44103351

SPHERICALLY-ACTUATED PLATFORM MANIPULATOR

A Thesis Presented to

The Faculty of the

Fritz J. and Dolores H. Russ
College of Engineering and Technology

Ohio University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

by

Dana B. Poling

June, 2000

Thesis
M
2000
POLI

OHIO UNIVERSITY
LIBRARY

copy 12 6/19/01

THIS THESIS ENTITLED
“SPHERICALLY-ACTUATED PLATFORM MANIPULATOR”

by Dana B. Poling

has been approved

for the Department of Mechanical Engineering
and the Russ College of Engineering and Technology

Robert L. Williams II
Assistant Professor

Warren K. Wray, Dean
Fritz J. and Dolores H. Russ
College of Engineering and Technology

TABLE OF CONTENTS

LIST OF TABLES	v
LIST OF FIGURES	vi
ABBREVIATIONS AND SYMBOLS.....	ix
 CHAPTER 1: INTRODUCTION.....	 1
1.1 Platform Manipulators	1
1.2 Literature Review.....	2
1.3 Thesis Objectives.....	5
 CHAPTER 2: SPAM SYSTEM CONFIGURATION	 8
2.1 System Hardware	8
2.1.1 SPAM Design and Construction	9
2.1.2 Omni-Wrist	11
2.1.3 Electronics Cabinet.....	13
2.1.4 MultiQ-3 I/O Boards.....	16
2.2 System Software	18
2.2.1 Matlab/Simulink	18
2.2.2 WinCon	19
 CHAPTER 3: KINEMATICS.....	 22
3.1 <i>SRU</i> SPAM Version.....	22
3.1.1 D-H Parameters.....	23
3.1.2 Forward Pose Kinematics	26
3.1.3 Inverse Pose Kinematics	31
3.1.4 Inverse Rate Kinematics	33
3.1.5 Singularities	35
3.2 <i>SPU</i> SPAM Version.....	36
3.2.1 D-H Parameters.....	37
3.2.2 Pose and Rate Kinematics.....	39
3.2.3 Singularities	40
 CHAPTER 4: CONTROL	 42
4.1 Single Joint Controller Design.....	42
4.2 Independent Joint Control.....	51
4.3 Cartesian Control	57
 CHAPTER 5: EXPERIMENTATION AND RESULTS.....	 68
5.1 Kinematics Examples	68
5.2 Controls Examples.....	82
 CHAPTER 6: CONCLUSION.....	 91
6.1 Conclusion	91

6.2 Future Work	95
REFERENCES.....	99
APPENDIX A: <i>SRU</i> SPAM CAD DRAWINGS.....	102
APPENDIX B: OPERATION PROCEDURE	111
APPENDIX C: KINEMATICS TERMS	114
ABSTRACT	

LIST OF TABLES

Tables	Page
1. Joint Limits for the Left <i>SRU</i> Chain	11
2. Gear Ratios for the Omni-Wrist.....	13
3. Left <i>SRU</i> Leg D-H Parameters.....	25
4. Left <i>SPU</i> Leg D-H Parameters.....	38
5. Omni-Wrist Physical Parameters for Controller Design	52

LIST OF FIGURES

Figure	Page
1. SPherically-Actuated platform Manipulator (SPAM) Concept.....	5
2. Overall SPAM System	8
3. Solid Model of the SPAM Robot.....	9
4. The SPAM Robot.....	10
5. Diagram of Left <i>SRU</i> Chain.....	11
6. The Omni-Wrist.....	12
7. Inside the Electronics Cabinet	14
8. Control Panel for Electronics Cabinet.....	16
9. The MultiQ-3 Boards.....	17
10. Simulink Diagram for Simulation of Position Control.....	19
11. Simulink Diagram for Position Control of Hardware.....	20
12. <i>SRU</i> Version of SPAM	23
13. Kinematic Diagram of Left <i>SRU</i> Chain.....	24
14. Example of a Singular Position	36
15. <i>SPU</i> Version of SPAM	37
16. Kinematic Diagram of Left <i>SPU</i> Chain.....	38
17. Truly Spherically-Actuated Version of SPAM.....	41
18. General System Diagram for Single Joint/Link Controller Design	43
19. Block Diagram of Open Loop System.....	44
20. Free Body Diagram of Single Joint System.....	46

21. Simulink Model for Simulation of PID Position Control of Independent Joint.....	51
22. Input Dialog Box for Simulation Program.....	53
23. PID Controller Details	54
24. Input Menu for Controller Gains	54
25. Simulink Model for Hardware Control of Independent Joint.....	56
26. Reduced Degree of Freedom SPAM Robot.....	58
27. Simulink Diagram for Simulating Control of Plunge Motion for SPAM.....	62
28. System Inputs Dialog Box for Cartesian Control	63
29. Inverse Kinematics Subsystem	63
30. Forward Kinematics Subsystem.....	66
31. Nominal Configuration for Forward Pose Kinematics Example.....	69
32. Counts to Convergence for Forward Pose Kinematics Example.....	70
33. Platform Cartesian Displacement for Forward Pose Kinematics Example.....	71
34. Platform Angular Displacement for Forward Pose Kinematics Example	72
35. Counts to Convergence for First Inverse Pose Kinematics Example	74
36. Left Leg Results for the First Inverse Pose Kinematics Example.....	75
37. Right Leg Results for the First Inverse Pose Kinematics Example	75
38. Counts to Convergence for Second Inverse Pose Kinematics Example.....	76
39. Left Leg Results for the Second Inverse Pose Kinematics Example	77
40. Right Leg Results for the Second Inverse Pose Kinematics Example.....	77
41. Four View Graphical Representation of SPAM After Following Trajectory	78
42. Left Leg Results of Inverse Rate Program	80

43. Right Leg Results of Inverse Rate Program.....	81
44. Initial Attempt at PID Position Control of Single Joint/Link System.....	84
45. Tuned PID Position Control of Single Joint/Link System.....	85
46. Simulation of Plunge Position Control.....	86
47. Simulated Control of θ_2 for Plunge Motion.....	87
48. Simulated Control of θ_5 for Plunge Motion.....	87
49. Hardware Plunge Position Control	89
50. Hardware Control of θ_2 for Plunge Motion	90
51. Hardware Control of θ_5 for Plunge Motion	90
52. Example of Full Cartesian Positioning of SPAM.....	96

ABBREVIATIONS AND SYMBOLS

a_{i-1}	Denavit-Hartenberg Parameter
$\{B\}$	Base coordinate frame
$c\alpha$	$\cos(\alpha)$
CAD	Computer Aided Design
$c\beta$	$\cos(\beta)$
$c\phi_i$	$\cos(\phi_i)$
$c\gamma$	$\cos(\gamma)$
c_i	$\cos(\theta_i)$
C_L	Rotational viscous friction coefficient for load
C_M	Rotational viscous friction coefficient for motor
DC	Direct Current
$D-H$	Denavit-Hartenberg
d_i	Denavit-Hartenberg parameter
dof	degree-of-freedom
EOM	Equation of Motion
F_i	Function used for Newton-Raphson method
I/O	Input/Output
$i_A(t)$	Motor armature current
J_E	Effective inertia for motor and load combined
J_L	Jacobian for left leg inverse rate kinematics
$J_L(t)$	Load inertia

J_{LA}	Jacobian for left leg inverse rate kinematics, active joints only
J_M	Motor inertia
J_{NR}	Newton-Raphson Jacobian
J_R	Jacobian for right leg inverse rate kinematics
J_{RA}	Jacobian for right leg inverse rate kinematics, active joints only
k_b	Motor's back emf constant
k_D	Derivative gain for PID controller
k_I	Integral gain for PID controller
k_P	Proportional gain for PID controller
k_T	Motor's torque constant
L	Motor armature inductance
LED	Light Emitting Diode
L_i	Robotic link length
M	Combination of inverse rate Jacobians, active joints only
N	Gear ratio
$NASA$	National Aeronautics and Space Administration
$\{P\}$	Platform coordinate frame
${}^A P_B$	Position of frame B relative to frame A, expressed in frame A terms
PC	Personal Computer
PID	Proportional, Integral, Derivative
PO	Percent Overshoot
ppr	pulses per revolution

R	Motor armature resistance
${}^B_A R$	Rotation matrix of frame B relative to frame A
r_{ij}	i,j entry of rotation matrix
RTW	Real-Time Workshop
$RUUR$	Revolute-Universal-Universal-Revolute
s_{1-3}	Roots of characteristic polynomial
$s\alpha$	$\sin(\alpha)$
$s\beta$	$\sin(\beta)$
$s\phi_i$	$\sin(\phi_i)$
$s\gamma$	$\sin(\gamma)$
s_i	$\sin(\theta_i)$
$SPAM$	SPherically-Actuated platform Manipulator
SPU	Spherical-Prismatic-Universal
SRU	Spherical-Revolute-Universal
${}^B_A T$	Homogeneous transformation matrix of frame B relative to frame A
T_s	Settling time
$V_B(t)$	Motor armature back emf
VGT	Variable Geometry Truss
$V_{ref}(t)$	Input voltage for motor
x,y,z	Cartesian positions
X_0	Initial guess for Newton-Raphson forward pose kinematics
$\dot{x}, \dot{y}, \dot{z}$	Cartesian velocities

$\dot{X}$	Vector of Cartesian rates and angular rates
X_k	Value of X at k th step
${}^i\hat{z}_i$	Unit vector in z direction of the i th frame, expressed in i frame terms
α	Euler angle
α_{i-1}	Denavit-Hartenberg parameter
β	Euler angle
$\delta\Theta_k$	Error vector for Newton-Raphson inverse kinematics
δX_k	Error vector for Newton-Raphson forward kinematics
ε	Tolerance for unknown variables in Newton-Raphson methods
ϕ_{1-6}	Passive joint angles
$\dot{\phi}_i$	Passive joint rates
γ	Euler angle
Θ_0	Initial guess for Newton-Raphson inverse pose kinematics
θ_{1-6}	Active joint angles
$\dot{\theta}_i$	Active joint rates
$\dot{\Theta}$	Vector of joint rates
Θ_k	Value of Θ at the k th step
θ_L	Load angle
$\dot{\Theta}_L$	Vector of left leg joint rates
$\dot{\theta}_L$	Load angular velocity

$\ddot{\theta}_L$	Load angular acceleration
θ_M	Motor angle
$\dot{\theta}_M$	Motor angular velocity
$\ddot{\theta}_M$	Motor angular acceleration
$\dot{\Theta}_R$	Vector of right leg joint rates
τ_L	Load torque
τ_M	Motor torque
ω_L	Load angular velocity
ω_M	Motor angular velocity
ω_n	Natural frequency
$\omega_x, \omega_y, \omega_z$	Angular velocities
ζ	Damping ratio

CHAPTER 1

INTRODUCTION

The SPherically-Actuated platform Manipulator (SPAM) is presented in this thesis. This chapter provides background information on platform manipulators, a literature review of related research, and the thesis objectives. The remaining chapters present the SPAM system configuration, kinematics problems and solutions, control, experimentation and results, and conclusions.

1.1 Platform Manipulators

The research and development of parallel robots is well documented. The parallel structure provides this class of robots with several attributes that make them more desirable than their serial counterparts. These include higher load bearing, acceleration, stiffness, and accuracy as well as lower moving mass. The major drawback associated with parallel robots is the substantial loss of workspace compared to that of serial robots. Hunt (1983) was one of the first investigators into parallel robotics and his studies have led to a whole new field of interest.

One of the most common forms of parallel robot architecture is the platform manipulator, which consists of two or more serial chains that connect a fixed base to a moving platform. The most well-known and researched form of platform manipulator is the Gough/Stewart platform (Stewart, 1966). The basis of many variations of platform manipulators, this architecture uses six actuated prismatic legs to provide the platform

with six degrees of freedom. A misnomer, the “Stewart” platform (as it is often called) was actually never proposed by Stewart (1966); he actually suggested a platform manipulator with three legs and active revolute joints to be used as a flight simulator. By design, prismatic actuators are used almost exclusively as the means of motion in platform manipulators and are meant to bear axial loading only. The SPAM robot, however, uses actuated spherical joints that must bear moment loads. In the more well-known parallel robots, passive universal (U) and spherical (S) joints are used to allow the necessary freedoms. The SPAM architecture has passive revolute (R) and U joints. The variable-geometry truss (VGT) is another heavily researched parallel robot that also uses prismatic actuators (Rhodes and Mikulas, 1985; Padmanabhan et al., 1992, Williams and Hall, 2000). Uses for these manipulators include flight simulators, machining centers, entertainment, medical robotics, space structure modules, and haptics.

1.2 Literature Review

Presented here is a literature review of research related to the SPAM robot. Included in these related areas are parallel robots, platform manipulators, spherical actuators, kinematic analysis, and control methods.

As stated earlier, parallel robots have a rich history. Entire journal issues have been dedicated to the research and development of parallel manipulators (Nguyen, 1995). Several researchers have proposed novel architectures for platform manipulators that have unique characteristics. Dafaoui et al. (1998) presented a novel, six degree-of-freedom manipulator adapted to perform force feedback control. Zhiming and Philip

(1998) proposed a modular platform manipulator that can be reconfigured according to the task at hand. Pennock and Matson (1995) studied a class of one degree-of-freedom parallel manipulators formed by multiple serial robots grasping a common payload. Modifications of the “Stewart” platform have also been presented (Williams, 1992; Stoughton and Arai, 1993).

In addition to creating physical variations of the “Stewart” platform, much work has been dedicated to analyzing this platform manipulator. The kinematic analysis of this design has been exhaustively studied (Williams, 1992; Liu et al., 1993, Liu et al., 1994). With a complete kinematic analysis, researchers have also turned their attention towards the dynamics and control of the platform. Several methods have been used in dealing with the dynamics of the “Stewart” platform. Dasgupta and Mruthyunjaya (1998) used the Newton-Euler approach to model the dynamics of the platform, including gravity and viscous friction at the joints. Liu and Li (1999) used Kane’s method to formulate a symbolic expression of the dynamic equations of motion for the platform with full effects of leg inertia. Extensive work by Liu and associates (Lbret et al., 1993; Liu et al, 1993; Liu et al., 1991) further examines the dynamic modeling and control of variations of the “Stewart” platform.

Many types of spherical actuators have been built or proposed, most of which are intended to be used as robotic wrists. For example, the spherical actuator used for SPAM (although not truly spherical due to an offset between the two universal joints) was proposed originally as a wrist mechanism, (Rosheim, 1987). Truly spherical actuators have been researched using various types of motors. Roth and Lee (1995) proposed

variable reluctance spherical motors as actuators. Wang et al. (1998) researched the use of permanent magnets in an actuator to create motion. Ultrasonic vibrations have also been used for actuation (Toyama, 1996). In addition to Rosheim's ongoing work (1985, 1986, 1987, 1988), several other researchers have developed the capabilities of spherical actuators (Chaware and Amarnath, 1984; Stanisic and Duta, 1990; Lee and Kwan, 1991; Sasae et al., 1996; Leguay-Durand and Reboulet, 1997).

Also building on Rosheim's design, Williams (1990, 1999) explored the kinematic analysis of the double universal joint Omni-Wrist. This kinematic analysis is complex due to an offset between the two universal joints. Much of the literature dealing with kinematics of parallel robots is dedicated to the solution of the forward pose problem and singularity analysis (e.g. Daniali et al., 1995; Innocenti and Parenti-Castelli, 1993; Raghavan, 1993; and Wang and Gosselin, 1997). Tsai (1999) has recently published a book that provides an overview of the mechanics of both serial and parallel manipulators. Craig (1989) also published a text that is quite useful in kinematic analysis, although it deals largely with serial robots. The text also discusses control of the robots, using independent joint controllers.

With a proper kinematic analysis, simple position control can be developed using independent joint controllers. The dynamics of each joint must be known to develop the control algorithm. Williams (2000) and Dorf and Bishop (1998) present a similar method for developing a controller for a motor coupled to a robotic link. This controller, usually a Proportional, Integral, Derivative (PID) controller, operates on each joint independently of the others (Craig, 1989). In actuality, the motion of any joint does have an effect on

the other joints; this effect is negligible when dealing with large gear ratios (Williams, 2000).

1.3 Thesis Objectives

Presented here are the objectives of this thesis. The thesis focuses on five areas of the SPherically-Actuated platform Manipulator (SPAM): design, construction, and implementation of the hardware, kinematic analysis, and control of the robot.

At the outset of the research, Ohio University had two robotic wrists that had no associated application. These wrists (Rosheim Omni-Wrists) are on loan from NASA Langley Research Center and had been wired to work with the personal computers in the Ohio University Robotics/Haptics Lab. John Ottersbach (an Ohio University graduate student alumnus) configured the wrists and their associated electronics so that position control of the wrists would be possible. With the wrists in operation, Dr. Robert Williams II and the author decided that the wrists could be used to actuate a platform manipulator (as shown in Figure 1), an idea that had not previously been presented.

Figure 1. SPherically-Actuated platform Manipulator (SPAM) Concept

The hardware objectives were to design the links, base, and platform so that they would fit the scale of the Omni-Wrists. The robot also had to be lightweight for its size so that the moment loading on the wrists was minimal. (The designs appear in Appendix A in the form of Solid Edge CAD drawings). With a proper design, the robot would then be constructed and connected to a personal computer for control. Also, NASA supplied the university with four amplifiers. For full Cartesian control, six amplifiers would be necessary to drive the six motors. With this setup, either the hardware would not be fully controlled in all six degrees of freedom or two more amplifiers and a power supply would have to be purchased. A combination of budget and time restrictions led to the former.

The largest part of the study would be invested in the kinematics of the manipulator. The Omni-Wrists create a complication in the kinematic analysis because they are not truly spherical wrists. Although Dr. Williams had already done the kinematic analysis of this wrist, incorporating it at the base of a serial chain would require further study. Most previous work only examines serial chains with a wrist at the end of the chain. The objectives for the kinematics were to develop this placement of the wrist at the base of the serial chain and find a reliable method for solution. The Newton-Raphson numerical procedure will be presented as the method used to solve the kinematics problems. Singularities, both physical and algorithmic, would also have to be examined and avoided where possible.

The computer control of the robot's position would be fairly simple due to previous work that the author had completed for graduate courses. The objectives for hardware control were to develop reliable programs to first control the position of a

single joint of the Omni-Wrist and then control a plunge motion of the SPAM robot.

Using Matlab and Simulink, the program to control one joint of the Omni-Wrist had been developed. This program was expanded to control any of the joints of the wrist. The program to control the plunge motion of the platform would have to be first tested in simulation and then linked to the hardware via MultiQ-3 input/output boards and Wincon software, both products of Quanser Consulting (1999). These programs, along with the theory behind the control scheme, will be presented in detail.

The following chapters present the work that Dr. Williams and the author have completed in designing, analyzing and controlling the SPAM robot. Chapter 2 focuses on the hardware and software of the robot system. Chapter 3 presents the forward and inverse pose kinematics as well as rate kinematics for two versions of SPAM. The programs used to control the manipulator and the theory behind them are presented in Chapter 4. Chapter 5 presents the experimentation and results of the kinematic analysis as well as simulated and real-world control. Conclusions and future work are discussed in Chapter 6. Four appendices present the CAD drawings, operation procedure, kinematics terms, and analytical methods for solution of the kinematics of one version of the manipulator.

CHAPTER 2

SPAM SYSTEM CONFIGURATION

This chapter introduces the configuration of the SPherically-Actuated platform Manipulator (SPAM), discussing the details of each component used in the operation of the robot system. This system may be viewed as two interdependent subsystems: the hardware and the software. Presented in the hardware subsystem is the physical platform manipulator, including its spherical actuators and their associated electronics. The software subsystem consists of the computer programs needed to operate and control the platform manipulator. Figure 2 shows the high-level complete system setup.

Figure 2. Overall SPAM System

2.1 System Hardware

There are four main components of the SPAM hardware: the robot itself, the spherical actuators, the electronics cabinet and the input/output boards. Each of the next four sections is dedicated to describing one of these components, discussing the parameters of each along with any problems or drawbacks associated with the component.

2.1.1 SPAM Design and Construction

The SPAM robot is a novel, six degree-of-freedom (dof), parallel platform manipulator. The platform manipulator consists of a fixed base and a moving platform connected by two identical *SRU* (spherical-revolute-universal joint) serial chain legs, as shown in Figure 3. The novelty in this design lies in the spherical actuation of the robot, which, to the author's knowledge and literature review, has not been previously proposed. This section will present a look at the structure of the SPAM robot, including its mobility and its limits.

Figure 3. Solid Model of the SPAM Robot

The Rosheim Omni-Wrists, on loan to Ohio University from the NASA Langley Research Center, act as the active spherical joints, although they are not purely spherical. With both wrists and one base for the wrist supplied, the author designed the remaining links, platform and base to fit the scale of the Omni-Wrists. The CAD drawings for this design are displayed in Appendix A. The entire robot (Figure 4), other than the wrists, is made of 6061 aluminum. The aluminum provides for a stiff, lightweight structure that

helps both in the accuracy of the platform positioning as well as the reduction of moment loading on the wrists. In the more well-known platform manipulator designs, the actuators are almost exclusively prismatic. This is done by design to eliminate the moment loading from which the SPAM robot suffers. However, the Omni-Wrists are specifically designed to handle this moment loading.

Figure 4. The SPAM Robot

The links were designed so that there is 14" between pivots. The platform is twice that length, at 28" between pivots, and the base is three times it, at 42" between the centers of the Omni-Wrists. The passive joints on the robot each consist of a bronze bearing with a 3/8" aluminum pin to provide the required pivot. These joints have also been lubricated to promote the smooth, unrestricted motion desired from a passive joint in any robot. This lubrication also prevents the aluminum links from wearing when rubbing against one another. Looking at the left leg kinematic diagram (Figure 5), the description of the joints and their limits are shown in Table 1. The right leg is identical to

the left (not a mirror image, but identical) with different variables representing the joints. Note that the limits on the pitch and yaw are 45° . This corresponds to a 90° limit on the overall pitch and yaw angles due to a doubling effect that the Omni-Wrists produce. The manipulator's mobility is calculated using the Kutzbach equation:

$$M = 6(N-1) - 5J_1 - 4J_2 - 3J_3$$

$$M = 6(6-1) - 5(2) - 4(2) - 3(2) = 6 \quad (1)$$

These six degrees-of-freedom can be provided by the two Omni-Wrists to produce general platform Cartesian poses (positions and orientations).

Table 1. Joint Limits for Left SRU Chain

Angle	Description	Angle Limits
θ_1	Omni-Wrist roll	$\pm$ Unlimited
θ_2	Omni-Wrist pitch	$\pm 45^\circ$
θ_3	Omni-Wrist yaw	$\pm 45^\circ$
ϕ_1	Elbow	$\pm 143^\circ$
ϕ_2	U-joint pitch	$+218, -38^\circ$
ϕ_3	U-joint roll	$\pm$ Unlimited

Figure 5. Diagram of Left SRU Chain

2.1.2 Omni-Wrist

The Omni-Wrist, shown in Figure 6, is a double universal joint wrist created by Ross-Hime Designs, Inc. (www.anthrobot.com) for both commercial and research applications. The wrist was initially developed for use with Space Station robots and is currently being used in industrial spray painting operations. This section presents the

Omni-Wrist as the spherical actuators for the SPAM robot and discusses the details of the wrist, including its drive train and sensors.

Figure 5 illustrates the nature of the double universal joint wrist. The universal joints represent the pitch and yaw joints of the Omni-Wrist. As shown in Figure 5, there is an offset between these two joints, which greatly complicates the kinematics of the robot. The mechanical coupling between the two universal joints produces a doubling of the input command received by the wrist for both the pitch and yaw angles, as mentioned earlier. For example, an input angle of 30° pitch results in an output of 60° . This is not a problem, but must be accounted for to receive proper position control. The roll joint for the Omni-Wrist is continuous and bi-directional. The three joints are kinematically independent of one another.

Figure 6. The Omni-Wrist

Ross-Hime has focused on low weight and high stiffness, as well as high speed reduction and high torque capabilities, in the design of the Omni-Wrist. All of these

attributes make the Omni-Wrist a good spherical actuator for a platform manipulator. The speed reduction and high load bearing are especially desirable for producing smooth, controlled motion. Both of these properties are directly related to the gearing involved in the wrist, which consists of a combination of stainless steel spur gears, helical gears, and bevel gears. The overall gear ratios are as shown in Table 2 (Rosheim, 1990).

Table 2. Gear Ratios for the Omni-Wrist

Joint	Gear Ratio
Pitch	259.2 : 1
Yaw	220 : 1
Roll	175.5 : 1

Each joint is driven by a brushless DC servomotor manufactured by Pacific-Scientific. The motors have a continuous stall torque of 4.8 in-lb and are thermally protected. These motors, in conjunction with the gearing, allow the Omni-Wrist to handle significant moment loading, as required by the SPAM robot. Each motor is equipped with a pancake resolver that mounts directly on the motor shaft. The resolver provides position feedback as well as commutation for the motors. Because resolver signals are difficult to deal with, the outputs are sent to an encoder emulator card mounted on the motor's amplifier. The amplifiers are mounted in the electronics cabinet.

2.1.3 Electronics Cabinet

The electronics cabinet, shown partially in Figure 7, houses the motor amplifiers and necessary wiring to connect the PC to the motors. The wiring, as with most of the

electronics for the Omni-Wrist, was set up by John Ottersbach, an Ohio University graduate student alumnus. The focus of Figure 7 is on the four amplifiers mounted inside the electronics cabinet. Like the motors, these amplifiers are a Pacific-Scientific product. They require a 115-volt AC supply and can output up to 1000 Watts. Each amplifier is wired to provide multiple types of protection, including thermal overload, bus overvoltage, and short-circuiting. For use with DC servomotors, the amplifiers are set up to run in current mode, which means that the amplifiers send a current to the motor that is proportional to the input voltage. This input voltage comes from the computer and ranges from -5 to +5 volts. Prior to operation, each motor must be calibrated to its respective amplifier to ensure proper motion (IMEC Corporation, 1987). Required in this procedure is the adjustment of the potentiometers located on the front of the amplifiers. These can be seen in Figure 7 directly below the LEDs near the top of the amplifiers.

Figure 7. Inside the Electronics Cabinet

The amplifiers each house an encoder emulator card that receives signals from the motor's resolver and converts them to encoder-like signals that can be recognized by the

computer. These signals are a marker pulse and two channels that are 90° out of phase, Channel A and Channel B. The marker pulse sends a signal once for every resolver revolution, but is not currently in use. Setting up the other two output channels 90° out of phase allows the computer to read the feedback at four times the resolution of the resolver, a feature known as quadrature. The pancake resolvers have a resolution of 1000 pulses per revolution (ppr). Therefore, the computer reads it at 4000 ppr by using quadrature. When reading this feedback from the computer, it is necessary to divide it by 4 and perform the transformations from encoder counts to joint angle, which includes the gear ratio and the pulses per revolution rating as follows:

$$\theta = \text{counts read} \left(\frac{1 \text{ pulse}}{4 \text{ counts}} \right) \left(\frac{1 \text{ revolution}}{1000 \text{ pulses}} \right) \left(\frac{2\pi \text{ radians}}{1 \text{ revolution}} \right) \quad (2)$$

Currently only two amplifiers are in operation. NASA supplied Ohio University with four amplifiers, but since then, two of the encoder emulator boards have stopped functioning properly. This can be seen in Figure 7, in which the two amplifiers on the right show a fault LED lit up. This LED lights whenever there is some type of problem with the wiring of the amplifier. In this case, the motors are not connected, producing the fault. There are plans to purchase the required components for full control of both wrists.

Figure 8 shows the control panel for the electronics cabinet. The four switches to control the power supply to the amplifiers are located on top of the cabinet. The “Power On” switch turns power on to the system, which is stopped by the “Off” switch. The “On-Off” switch enables and disables the amplifiers. The “E-Stop” switch can be used to stop all motor movement in emergency situations (Rosheim, 1990). There is a fifth

switch located at the top right of the cabinet door that provides power to the cabinet. In addition, John Ottersbach placed a safety switch under the top lid of the cabinet that requires that the proper wires be jumpered to enable power to the cabinet. Finally, there is a switch located beside the PC labeled Omni-Wrist that allows outputs to be sent to the wrist.

Figure 8. Control Panel for Electronics Cabinet

2.1.4 MultiQ-3 I/O Boards

The MultiQ-3 boards, produced by Quanser Consulting (1999), are data acquisition and control boards used to connect the robot to the computer. There are two boards in the system, as shown in Figure 9. The larger board fits into an ISA slot in a personal computer; it is connected to the smaller board (the terminal board) via the two flat ribbon cables. The final component in this system is the software necessary to read the inputs and generate the outputs.

Figure 9. The MultiQ-3 Boards

The terminal board houses eight analog inputs, eight analog outputs, sixteen bits of digital input, sixteen bits of digital output, three programmable timers and up to eight encoder inputs decoded in quadrature (Quanser Consulting, 1999). The boards have a few limitations that must be considered when operating them. The analog connections, both input and output, are restricted to a range of ± 5 volts. The encoder decoders receive the Channel A and Channel B signals as discussed in the previous section and increment a 24 bit counter to provide a maximum of 16,777,215 counts (Quanser Consulting, 1999). With the resolvers rated at 1000 ppr, it is possible to measure up to ± 2097 revolutions without overflowing the counters, which can be an issue when measuring the continuous Omni-Wrist roll.

This section has presented all of the hardware components of the SPAM robot. Included in this discussion was a presentation of the Omni-Wrist and its associated electronics, which consist of the motors, resolvers, and amplifiers equipped with encoder emulator cards. Also examined were the overall platform manipulator and the MultiQ-3 I/O boards. Features, problems, and limitations of each component have been cited where relevant.

2.2 System Software

There are two separate software packages required to operate the SPAM robot, the Matlab/Simulink software and the WinCon software. Here Matlab and Simulink are considered as one because Simulink is a Matlab add-on. The following two sections will present these programs and examine their properties as they relate to SPAM real-time operation and control.

2.2.1 Matlab/Simulink

Presented in this section is a discussion of the Matlab/Simulink software package created by MathWorks, Inc. This software is a general programming package used to generate code that controls the movement of the SPAM robot. Simulink is a dependent add-on program to the Matlab package; therefore some background into Matlab must also be discussed. In this work, the software versions are Matlab 5.2 and Simulink 2.2.

Matlab is a mathematically-focused computing environment that employs a high-level programming language to perform numeric and symbolic computation and produce visualizations. Matlab includes hundreds of functions to make programming simpler and more productive. The software is used in a variety of application areas including control system design, financial engineering, and medical research (MathWorks, 2000).

Simulink is a graphical interface to Matlab. It uses block diagrams to model and simulate dynamic systems. This simulation is a very effective way to evaluate design concepts without first having to make a hardware prototype. Figure 10 shows a sample program used for simulating the position control of the pitch joint of the Omni-Wrist.

This figure demonstrates the nature of the Simulink environment. Provided in the Simulink package is a block library that has various types of blocks, allowing the user to create a Matlab program using these blocks rather than lines of code. Shown in Figure 10 are blocks for inputs and outputs, conversions, gains (multiplication and division), summing junctions (addition and subtraction), integration, and controllers. All of these are predefined blocks that the user can drag and drop into the diagram and then make the necessary modifications. Numerous other blocks are included in the libraries that allow for the graphical programming of most anything that can be programmed in Matlab code.

Figure 10. Simulink Diagram for Simulation of Position Control

This discussion of the simulation of position control is followed by with one on the actual hardware control, in which Simulink is still the programming environment. However, some revisions to the program of Figure 10 must be made to produce a connection to the hardware. This connection comes in the form of the Wincon software.

2.2.2 WinCon

WinCon is a real-time application produced by Quanser Consulting that is used to run hardware via Simulink-generated code. Version 3.0.2c is used for all programming

involved with the SPAM robot and runs in the Windows 95/98 operating system (WinCon, 1999). In order for WinCon to operate, Matlab must also have the Real-Time Workshop (RTW) add-on installed.

Figure 11 shows the same position control program as Figure 10 with the simulated joint and the simulated resolver feedback replaced by WinCon and the hardware. The two Quanser Consulting blocks in this figure, which are part of the WinCon library, achieve this replacement. The block in the upper right that is labeled “Analog Output1” creates a voltage output proportional to the desired input angle that is calculated continuously by the PID controller. This voltage is sent to the internal MultiQ-3 board and then follows the routine as illustrated in Figure 2. The block on the lower left represents the encoder feedback signal. This signal can also be followed in Figure 2 and returns to the “Software” block via this encoder feedback block. The closed-loop control diagram of Figure 11 appears to be broken; the real-world hardware and sensor close the loop.

Figure 11. Simulink Diagram for Position Control of Hardware

In order for Simulink to use the WinCon blocks, the diagram must be built using the “build” command under the WinCon drop-down menu in the Simulink environment. When the build command is given, WinCon uses the Matlab RTW to create C-code and link the Simulink diagram to the MultiQ-3 boards through various other files. With this done, a WinCon server window appears that allows the user to execute the Simulink program, incorporating the hardware in the control. While the program is running, the input parameters may be changed for analysis and verification of the controlled hardware.

Currently, there are some limitations to using this software for control. There are no less than four blocks that the RTW (and therefore WinCon) cannot handle. These blocks are the s-function, Matlab function, write to file, and read to file (Nyzen, 1999). The first two are important in controlling complex robots, such as SPAM. This is because the s-function provides a way to program the forward and inverse kinematics each in one block. Without this capability, a combination of many function blocks must be networked to provide the same output that one s-function block could produce. This larger network of blocks requires longer processing time and holds greater potential for error.

The details of the software used in controlling the position of the SPAM robot have been presented in this section. The Matlab/Simulink package provides a simple programming environment that allows for system analysis through simulation. Once this simulation is satisfactory, the WinCon software connects the hardware-ready Simulink program to the MultiQ-3 I/O boards, which communicate with the motors and resolvers for hardware control.

CHAPTER 3

KINEMATICS

This chapter presents the kinematic problems and their methods of solution for the SPAM robot. The kinematic study is done to relate Cartesian poses to active joint variables, which allows for platform simulation and control. The kinematic analysis has been partitioned in to five sections for the *SRU* version of SPAM: the Denavit-Hartenberg (D-H) parameters, the forward pose kinematics, the inverse pose kinematics, the inverse rate kinematics, and the singularities. The chapter introduces a variation on the previously discussed SPAM robot. This variation replaces the passive revolute elbow joint with a passive prismatic joint. Dr. Robert Williams II has done the majority of the work in this area and has presented it in both a conference paper and a journal paper (Williams and Poling, 2000). The section dedicated to the *SPU* (spherical-prismatic-universal) version of SPAM will summarize his results. The methods discussed in the *SRU* kinematics sections are the same as the ones used for the *SPU* version, therefore only a summary of the *SPU* kinematics is presented along with the D-H parameters and singularities.

3.1 *SRU* SPAM Version

The *SRU* SPAM, shown in Figure 12, has been the subject of the previous discussions. This version differs from the *SPU* version only in that it has a passive

revolute joint at the elbow. This revolute joint presents singularity problems that the *SPU* version eliminates.

Figure 12. SRU version of SPAM

3.1.1 D-H Parameters

The Denavit-Hartenberg (D-H) parameters are used to define the geometry of the robot. Each line of the parameters describes the position and orientation of that coordinate frame with respect to the previous one. There are differing conventions in defining these parameters; this study uses the Craig (1989) convention. The full kinematic diagram for the left leg is shown in Figure 13.

L_1 is the distance from the fixed base frame $\{B\}$ origin to the origin of the first U-joint of the left Omni-Wrist. L_2 represents the offset in the wrist, as discussed in the hardware section. L_3 is the distance from the second universal joint of the Omni-Wrist to the revolute elbow joint. L_4 is the distance from the elbow to the passive U-joint at the platform. L_5 is the distance from that U-joint to the origin of the moving platform frame

Table 3. Left *SRU* Leg D-H Parameters

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	$\theta_1 + 90$
2	90	0	0	$\theta_2 + 90$
3	90	0	0	θ_3
4	0	L_2	0	θ_3
5	-90	0	0	θ_2
6	0	L_3	0	ϕ_1
7	0	L_4	0	$\phi_2 + 90$
8	90	0	0	ϕ_3

Lengths L_1 and L_5 are not present in the D-H parameters. These two lengths connect the *SRU* chain to the base and the platform, respectively. They are later added to the kinematics via homogenous transformation matrices. For full kinematic analysis, the right leg must also be considered. Since the right leg is identical to the left, the following variable substitutions are needed to produce the D-H parameters for the right leg:

$$\begin{aligned}
 \theta_4 &\rightarrow \theta_1 & \phi_4 &\rightarrow \phi_1 \\
 \theta_5 &\rightarrow \theta_2 & \phi_5 &\rightarrow \phi_2 \\
 \theta_6 &\rightarrow \theta_3 & \phi_6 &\rightarrow \phi_3
 \end{aligned}$$

This study uses a combination of serial chain and parallel chain methods to solve the pose kinematics problems. The objective is to relate the six Omni-Wrist joint angles $\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\}$ to the Cartesian pose variables $\{x, y, z, \alpha, \beta, \gamma\}$ of the moving platform frame $\{P\}$ with respect to the fixed base frame $\{B\}$. (Note that α here is not the α of the D-H parameters.) The passive joint angles $\{\phi_1, \phi_2, \phi_3, \phi_4, \phi_5, \phi_6\}$ must also be included in the analysis. Z-Y-X Euler angles α, β , and γ (Craig, 1989) are used to describe the orientation of the platform. Homogeneous transformation matrices are used as shown in (3).

$$\begin{aligned}
{}^B_P T &= \begin{bmatrix} \begin{bmatrix} {}^B_P R \end{bmatrix} & \begin{Bmatrix} {}^B_P P \end{Bmatrix} \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
{}^B_P T &= \begin{bmatrix} c\alpha c\beta & -s\alpha c\gamma + c\alpha s\beta s\gamma & s\alpha s\gamma + c\alpha s\beta c\gamma & x \\ s\alpha c\beta & c\alpha c\gamma + s\alpha s\beta s\gamma & -c\alpha s\gamma + s\alpha s\beta c\gamma & y \\ -s\beta & c\beta s\gamma & c\beta c\gamma & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)
\end{aligned}$$

Using this information we can solve the forward and inverse pose kinematics problems, which are a precursor to the rate kinematics problems. The methods used to obtain these solutions are the focus of the next three sections.

3.1.2 Forward Pose Kinematics

This section presents the solution of the SPAM forward pose kinematics problem, which is stated: Given the Omni-Wrist angles $\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\}$, calculate the resulting Cartesian platform position and orientation $\{x, y, z, \alpha, \beta, \gamma\}$, which is equivalent to calculating ${}^B_P T$. From this statement, it is shown that there are 6 givens and 6 unknowns. However, the passive joints for the left and right leg, $\{\phi_1, \phi_2, \phi_3\}$ and $\{\phi_4, \phi_5, \phi_6\}$ respectively, are also unknown. Due to these additional unknowns, this problem cannot be solved by treating the left and right legs separately and using standard serial robotics techniques.

The forward pose problem is a coupled problem because the two legs share the same Cartesian unknowns (Williams and Poling, 2000). To obtain the forward pose solution, the Newton-Raphson numerical method is used. The numerical method requires as many functions as unknowns; hence, with the six unknowns for each leg listed above,

the method requires six functions. To form these functions, compare the elements of the numerical B_pT to that of the symbolic B_pT as follows:

$${}^B_pT_{NUM} = {}^B_pT(\theta_1, \theta_2, \theta_3, \phi_1, \phi_2, \phi_3) \quad (4)$$

The right-hand-side of (4) is the symbolic kinematics expressions for the left leg only, formed using the procedure outlined in Craig (1989). The procedure transforms the D-H parameters presented in Table 3 into neighboring homogeneous transformation matrices as:

$${}^{i-1}_iT = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

The neighboring homogeneous transformation matrices are multiplied together to form 0_8T . To include the base and platform frames, the following matrix multiplication must be used:

$${}^B_pT = {}^B_0T {}^0_8T {}^8_pT \quad (6)$$

where, from Figure 13:

$${}^B_0T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -L_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^8_pT = \begin{bmatrix} 0 & -1 & 0 & -L_5 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

Equation (6) now includes the base and platform lengths in the analysis. Still looking at only the left leg, there are six unknowns as shown in (4). Therefore, the Newton-Raphson method requires six functions. Three of these functions come from the

translational part of (6), and three from the rotational part. Following the form in (3), the translational terms are:

$$\begin{Bmatrix} X_1 \\ Y_1 \\ Z_1 \end{Bmatrix} = {}^B P_P(\theta_1, \theta_2, \theta_3, \phi_1, \phi_2, \phi_3) \quad (8)$$

From the rotation part of (6), there are nine possible functions to choose from. However, only three may be selected due to the orthonormal constraints present in rotation matrices. That is, each row or column has entries that, when squared and added, equal one and the rows or columns representing the x , y , and z values are orthogonal to one another. Therefore, three elements from any one row or column may not be selected. The (1,2), (2,2), and (3,3) rotation terms have been selected for this analysis. The numerical counterparts for these six functions come from evaluating (3) using an initial guess required by the Newton-Raphson procedure. The comparison of the symbolic expressions to the numeric entries yields the following six functions:

$$\begin{aligned} F_1 : X_1 - x &= 0 \\ F_2 : Y_1 - y &= 0 \\ F_3 : Z_1 - z &= 0 \\ F_4 : r_{12}^{(1)} + s\alpha c\gamma - c\alpha s\beta s\gamma &= 0 \\ F_5 : r_{22}^{(1)} - c\alpha c\gamma - s\alpha s\beta s\gamma &= 0 \\ F_6 : r_{33}^{(1)} - c\beta c\gamma &= 0 \end{aligned} \quad (9)$$

The functions for the right side follow almost identically, replacing the symbolic expressions as follows:

$$\begin{aligned}
F_7 : X_2 - x &= 0 \\
F_8 : Y_2 - y &= 0 \\
F_9 : Z_2 - z &= 0 \\
F_{10} : r_{12}^{(2)} + s\alpha c\gamma - c\alpha s\beta s\gamma &= 0 \\
F_{11} : r_{22}^{(2)} - c\alpha c\gamma - s\alpha s\beta s\gamma &= 0 \\
F_{12} : r_{33}^{(2)} - c\beta c\gamma &= 0
\end{aligned} \tag{10}$$

where B_pT has been formed as in (6), only the signs of L_1 and L_5 must be changed to reflect the proper translations in the right leg B_0T and 8_pT :

$${}^B_0T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & L_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^8_pT = \begin{bmatrix} 0 & -1 & 0 & L_5 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

These twelve functions are complex symbolic formulas that are facilitated by the use of Matlab for derivation and manipulation. The translational subscript 1 and rotational superscript (1) in Equations (9) indicate the left *SRU* chain; the right chain is represented by subscripts and superscripts of 2, as in (10).

Allowing $X = \{x \ y \ z \ \phi_1 \ \phi_2 \ \phi_3 \ \alpha \ \beta \ \gamma \ \phi_4 \ \phi_5 \ \phi_6\}^T$, the Newton-Raphson iterative method for solving (9) and (10) together, as it must be due to the coupling, is now presented as in Williams and Poling (2000). Starting from the initial guess for the solution, X_0 , the Newton-Raphson process for step $k+1$ is as follows:

- Solve $J_{NR}\delta X_k = -F(X_k)$ for δX_k : $\delta X_k = -J_{NR}^{-1}F(X_k)$
- Then $X_{k+1} = X_k + \delta X_k$
- Iterate until $\|\delta X_k\| < \varepsilon$

where $F(X) = \{F_i(X)\}$, $i = 1, 2, \dots, 12$ is given in (9) and (10) and ϵ is a user-defined tolerance for each unknown. The Newton-Raphson Jacobian matrix is:

$$J_{NR} = \begin{bmatrix} \frac{\partial F_i}{\partial X_j} \end{bmatrix} \quad j = 1, 2, \dots, 12 \quad (12)$$

$$J_{NR} = \begin{bmatrix} -1 & 0 & 0 & J_{1,4} & J_{1,5} & J_{1,6} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & J_{2,4} & J_{2,5} & J_{2,6} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & J_{3,4} & J_{3,5} & J_{3,6} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & J_{4,4} & J_{4,5} & J_{4,6} & J_{4,7} & J_{4,8} & J_{4,9} & 0 & 0 & 0 \\ 0 & 0 & 0 & J_{5,4} & J_{5,5} & J_{5,6} & J_{5,7} & J_{5,8} & J_{5,9} & 0 & 0 & 0 \\ 0 & 0 & 0 & J_{6,4} & J_{6,5} & 0 & 0 & J_{6,8} & J_{6,9} & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & J_{7,10} & J_{7,11} & J_{7,12} \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & J_{8,10} & J_{8,11} & J_{8,12} \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & J_{9,10} & J_{9,11} & J_{9,12} \\ 0 & 0 & 0 & 0 & 0 & 0 & J_{10,7} & J_{10,8} & J_{10,9} & J_{10,10} & J_{10,11} & J_{10,12} \\ 0 & 0 & 0 & 0 & 0 & 0 & J_{11,7} & J_{11,8} & J_{11,9} & J_{11,10} & J_{11,11} & J_{11,12} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & J_{12,8} & J_{12,9} & J_{12,10} & J_{12,11} & 0 \end{bmatrix}$$

This twelve-by-twelve matrix is fairly simple to form with the aid of Matlab. 88 of the 144 terms are zero and six are negative one (partial derivatives of functions 1,2,3,7,8, and 9 with respect to x , y , and z). Also, the partial derivative of functions 4, 5, and 6 with respect to α , β , and γ are the same as those for functions 10, 11, and 12 (Williams and Poling, 2000).

Using this method, the forward pose kinematics problem for SPAM is solved.

Due to the complexities involved with the SPAM robot, the numerical method is the best choice for solution. The drawbacks to the method are that it requires a good initial guess and produces only one of the many possible solutions. However, in real-world control, these are not issues because the previous control cycle configuration provides a good

initial guess and the solution that nearest fits the present configuration is generally the only one that is needed.

3.1.3 Inverse Pose Kinematics

The inverse pose kinematics problem is stated: Given the desired platform pose $\{x, y, z, \alpha, \beta, \gamma\}$ (or, equivalently, B_pT), calculate the required Omni-Wrist actuator joint variables $\{\theta_1, \theta_2, \theta_3\}$ and $\{\theta_4, \theta_5, \theta_6\}$. The left- and right-leg passive joint variables $\{\phi_1, \phi_2, \phi_3\}$ and $\{\phi_4, \phi_5, \phi_6\}$ are again unknowns. The passive variables are not required for real-time pose control, but they may be used in on-line rate and dynamics equations, plus off-line for simulation.

The inverse pose problem can be solved by considering the left and right *SRU* serial chain legs separately, unlike the forward pose problem. The given Cartesian pose B_pT must be reached by both legs independently, each with six degrees of freedom provided by the three active and three passive joints. Thus, the problem decouples between the left and right legs. Due to the complexity involved with the *SRU* serial chain using the offset Omni-Wrist actuator, analytical solutions may not exist. Because of this, the Newton-Raphson numerical method will be used to solve the inverse kinematics problem. Another possible method for solution is the combined analytical/iterative approach presented in Williams (1999).

The Newton-Raphson method follows the same procedure as the forward problem, but the angles for the left leg $\{\theta_1, \theta_2, \theta_3, \phi_1, \phi_2, \phi_3\}$ and right leg $\{\theta_4, \theta_5, \theta_6, \phi_4, \phi_5, \phi_6\}$ are now the unknowns. The twelve functions are the same as the ones used in the

forward problem, but only six are used at a time; F_1 through F_6 are used to solve the left leg and F_7 through F_{12} are used for the right leg. Allowing Θ to represent either the left or right unknown vectors, the Newton-Raphson procedure is repeated as it was in the forward pose problem. Starting from an initial guess for the solution, Θ_0 , the Newton-Raphson process for step $k+1$ of the left leg is summarized below:

- Solve $J_{NR} \delta\Theta_k = -F(\Theta_k)$ for $\delta\Theta_k$: $\delta\Theta_k = -J_{NR}^{-1} F(\Theta_k)$
- Then $\Theta_{k+1} = \Theta_k + \delta\Theta_k$
- Iterate until $\|\delta\Theta_k\| < \varepsilon$

where the inverse kinematics Newton-Raphson Jacobian is formed for each leg as

$$J_{NR} = \begin{bmatrix} \frac{\partial F_i}{\partial \Theta_j} \end{bmatrix} \quad (13)$$

Due to the specific *SRU* leg kinematics, the six-by-six Jacobian matrix has zero elements in the (3,1), (6,1), and (6,6) terms.

With the use of the Newton-Raphson numerical procedure, the inverse pose kinematics problem for SPAM is solved as two separate serial chains. As was the case in the forward problem, only one of the multiple solutions is found and a good initial guess is critical. Again, these are not issues when evaluating the usefulness of the Newton-Raphson procedure in real-time control because the initial guess is provided via the previous control cycle and multiple solutions are not desired.

3.1.4 Inverse Rate Kinematics

Rate kinematics consists of relating the active joint rates to the resulting platform Cartesian rates. The pose of the manipulator must be known to form this relationship.

The inverse velocity solution, which is the basis of the resolved-rate control scheme, is

stated: Given the commanded platform Cartesian rates $\dot{X} = \{\dot{x} \quad \dot{y} \quad \dot{z} \quad \omega_x \quad \omega_y \quad \omega_z\}^T$,

calculate the associated active joint rates $\dot{\Theta} = \{\dot{\theta}_1 \quad \dot{\theta}_2 \quad \dot{\theta}_3 \quad \dot{\theta}_4 \quad \dot{\theta}_5 \quad \dot{\theta}_6\}^T$. The

Cartesian rate $\dot{X}$ gives the translational and rotational rates of the platform coordinate

frame $\{P\}$ with respect to the base frame $\{B\}$, expressed in the $\{B\}$ frame. As in the

inverse pose solution, the inverse rate solution decouples between the left and right legs.

Using standard serial robotics methods, two Jacobian matrices may be formed:

$$\dot{X} = J_L \dot{\Theta}_L \quad \dot{X} = J_R \dot{\Theta}_R \quad (14)$$

where $\dot{\Theta}_L = \{\dot{\theta}_1 \quad \dot{\theta}_2 \quad \dot{\theta}_3 \quad \dot{\phi}_1 \quad \dot{\phi}_2 \quad \dot{\phi}_3\}^T$ and $\dot{\Theta}_R = \{\dot{\theta}_4 \quad \dot{\theta}_5 \quad \dot{\theta}_6 \quad \dot{\phi}_4 \quad \dot{\phi}_5 \quad \dot{\phi}_6\}^T$.

$\dot{X}$, defined above, is the platform Cartesian velocity that must be achieved

simultaneously by both serial *SRU* chains. J_L is the left-leg Jacobian matrix mapping the

left leg joint rates (active and passive) to the commanded Cartesian rates and J_R is the

right-leg Jacobian matrix mapping the right leg joint rates (active and passive) to the

same Cartesian rates. These Jacobians are formed using the column-based method. In

this method the i^{th} column of the Jacobian is calculated as:

$$\{J_i\} = \begin{Bmatrix} {}^0_i R \left({}^i \hat{z}_i x^i P_P \right) \\ {}^0_i R {}^i \hat{z}_i \end{Bmatrix} \quad (15)$$

where 0_iR is the rotation matrix of the i^{th} frame with respect to the $\{0\}$ frame, ${}^i\hat{z}_i$ is the unit vector in the direction of the z-axis for the i^{th} frame, expressed in the i^{th} frame coordinates ($= \{0 \ 0 \ 1\}^T$), and iP_p is the position vector of the $\{P\}$ frame with respect to the i^{th} frame, also expressed in the i^{th} frame coordinates. Each Jacobian is six-by-six. However, there are 8 coordinate frames. The six columns for the left leg Jacobian come from combining the two columns for $\dot{\theta}_2$ as one and the two columns for $\dot{\theta}_3$. These columns are combined because of the doubling effect that the Omni-Wrist's gearing produces. The right leg follows in the same manner, combining the columns of $\dot{\theta}_5$ and $\dot{\theta}_6$. The decoupled inverse rate solutions are:

$$\dot{\Theta}_L = J_L^{-1} \dot{X} \quad \dot{\Theta}_R = J_R^{-1} \dot{X} \quad (16)$$

Equations (16) are sufficient for use in real-time resolved-rate control. This is a semi-analytical solution since the Jacobian matrices are available symbolically, but numerical inversion is used for (16). This is done for simplicity, as symbolic inversion results in long, complex terms.

Using a fully symbolic approach, the inverse rate problem may be solved in one step. This approach inverts the Jacobian matrices symbolically off-line. This is done to save time in the real-world control program. Equations (16) map the common Cartesian velocities to the joint rates; only the active joint rates are the ones of interest. Therefore the first three rows of each symbolic inverse Jacobian matrix are selected, mapping the Cartesian velocities only to these joint rates. Denote the top three rows of the symbolic

inverse Jacobian matrices as J_{LA}^{-1} and J_{RA}^{-1} , respectively (A indicates active joints only).

The one-step inverse rate solution is then:

$$\dot{\Theta} = M\dot{X} \quad (17)$$

where: $M = \begin{bmatrix} J_{LA}^{-1} \\ J_{RA}^{-1} \end{bmatrix}$.

Even though the symbolic Jacobian matrices for the left and right legs of SPAM are relatively simple (few terms), this approach yields very complex results when using a symbolic computer program. Therefore, the partial-analytical method described above is used with (16) for solving the inverse rate kinematics problem.

3.1.5 Singularities

As mentioned in the previous section, the *SRU* version of SPAM suffers from singularities due to the passive revolute elbow joint. At any configuration of the platform manipulator in which the platform lies in the same vertical plane as the base and has zero β or γ rotation, there is an unwanted, uncontrollable extra freedom. This can be seen by examining Figure 14. In this particular type of pose, the passive *RUUR* chain in the center of the robot acts as a four-bar mechanism. This means that, in this position, one could rock the platform from side to side, despite the fact that the six joint inputs are locked.

The numerical methods used to solve the inverse kinematics are capable of working through these singularities, but require more iterations to drive the robot away from singular conditions. However, if the Newton-Raphson method receives an input

that has zero α , β , and γ values, it will not converge. When the Newton-Raphson method approaches the area of a singularity in a trajectory generation algorithm, it requires more iterations to converge on the correct answer. This convergence could result in jumping from one solution to another valid, but unwanted branch. Solution jumping produces a sharp change in the angles, which yields jerky, unreliable motion. These singularities may be avoided by replacing the passive revolute elbow joint with a passive prismatic joint, as has been done in the *SPU* version of SPAM. Otherwise, the user must avoid giving the robot any input that may result in a singular condition.

Figure 14. Example of a Singular Position

3.2 SPU SPAM Version

The *SPU* version of the SPAM robot is shown in Figure 15. Note that there is now a passive prismatic joint instead of a revolute joint between the *S* and *U* joints of each leg. This version of the platform manipulator is a better design than the *SRU* version for two reasons: 1.) the singularities are reduced to two specific configurations of the robot instead of an entire set of configurations; and 2.) the kinematics are less complex. Dr. Robert Williams II (Williams and Poling 2000) has done the analysis for this version

of SPAM. He has also developed the analysis for a version of the SPAM robot which replaces the Omni-Wrist with a truly spherical wrist. This simplification allowed him to develop analytical inverse pose solutions in place of the Newton-Raphson solutions. His work will be summarized in the following three sections.

Figure 15. SPU Version of SPAM

3.1.1 D-H Parameters

Craig (1989) convention is again used to define the Denavit-Hartenberg parameters for the *SPU* version of SPAM. The full kinematic diagram for the left leg is shown in Figure 16. As with the *SRU* version, the design is symmetrical with both legs being identical, not mirror images.

From Figure 16, one can see the differences in the two versions. L_1 , L_2 , and L_5 still represent the same distances as they did in the *SRU* version. However, L_3 and L_4 are no longer fixed lengths. They now represent the variable lengths between the Omni-Wrist and the *U*-joint at the platform for the left and right legs, respectively. As before, the *U*-joint must be aligned as shown to prevent the platform from spinning

uncontrollably. Also, ϕ_1 (and therefore ϕ_4 on the right side) is no longer present; they are replaced by passive prismatic joint variables L_3 (left leg) and L_4 (right leg). The D-H parameters for the left *SPU* chain are given in Table 4 (angles in degrees).

Figure 16. Kinematic Diagram of Left SPU Chain

Table 4. Left SPU Leg D-H Parameters

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	$\theta_1 + 90$
2	90	0	0	$\theta_2 + 90$
3	90	0	0	θ_3
4	0	L_2	0	θ_3
5	-90	0	0	θ_2
6	0	L_3	0	0
7	0	0	0	$\phi_2 + 90$
8	90	0	0	ϕ_3

Once again L_1 and L_5 are not present in the D-H parameters and are included in the kinematics via homogenous transformation matrices. The variable substitutions necessary for analyzing the right leg are now:

$$\begin{array}{ll}
\theta_4 \rightarrow \theta_1 & L_4 \rightarrow L_3 \\
\theta_5 \rightarrow \theta_2 & \phi_5 \rightarrow \phi_2 \\
\theta_6 \rightarrow \theta_3 & \phi_6 \rightarrow \phi_3
\end{array}$$

Using this set of D-H parameters, the forward and inverse pose and inverse rate kinematics problems are solved.

3.2.2 Pose and Rate Kinematics

Using precisely the same methods as discussed in the *SRU* version section, the pose and rate kinematics problems for the SPAM robot may be solved. The only difference in these methods is that the unknown passive variables being solved for and supplied as guesses are different: L_3 replaces ϕ_1 and L_4 replaces ϕ_4 . The problem statements are identical to those of the *SRU* version. Also, the twelve functions used for the forming the Newton-Raphson Jacobian matrix in the pose kinematics problems are the same.

However, there are small differences in these Jacobian matrices. Due to changes in the structure of the robot, the twelve-by-twelve Jacobian (formed by combining the two six-by-six Jacobians for each leg) for the forward pose problem now has 94 zero terms, as compared to the 88 zeroes in the *SRU* version's method. The six-by-six Jacobians used in the inverse pose problem now have zeroes in the (3,1), (6,1), (4,4), (5,4), (6,4), and (6,6) terms, instead of only the (3,1), (6,1), and (6,6) terms as before. The rate kinematics follows exactly as discussed in Section 3.1.4, except that now

$$\dot{\Theta}_L = \{\dot{\theta}_1 \quad \dot{\theta}_2 \quad \dot{\theta}_3 \quad \dot{L}_3 \quad \dot{\phi}_2 \quad \dot{\phi}_3\}^T \text{ and } \dot{\Theta}_R = \{\dot{\theta}_4 \quad \dot{\theta}_5 \quad \dot{\theta}_6 \quad \dot{L}_4 \quad \dot{\phi}_5 \quad \dot{\phi}_6\}^T.$$

3.2.3 Singularities

The work that Dr. Williams did with the truly spherically-actuated *SPU* version is somewhat less complex. This version is shown in Figure 17. In this case, the rate kinematics allow for an analytical singularity analysis that Matlab can handle. By setting the determinants of the rate Jacobians to zero, singular configurations can be discovered as followed:

$$|J_L| = L_2^2 c_2 s \phi_2 = 0 \qquad |J_R| = L_3^2 c_3 s \phi_3 = 0 \qquad (18)$$

From (18) singular conditions for the *SPU* version of SPAM are:

$$\begin{array}{lll} L_2 = 0 & \theta_2 = \pm 90^\circ & \phi_2 = 0, 180^\circ \\ L_3 = 0 & \theta_3 = \pm 90^\circ & \phi_3 = 0, 180^\circ \end{array} \qquad (19)$$

The left-most conditions of (19) can never occur since L_2 and L_3 must have some minimum length for the platform to assemble. (Note that L_2 now represents the variable length between the spherical wrist and the universal joint.) The restrictions on the two spherical wrist angles (the middle conditions) are at the edge of the wrists' workspace, as described in Table 1 from Chapter 2. Therefore, the first parts of the right-most conditions are the only singular conditions for the *SPU* version of SPAM. These happen when the left ($\phi_2 = 0$) or right ($\phi_3 = 0$) leg is positioned so that the platform is normal to it. In this configuration, the Jacobian row corresponding to ω_y is not independent, which means that rotation about Y_P is not possible.

The determinants of the Jacobians should be monitored on-line; when one approaches zero, the pseudoinverse for that Jacobian should be used. This method

provides a stable means of moving through a singular position, however, the motion will not be exactly as commanded in the neighborhood of the singularity.

Figure 17. Truly Spherically-Actuated Version of SPAM

The solution methods for the forward and inverse pose kinematics problem as well as the inverse rate kinematics problem have been presented. The Newton-Raphson numerical procedure is used for the pose kinematics problems due to the complexity that the Omni-Wrists present. The kinematics terms for these methods are presented in Appendix C. Singularity analyses have also been given. For the *SRU* version of SPAM, this singularity analysis takes the form of a physical testing of the manipulator. The simplified *SPU* version of the SPAM robot eliminates the type of singularity that the *SRU* version suffers from. In addition, since its Jacobians are less complex, the *SPU* version lends itself to a analytical singularity analysis, that reveals two singular conditions for the manipulator.

CHAPTER 4

CONTROL

This chapter presents the theory and implementation for the programs used to control the SPAM robot. The discussion will begin with the design of a linear single joint/link controller. The concepts for this design are the basis for the development of the two control methods that will be presented in the chapter's final two sections. The first of these methods is used for controlling only one joint of the Omni-Wrist. The second method is used for Cartesian control of the platform manipulator. However, as discussed earlier, the Cartesian control is restricted due to the lack of hardware.

4.1 Single Joint Controller Design

This section presents the theory involved in designing a linear controller used for positioning a robotic joint. This is applied to the pitch joint of the Omni-Wrist in this thesis. The wrist amplifiers are set to run in current mode, which means that torque control of the brushless DC servomotors will be used due to the direct relationship between torque and current. These methods are presented in Dr. Williams' website for his ME 401 notes (<http://www.ent.ohio.edu/~bow/html/Courses.html>). The system diagram used for this analysis is shown in Figure 18. (Figures 18 and 20 were developed by Robert Nyzen (1999)).

Figure 18. General System Diagram for Single Joint/Link Controller Design

Shown in Figure 18 are representations of the amplifier, armature circuit, motor, gearing, robotic link, and resolver. $V_{ref}(t)$ is the voltage supplied to the amplifier, where (t) shows that the DC voltage may be changed with time. In current mode, the system's input is the current supplied to the amplifier, denoted as $i_A(t)$. L is the armature inductance and R is the armature resistance. $V_B(t)$ represents the back emf present in the motor's circuit. J_M and J_L are the inertias of the motor and the load (the end of the Omni-Wrist in this case), respectively. C_M and C_L are the coefficients representing the rotational viscous friction present in the motor and the robotic joint, respectively. The gear ratio of the axis is denoted by n . The remaining variables on the diagram are the motor's outputs (denoted with a subscript M) and the overall system outputs (denoted with a subscript L). In these output variables, θ represents the angle, ω the angular velocity, and τ the torque, all of which are functions of time.

To begin the design process, a model of the system must be developed. Modeling in this sense means deriving the system's equations. Figure 19 shows an open loop diagram of the system, beginning with the input voltage and following through to the load angle. The $G(s)$ block in Figure 19 represents the system transfer function. The

system's equations are developed by examining the relationship between the input and output of this transfer function.

Figure 19. Block Diagram of Open Loop System

Laplace transforms are used to convert relationships in the time domain to the frequency domain, replacing the time variable t with s , the Laplace frequency variable. Also, the lowercase symbols representing variables are changed to uppercase to denote frequency domain. Transfer functions are defined as the Laplace transform of the output variable divided by the Laplace transform of the input variable with zero initial conditions (Dorf, 1998).

The modeling is started by examining the left side of Figure 18, the electrical circuit. The inductance in the circuit is ignored in modeling. The voltage drop across the inductor is $L \frac{di}{dt}$. The electrical time constant is neglected because the electrical response is many times faster than the mechanical response and therefore, the inductor's voltage drop is not significant. Applying Kirchoff's Voltage Law yields:

$$V_{ref} = V_b + iR \quad (20)$$

The motor's back emf, V_b , is calculated using the back emf constant, k_b , and the motor's angular velocity as:

$$V_b = k_b \dot{\theta}_M \quad (21)$$

Equations for the mechanical side of the circuit are developed by examining the free body diagrams of the motor and load (Figure 20). For this thesis, the rotational viscous friction has been ignored due to its low impact on the system. The friction on the load side of Figure 20 has especially little effect due to the gearing ratio. With this exclusion, summing moments on the motor side of the diagram gives:

$$\tau_M = J_M \ddot{\theta}_M + \tau_{LR} \quad (22)$$

On the load side, the analysis shows

$$\tau_L = J_L \ddot{\theta}_L \quad (23)$$

The load torque reflected to the motor shaft is:

$$\tau_{LR} = \frac{\tau_L}{n} \quad (24)$$

Also, the motor angle and load angle (as well as angular velocity and acceleration) are related through the gear ratio as:

$$\theta_M = \theta_L n \quad (25)$$

By substituting (23) and (24) into (22):

$$\tau_M = J_M \ddot{\theta}_M + \frac{\tau_L}{n} = J_M \ddot{\theta}_M + J_L \frac{\ddot{\theta}_L}{n} \quad (26)$$

Substituting (25) into (26) yields

$$\tau_M = J_M \ddot{\theta}_M + \frac{J_L \ddot{\theta}_M}{n^2} \quad (27)$$

which yields the system's inertia

$$J_E = J_M + \frac{J_L}{n^2} \quad (28)$$

and the relationship between the motor torque and angular acceleration is

$$\tau_M = J_E \ddot{\theta}_M \quad (29)$$

Figure 20. Free Body Diagram of Single Joint System

Because the Omni-Wrist amplifiers are set to run in current mode in the, the motor's torque must be related to the motor's input current. To do this, the motor's torque constant, k_T , is used:

$$\tau_M = k_T i_A \quad (30)$$

From the above relationships, the input voltage is:

$$V_{ref} = k_b \dot{\theta}_M + \frac{J_E \ddot{\theta}_M}{k_T R} \quad (31)$$

which can be rearranged to give the system's equation of motion (EOM):

$$\ddot{\theta}_M + \frac{k_b k_T \dot{\theta}_M}{R J_E} = \frac{V_{ref} k_T}{R J_E} \quad (32)$$

The Laplace transform of (32) is:

$$s^2 \Theta_M(s) + \frac{k_b k_T}{R J_E} s \Theta_M(s) = \frac{k_T}{R J_E} V_{ref}(s) \quad (33)$$

Equation (33) yields the system's transfer function $G(s)$:

$$G(s) = \frac{\Theta_M(s)}{V_{ref}(s)} = \frac{\frac{k_T}{RJ_E}}{s^2 + \left(\frac{k_b k_T}{RJ_E}\right)s} \quad (34)$$

The second block in the open loop diagram of Figure 19 is $G_R(s)$. This represents the gear ratio between the motor's output angle and the system's output angles. $G_R(s)$ is defined:

$$G_R(s) = \frac{\Theta_L(s)}{\Theta_M(s)} = \frac{1}{n} \quad (35)$$

Equations (20) through (35) provide a model for the DC servomotor/robotic link system. This model is then transformed into a Simulink diagram that is used to control the system. This method requires a system transfer function be developed using $G(s)$ and the Laplace representation of a PID (Proportional, Integral, Derivative) controller. Dorf and Bishop (1998) give this controller transfer function as:

$$G_C(s) = K_P + \frac{K_I}{s} + K_D s \quad (36)$$

where the proportional gain is K_P , the integral gain is K_I , and the derivative gain is K_D . The three terms work together to form a complete controller, but, in general, they each have an effect on a certain part of the system's performance. The proportional term acts on the system's rise time; higher gains mean faster responses. The derivative terms has a damping effect on the system output; large derivative gains yield a small envelope for

oscillation of the system output. The integral term (often times not necessary) is used to reduce or eliminate any steady state error in the system's performance.

The final part in designing a controller for the system is determining these gains. As stated before, this is done using the ITAE criterion. To begin, the overall system transfer function must be defined as:

$$T(s) = \frac{G_C(s)G(s)}{1 + G_C(s)G(s)} \quad (37)$$

where the system is assumed to have negative unity feedback from the resolver. $G(s)$ can be simplified by allowing

$$\alpha = \frac{k_b k_T}{R J_E} \quad \text{and} \quad \beta = \frac{k_T}{R J_E} \quad (38)$$

From Equations (34), (36), (37), and (38), the system transfer function is:

$$T(s) = \frac{\beta(k_D s^2 + k_P s + k_I)}{s^3 + (\alpha + \beta k_D)s^2 + \beta k_P s + \beta k_I} \quad (39)$$

The denominator of (39) is the characteristic polynomial of the closed loop system. The coefficients of this polynomial are used to determine the controller gains by equating them to the coefficients of the desired characteristic polynomial. The desired characteristic polynomial may be provided by two different methods. The first method uses the ITAE criterion for a 3rd order polynomial with a step input:

$$\Delta s = s^3 + 1.75\omega_n s^2 + 2.15\omega_n^2 s + \omega_n^3 \quad (40)$$

where ω_n is introduced as the system's natural frequency. The second method requires that system design parameters be set. Design parameters for the system's performance determine the value of its natural frequency. These parameters include rise time, settling

time, and percent overshoot. Definitions of these parameters may vary. Settling time (T_S) is the amount of time required for the system to settle into a range of $\pm 2\%$ of steady state. Percent overshoot (PO) is a measure of the distance that the system travels past the steady state value. The equations for determining these values are as follows:

$$T_S = \frac{4}{\zeta \omega_n} \quad (41)$$

$$PO = 100e^{-\frac{\pi \zeta}{\sqrt{1-\zeta^2}}} \quad (42)$$

Present in all three of these equations is ζ , which is the system's damping ratio. This is another unknown that can be solved along with the natural frequency by specifying the design parameters. Using the two equations above to solve for ζ and ω_n , the roots for the desired characteristic polynomial are found as:

$$s_{1,2} = -\zeta \omega_n \pm j \omega_n \sqrt{1-\zeta^2} \quad (43)$$

Because the characteristic polynomial of (39) is a third order polynomial, a third root must be added to find the desired characteristic polynomial. This third root is given as

$$s_3 = 10(-\zeta \omega_n) \quad (44)$$

Using the Matlab “convolve” command, the three roots are used to find the coefficients of the desired characteristic polynomial. These coefficients are then compared to those of the denominator in (39). This comparison yields the values for the PID controller gains (K_P , K_I , and K_D). Once determined, these gains are entered into the “PID Controller” block, which is a predefined Simulink block.

In most cases, these gains will not provide the best possible performance for the given system. It does, however, provide a good starting point from which to tune the PID controller so that it will produce the desired output. It is best to start with simulation and examine the controller's performance and then move on to hardware implementation. With an accurate model, the simulation should yield results near that of the actual hardware implementation. Once the simulation performance is satisfactory, tuning the PID gains to produce the desired hardware performance should be carefully done. Making too large a change in any gain may cause unstable behavior.

To begin the tuning, the proportional gain K_P should be adjusted. Increasing this gain decreases the system's rise time and increases the overshoot and steady-state error. The derivative gain K_D should be the next adjustment. An increase in this gain will yield lower overshoot and settling time. The integral gain should be the last gain adjustment and it often requires no adjustment. K_I is used mainly for eliminating steady-state error, but increasing it will also decrease the rise time and increase the settling time and overshoot. Finding the proper combination of these gains is the key to PID tuning, as each change will likely affect previous behavior.

The theory behind controller design for a single joint/link system has been presented in this section. The theory provides the methodology to create a valid model for the system. This model can then be used to calculate the appropriate proportional, integral, and derivative gains for a PID controller. This controller is simulated before being used with hardware and then adjusted to provide the desired hardware performance

in a process called PID tuning. Implementation of these methods is presented in the next section.

4.2 Independent Joint Control

This section presents the implementation of a PID controller designed using the theory presented in the previous section. The controller is used both in simulation as well as with hardware. In this case, the controller operates on only one joint of the Omni-Wrist. This procedure ignores any effects that another joint of the wrist may have on the controlled joint. This is possible because of the high gear ratios involved with the Omni-Wrist. For simulation and hardware implementation, Matlab's Simulink and Quanser's WinCon software are used (discussed in Section 2.2). Figure 21 shows the closed loop system.

Figure 21. Simulink Model for Simulation of PID Position Control of Independent Joint

The controller shown in Figure 21 is designed for the pitch joint of the Omni-Wrist. The physical parameters used in this design process are listed in Table 5. They were taken from work done by John Ottersbach. Notice that a few of the parameters

involved in the previous discussion are absent from Table 5. The first of these, J_L , is left out because it is not significant in a system that has a large gear ratio. This can be seen in (27) and (28) where it is divided by n^2 . The other two terms missing from Table 5 are the friction coefficients C_M and C_L . As with J_L , one could argue that C_L is also insignificant due to the large gear ratio. However, in this study, both motor and load friction have been ignored. This allowance can be corrected with PID tuning. The performance specifications for the Omni-Wrist were set as a two second settling time and less than five percent overshoot to reach any pose within its limits. Results of this controller design process will be given in Chapter 5.

Table 5. Omni-Wrist Physical Parameters for Controller Design

Parameter	Value
τ_M	0.542 N·m
i	1.7 Amps
K_b	0.363 Volts/(rad/sec)
R	22 Ω
K_T	0.319 N·m/Amp
J_M	2.34×10^{-4} kg·m ²
n	259.2

Figure 21 shows the Simulink program used for simulating the closed-loop PID control developed to control the position of the pitch joint of the Omni-Wrist. The model was developed from the equations presented in Section 4.1. The model flows from left to right, as indicated by the arrows (feedback flows right to left). It begins with the user entering the required fields in the block labeled “Step”. By double clicking on this box, the user brings up the window shown in Figure 22.

Figure 22. Input Dialog Box for Simulation Program

The first value in this selection box is the “Step time.” This is the time from the beginning of the control cycle at which the input command is given. This produces a pause at the beginning of the control cycle for the specified amount of time. The “Initial Value” input is used for giving the output a starting place, usually zero. The “Final Value” command is the input that allows the user to define the final state of the output; in this case, this value is the desired load angle. The “Sample time” block sets the sampling rate of the step.

The next block, labeled “Degrees to Radians”, converts the load angle input from degrees to radians. This value is then converted to a motor angle via the “Gear Ratio” block. Now the controller has a commanded angle in radians. However, the controller does not operate on the input alone; rather, it operates on the error between this input and the feedback from the previous control cycle. This error is the output of the summing junction. This error is then sent to the PID controller. The controller is a predefined Simulink block that is actually a mask of the system shown in Figure 23. By left-clicking

on the “PID Controller” block and selecting “Look Under Mask”, the user can see the controller details.

Figure 23. PID ControllerDetails

This figure shows that the error is sent in and operated on by the controller using the proportional, integral and derivative gains and the controlled signal is sent on through the cycle. The gains are user-defined and can be entered by double clicking on the “PID Controller” block. This action brings up the input window shown in Figure 24. The window has inputs for each of the gains and provides the user with an easy interface to tune the controller.

Figure 24. Input Menu for Controller Gains

The output from the controller is then sent through a block diagram representation of the system modeling that was presented in the previous section. The “Beta” and “Alpha” constants are from Equation (38) and the system of blocks between the controller and the “1/n” block represent the simplified equation of motion (EOM) for the system, as developed in Equation (32). After the motor’s angular acceleration is integrated, the angular velocity is sent back for back emf simulation. Another integration yields the motor angle, which is fed back to the summing junction to create the error for the next control cycle (perfect angle sensing is assumed for the simulation). The motor angle is divided by the gear ratio and converted back to degrees for final output of the controlled load angle. This angle is then plotted versus time using a Simulink “Scope” block.

After the simulation yields acceptable performance, the final goal is hardware control. To transform this simulated control program into a real-world control program, the Quanser products and the Matlab Real-Time Workshop are used. Figure 25 shows the Simulink diagram for hardware control. Notice that the equation of motion is no longer necessary because the physical system is present in the cycle, although this is not evident by looking at Figure 25. As described in Chapter 2, the Quanser MultiQ-3 blocks connect the computer with the hardware. The voltage output from the controller is sent to the MultiQ-3 boards (Analog Output1 in Figure 25) and the system then follows the routine discussed in Chapter 2. Also as discussed in Chapter 2, the feedback comes back from the resolver and encoder emulator board (Encoder Feedback in Figure 25) and is converted from counts to motor angle (in radians) by using Equation (2). The block

labeled “Gain1” handles this conversion. As in the simulation program, this value is converted from the motor angle to the load angle and then from radians to degrees.

Figure 25. Simulink Model for Hardware Control of Independent Joint

The MultiQ-3 blocks allow the user to substitute the hardware for the system modeling. By double clicking on either the “Analog Output1” block or the “Encoder Feedback” block, the user must define the parameters required by the WinCon software. These parameters include the number of the MultiQ-3 board being used and the channel that the input or output is connected to. With all parameters properly set, PID tuning can then be used to achieve the desired performance from the hardware by experiment.

This section has presented the implementation of Simulink control programs to perform the PID control of the pitch joint of the Omni-Wrist. This implementation has been used for both the simulation of controlling the joint as well as the real world control of the joint, using the Quanser products and the Matlab Real Time Workshop as described in Chapter 2 to make the transformation from simulation to the real world. Results of this implementation will be shown in the next chapter. Independent joint control is the simplest form of control and operates strictly in joint space in these examples. The schemes developed in this section will be used as the basis for a simple

Cartesian control algorithm that will control the plunge motion of the SPAM robot. This algorithm is the subject of the following section.

4.3 Cartesian Control

In this section, the methods used for Cartesian control of the SPAM robot are discussed. Cartesian control involves independently controlling each joint to produce a desired pose for the platform. Forward and inverse kinematics must be used within the control algorithm to map joint space to Cartesian space and vice versa. Because of the aforementioned hardware restrictions, only two of the six active joints of the robot are currently operable. This leads to a reduced Cartesian control scheme in which only two degrees of freedom may be controlled. The motion described in this section is a vertical plunge motion, in which the pitch joints of both wrists are identically controlled to produce varying z heights while keeping the platform level with the base. This motion is a one-degree-of-freedom motion.

Controlling this reduced motion calls for a simpler kinematics analysis to be performed. The six degree-of-freedom SPAM robot has been essentially transformed into a two degree-of-freedom manipulator. Requiring the platform to have no rotation about the x -axis of the base frame further constrains the robot to a single-degree-of-freedom mechanism, computing only θ_2 or θ_5 with simplified kinematics. The other angle is then set to the negative of the computed angle by symmetry, and identical PID controllers control each of the angles independently. Figure 26 shows the reduced degree of freedom manipulator.

Figure 26. *Reduced Degree of Freedom SPAM Robot*

The full Newton-Raphson method presented in Chapter 3 could be used here instead of the simplified version. However, the current version of the Real-Time Workshop (RTW) has a limitation that makes this option infeasible. The current RTW cannot process Simulink s-functions. These s-functions allow users to implement a Matlab m-file that is formatted in a specific manner so that Simulink can pass inputs to the file and receive outputs from it. With this option available, the existing kinematics code could be used to go between joint space and Cartesian space. Instead, this limitation makes programming full kinematics in Simulink difficult and hard to understand because series of simpler math function blocks must be used to achieve the same effect that could be produced by a single s-function block.

Examining the right leg only, the forward pose kinematics problem for the plunge motion is stated: Given θ_5 , calculate z_p . Only one leg is needed since the robot is symmetrical in this simplified case. The y position of the platform is known to be L_5 from the base frame because the desired motion is centered between the two wrists. The

y position of each joint, beginning at the second universal joint of the Omni-Wrist, is calculated as:

$$\begin{aligned} y_1 &= L_1 - L_2 \sin(\theta_5) \\ y_2 &= y_1 - L_3 \sin(2\theta_5) \\ y_3 &= L_5 \end{aligned} \quad (45)$$

Using this information, the value for the passive revolute joint angle is found to be:

$$\phi_4 = \sin^{-1}\left(\frac{y_3 - y_2}{L_4}\right) - 2\theta_5 + \pi \quad (46)$$

Finally, the z values for the joints are calculated:

$$\begin{aligned} z_1 &= L_2 \cos(\theta_5) \\ z_2 &= z_1 + L_3 \cos(2\theta_5) \\ z_3 &= z_2 + L_4 \cos(\phi_4 + 2\theta_5) \end{aligned} \quad (47)$$

The platform height z_p is equal to z_3 .

The inverse pose kinematics problem is stated: Given z_p , calculate θ_5 . This is done by rearranging the above equations and adding that $y_3 = y_2 - L_4 \sin(\phi_4 + 2\theta_5)$.

The equations for locating the passive universal joint (joint 3), and therefore the platform height, are rearranged as follows:

$$\begin{aligned} y_3 - L_1 + L_2 \sin(\theta_5) + L_3 \sin(2\theta_5) &= L_4 \sin(\phi_4 + 2\theta_5) \\ z_3 - L_2 \cos(\theta_5) - L_3 \cos(2\theta_5) &= L_4 \cos(\phi_4 + 2\theta_5) \end{aligned} \quad (48)$$

These equations are squared and added together to eliminate ϕ_4 , yielding:

$$\begin{aligned} y_3^2 - 2L_1y_3 + 2L_2y_3 \sin(\theta_5) + 2L_3y_3 \sin(2\theta_5) + L_1^2 - 2L_1L_2 \sin(\theta_5) - 2L_1L_3 \sin(2\theta_5) + L_2^2 \\ + 2L_2L_3 \cos(\theta_5) + L_3^2 + z_3^2 - 2L_2z_3 \cos(\theta_5) - 2z_3L_3 \cos(2\theta_5) = L_4^2 \end{aligned} \quad (49)$$

Equation (49) is expanded using the substitutions $\sin(2\theta) = 2 \sin(\theta) \cos(\theta)$ and $\cos(2\theta) = \cos^2(\theta) - \sin^2(\theta)$. Furthermore, the tangent-half angle substitutions are made such that:

$$\begin{aligned} t &= \tan\left(\frac{\theta_5}{2}\right) \\ \cos(\theta_5) &= \frac{1-t^2}{1+t^2} \\ \sin(\theta_5) &= \frac{2t}{1+t^2} \end{aligned} \quad (50)$$

The final substitution is setting y_3 equal to L_5 . These substitutions produce a quartic equation in terms of t . This equation can be solved for the four values of t and θ_5 is then solved as

$$\theta_5 = 2 \tan^{-1}(t) \quad (51)$$

Normalizing the equation with the above substitutions and inserting the values for the lengths, Equation (49) is transformed to

$$t^4 + at^3 + bt^2 + ct + d = 0 \quad (52)$$

where

$$\begin{aligned} a &= \frac{738.5}{z_3^2 - 24.75z_3 + \frac{393}{64}} & b &= \frac{2z_3 + 168z_3 + \frac{6610}{64}}{z_3^2 - 24.75z_3 + \frac{393}{64}} \\ c &= \frac{829.5}{z_3^2 - 24.75z_3 + \frac{393}{64}} & d &= \frac{z_3^2 + 31.25z_3 + \frac{6217}{64}}{z_3^2 - 24.75z_3 + \frac{393}{64}} \end{aligned} \quad (53)$$

Dr. Williams presented a method for solving this type of equation in his master's thesis (1985). This method requires that the roots of the quartic's resolvent cubic be given to find the equation's four solutions. The resolvent cubic is:

$$x^3 - bx^2 + (ac - 4d)y - a^2d + 4bd - c^2 = 0 \quad (54)$$

An offline symbolic Matlab program is used to find the roots of the resolvent cubic given z_3 . Using one of the real roots, y_0 , the four values for t are found using the method described by Dr. Williams in his thesis. This method proceeds as follows:

$$R = \frac{a^2}{4} - b + y_0 \quad (55)$$

For R not equal zero:

$$\begin{aligned} D &= \frac{3a^2}{4} - R^2 - 2b + \frac{4ab - 8c - a^3}{4R} \\ E &= \frac{3a^2}{4} - R^2 - 2b - \frac{4ab - 8c - a^3}{4R} \end{aligned} \quad (56)$$

For R equal zero:

$$\begin{aligned} D &= \frac{3a^2}{4} - 2b + 2y_0^2 - 4d \\ E &= \frac{3a^2}{4} - 2b - 2y_0^2 - 4d \end{aligned} \quad (57)$$

The roots of the quartic are:

$$\begin{aligned} t_{1,2} &= \frac{-a}{4} + \frac{R}{2} \pm \frac{D}{2} \\ t_{3,4} &= \frac{-a}{4} - \frac{R}{2} \pm \frac{E}{2} \end{aligned} \quad (58)$$

These roots are then used to find θ_5 as given in (51) and the inverse pose kinematics problem is solved.

Figure 28. System Inputs Dialog Box for Cartesian Control

Figure 29. Inverse Kinematics Subsystem

The inverse kinematics routine solves the quartic equation by working through several function blocks and then reducing all of the possible solutions to the one desired solution for θ_2 . The desired solution is the one that produces the proper configuration of the manipulator. In this case, it is the value of θ_2 that produces the “elbow up” branch. This sorting is done using the switch blocks, which each have a threshold value that the center input is compared to. If the value of the center input is equal to or greater than the threshold value, the top path is chosen. Otherwise, the bottom path is chosen.

The switches labeled “Switch” and “Switch1” filter out the appropriate value for θ_2 , depending on the value of the desired height. If z is greater than or equal to 0.25 inches, solution 3 (or 7) is the solution that corresponds to the desired branch of the SPAM robot. Otherwise, solution 2 (or 6) is used. The switch labeled “Switch2” chooses between the top and bottom solution sets once the branch filtering has been done. This switch decides whether Equation (56) or (57) is used. The output of this switch is a value for θ_2 that must undergo one last important operation before being controlled. The Omni Wrists have no brakes or other means of overcoming gravity to keep the pitch joints at an upright position. Because of this, θ_2 actually starts at a horizontal position, which corresponds to 45 degrees in the coordinate system used for the kinematic analysis in Chapter 3. However, absolute encoders and resolvers always begin at zero. Therefore, the final summing junction is in place to accommodate this difference between the hardware and kinematic configuration. The output of the “Inverse Kinematics” block is the commanded value of θ_2 in degrees.

The symmetrical plunge motion requires that θ_2 and θ_5 be equal and opposite. Therefore, the value of the θ_2 is negated and sent to a separate independent joint control scheme as θ_5 . These independent joint controllers are the routines from Section 4.2. The only difference in the simulation program (Figure 27) and the hardware program is the same difference that separates Figures 21 and 25. The program for controlling the hardware looks exactly like Figure 27 at the top level. All subsystems are the same except the independent joint controller, where Figure 25 is substituted as in Section 4.2 to link the computer to the hardware.

It should also be noted that the load inertia has been greatly increased by adding the aluminum links to the wrists, but this effect is negligible in designing the controller due to the large gear ratios. This allows the Cartesian control program to use the same controller gains as the independent joint control program. From the two controllers, the controlled θ_2 and θ_5 are output in degrees and sent to scopes along with the commanded values for these angles. This is done to monitor the controller's performance and help in tuning the PID gains if necessary.

From this point, the program operates only on θ_2 to return to Cartesian space. The first operation is to return the angle to the proper form that the kinematic analysis has been developed to handle. Recall that it was transformed due to hardware restrictions so that θ_2 equal to zero meant that the pitch joint was horizontal. Returning to the proper orientation means that the pitch joint is vertical at θ_2 equal to zero. This is the function of the summing block after the independent joint controller in Figure 27. This value is sent

to the forward kinematics to return to Cartesian space to assess the controller's performance. The forward kinematics subsystem is shown in Figure 30.

Figure 30. Forward Kinematics Subsystem

Figure 30 shows that θ_2 is first converted from degrees to radians and then used to solve for the elbow angle, ϕ_1 . The equations that are placed in the function blocks “phi1” and “Actual z” come from Equations (46) and (47). The elbow angle is sent to a scope for monitoring and into a vector along with the actual controlled value for θ_2 . This vector is then passed to the final function, which calculates the actual z height of the platform based on the values of the Omni-Wrist pitch angle and the elbow angle. Finally, at the far right side of Figure 27, the actual platform height value is placed into a vector along with the desired platform height value and plotted to evaluate the system's performance. This evaluation is included in the following chapter.

This chapter has presented the methodology and implementation of PID controller design. The independent joint control scheme, developed and tested first in simulation, served as the basis for the more complex Cartesian control scheme. The Cartesian control scheme includes subsystems for evaluating the forward and inverse kinematics problems of the SPAM robot. However, in this case, the robot (and therefore the

kinematic analysis) has been simplified due to hardware restrictions. Results of this design and implementation are presented in Chapter 5.

CHAPTER 5

EXPERIMENTATION AND RESULTS

This chapter presents the experimentation and results associated with operating and controlling the SPAM manipulator. The first section discusses the experimentation with the forward and inverse kinematics algorithms. To test the programming for the forward and inverse pose problems, an algorithm was developed to generate a trajectory for the manipulator. This algorithm allows the user to input the initial configuration of the robot and then increments the input by a predefined amount. A program was also developed to test the inverse rate kinematics solution. This program requires the user to input the desired Cartesian rates and initial configuration of the robot and produces the resulting joint rates. The second section presents the experimentation with the controls programs, both in simulation and in hardware implementation.

5.1 Kinematics Examples

The forward pose kinematics trajectory generation program prompts the user to enter the Omni-Wrist angles (θ_1 through θ_6). Because the algorithm is based on the Newton-Raphson method, the user must also enter the initial guess for the Cartesian unknowns and the passive joint variables ($x, y, z, \alpha, \beta, \gamma$, and ϕ_1 through ϕ_6). This guess must be close enough to the actual values so that the Newton-Raphson method converges. From these inputs, the algorithm develops a value for incrementing the Omni-Wrist angles; at each step, the Newton Raphson method is used to compute the resulting values

for the Cartesian unknowns and passive joint variables. A graphical representation of the platform and the results of the algorithm are plotted for the user to view and evaluate.

The user can choose to see a single isometric view of the platform, or a set of four views, including the isometric view as well as a front, top, and right side view.

The example for the forward pose kinematics commands the robot to a nominal position. This position has little rotation of the platform in any direction and locates the platform near the middle of its x - y workspace and near the top of its plunge height. Note that the rotations α , β , and γ are not zero, but are very close. This is done to avoid the singularity present when these angles are equal to zero. The input values for the initial Omni-Wrist angles are $\Theta = \{1, 5, 1, -10.281, -2.847, 1.2053\}^\circ$ (θ_1 through θ_6) and the corresponding initial guess for the Cartesian unknowns and the passive joint variables is $X = \{.210, -.766, 25.792, 2.174, 2.027, -1.962\}$ and $\Phi = \{-49, 37, 1, 45, -38.546, 12.694\}$ where $X = (x, y, z, \alpha, \beta, \gamma)$ (in, deg) and $\Phi = (\phi_1, \phi_2, \phi_3, \phi_4, \phi_5, \phi_6)$ (deg). This configuration is shown in Figure 31.

Figure 31. Nominal Configuration for Forward Pose Kinematics Example

The next three figures show the results of the trajectory generation. The delta value set for this program was a 0.2° increment per step for the Omni-Wrist angles. If this trajectory generation were to be used with hardware, this increment would depend on the sampling rate of the control program. Programs running at lower frequencies are capable of handling larger increments. The time axis is labeled “Time Steps” instead of “Time” for this reason. These time steps could correspond to any amount of time, depending on the sampling rate of the control program.

A tolerance was specified for allowable error in the Newton Raphson procedure. This tolerance was set at 0.0001 inches for the linear unknowns and 0.000001 radians for the angular unknowns. Figure 32 shows the number of times the Newton-Raphson iteration takes to converge within the specified error for each variable. The figure shows that three or four counts are required for convergence inside of these limits at each time step.

Figure 32. Counts to Convergence for Forward Pose Kinematics Example

Figure 33 shows the change in the position of the platform coordinate frame that results from driving the Omni-Wrists using the aforementioned incremental values. The figure shows that after ten time steps, the platform has moved approximately two inches in the positive x direction, 1.5 inches in the negative y direction and just under an inch in the negative z direction. This movement may happen in any amount of time, depending on the sampling rate of the control software. A two-inch movement would likely be too large for a very small amount of time, but is certainly acceptable for something on the order of one or two seconds. In addition, it may be too small a movement for ten seconds. The application would determine the appropriate magnitude of the movement.

Figure 33. Platform Cartesian Displacement for Forward Pose Kinematics Example

Figure 34 shows the resulting orientation change for the platform. The figure shows that platform rotates approximately minus one degree about the z -axis (α), four degrees about the y -axis (β), and minus two degrees about the x -axis (γ). As with the Cartesian displacements, the magnitude of these movements over time depend on the control cycle's sampling rate and the application of the manipulator.

Figure 34. Platform Angular Displacement for Forward Pose Kinematics Example

The inverse pose kinematics algorithm follows the same routine as the forward pose kinematics program. The user must first enter the desired Cartesian values for the platform ($x, y, z, \alpha, \beta, \gamma$) and a guess for the initial configuration for the left ($\theta_1, \theta_2, \theta_3, \phi_1, \phi_2$, and ϕ_3) and right legs ($\theta_4, \theta_5, \theta_6, \phi_4, \phi_5$, and ϕ_6). With this information, the algorithm uses a predefined incremental value to increase the platform's position and orientation at

each step. The program uses the Newton-Raphson inverse pose kinematics code to compute the associated joint values for all joints of the robot at each of these steps. As described in Chapter 3, the inverse pose problem is decoupled between the left and right legs; therefore, each leg is solved independently of the other. As with the forward pose trajectory generation, the graphical representation of the manipulator is made available for verification and evaluation.

The first example for the inverse pose kinematics trajectory generation uses the same nominal configuration as the forward pose example, which is shown in Figure 31. From this pose, the manipulator is commanded to increase the x , y , and z positions of the platform by 0.01 inches per time step. The Euler rotation angles, α and β , are to stay constant and γ is commanded to increase by 0.5° per time step. Figure 35 shows the counts to convergence for the Newton-Raphson method to find the inverse pose solution within the specified error tolerances, defined as they were in the forward kinematics example.

Figure 35. Counts to Convergence for First Inverse Pose Kinematics Example

Figures 36 and 37 show the results of the inverse pose kinematics trajectory generation program for this first example. These results are the left and right leg Omni-Wrist angles that correspond to the desired Cartesian values. Notice the sharp change in the angles at the fourth time step. At this time, the combination of α , β , and γ is near enough to zero to constitute a singular position for the SPAM robot. However, the Newton-Raphson method is capable of working through the singularity with a relatively small number of iterations. The problem with the Newton-Raphson method is that it has jumped solution branches, represented by the sharp change in the angles. The second example will show how the user can avoid commanding the robot into one of these singular positions. This should be done because the sharp change in angles could result in dangerous, unstable motions from the manipulator.

Figure 36. Left Leg Results for the First Inverse Pose Kinematics Example

Figure 37. Right Leg Results for the First Inverse Pose Kinematics Example

To avoid sending the robot to a singular condition, the user must learn and understand where these conditions exist. In the first example, the nominal configuration started with γ at approximately -2° . Commanding this angle closer to zero by 0.5° per time step made the platform approach the singular condition. After four time steps, the platform was close enough to the singular position to allow the Newton-Raphson method to jump solution branches. By simply commanding γ further away from zero, this singularity can be avoided, which has been done in the second example. In this example, γ has been decreased by 0.5° per time step, sending it further away from zero. Figure 38 shows the counts for convergence for this example. Figures 39 and 40 show the smooth curves representing the Omni-Wrist angles' values, where there are no sharp changes. Figure 41 shows the four views of SPAM after going through the commanded trajectory.

Figure 38. Counts to Convergence for Second Inverse Pose Kinematics Example

Figure 39. Left Leg Results for the Second Inverse Pose Kinematics Example

Figure 40. Right Leg Results for the Second Inverse Pose Kinematics Example

Figure 41. Four View Graphical Representation of SPAM After Following Trajectory

The final kinematics example focuses on the routines developed to solve the inverse rate kinematics problem. The program prompts the user to input the desired Cartesian rates $\dot{X} = \{\dot{x} \quad \dot{y} \quad \dot{z} \quad \omega_x \quad \omega_y \quad \omega_z\}^T$ and the initial guess for the joint configurations of each of the legs. Like the inverse pose kinematics problems, the algorithm solves each leg separately because the problems are decoupled.

The nominal configuration shown in Figure 31 is again used as the starting point for this example. From this position, the robot is commanded to move 0.1 inches per time step in each of the positive x , y , and z directions and rotate $\omega_z = 0.02$ degrees per time step about the z -axis of the base frame. The inverse rate kinematics program calculates the resulting joint rates for the active and passive joints for each leg. If the

program were being used with hardware, these values would have to be integrated so that they could be sent to the hardware as angular values instead of angular rates. This integration has also been done in simulation. Both the joint rate values and angular values are plotted for the user to monitor and evaluate. The simulation program again provides graphical animation for the platform.

Figure 42 shows the program output for the left leg. The left half shows the angular rates and the right half shows the integral of those curves (the angle values.) The top two graphs represent the values for the Omni-Wrist angles and the bottom two represent the passive joint angles. Figure 43 shows the same output for the right leg. The smooth curves shown in this example are necessary if any hardware was to be used with this program. A sharp turn in either the angular rates or angle values could cause unsteady behavior from the robot. These outputs should always be tested and evaluated before being used with hardware.

Figure 42. Left Leg Results of Inverse Rate Program

Figure 43. Right Leg Results of Inverse Rate Program

This section has presented examples for the simulation of trajectory generation programs for the forward and inverse pose kinematics problem as well as for the inverse rate kinematics problem. These examples have been provided to validate the code used to solve these three problems. It has been shown that the Newton-Raphson is capable of working through singular positions, although it may produce unwanted jumping of solution branches. This shows up in the simulation as a sharp change in the angles or angular rates. The changing of solution branches may cause unsteady behavior from the robot and must be avoided by the user. The section also discussed the relationship of the

demonstrated movements to actual time. Each graph shows the number of time steps on the independent axis instead of a number of seconds because the actual time depends on the sampling rate of the software.

5.2 Controls Examples

This section presents the results of experimentation with the PID controller for both a single joint/link system and a one degree-of-freedom plunge motion as discussed at the end of Chapter 4. The process begins with finding the PID gains using the performance specifications. These gains are tested and adjusted to give the desired performance in simulation for the single joint/link system. Using the adjusted gains, the plunge motion is simulated and evaluated. Finally, the controller is implemented with the hardware for position control of the plunge motion.

To calculate the PID gains, the design parameters set forth in Chapter 4 were used to define the system's natural frequency, ω_n , and damping ratio, ζ . These gains were set to be a settling time, T_s , less than two seconds and a percent overshoot, PO , less than five percent. From these parameters, ω_n and ζ are calculated using Equations (41) and (42):

$$\begin{aligned}
 PO = 5 &= 100e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}} \\
 \ln(.05) &= \frac{-\zeta\pi}{\sqrt{1-\zeta^2}} \\
 \zeta &= 0.691
 \end{aligned}
 \tag{59}$$

$$T_s = 2 = \frac{4}{\zeta \omega_n}$$

$$\omega_n = \frac{2}{\zeta} = 2.9 \text{ rad / s}$$
(60)

The roots of the desired third-order characteristic polynomial are found using Equations (43) and (44):

$$s_{1,2} = -\zeta \omega_n \pm j \omega_n \sqrt{1 - \zeta^2} = -.691(2.9) \pm j 2.9 \sqrt{1 - .691^2} = -2 \pm 2.1j$$

$$s_3 = 10(-\zeta \omega_n) = 10(-2) = -20$$
(61)

These three roots are then used to find the desired characteristic polynomial, which is

$$s^3 + 24s^2 + 88.41s + 168.2 = 0$$
(62)

The coefficients of this desired polynomial are compared to the coefficients of the denominator of the system transfer function, shown in Equation 39 to be:

$$s^3 + (\alpha + \beta k_D)s^2 + \beta k_P s + \beta k_I$$
(63)

This comparison yields

$$\begin{aligned} \alpha + \beta k_D &= 24 \\ \beta k_P &= 88.41 \\ \beta k_I &= 168.2 \end{aligned}$$
(64)

where $\alpha = 11.25$ and $\beta = 30.98$ from Equation (38). Solving Equation (64) gives the PID gains as:

$$k_P = 2.854 \quad k_I = 5.429 \quad k_D = 0.411$$

These values provide a starting point for tuning the controller gains.

To begin, the values calculated above are put into the PID controller in the single joint/link simulation program to evaluate the controller's performance. The program is

given a step input of 30° and then the position control is simulated. Figure 44 shows the results of this position control. The controller has come very close to meeting the settling time requirement of two seconds. However, this controller has approximately 25% overshoot. PID tuning must now be used to correct the controller's performance. Another item to be noted is the x -axis of Figure 44. The axis now shows "Time" in seconds instead of "Time Steps" as before. This is because the program's sampling rate is now known. The Simulink program is running at 500 Hz. Therefore, the program has cycled 5000 times to give this ten second result.

Figure 44. Initial Attempt at PID Position Control of Single Joint/Link System

The PID tuning process may be a lengthy one. In this case, experimentation was done for one to two hours to arrive at PID gains that produced results that not only met the design parameters, but exceeded them. Using gains of $k_P = 25$, $k_I = 1$, and $k_D = 1$, the system's performance is nearly ideal, as shown in Figure 45. The figure reveals that the system settles into steady-state at under 0.5 seconds and has just over 2 percent overshoot. Proper tuning of the single joint/link system allows for simulation of the one-degree-of-freedom plunge motion. Note that the gains have been tuned for the robot itself. Adding additional weight would require different PID gains to produce the desired motion.

Figure 45. Tuned PID Position Control of Single Joint/Link System

With the satisfactory simulation of the position control of a single Omni-Wrist, the next objective is to achieve the same performance from the two wrists working together to produce the plunge motion. In this case, the desired z height of the platform is given as -5 inches. A real root of the resolvent cubic (given in Chapter 4, Equation (4)) for this configuration is 5.3145. These two inputs are given to the system using the tuned PID gains. Figures 46 through 48 show the simulation's results. Figure 46 shows the results that are the primary concern for the control of the plunge position: the movement of the platform in the z direction. The controller performs well; it places the platform at the desired height in approximately 0.2 seconds and has very little overshoot. Figures 47 and 48 show the movement of the pitch joints required to produce this positioning of the platform. Note that the pitch joints move exactly opposite one another, as required.

Figure 46. Simulation of Plunge Position Control

Figure 47. Simulated Control of θ_2 for Plunge Motion

Figure 48. Simulated Control of θ_5 for Plunge Motion

With all simulations of the controlled system properly functioning and meeting the design parameters, the final objective is to control the hardware. For comparison, the same inputs used in the previous example are used. The platform is commanded to five inches below the center of the first universal joint on the Omni-Wrists. Figure 49 shows the system's response to this command. The hardware responds almost exactly as the simulation did. The rise time is a fraction of a second longer with the hardware, but still well under the required settling time of two seconds.

Another difference with the hardware is the platform's position at the beginning of the cycle. Figure 49 shows the platform starting at a little more than nine inches below the Omni-Wrists while the simulation shows the platform starting at just more than ten inches below the wrists. Comparing Figures 50 and 51 with Figures 47 and 48 explains this discrepancy. The simulation assumes that the pitch angles start at a horizontal position; the assumption comes from the Omni-Wrist joint limits that state that the pitch joints can travel to this position. The hardware shows otherwise. Both pitch joints are fully rotated to hit their physical limit at the beginning of the cycle. However, Figures 50 and 51 show this position to be two degrees and four degrees away from zero for θ_2 and θ_3 , respectively. This could have been corrected in the hardware control program by adjusting the 45-degree offset. This adjustment has been left out in light of a suggestion for future work that places the robot in a home position.

This chapter has presented results for the verifying the solution of the SPAM robot's kinematic problems. Examples were given using trajectory generations for the forward and inverse pose kinematics problems as well as the inverse rate kinematics

problem. Simulations were also presented to verify both the control of a single joint of the Omni-Wrist alone as well as the control of the plunge motion. A hardware example for the plunge motion was demonstrated for comparison to the simulation. This comparison and verification of the solution of the kinematics problems have shown the control of the SPAM system to be successful.

Figure 49. Hardware Plunge Position Control

Figure 50. Hardware Control of θ_2 for Plunge Motion

Figure 51. Hardware Control of θ_5 for Plunge Motion

CHAPTER 6

CONCLUSION

The SPherically-Actuated platform Manipulator (SPAM) has been presented in this thesis. This final chapter will present concluding statements along with ideas for further developing the research that the author has begun. The conclusion reflects back on the work done on the manipulator. The future work is concerned with improving the performance and analysis of the robot.

6.1 Conclusion

Chapter 2 of this thesis presented the hardware and software for the SPAM system. The objectives related to this part of the study were to design a manipulator that fit the scale of the Omni-Wrists. This included designing the links, a base, and a platform. The manipulator was then to be linked to a personal computer via the MultiQ-3 I/O boards. These boards interface with Matlab and Simulink, which are the software tools used to operate the robot. The programming objective was to create programs that were user-friendly and easily understood.

Section 2.1.1 presented the structure of hardware, discussing the novelty of the design, which lies in the spherical actuation. The mobility of the manipulator was discussed along with its joint limits. Details concerning material used for the links as well as the structure of the joints were also addressed. Section 2.1.2 introduced the Omni-Wrist as the spherical actuator for the manipulator and the complications

associated with the mechanical coupling and the offset between the wrist's universal joints. The section also provided justification for using the Omni-Wrist as a spherical actuator, stating that it could handle the moment loading required of such an actuator. The wrist's physical parameters were further discussed, presenting the gear ratios for each joint of the wrist and data for the motors used to drive those joints. The remainder of Section 2.1 focused on the electronics for operating the Omni-Wrists and the MultiQ-3 boards. This area dealt mainly with the amplifiers used for the motors as well as the feedback system. The position feedback is provided by a resolver on each motor. The resolver sends a signal to an encoder emulator board mounted on the amplifier that alters the signal and sends it to the MultiQ-3 boards. The boards connect with the system software, which was discussed in Section 2.2.

Matlab, Simulink and WinCon were presented as the software programs used to operate SPAM. Simulink and WinCon provide an easy way to program using block diagrams. Simulink can be used without WinCon to simulate the system's performance; adding the WinCon software allows the user to connect to the real world system. The Real-Time Workshop provided by Matlab must also be installed for real world applications. Although limitations were found with this setup (due mainly to the Real-Time Workshop), the software is adequate and the programming has been successful. Proper position control of the Omni-Wrist has been obtained.

Chapter 3 presented the methods for solving the forward and inverse pose and inverse rate kinematics problems for both the *SRU* and *SPU* versions of the SPAM manipulator. The objectives for this portion of the study were to solve these problems for

a robot that has a spherical joint at the base of a serial chain, as opposed to the common placement at the end of a serial chain. Methods for the latter have been developed on numerous occasions while methods for the former had not been previously presented.

Due to the complications presented by the Omni-Wrist, the Newton-Raphson numerical method was used to solve both pose problems for the manipulator. The analysis showed that what was to be a nominal configuration for the *SRU* version of the robot was actually a singular one that had to be avoided. The numerical method's sensitivity to the initial guess was also discovered, as the Newton-Raphson procedure required a very good initial guess for convergence. Aside from these limitations, the numerical method proved successful in solving the pose kinematics problems. With the pose problems solved, analytical methods were developed to successfully solve the inverse rate kinematics problems. In addition, an analytical singularity analysis for the *SPU* version of the robot was discussed. The *SPU* SPAM does not suffer from the same nominal-configuration-singularity as the *SRU* SPAM.

The efforts put forth to control SPAM are described in Chapter 4. The chapter presented the theory behind designing a controller for a single joint/link system. Described in this section was the method for determining the gains for a PID controller and the procedure for adjusting those gains to produce the desired performance from the system. This followed with discussions of Simulink programs used for controlling first the pitch joint of the Omni-Wrist and then a Cartesian plunge motion of the platform. The plunge motion simplified the manipulator to a one-degree-of-freedom mechanism that could be controlled with the present hardware. Controlling this motion involved

coupling the forward and inverse kinematics solutions with the independent joint control to produce a desired motion from the manipulator. Control of the plunge motion has proven successful and is adequate for demonstrating a sample of the manipulator's capabilities.

Chapter 5 displayed and discussed results for all of the procedures relating to the *SRU* version of the manipulator as presented in Chapters 3 and 4. The pose kinematics problems were simulated to validate the numerical procedure. This followed with a simulation of the rate kinematics problem. Section 5.2 presented the results for controlling the manipulator, beginning with the controller design. The controller was adjusted to achieve a desired performance from the single joint/link system and then implemented with the plunge motion program. This program was simulated and performed satisfactorily. The chapter concludes with results of controlling the plunge motion of the hardware.

This thesis has presented the process of conceptualizing a novel robot architecture and converting the idea into a hardware prototype that was simulated and controlled. All objectives as set forth at the beginning of the research have been met. Included in this process was examination of relative theories and methods used to develop the *SP*herically-Actuated platform Manipulator. The idea has become a functioning, controllable robot with the following pros and cons:

Pros:

- High stiffness and accuracy
- Fast movement

- Low weight
- Large workspace
- High load-bearing capability

Cons:

- Singular position in center of workspace with *SRU* version
- Moment loading on Omni-Wrists

6.2 Future Work

This section will present ideas that may be used to further the development of the SPAM robot. The improvements focus mainly on the structure of the manipulator.

Several ideas for improvement have come about throughout the course of this work. These improvements are almost exclusively in the architecture of the robot. The most prominent concern for position control of the robot is to have all joints fully functional. To do this, six amplifiers and encoder emulator boards must be used. The robot is currently using the only two functioning amplifiers to produce the plunge motion. There are two other amplifiers housed in the electronics cabinet that seem to have bad encoder emulator boards on them. The author exchanged encoder emulator boards between a known good amp and an amp that was not functioning properly. This exchange allowed the bad amp to work properly. The wiring has been inspected and all of the connections are secure. A Pacific Scientific representative commented that it would be possible that the boards are bad and need to be replaced.

In addition to those two amps, Dr. Williams has purchased two amplifiers and an accompanying power supply from Servo Systems and placed them in the Ohio University Robotics/Haptics Lab. These need to be connected and installed in the electronics cabinet. The controllers in the top portion of the cabinet may be removed to provide mounting space. However, these controllers are fully functional and should be stored in a safe place. With six functioning amplifiers, full Cartesian control of the SPAM manipulator would be possible. The manipulator would then be capable of achieving positions like the one shown in Figure 52.

Figure 52. Example of Full Cartesian Positioning of SPAM

Several structural changes to the manipulator have been explored. The most researched and promising of these is the *SPU* version of the manipulator. Dr. Williams has done extensive work in developing the kinematics for this architecture, which replaces the passive revolute elbow joint with a passive prismatic joint. The main advantage of this architecture is the elimination of the singularity that occurs in the

nominal position. The kinematic analysis is also somewhat simpler than the *SRU* version. In addition to this work, Dr. Williams has explored the idea of using a truly spherical wrist (without the Omni-Wrist offset) for actuating the platform.

The kinematic diagram for the *SPU* SPAM robot using a truly spherical wrist is shown in Figure 17 in Chapter 3. This design is much simpler than the previously discussed *SPU* version of SPAM due to the elimination of the offset in the wrist. Dr. Williams has developed methods for solving the kinematics problems for this architecture and has presented them in a conference paper (Williams and Poling, 2000).

A third variation of the manipulator involves adding a third, completely passive leg to the manipulator. This third leg may add some rigidity to the platform, although it could limit the workspace at the same time. The initial idea for the manipulator was to be a three-legged robot, but the third leg was dropped because the two leg version is simpler and hence better for practical implementation. This may still be the case, but the addition of another leg could be beneficial for some applications. The platform and base would then be triangular in shape and the singularity condition present with the *SRU* SPAM robot would be avoided (if passive revolute joints are at the elbow; otherwise, the singularities do not exist).

In addition to the structural improvements, the software could be improved in largely two areas. The first improvement involves the Matlab Real-Time Workshop (RTW). It currently does not recognize S-functions created in Simulink. This causes the user to develop a series of simpler function blocks that the RTW recognizes. If future versions of the RTW can handle S-functions, it would allow for a cleaner control

program. The other software-related improvement concerns developing the program for controlling full Cartesian motion. With all six motors operational, the control program would have to be revised to replace the forward and inverse kinematics subsystems for the plunge motion with that for full Cartesian motion. The S-function block would be a great help in doing this if future versions of the RTW can handle it.

Another lesser issue that involves both the hardware and software is creating a home position for the manipulator. Currently, the manipulator begins with the two bottom links near horizontal and pointing at one another. This position is zero for the roll and yaw angles, but it is approximately 45° from zero for the pitch angles. Because the resolvers begin at zero when powered on, a method to position the manipulator at its true zero position should be developed once all six motors are operational.

Several improvements for future work on the SPAM manipulator have been discussed. These improvements could possibly transform the SPAM manipulator from a research project involving a hardware prototype used for experimentation and visualization into a reliable parallel platform manipulator. Spherical actuation is a developing concept that may have unrealized potential which could also further this research and the research of platform manipulators in general.

This thesis has presented the hardware, software, theory and implementation for the SPHERICALLY-ACTUATED platform Manipulator. The objectives of the thesis have been met and thoroughly explored. Suggestions for furthering this research have been provided to improve the performance of the SPAM robot.

REFERENCES

- Chaware, P. R. & C. Amarnath (1984). Novel Wrist Joint Design for Manipulators. *International Symposium on Design and Synthesis*. 479-482.
- Craig, J. J. (1989). Introduction to Robotics: Mechanics and Control (2nd ed.). Reading, Massachusetts: Addison-Wesley Publishing Co.
- Dafaoui, El-Mouloudi, Yacine Amirat, & Jean Pontnau (1998). Analysis and Design of a Six-DOF Parallel Manipulator, Modeling, Singular Configurations, and Workspace. *IEEE Transactions on Robotics and Automation*, Volume 14 (February, 1998). 78-92.
- Dasgupta, Bh. & T. S. Mruthyunjaya (1998). Newton-Euler Formulation for the Inverse Dynamics of the Stewart Platform Manipulator. *Mechanism & Machine Theory*, Volume 33, No. 8. 1135-1152.
- Dorf, R. C. & R. H. Bishop (1998). Modern Control Systems (8th ed.). Menlo Park, California: Addison-Wesley Publishing Co.
- IMEC Corporation (1987). 400 Series Brushless Servo Controller Instruction Manual. Encoder Emulator Card Insert.
- Ji, Zhiming & Philip Song (1998). Design of a Reconfigurable Platform Manipulator. *Journal of Robotic Systems*, Volume 15, No. 6. 341-346.
- Lbret, G., K. Liu, & F. L. Lewis (1993). Dynamic Analysis and Control of a Stewart Platform Manipulator. *Journal of Robotic Systems*, Volume 10, No. 5. 629-655.
- Lee, Kok-Meng & Chi-Kong Kwan (1991). Design Concept Development of a Spherical Stepper for Robotic Applications. *IEEE Transactions on Robotics and Automation*. Volume 7, No. 1. 175-181.
- Leguay-Durand, Sylvie & Claude Reboulet (1997). Optimal Design of a Redundant Spherical Parallel Manipulator. *Robotica*, Volume 15, No. 4. 399-405.
- Liu, K., F. L. Lewis, & M. Fitzgerald (1994). Solution of Nonlinear Kinematics of a Parallel-Link Constrained Stewart Platform Manipulator. *Circuits, Systems, and Signal Processing*, Volume 13, No. 2-3. 167-183.
- Liu, Min-Jie & Cong-Xin Li (1999). Dynamics Analysis of the Stewart Platform Manipulator. *Proceedings of the 1999 3rd International Symposium on Test and Measurement*. 920-923.

- Liu, K., F. L. Lewis, G. Lebet, & D. Taylor (1993). Singularities and Dynamics of a Stewart Platform Manipulator. *Journal of Intelligent and Robotic Systems*, Volume 8, No. 3. 287-308.
- Liu, K., J. M. Fitzgerald, & F. L. Lewis (1993). Kinematics analysis of a Stewart Platform Manipulator. *IEEE Transactions on Industrial Electronics*, Volume 40, No. 2. 282-293.
- Liu, K., M. Fitzgerald, & D. M. Dawson (1991). Modeling and Control of a Stewart Platform Manipulator. *American Society of Mechanical Engineers, Dynamic Systems and Control Division*, Volume 33. 83-89.
- MathWorks website (www.mathworks.com).
- Nguyen, Charles C. (1995). [Special issue on parallel manipulators: research, development, and application]. *Journal of Robotic Systems*, Volume 12 (December, 1995). 781-893.
- Nyzen, Robert J. (1999). Analysis and Control of an Eight Degree-of-Freedom Manipulator. Master's Thesis. Fritz J. and Dolores H. Russ College of Engineering and Technology, Ohio University. Athens, Ohio.
- Quanser Consulting, Inc. (1999) MultiQ-3 Programming Manual. Ontario, Canada.
- Rosheim, M. E. (1990). Final Report on Further Development of Computer Controlled Telerobot Wrist Module. NASA SBIR Contract, NASA-18673.
- Rosheim, M. E. (1985). New Pitch-Yaw-Roll Mechanical Robot Wrist Actuator. *Robots 9, Conference Proceedings*, Volume 2. 15.20 – 15.42.
- Roth, R. B. & Kok-Meng Lee (1995). Design Optimization of a Three Degrees-of-Freedom Variable-Reluctance Spherical Wrist Motor. *Journal of Engineering for Industry, Transactions of the ASME*, Volume 117, No. 3. 378-388.
- Sasae, K., K. Ioi, Y. Ohtsuki, & Y. Kurosaki (1996). Development of a Small Actuator with Three Degrees of Rotational Freedom (3rd report) – Design and Experiment of a Spherical Actuator. *Journal of the Japan Society for Precision Engineering*, Volume 62, No. 4. 599-603.
- Stroughton, R. S. & T. Arai (1993). Modified Stewart Platform Manipulator with Improved Dexterity. *IEEE Transactions on Robotics and Automation*, Volume 9, No. 2. 166-172.

- Toyama, Shigeki (1996). Development of Spherical Ultrasonic Motor. *CIRP Annals – Manufacturing Technology*, Volume 45, No. 1. 27-30.
- Tsai, Lung-Wen (1999). Robot Analysis: The mechanics of Serial and Parallel Manipulators. New York, New York: John Wiley & Sons, Inc.
- Wang, Jiabin, Weiya Wang, G.W. Jewell, & D. Howe (1998). Novel Spherical Permanent Magnet Actuator with Three Degrees-of-Freedom. *IEEE Transactions on Magnetics*, Volume 34, No. 4, part 1. 2078-2080.
- Williams, R. L., II (1985) Synthesis and Design of the RSSR Spatial Mechanism for Function Generation. Appendix D. – Rooting a Quartic in Closed Form. Master's Thesis, Virginia Polytechnic Institute and State University. Blacksburg, Virginia.
- Williams, R.L., II. (1990). Forward and Inverse Kinematics of Double Universal Joint Robot Wrists. Presented at the Space Operations, Applications and Research (SOAR) Symposium. June 26-28. Albuquerque, New Mexico.
- Williams, R. L., II (1992) Kinematics of an In-Parallel Actuated Manipulator Based on the Stewart Platform Mechanism. NASA Technical Memorandum 107585. NASA Langley Research Center.
- Williams, R. L., II (1999) Inverse Kinematics and Singularities of Manipulators with Offset Wrist. *IASTED International Journal of Robotics and Automation*, Volume 14, No. 1. 1-8.
- Williams, R. L., II (2000) ME 401 Class Notes.
<http://www.ent.ohiou.edu/~bobw/html/Courses.html>
- Williams, R. L., II & D. B. Poling (2000). Spherically-Actuated Platform Manipulator. Submitted to the 26th ASME Mechanisms Conference, Baltimore, Maryland.
- Williams, R. L., II & D. B. Poling (2000). Spherically-Actuated Platform Manipulator. Submitted to the Journal of Mechanical Design.
- Wincon 3.0.2a [computer program]. (1999). Quanser Consulting, Inc. Ontario, Canada

APPENDIX A

SRU SPAM CAD DRAWINGS

This appendix presents the Solid Edge drawings developed by the author for construction of the SPAM manipulator. These drawings were given to Tom Chambers of Ohio University, who machined the parts and helped in the construction of the manipulator.

PART NAME:		FIN	
MATERIAL	ALUMINUM 6061	SCALE	2:1
DESIGNED BY	DANA POLING	DATE	10/20/99
MECHANICAL ENGINEERING DEPARTMENT - OHIO UNIVERSITY

PART NAME		U-JOINT FIN	
MATERIAL	ALUMINUM 6061	SCALE	2:1
DESIGNED BY	DANIA POLING	DATE	10/20/23
APPROVALS			

APPENDIX B

OPERATION PROCEDURE

This appendix discusses the operation procedure for the SPAM manipulator. The methods for both starting and shutting down the robot are presented.

Start-up:

1. Make sure that the power cord is plugged in to the outlet.
2. Open the top of the electronics cabinet. Verify that the three connectors that are wired together are connected the three jumper connectors. If the jumper connectors are not plugged in, take them from under the control panel on the electronics box and plug them in.
3. Ensure that the switch on the right side of the door to the electronics cabinet is turned to "On."
4. Plug in all plugs necessary for communication between the encoder emulator boards, motors, and MultiQ-3 boards. Currently, only two of these are needed to operate the two motors in use. The author's programs are configured so that the pitch joint of the left wrist is Channel 0 on the 340 MultiQ-3 board and the pitch joint of the right wrist is Channel 1 on that same board. Ensure that both the encoder plug and analog output plug are connected for each motor.
5. On the control panel, press the "Power On" button, turn the selector switch from "Off" to "On", and raise the "E-Stop" button.
6. Power on the personal computer that the MultiQ-3 boards are connected to.

7. Start Matlab 5.2. If any changes are to be made to a Simulink diagram, type “wc_setoptions” to initialize the build properties. Also, type “winlib” if the WinCon block library is needed.
8. Once Matlab has started, start Simulink by either typing “simulink” at the prompt or selecting the Simulink icon at the top center of the Matlab window.
9. Open the desired Simulink program.
10. Use the WinCon pull-down menu to build the program once the program is complete. The software will then generate all necessary code to run the program.
11. Once all the code is generated, a window appears. The window contains a “Start” button and buttons that allow viewing of any plots. Select any plots that are to be viewed.
12. Open the system inputs box and enter the initial configuration for the robot.

These inputs should represent the configuration that the robot begins in to avoid any motion at startup.
13. To operate the robot, press the start button on the WinCon window and wait until the button turns to a red “STOP” button. Switch the Omni-Wrist emergency switch (located next to the personal computer) to “ON.” This final switch allows the motors to run and receive commands from the Simulink program.
14. Leaving the program running, the user may change the position of the Omni-Wrists relative to the start position by changing the system inputs. If the simulation is stopped, the robot loses its start position and restarts as if the last position were zero.

Shut-down:

1. To stop the real-time control, switch the emergency switch next to the computer to “OFF.” Press the “STOP” button on the WinCon window.
2. Close the WinCon window and shut down Simulink and Matlab, making sure that any changes to the Simulink diagram have been saved (if desired).
3. Press the “E-Stop” button on the control panel of the electronics box. Switch the selector switch to “Off” and press the “Off” switch.
4. For added safety, the switch on the door of the electronics cabinet should be turned off, the three jumper connectors should be unplugged and placed inside the electronics cabinet under the control panel, and the power line should be unplugged.

APPENDIX C

KINEMATICS TERMS

Presented here are the kinematics terms for the SPAM manipulator. The functions for forming the Newton-Raphson Jacobians of Chapter 3 are presented along with their substitution terms. To form the functions for the forward pose Newton-Raphson Jacobian, the terms for Equation (8) are:

$$\begin{aligned} X_1 &= -((-K_A sp_2 + K_B cp_2)cp_3 + (-K_1 s_3 + K_2 c_3)sp_3)L_5 + K_A L_4 + K_6 L_3 + K_1 L_2 \\ Y_1 &= -((-K_C sp_2 + K_D cp_2)cp_3 + (-K_3 s_3 + K_4 c_3)sp_3)L_5 + K_C L_4 + K_8 L_3 + K_3 L_2 - L_1 \\ Z_1 &= -((-K_E sp_2 + K_F cp_2)cp_3 - 2c_2 c_3 s_3 sp_3)L_5 + K_E L_4 + K_{10} L_3 + c_2 c_3 L_2 \end{aligned}$$

The rotation elements used in Equation (9) are:

$$\begin{aligned} r_{12} &= -(-K_A sp_2 + K_B cp_2)cp_3 - (-K_1 s_3 + K_2 c_3)sp_3 \\ r_{22} &= -(-K_C sp_2 + K_D cp_2)cp_3 - (-K_3 s_3 + K_4 c_3)sp_3 \\ r_{33} &= K_E cp_2 + K_F sp_2 \end{aligned}$$

The substitution values used in the above equations are:

$$\begin{aligned} K_1 &= s_1 s_2 c_3 + c_1 s_3 \\ K_2 &= -s_1 s_2 s_3 + c_1 c_3 \\ K_3 &= -c_1 s_2 c_3 + s_1 s_3 \\ K_4 &= c_1 s_2 s_3 + s_1 c_3 \\ K_5 &= c_2 c_3^2 - c_2 s_3^2 \\ K_6 &= (K_1 c_3 + K_2 s_3)c_2 + s_1 c_2 s_2 \\ K_7 &= -(K_1 c_3 + K_2 s_3)s_2 + s_1 c_2^2 \\ K_8 &= (K_3 c_3 + K_4 s_3)c_2 - c_1 c_2 s_2 \\ K_9 &= -(K_3 c_3 + K_4 s_3)s_2 - c_1 c_2^2 \\ K_{10} &= K_5 c_2 - s_2^2 \\ K_{11} &= -K_5 s_2 - s_2 c_2 \\ K_A &= K_6 cp_1 + K_7 sp_1 \\ K_B &= -K_6 sp_1 + K_7 cp_1 \\ K_C &= K_8 cp_1 + K_9 sp_1 \\ K_D &= -K_8 sp_1 + K_9 cp_1 \\ K_E &= K_{10} cp_1 + K_{11} sp_1 \\ K_F &= -K_{10} sp_1 + K_{11} cp_1 \end{aligned}$$

To form the functions defined in Equation (10), the appropriate variable replacements are needed in the above equations to change the variables representing the left leg to those representing the right leg. In addition, the sign of L_1 in the Y_1 equation must be changed to positive when dealing with the right leg. These equations allow the forming of the functions used for the forward pose Newton-Raphson Jacobian. From Equation (12), the terms used in forming the Newton-Raphson Jacobian are:

$$\begin{aligned}
J_{1,4} &= -(-K_Bsp_2 + (-K_6cp_1 - K_7sp_1)cp_2)cp_3L_5 + K_BL_4 \\
J_{2,4} &= -(-K_Dsp_2 + (-K_8cp_1 - K_9sp_1)cp_2)cp_3L_5 + K_DL_4 \\
J_{3,4} &= -(-K_Fsp_2 + (-K_{10}cp_1 - K_{11}sp_1)cp_2)cp_3L_5 + K_FL_4 \\
J_{4,4} &= -(-K_Bsp_2 + (-K_6cp_1 - K_7sp_1)cp_2)cp_3 \\
J_{5,4} &= -(-K_Dsp_2 + (-K_8cp_1 - K_9sp_1)cp_2)cp_3 \\
J_{6,4} &= K_Fcp_2 + (-K_{10}cp_1 - K_{11}sp_1)sp_2 \\
J_{1,5} &= -(-K_Acp_2 - K_Bsp_2)cp_3L_5 \\
J_{2,5} &= -(-K_Ccp_2 - K_Dsp_2)cp_3L_5 \\
J_{3,5} &= -(-K_Ecp_2 - K_Fsp_2)cp_3L_5 \\
J_{4,5} &= -(-K_Acp_2 - K_Bsp_2)cp_3 \\
J_{5,5} &= -(-K_Ccp_2 - K_Dsp_2)cp_3 \\
J_{6,5} &= -K_Esp_2 + K_Fcp_2 \\
J_{1,6} &= -(-(-K_Asp_2 + K_Bcp_2)sp_3 + (-K_1s_3 + K_2c_3)cp_3)L_5 \\
J_{2,6} &= -(-(-K_Csp_2 + K_Dcp_2)sp_3 + (-K_3s_3 + K_4c_3)cp_3)L_5 \\
J_{3,6} &= -(-(-K_Esp_2 + K_Fcp_2)sp_3 - 2c_2c_3s_3cp_3)L_5 \\
J_{4,6} &= (-K_Asp_2 + K_Bcp_2)sp_3 - (-K_1s_3 + K_2c_3)cp_3 \\
J_{5,6} &= (-K_Csp_2 + K_Dcp_2)sp_3 - (-K_3s_3 + K_4c_3)cp_3 \\
J_{4,7} &= c\alpha\gamma + s\alpha s\beta s\gamma \\
J_{5,7} &= s\alpha c\gamma - c\alpha s\beta s\gamma \\
J_{4,8} &= -c\alpha c\beta s\gamma \\
J_{5,8} &= -s\alpha c\beta s\gamma \\
J_{6,8} &= s\beta c\gamma \\
J_{4,9} &= -s\alpha s\gamma - c\alpha s\beta c\gamma \\
J_{5,9} &= c\alpha s\gamma - s\alpha s\beta c\gamma \\
J_{6,9} &= c\beta s\gamma \\
J_{7,10} &= (-KK_Bsp_5 + (-KK_6cp_4 - KK_7sp_4)cp_5)cp_6L_5 + KK_BL_4 \\
J_{8,10} &= (-KK_Dsp_5 + (-KK_8cp_4 - KK_9sp_4)cp_5)cp_6L_5 + KK_DL_4 \\
J_{9,10} &= (-KK_Fsp_5 + (-KK_{10}cp_4 - KK_{11}sp_4)cp_5)cp_6L_5 + KK_FL_4 \\
J_{10,10} &= -(-KK_Bsp_5 + (-KK_6cp_4 - KK_7sp_4)cp_5)cp_6 \\
J_{11,10} &= -(-KK_Dsp_5 + (-KK_8cp_4 - KK_9sp_4)cp_5)cp_6 \\
J_{12,10} &= KK_Fcp_5 + (-KK_{10}cp_4 - KK_{11}sp_4)sp_5
\end{aligned}$$

$$\begin{aligned}
J_{7,11} &= (-KK_A cp_5 - KK_B sp_5) cp_6 L_5 \\
J_{8,11} &= (-KK_C cp_5 - KK_D sp_5) cp_6 L_5 \\
J_{9,11} &= (-KK_E cp_5 - KK_F sp_5) cp_6 L_5 \\
J_{10,11} &= -(-KK_A cp_5 - KK_B sp_5) cp_6 \\
J_{11,11} &= -(-KK_C cp_5 - KK_D sp_5) cp_6 \\
J_{12,11} &= -KK_E sp_5 + KK_F cp_5 \\
J_{7,12} &= (-(-KK_A sp_5 + KK_B cp_5) sp_6 + (-KK_1 s_6 + KK_2 c_6) cp_6) L_5 \\
J_{8,12} &= (-(-KK_C sp_5 + KK_D cp_5) sp_6 + (-KK_3 s_6 + KK_4 c_6) cp_6) L_5 \\
J_{9,12} &= (-(-KK_E sp_5 + KK_F cp_5) sp_6 - 2c_5 c_6 s_6 cp_6) L_5 \\
J_{10,12} &= (-KK_A sp_5 + KK_B cp_5) sp_6 - (-KK_1 s_6 + KK_2 c_6) cp_6 \\
J_{11,12} &= (-KK_C sp_5 + KK_D cp_5) sp_6 - (-KK_3 s_6 + KK_4 c_6) cp_6
\end{aligned}$$

In the above terms, variables beginning with “KK” represent substitution terms that use the substitutions for right leg angles in place of left leg angles. These terms form the forward pose Newton-Raphson Jacobian, as shown on page 30. To form the Jacobian for inverse pose kinematics as shown on page 32, the same functions are used as before. The entries into the 6x6 matrix for the left leg are:

$$\begin{aligned}
J_{1,1} &= -r_{22} L_5 - K_C L_4 + K_8 L_3 - K_3 L_2 \\
J_{2,1} &= r_{12} L_5 + K_A L_4 + K_6 L_3 + K_1 L_2 \\
J_{4,1} &= -r_{22} \\
J_{5,1} &= -(K_A sp_2 + K_B cp_2) cp_3 - (-K_1 s_3 + K_2 c_3) sp_3 \\
J_{1,2} &= -((- (K_{14} cp_1 + K_{15} sp_1) sp_2 + (-K_{14} sp_1 + K_{15} cp_1) cp_2) cp_3 - s_1 c_2 c_3 s_3 sp_3) L_5 \\
&\quad + (K_{14} cp_1 + K_{15} sp_1) L_4 + K_{14} L_3 + s_1 c_2 c_3 L_2 \\
J_{2,2} &= -((- (K_{16} cp_1 + K_{17} sp_1) sp_2 + (-K_{16} sp_1 + K_{17} cp_1) cp_2) cp_3 + 2c_1 c_2 c_3 s_3 sp_3) L_5 \\
&\quad + (K_{16} cp_1 + K_{17} sp_1) L_4 + K_{16} L_3 - c_1 c_2 c_3 L_2 \\
J_{3,2} &= -((- (K_{19} cp_1 + K_{20} sp_1) sp_2 + (-K_{19} sp_1 + K_{20} cp_1) cp_2) cp_3 + 2s_2 c_3 s_3 sp_3) L_5 \\
&\quad + (K_{19} cp_1 + K_{20} sp_1) L_4 + K_{19} L_3 - s_2 c_3 L_2 \\
J_{4,2} &= -((- (K_{14} cp_1 + K_{15} sp_1) sp_2 + (-K_{14} sp_1 + K_{15} cp_1) cp_2) cp_3 + 2s_1 c_2 c_3 s_3 sp_3 \\
J_{5,2} &= -((- (K_{16} cp_1 + K_{17} sp_1) sp_2 + (-K_{16} sp_1 + K_{17} cp_1) cp_2) cp_3 - 2c_1 c_2 c_3 s_3 sp_3 \\
J_{6,2} &= (K_{19} cp_1 + K_{20} sp_1) cp_2 + (-K_{19} sp_1 + K_{20} cp_1) sp_2 \\
J_{1,3} &= -((- (K_{22} sp_2 + K_{23} cp_2) cp_3 + (-K_1 c_3 - 2K_2 s_3 + (-s_1 s_2 c_3 - c_1 s_3) c_3) sp_3) L_5 \\
&\quad + K_{22} L_4 + K_{21} c_2 L_3 + K_2 L_2 \\
J_{2,3} &= -((- (K_{25} sp_2 + K_{26} cp_2) cp_3 + (-K_3 c_3 - 2K_4 s_3 + (c_1 s_2 c_3 - s_1 s_3) c_3) sp_3) L_5 \\
&\quad + K_{25} L_4 + K_{24} c_2 L_3 + K_4 L_2 \\
J_{3,3} &= -((- (K_{27} sp_2 + K_{28} cp_2) cp_3 + 2c_2 s_3^2 sp_3 - 2c_2 c_3^2 sp_3) L_5 + K_{27} L_4 - 4c_2^2 c_3 s_3 L_3 - c_2 s_3 L_2 \\
J_{4,3} &= -((- (K_{22} sp_2 + K_{23} cp_2) cp_3 - (-K_1 c_3 - 2K_2 s_3 + (-s_1 s_2 c_3 - c_1 s_3) c_3) sp_3 \\
J_{5,3} &= -((- (K_{25} sp_2 + K_{26} cp_2) cp_3 - (-K_3 c_3 - 2K_4 s_3 + (c_1 s_2 c_3 - s_1 s_3) c_3) sp_3 \\
J_{6,3} &= K_{27} cp_2 + K_{28} sp_2
\end{aligned}$$

$$\begin{aligned}
J_{1,4} &= -(-K_B sp_2 + (-K_6 cp_1 - K_7 sp_1) cp_2) cp_3 L_5 + K_B L_4 \\
J_{2,4} &= -(-K_D sp_2 + (-K_8 cp_1 - K_9 sp_1) cp_2) cp_3 L_5 + K_D L_4 \\
J_{3,4} &= -(-K_F sp_2 + (-K_{10} cp_1 - K_{11} sp_1) cp_2) cp_3 L_5 + K_F L_4 \\
J_{4,4} &= -(-K_B sp_2 + (-K_6 cp_1 - K_7 sp_1) cp_2) cp_3 \\
J_{5,4} &= -(-K_D sp_2 + (-K_8 cp_1 - K_9 sp_1) cp_2) cp_3 \\
J_{6,4} &= K_F cp_2 + (-K_{10} cp_1 - K_{11} sp_1) sp_2
\end{aligned}$$

$$\begin{aligned}
J_{1,5} &= -(K_A cp_2 - K_B sp_2) cp_3 L_5 \\
J_{2,5} &= -(K_C cp_2 - K_D sp_2) cp_3 L_5 \\
J_{3,5} &= -(K_E cp_2 - K_F sp_2) cp_3 L_5 \\
J_{4,5} &= -(K_A cp_2 - K_B sp_2) cp_3 \\
J_{5,5} &= -(K_C cp_2 - K_D sp_2) cp_3 \\
J_{6,5} &= -K_E sp_2 + K_F cp_2
\end{aligned}$$

$$\begin{aligned}
J_{1,6} &= -(-K_A sp_2 + K_B cp_2) sp_3 + (-K_1 s_3 + K_2 c_3) cp_3 L_5 \\
J_{2,6} &= -(-K_C sp_2 + K_D cp_2) sp_3 + (-K_3 s_3 + K_4 c_3) cp_3 L_5 \\
J_{3,6} &= -(-K_E sp_2 + K_F cp_2) sp_3 - 2c_2 c_3 s_3 cp_3 L_5 \\
J_{4,6} &= (-K_A sp_2 + K_B cp_2) sp_3 - (-K_1 s_3 + K_2 c_3) cp_3 \\
J_{5,6} &= (-K_C sp_2 + K_D cp_2) sp_3 - (-K_3 s_3 + K_4 c_3) cp_3
\end{aligned}$$

Where the new substitution terms are needed:

$$\begin{aligned}
K_{12} &= s_1 c_2 c_3^2 - s_1 c_2 s_3^2 \\
K_{13} &= -c_1 c_2 c_3^2 + c_1 c_2 s_3^2 \\
K_{14} &= K_{12} c_2 - (K_1 c_3 + K_2 s_3) s_2 - s_1 s_2^2 + s_1 c_2^2 \\
K_{15} &= -K_{12} s_2 - (K_1 c_3 + K_2 s_3) c_2 - 2s_1 c_2 s_2 \\
K_{16} &= K_{13} c_2 - (K_3 c_3 + K_4 s_3) s_2 + c_1 s_2^2 - c_1 c_2^2 \\
K_{17} &= -K_{13} s_2 - (K_3 c_3 + K_4 s_3) c_2 + 2c_1 c_2 s_2 \\
K_{18} &= -s_2 c_3^2 + s_2 s_3^2 \\
K_{19} &= K_{18} c_2 - K_5 s_2 - 2s_2 c_2 \\
K_{20} &= -K_{18} s_2 - K_5 c_2 - c_2^2 + s_2^2 \\
K_{21} &= (-K_1 s_3 + 2K_2 c_3 + (-s_1 s_2 c_3 - c_1 s_3) s_3) \\
K_{22} &= K_{21} c_2 cp_1 - K_{21} s_2 sp_1 \\
K_{23} &= -K_{21} c_2 sp_1 - K_{21} s_2 cp_1 \\
K_{24} &= (-K_3 s_3 + 2K_4 c_3 + (c_1 s_2 c_3 - s_1 s_3) s_3) \\
K_{25} &= (K_{24} c_2 cp_1 - K_{24} s_2 sp_1) \\
K_{26} &= (-K_{24} c_2 sp_1 - K_{24} s_2 cp_1) \\
K_{27} &= (-4c_2^2 c_3 s_3 cp_1 + 4c_2 c_3 s_3 s_2 sp_1) \\
K_{28} &= (4c_2^2 c_3 s_3 sp_1 + 4c_2 c_3 s_3 s_2 cp_1)
\end{aligned}$$

The 6x6 right leg inverse pose Newton-Raphson Jacobian is formed by changing the left leg angles to the right leg angles and minding the signs of L_1 and L_5 .