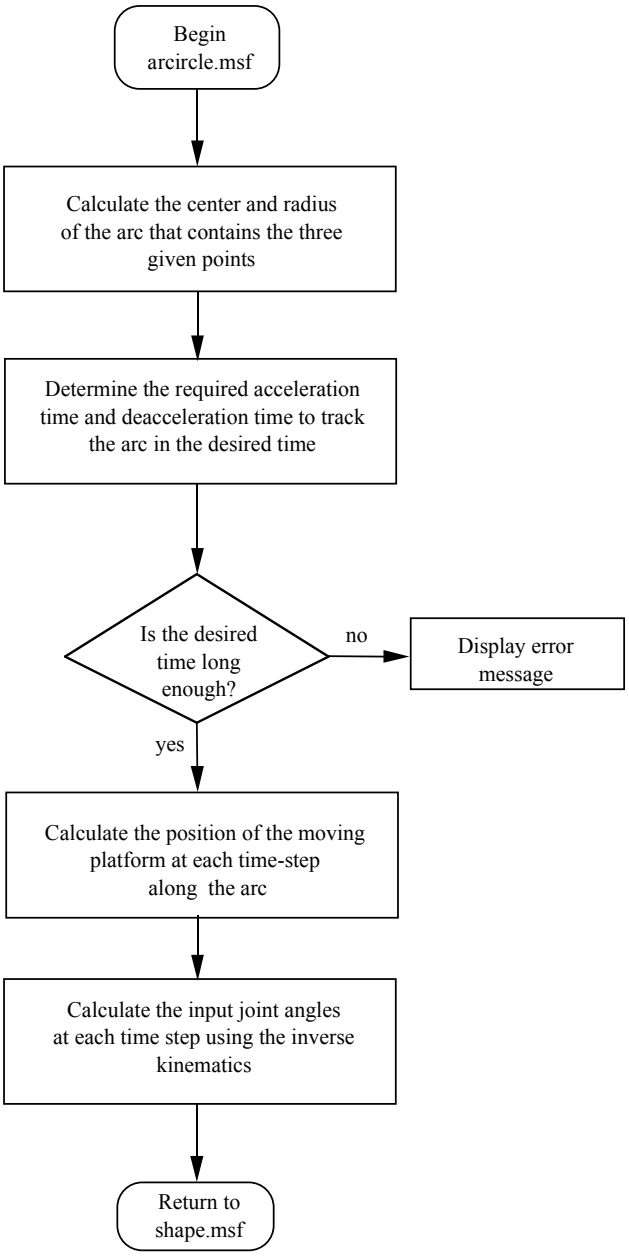
# Complete Single Leg PID Block Diagram



# Appendix F
# Trajectory Generator Flowcharts




Begin
polylin.msf
Calculate the unit vector formed by the two endpoints of the line segment, and the distance between the two endpoints
Determine the required acceleration time and deacceleration time to track the arc in the desired time
Is the desired time long enough?
no
Display error message
yes
Calculate the position of the moving platform at each time-step along the arc
Calculate the input joint angles at each time step using the inverse kinematics
Return to shape.msf


Begin
arcircle.msf
Calculate the center and radius
of the arc that contains the three
given points
Determine the required acceleration
time and deacceleration time to track
the arc in the desired time
Is the desired
time long
enough?
no
Display error
message
yes
Calculate the position of the moving
platform at each time-step
along the arc
Calculate the input joint angles
at each time step using the inverse
kinematics
Return to
shape.msf

# Appendix G

# Trajectory Generation Code

This is the trajectory generation code for the prototype manipulator written in Xmath math-script. The author of the code is Remi Taulemesse. The first listed function is *shape.msf*, which is the function that is envoked by the user to define the trajectory. All the other functions are called from *shape.msf*.

```
function [theta1,theta2,theta3,veloci,time,tt,gra,space,pt_user,
g1,g2,g3]=shape()

#################################################################
#{ The meaning of these output variables are the following:
theta1, theta2, theat3 : values of theta1(2 and 3) as a function
of time
veloci: velocity as a function of time
time: time increasing of the step time
tt: total time of the motion
space: cartesian coordinates of points followed by the moving
platform
graph: graph which shows the three angles as a function of
time and the path in the space
pt_user: coordinates of points given by the user
g1, g2, g3: initial value of each joints
To run this program, type the following in Xmath commands:
 [theta1,theta2,theta3,veloci,time,tt,gra,space,pt_user,g1,g2,g3]
 = shape();
}#
#################################################################

acc=getline("Enter the acceleration required:")
acc=makematrix(acc)
velocmax=getline("Enter the maximum velocity:")
velocmax=makematrix(velocmax)
dt = getline("Enter the step time wanted :")
dt = makematrix(dt)
#{ Input the first coordinates and turn them into a row vector }#
temp = getline("Enter the coordinates of the first point:")
```

```
x = index(temp,",")
px = makematrix(stringex(temp, 1,x-1))
temp = stringex(temp,x+1,length(temp))
y = index(temp,",")
py = makematrix(stringex(temp, 1,y-1))
temp = stringex(temp,y+1,length(temp))
pz = makematrix(temp)
p(1,:)=[px, py, pz]

#{
#################################################################
   Initialisation cur_pt: current point which will be the first
   point of each curve shape,
nbr: Total Number of points,
time: Vector which is the discete time,
tt: Total Time,
pt_user: Coordinates of points given by the user,
nb_user: Number of points given by the user,
theta: Points in angular coordinates.
#################################################################
}#
cur_pt = p(1,:)
what = 0
nbr = 0
tt = 0
pt_user = p(1,:)
nb_user = 1

#{ First Point }#
nbr = nbr+1
time(nbr) = tt
veloci(nbr) = 0
space(nbr,:) = p(1,:)
#{ Finding the angular coordinates of the first point }#
for i = 1:3
[angle,err] = inv_kin(p(1,:),i)
if (err == 1)
error("There are points out of the workspace","S")
endif
theta(nbr,i) = angle
endfor;


#{ creating the linear-circular path }#
while (what<>3)
what = getchoice("Three DOF Simulation", [ "Line Shape";...
"Circle Shape";"No more points"], {defaultchoice=1})
     if (what == 1)
       [last_pt,pt_user,theta,veloci,time,tt,nbr,nb_user,space]=
  polylin (cur_pt,pt_user,theta,veloci,time,tt,nbr,nb_user,
space,acc,velocmax,dt)
```

```
     elseif (what == 2)
       [last_pt,pt_user,theta,veloci,time,tt,nbr,nb_user,space]=
arcircle(cur_pt, pt_user, theta, veloci, time,tt,
nbr, nb_user, space, acc, velocmax, dt)
     else
 open = getchoice("Type of shape",["Open polygon";
        "Close Polygon"], {defaultchoice=1})
     endif

     if (nbr <> 1)
      cur_pt = last_pt
     endif

endwhile
#{ A last path will be done if necessary (close polygon case) }#
if (open == 2)
   final = getchoice("Three DOF Simulation", [ "Line Shape";
   "Circle Shape"], {defaultchoice=1})
   if (final == 1)
     [last_pt,theta,veloci,time,tt,nbr,space] = onelin(cur_pt,
space(1,:),theta,veloci,time,tt,nbr,space,acc,
velocmax, dt)
   else
     [last_pt,pt_user,theta,veloci,time,tt,nbr,nb_user,space] =
last_circle (cur_pt, space(1,:), pt_user,theta,
veloci, time,tt,nbr,nb_user,space, acc,velocmax,dt)
   endif
endif

where = getchoice("Use of datas",["Datas used for ac100 model";
                "Datas used for DADS model"],{defaultchoice=1})

#{ If open polygon, back to the first point, necessary condition
   for ac100 control }#
if ( (open == 1) & (where == 1))
   time(nbr+1) = tt+dt
   veloci(nbr+1) = 0
   for i = 1:3
theta(nbr+1,i) = theta(1,i)
   endfor
endif
theta1 = theta(:,1)
theta2 = theta(:,2)
theta3 = theta(:,3)
#{ If wanted change angles values to fit with Dads model and Send
    datas to Dads directory   }#
if (where ==2)
theta1 = theta1 - (pi/2)
theta2 = theta2 - (pi/2)
```

```
theta3 = theta3 - (pi/2)
result = [time, theta1]
print result file="/homes/rstamper/proto3/dav1.dat"
result = [time, theta2]
print result file="/homes/rstamper/proto3/dav2.dat"
result = [time, theta3]
print result file="/homes/rstamper/proto3/dav3.dat"
endif

#{
################################################################
Creating a graph with all datas
################################################################
}#

gra=plot(time, theta1, {rows=2,columns=2, title="THETA1"})
plot(time, theta2, {keep=gra, row=1, column=2, title="THETA2"})
plot(time, theta3, {keep=gra, row=2, title="THETA3"})
gr = plot(space(:,1),space(:,2),space(:,3),...
    {row=2, column=2, keep=gra, title="TRAJECTORY",...
  line_color=[1], line_width=[2], x_lab="x", y_lab="y",
             zlab="z"})
plot(pt_user(:,1), pt_user(:,2), pt_user(:,3),...
{keep=gr, row=2, column=2, marker_size=[1],...
marker_style=[2],marker_color=[48],...
line_style=[0], line_color=[1]})

#{
################################################################
Coefficient usefull for the simulation on ac100
################################################################
}#

g1=theta1(nbr)
g2=theta2(nbr)
g3=theta3(nbr)

endfunction
```

This is the function *polylin.msf* that is called from *shape.msf*.

```
function [last_pt,pt_userout,thetaout,velociout,timeout,ttout,nbrout,
nb_userout,spaceout] = polylin(first_pt,pt_user,theta,veloci,time,tt,
nbr,nb_user,space,acc,velocmax,dt)
##############################################################
#{ This function create a polygon thanks to the coordinates of points
given by the user.  All these output and input are useful to fill
out datas matrix.  Output variables whose name end by "out" are
the same than input variables without "out" at the end.
}#
##############################################################

set format fixed
set precision 7

#{ Taking datas from the user }#

p(1,:) = first_pt

#{ Input the coordinates of the points which will create the
polygon. THE FIRST POINT IS ALREADY TAKEN IN ACOUNT. }#

nbpts = getline("Enter the number of points wanted:")
nbpts = makematrix(nbpts)
nbpts = nbpts + 1  # Adding the first point

loop = nbpts - 1

nb_userout = nb_user + nbpts

for i = 2:nbpts
temp = getline("Enter the coordinates of the point:")
x = index(temp,",")
px = makematrix(stringex(temp, 1,x-1))
temp = stringex(temp,x+1,length(temp))
y = index(temp,",")
py = makematrix(stringex(temp, 1,y-1))
temp = stringex(temp,y+1,length(temp))
pz = makematrix(temp)
p(i,:)=[px, py, pz]
endfor

for i=1:loop

#{ Finding the magnitude of P1P2 vector }#

magn(i) = sqrt((p(i+1,1)-p(i,1))^2 + (p(i+1,2)-p(i,2))^2 +
(p(i+1,3)-p(i,3))^2)


#{ Finding the unit vector supporting the line we are suppose to follow }#
for j=1:3
```

```
norm_vec(i,j) = (p(i+1,j)-p(i,j)) / magn(i)
endfor

#{ Reading of the trajectroy time and testing if it fits }#

temp = getline("Enter the trajectory time wanted :")
tf(i) = makematrix(temp)

tacc(i) = (tf(i)/2)-(sqrt(acc^2*tf(i)^2-4*acc*magn(i))/(2*acc))

while((tf(i)<(2*sqrt(magn(i)/acc))) | (velocmax<(acc*tacc(i))))

if (tf(i)<2*sqrt(magn(i)/acc))
beep "Time too short ... Unreachable"
else
beep "With this duration, velocity
will be greater than the
maximum velocity given."
endif

temp = getline("Enter the trajectory time wanted :")
tf(i) = makematrix(temp)
endwhile

endfor



#{ Initialisation }#

tlast = 0 # for time at the end of each line
veloc = 0 # for velocity



for i = 1:loop

   desacc = 0 # for desacceleration
   tempt = 0 # for temporary time

#{ Motion and calculation along the arc circle.
   This loop starts at dt because the initial point has
already been compute }#

   for t = dt:dt:tf(i)
       [s_cur, veloc, desacc, tempt] = next_point(acc,
tf(i), dt, t, tacc(i), tempt, desacc)
  #{ s_cur is for current curvilinear abscissa }#


       if (tempt>=0)
#{ Necessary condition to eradicate numerical errors }#

  pt(1) = p(i,1) + s_cur*norm_vec(i,1)
          pt(2) = p(i,2) + s_cur*norm_vec(i,2)
          pt(3) = p(i,3) + s_cur*norm_vec(i,3)

  nbr = nbr+1
  space(nbr,:) = pt'

        time(nbr) = t + tlast + tt
  veloci(nbr) = veloc
  for j = 1:3
```

```
#{ Calculating the inverse kinematic for each point }#
     [angle,err] = inv_kin(pt, j)
     if ( err == 1)
error("There are points out of the workspace","S")
     endif
     theta(nbr,j) = angle
  endfor
       endif
   endfor
#{ Last Point }#
   nbr = nbr+1
   time(nbr) = tf(i) + tlast + tt
   veloci(nbr) = 0
   space(nbr,:) = p(i+1,:)
   for j = 1:3
[angle,err] = inv_kin(p(i+1,:),j)
if ( err == 1)
error("There are points out of the workspace","S")
endif
theta(nbr,j) = angle
   endfor
   tlast = tlast + tf(i)
endfor
last_pt = space(nbr,:)
for i = 1:nbpts
pt_user(nb_user + i,:) = p(i,:)
endfor
#{ Puting datas in output variables }#
pt_userout = pt_user
spaceout = space
thetaout = theta
velociout = veloci
timeout = time
nbrout = nbr
ttout = tt + tlast
endfunction
```

This is the function *arcircle.msf* that is called from *shape.msf*.

```
function [last_pt,pt_userout,thetaout,velociout,timeout,ttout,
nbrout,nb_userout,spaceout] = arcircle(first_pt,pt_user,theta,
veloci,time,tt,nbr,nb_user,space,acc,velocmax,dt)
################################################################
#{ This function create a circle thanks to the coordinates of
three points given by the user.  All these output and input are
useful to fill out datas matrix.
Output variables whose name end by "out" are the same than input
variables whithout "out" at the end.
}#
################################################################
set format fixed
set precision 7
#{ Taking datas from the user }#

p(1,:) = first_pt
for i = 2:3
temp = getline("Enter the coordinates of the point:")
x = index(temp,",")
px = makematrix(stringex(temp, 1,x-1))
temp = stringex(temp,x+1,length(temp))
y = index(temp,",")
py = makematrix(stringex(temp, 1,y-1))
temp = stringex(temp,y+1,length(temp))
pz = makematrix(temp)
p(i,:)=[px, py, pz]
endfor
nb_userout = nb_user + 3

#{ Finding coordinates of the center of the circle }#
c1 = (p(2,1)-p(1,1))^2 + (p(2,2)-p(1,2))^2 + (p(2,3)-p(1,3))^2
c2 = (p(3,1)-p(1,1))^2 + (p(3,2)-p(1,2))^2 + (p(3,3)-p(1,3))^2
c3 = (p(2,1)-p(1,1))*(p(3,1)-p(1,1)) + ...
(p(2,2)-p(1,2))*(p(3,2)-p(1,2)) + ...
(p(2,3)-p(1,3))*(p(3,3)-p(1,3))
for i = 1:3
   cen(i) = p(1,i) + (c2*(c1-c3))/(2*(c1*c2-c3^2))
*(p(2,i)-p(1,i))
+ (c1*(c2-c3))/(2*(c1*c2-c3^2))*(p(3,i)-p(1,i))
endfor
#{ Finding the radius of the circle }#
r = sqrt((cen(1)-p(1,1))^2 + (cen(2)-p(1,2))^2
+ (cen(3)-p(1,3))^2)
```

```
#{ Finding the vector which will define this circle }#
mn = [(p(2,2)-p(1,2))*(p(3,3)-p(1,3))
- (p(2,3)-p(1,3))*(p(3,2)-p(1,2));
     (p(2,3)-p(1,3))*(p(3,1)-p(1,1))
- (p(2,1)-p(1,1))*(p(3,3)-p(1,3));
      (p(2,1)-p(1,1))*(p(3,2)-p(1,2))
- (p(2,2)-p(1,2))*(p(3,1)-p(1,1))]
magn_mn = sqrt( mn(1)^2 + mn(2)^2 + mn(3)^2)
ext = cen + (r/magn_mn) * mn


#{ Finding the magnitude of P1P2 vector and the changing base
homogen matrix }#

if ( (ext(1) <> cen(1)) | (ext(2) <> cen(2)) )
a = - (ext(2)-cen(2)) / (sqrt((ext(1)-cen(1))^2 +
(ext(2)-cen(2))^2))
b = (ext(1)-cen(1)) / (sqrt((ext(1)-cen(1))^2 +
(ext(2)-cen(2))^2))
mat(1,1) = (b*(ext(3)-cen(3))) / r
mat(2,1) = (-a*(ext(3)-cen(3))) / r
mat(3,1) = ( a*(ext(2)-cen(2))) / r
- (b*(ext(1)-cen(1)) ) / r
mat(4,1) = 0
mat(1,2) = a
mat(2,2) = b
mat(3,2) = 0
mat(4,2) = 0
else # if z vector coordinates and z axis are parallel
mat(1,1) = 1
mat(2,1) = 0
mat(3,1) = 0
mat(4,1) = 0
mat(1,2) = 0
mat(2,2) = sign(ext(3)-cen(3))
mat(3,2) = 0
mat(4,2) = 0
endif

mat(1,3) = (ext(1)-cen(1)) / r
mat(2,3) = (ext(2)-cen(2)) / r
mat(3,3) = (ext(3)-cen(3)) / r
mat(4,3) = 0
for i = 1:3
mat(i,4) = cen(i)
endfor
mat(4,4) = 1
```

```
#{ Finding the angle and the length of the arc }#
inv_mat = mat(1:3,1:3)'
ini_pt = p(1,:) - cen'
ini_pt = (1/r)*inv_mat*ini_pt'
fin_pt = p(3,:) - cen'
fin_pt = (1/r)*inv_mat*fin_pt'
if (ini_pt(2)>=0)
ini_angle = acos(ini_pt(1))
else
ini_angle = 2*pi - acos(ini_pt(1))
endif
if (fin_pt(2)>=0)
fin_angle = acos(fin_pt(1))
else
fin_angle = 2*pi - acos(fin_pt(1))
endif
if (fin_angle<ini_angle)
fin_angle = fin_angle + 2*pi
endif
ang = fin_angle - ini_angle
len = ang*r

#{ Reading of the trajectroy time and testing if it fits }#
temp = getline("Enter the trajectory time wanted :")
tf = makematrix(temp)
tacc = (tf/2)-(sqrt(acc^2*tf^2-4*acc*len)/(2*acc))
while((tf<(2*sqrt(len/acc))) | (velocmax<(acc*tacc)))
if (tf<2*sqrt(len/acc))
beep "Time too short ... Unreachable"
else
beep "With this duration,
velocity will be greater than the
maximum velocity given."
endif
temp = getline("Enter the trajectory time wanted :")
 tf = makematrix(temp)
endwhile


#{ Initialisation }#
veloc = 0 # for velocity
tempt = 0 # for temporary time
desacc = 0 # for desacceleration


#{ Motion and calculation along the arc circle.
This loop starts at dt because the initial point has
already been compute }#
```

```
for t = dt:dt:tf
       [s_cur, veloc, desacc, tempt] = next_point ...
(acc, tf, dt, t, tacc, tempt, desacc)
    #{ s_cur is for current curvilinear abscissa }#


       if (tempt>=0)
#{ Necessary condition to eradicate numerical errors }#
  alpha = s_cur / r  # Angle calculate from the origin
  nbr = nbr+1
  temp = (mat * [r*cos(alpha+ini_angle);
r*sin(alpha+ini_angle); 0;1])'
  space(nbr,:) = temp(1:3)
        time(nbr) = t + tt
  veloci(nbr) = veloc
  for j = 1:3
#{ Calculating the inverse kinematic for each point }#
     [angle,err] = inv_kin(space(nbr,:), j)
     if ( err == 1)
error("There are points out of the workspace","S")
     endif
     theta(nbr,j) = angle
  endfor
       endif
endfor

#{ Last Point }#
nbr = nbr+1
time(nbr) = tf + tt
veloci(nbr) = 0
temp = (mat * [r*fin_pt;1])'
space(nbr,:) = temp(1:3)

for j = 1:3
[angle,err] = inv_kin(space(nbr,:),j)
if ( err ==1 )
error("There are points out of the workspace","S")
endif
theta(nbr,j) = angle
endfor

last_pt = space(nbr,:)

for i = 1:3
pt_user(nb_user + i,:) = p(i,:)
endfor

#{ Puting datas in output variables }#

pt_userout = pt_user
spaceout = space
thetaout = theta
velociout = veloci
timeout = time
nbrout = nbr
ttout = tt + tf
endfunction
```

```
function [s, veloc, desaccout, temptout] = next_point
(cur_acc, tf, dt, t, tacc, tempt, desacc)

#{ Acceleration part }#
if (t<tacc)
s = 0.5 * cur_acc*t*t
veloc = cur_acc*t

#{ Desacceleration part }#
elseif (t>=tf-tacc)

#{ first loop of desacceleration }#
if (desacc == 0)
desacc = 1
tempt = tacc - dt
endif

#{ we use the symetrical aspect of the curve }#
if (tempt>=0)
s = (- cur_acc)*(tacc*tacc - tf*tacc + 0.5*tempt*tempt)
veloc = cur_acc*tempt
tempt = tempt - dt
endif

#{ Linear part }#
else
s = (- 0.5*cur_acc*tacc*tacc) + cur_acc*tacc*t
veloc = cur_acc*tacc

endif

desaccout = desacc
temptout = tempt

endfunction
```

```
function [angle,err]=inv_kin(pt,i)
#{ Inverse kinematic function:
i is the joint number,
pt is the cartesian coordinates of the point
   The output are:
the considered angle
an error message: err = 1 if point is out of the workspace
  else err = 0}#

#{ Length of the differents legs }#
r = 5
a = 8
e = 0.625
b = 10
d = 0.625
c = 5
#{ Orientation of the legs and conversion into radian }#
teta0(1)=0
teta0(2)=120
teta0(3)=240
for j=1:3
teta0(j)=(teta0(j)/360)*2*pi
endfor

pu = pt(1)*cos(teta0(i)) + pt(2)*sin(teta0(i)) - r
pv = -pt(1)*sin(teta0(i)) + pt(2)*cos(teta0(i))
pw = pt(3)

#{ If points out of the workspace }#

if ((pv/b)>1)
err = 1
else
teta3 = acos(pv/b)
err = 0
endif

l0 = pu*pu + pw*pw + 2*c*pu - 2*a*pu - b*b*sin(teta3)*sin(teta3)
l0 = l0 - 2*b*e*sin(teta3) - 2*b*d*sin(teta3) - 2*d*e - 2*a*c
l0 = l0 + a*a + c*c - d*d - e*e
l1 = -4*a*pw
l2 = pu*pu + pw*pw + 2*c*pu + 2*a*pu - b*b*sin(teta3)*sin(teta3)
l2 = l2 - 2*b*e*sin(teta3) - 2*b*d*sin(teta3) - 2*d*e + 2*a*c
l2 = l2 + a*a + c*c - d*d - e*e
discr = l1*l1-4*l2*l0
temp1 = 2*atan((-l1+sqrt(discr))/(2*l2))
temp2 = 2*atan((-l1-sqrt(discr))/(2*l2))

if (abs(temp1) < abs(temp2))
angle = temp1
else
angle = temp2
endif

#{ If points out of the workspace }#

if ( is(angle, {!real})==1)
err = 1
else
err = 0
endif
endfunction
```

# REFERENCES


- Alciatore, D., and Ng, C., 1994, "Determining Manipulator Workspace Boundaries Using the Monte Carlo Method and Least Squares Segmentation," *ASME Robotics: Kinematics, Dynamics, and Controls,* DE-Vol. 72, pp. 141-146.

- Angeles, J., and Lopez-Cajun, C., 1988, "The Dexterity Index of Serial-Type Robotic Manipulators," *ASME Trends and Developments in Mechanisms, Machines, and Robotics,* pp. 79-84.

- Aria, T., Cleary, K., Homma, K., Adachi, H., and Nakamura, T., 1991, "Development of Parallel Link Manipulator for Underground Excavation Task," 1991 International Symposium on Advanced Robot Technology, pp. 541-548.

- Arimoto, S., Kawamura, S., and Miyazaki, F., 1984, "Bettering Operation of Dynamic Systems by Learning: A New Control Theory for Servomechanism or Mechatronic Systems," *Proceedings of 23rd Conference on Decision and Control,* pp. 1064-1069.

- Aronson, R., 1996, "A Bright Horizon for Machine Tool Technology," *Manufacturing Engineering,* March, pp. 57-70.

- Åström, K., and Hägglund, T., 1995, *PID Controllers: Theory, Design, and Tuning,* $2^{nd}$ *Ed.,* Instrument Society of America.

- Åström, K., and Rundqwist, L., 1989, "Integrator Windup and How to Avoid It," *Proceedings of the American Control Conference,* pp. 1693-1698.

- Clavel, R., 1988, "Delta, A Fast Robot with Parallel Geometry," *Proceedings of the 18th International Symposium on Industrial Robots,* pp. 91-100.

- Fertik, H., and Ross, C., 1967, "Direct Digital Control Algorithm with Anti-Windup Feature." *ISA Transactions,* Vol. 6, No. 4, pp. 317-328.

- Fichter, E., 1986, "A Stewart Platform Based Manipulator: General Theory and Practical Construction," *International Journal of Robotics Research,* Vol. 5, pp. 157-182.

- Gosselin, C., 1996, "Analysis and Synthesis of Underactuated Force Generating Mechanisms," Proceedings of the 1996 ASME Design Engineering Technical Conference, MECH:1564.

- Gosselin, C., and Hamel, J., 1994, "The Agile Eye: A High-Performance Three-Degree-of-Freedom Camera-Orienting Device," *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 781-786.

- Gosselin, C., Lavoie, E., and Toutant P., 1992, "An Efficient Algorithm for the Graphical Representation for the Three-Dimensional Workspace of Parallel Manipulators," *ASME Robotics, Spatial Mechanisms, and Mechanical Systems,* DE-Vol.45, pp. 323-328.

- Gosselin, C., and Sefrioui, J., 1992, "Determination of the Singularity Loci of Spherical Three-Degree-of-Freedom Parallel Manipulators," *Proceedings of the 1992 ASME Design Engineering Technical Conference,* DE-Vol. 45, pp. 329-336.

- Gosselin, C., and Angeles, J., 1991, "A Global Performance Index for the Kinematic Optimization of Robotic Manipulators," *Journal of Mechanical Design,* Vol. 113, pp. 220-226.

- Gosselin, C., 1990a, "Stiffness Mapping for Parallel Manipulators," *IEEE Transactions on Robotics and Automation,* Vol. 6, pp. 377-382.

- Gosselin, C., 1990b, "Determination of the Workspace of 6-DOF Parallel Manipulators," *ASME Journal of Mechanical Design,* Vol. 112, pp. 331-336.

- Gosselin, C., and Angeles, J., 1990, "Singularity Analysis of Closed-Loop Kinematic Chains," *IEEE Transactions on Robotics and Automation,* Vol.6, pp. 281-290.

- Gosselin, C., and Angeles, J., 1989, "The Optimum Kinematic Design of a Spherical Three-Degree-of-Freedom Parallel Manipulator," *Journal of Mechanisms, Transmissions, and Automation in Design,* Vol. 111, pp. 202-207.

- Gosselin, C., and Angeles, J., 1988, "The Optimum Kinematic Design of a Planar Three-Degree-of-Freedom Parallel Manipulator," *Journal of Mechanisms, Transmissions, and Automation in Design,* Vol. 110, pp. 35-41.

- Griffis, J., 1985, *The Theory of Classical Dynamics,* Cambridge University Press.

- Griffis, M., and Duffy, J., 1989, "A Forward Displacement Analysis of a Class of Stewart Platforms," *Journal of Robotic Systems,* Vol. 6, pp. 703-720.

- Guglielmetti, P., and Longchamp, R., 1994, "A Closed Form Inverse Dynamics Model of the Delta Parallel Robot," *Proceedings from the International Federation of Automatic Control Conference on Robot Control 1994,* pp. 39-44.

- Hauser, J., 1987, "Learning Control for a Class of Nonlinear Systems," *Proceedings of the 26th Conference on Decision and Control,* pp. 859-860.

- Hunt, K., 1983, "Structural Kinematics of In-Parallel Actuated Robot-Arms," *ASME Journal of Mechanisms, Transmissions, and Automation in Design,* Vol. 105, pp. 705-712.

- Husain, M., and Waldron, K., 1994, "Position Kinematics of a Three-Limbed Mixed Mechanism," *ASME Journal of Mechanical Design,* Vol. 116, pp. 924-929.

- Innocenti, C., and Parenti-Castelli, V., 1993, "Closed-Form Direct Position Analysis of a 5-5 Parallel Mechanism," *ASME Journal of Mechanical Design,* Vol. 115, pp. 515-521.

- Kantor, G., 1995, "Linear Control Theory as Applied to Smart Structures," *Master's Thesis,* University of Maryland at College Park, pp. 35-41.

- Klein, C., and Blaho, B., 1987, "Dexterity Measures for the Design and Control of Kinematically Redundant Manipulators," *The International Journal of Robotics Research,* Vol. 6, pp. 72-82.

- Lazard, D., and Merlet, J.P., 1994, "The (True) Stewart Platform has 12 Configurations," *Proceedings of the 1994 IEEE Robotics and Automation Conference,* pp. 2160-2165.

- Lewis, F., Abdallah, C., and Dawson, D., 1993, *Control of Robot Manipulators,* MacMillan Publishing Company, pp. 125-130.

- Lin, W., Crane, C., and Duffy, J., 1994, "Closed Form Displacement Analysis of the 4-5 In Parallel Platforms," *ASME Journal of Mechanical Design,* Vol. 116, pp. 47-53.

- Lin, W., Griffis, M., and Duffy, J., 1992, "Forward Displacement Analyses of the 4-4 Stewart Platforms," *ASME Journal of Mechanical Design,* Vol. 114, pp. 444-450.

- Ma, O., and Angeles, J., 1991, "Architecture Singularities of Platform Manipulators," *Proceedings of the 1991 IEEE International Conference on Robotics and Automation,* pp. 1542-1547.

- Merlet, J-P., 1996, "Workspace-oriented Methodology for Designing a Parallel Manipulator," *Proceedings of the 1996 IEEE International Conference on Robotics and Automation,*, pp. 3726-3731.

- Merlet, J-P., 1989, "Singular Configurations of Parallel Manipulators and Grassman Geometry," *International Journal of Robotics Research,* Vol. 8, pp. 45-56.

- Miller, K., 1995, "Experimental Verification of Modeling of DELTA Robot Dynamics by Direct Application of Hamilton's Principle," *IEEE International Conference on Robotics and Automation,* pp. 532-537.

- Miller, K., and Clavel, R., 1992, "The Lagrange-based Model of Delta-4 Robot Dynamics," *Robotersysteme 8,* pp. 49-54.

- Mouly, N., and Merlet, J., 1992, "Singular Configurations and Direct Kinematics of a New Parallel Manipulator," *Proceedings of the 1992 IEEE International Conference on Robotics and Automation,* Vol. 1, pp. 338-343.

- Nanua, P., Waldron, K., and Murthy, V., 1990, "Direct Kinematic Solution of a Stewart Platform," *IEEE Transactions on Robotics and Automation,* Vol. 6, pp. 438-444.

- Oblak, D., Kohli, D., 1988, "Boundry Surfaces, Limit Surfaces, Crossable and Noncrossable Surfaces in Workspace of Mechanical Manipulators," *Journal of Mechanisms, Transmissions, and Automation in Design,* Vol. 110, pp. 389-396.

- Park, F., and Brockett, R., 1994, "Kinematic Dexterity of Robotic Mechanisms," *The International Journal of Robotics Research,* Vol. 13, pp. 1-15.

- Pierrot F., Reynaud, C., and Fournier, A., 1990, "Delta: A Simple and Efficient Parallel Robot," *Robotica*, Vol. 6, pp. 105-109.

- Ragahavan, M., and Roth, B., 1995, "Solving Polynomial Systems for the Kinematic Analysis and Synthesis of Mechanisms and Robot Manipulators," *Transactions of the ASME Special 50th Anniversary Design Issue,* Vol. 117, pp. 71-79.

- Ragahavan, M., and Roth, B., 1993, "Inverse Kinematics of the 6R Manipulator of General Geometry," *ASME Journal of Mechanical Design,* Vol. 115, No. 3, pp. 502-508.

- Raster, J., and Perel, D., 1988, "Generation of Manipulator Workspace Boundry Geometry Using the Monte Carlo Method and Interactive Computer Graphics," *ASME Trends and Developments in Mechanisms, Machines, and Robotics,* pp. 299-305.

- Salmon, G., 1964, *Lessons Introductory to the Modern Higher Algebra,* Chelsea Publishing Co., NY.

- Salisbury, J., and Craig, J., 1982, "Articulated Hands: Force Control and Kinematic Issues," *The International Journal of Robotics Research,* Vol. 1, pp. 4-17.

- Shi, X., and Fenton, R., 1992, "Structural Instabilities in Platform-Type Parallel Manipulators Due to Singular Configurations," *ASME Proceedings of the 1992 ASME Design Engineering Technical Conference,* DE-Vol. 45, pp. 347-352.

- Sternheim, F., 1988, "Tridimensional Computer Simulations of a Parallel Robot. Results for the Delta4 Machine," *Proceedings of the 18th International Symposium on Industrial Robots,*, pp. 333-340.

- Stewart, D., 1965, "A Platform with Six Degrees of Freedom," *Proc. Institute of Mechanical Engineering,* Vol. 180, pp. 371-386.

- Tahmasebi, F., and Tsai, L.W., 1995, "On the Stiffness of a Novel Six-DOF Parallel Minimanipulator," *Journal of Robotic Systems,* Vol. 12, No. 12, pp. 845-856.

- Tomizuka, M., Tsao, T.C., Chew, K.K., "Analysis and Synthesis of Discrete-Time Repetitive Controllers," *ASME Journal of Dynamic Systems, Measurement, and Control,* Vol. 111, pp. 353-358.

- Tsai, L.W., 1997, "Multi-degree-of-freedom Mechanisms for Machine Tools and the Like," U.S. Patent, No. 5,656,905.

- Waldron, K., and Hunt, K., 1988, "Series-Parallel Dualities in Actively Coordinated Mechanisms," *Proceedings of the 4th International Symposium on Robotic Research,* MIT Press, pp. 175-181.

- Wang, D., 1995, "A Simple Iterative Learning Controller for Manipulators with Flexible Joints," *Automatica*, Vol. 31, pp. 1341-1344.

- Wang, J., and Gosselin, C., 1996, "Kinematic Analysis and Singularity Loci of Spatial Four-Degree-of-Freedom Parallel Manipulators," *Proceedings of The 1996 ASME Design Engineering Technical Conference,* MECH:1106.

- Whitney, D., Lozinski, C., and Rourke, J., 1986, "Industrial Robot Forward Calibration Method and Results," *Journal of Dynamic Systems, Measurements, and Control,* Vol. 108, pp. 1-8.

- Yamamoto, S., and Hashimoto, I., 1991, "Present Status and Future Needs: The View from Japanese Industry," *Proceedings of the Fourth International Conference on Chemical Process Control*, pp. 1-28.

- Zhang, C., and Song, S., 1994, "Forward Position Analysis of Nearly General Stewart Platforms," *ASME Journal of Mechanical Design,* Vol. 114, pp. 444-450.

- Zlatanov, D., Fenton, R., and Benhabib, B., 1995, "A Unifying Framework for Classification and Interpretation of Mechanism Singularities," *ASME Journal of Mechanical Design,* Vol. 117, pp. 566-572.