NUMERICAL ANALYSIS OF A REDUNDANT COMPLIANT SPATIAL MECHANISM

By

JEAN-FRANÇOIS AJIT KAMATH

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2005

This dissertation is dedicated to my parents for their support and patience through the years and to Dr. Carl D. Crane III for his guidance and invaluable input. It is also dedicated to my uncle and aunt for their insight and experience.

ACKNOWLEDGMENTS

The author acknowledges the Center for Intelligent Machines and Robotics at the University of Florida for providing a research assistantship in addition to the resources and faculty that enabled completion of this project. The Department of Energy is also acknowledged for its support via grant to the University of Florida.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

NUMERICAL ANALYSIS OF A REDUNDANT COMPLIANT SPATIAL MECHANISM

By

Jean-Francois Ajit Kamath

August 2005

Chair: Carl D. Crane, III
Major Department: Mechanical and Aerospace Engineering

This thesis presents the development and testing of a generalized controls strategy that was applied to a redundant spatial mechanism. Three systems were analyzed: a planar, non-redundant, non-compliant mechanism, a redundant, compliant spatial mechanism, and a modification of the spatial mechanism which changed the configuration slightly. The planar mechanism was used to verify the controls strategy and general approach. The modified spatial mechanism was used to investigate the effects that changes in geometry would have on rotational performance.

Simulations of the systems and their controllers were performed in MATLAB 6.5 using SimuLink and the SimMechanics package. A numerical forward analysis was developed as a flexible and simple solution to the problem of position determination. This analysis can be readily adapted to virtually any parallel mechanism, regardless of configuration or number of legs. Control of the mechanisms was accomplished by calculating the net wrench that needed to be applied to the end effector, then performing a

conversion to determine the required forces in the individual actuators that will generate the desired motion.

The controls strategy worked well, effectively decoupling the motions of the system along its various degrees of freedom. While the coupling could not be completely eliminated, the disturbances that resulted were extremely small and had little to no impact on system performance. A method was developed to decouple the compliant elements from the legs, allowing a wide range of leg masses, spring rates, and damping rates to be used in conjunction with the controls strategy that was implemented.

This research demonstrated that even relatively simple methods can be used to effectively control highly cross coupled systems. It also showed that an effective process for developing a controller is to separate the system into smaller pieces that can be analyzed easily. Creating a modular controller is a versatile approach to solving a wide range of problems. Finally, this project revealed that redundancy in a parallel mechanism may be highly desirable as it can dramatically improve the performance of the system by expanding the useable workspace.

CHAPTER 1
INTRODUCTION

Research into control of parallel mechanisms is an ongoing process that still presents many challenges due to the dynamic nature of the systems. The main goal of this project was to develop a generalized control methodology to investigate the dynamic response of an over-constrained, spatial mechanism that contained compliant elements. The system was a modified 3-3 mechanism that incorporated an additional central leg for potentially added stability. Motion was generated using force actuators that were integrated with the controllers.

The basis for many of the analyses in this project came from Duffy [1], Crane and Duffy [2], and Baiges-Valentin [3]. The first reference, *Statics and Kinematics with Applications to Robotics*, focused on the analysis of planar mechanisms in both static and dynamic situations, including configurations requiring compliance. It served as a valuable resource when considering force applications.

Kinematic Analysis of Robot Manipulators was indispensable since it laid out all of the methods for position analysis of parallel mechanisms as well as providing the force convention used to determine the behavior of the systems. Forward and reverse analyses for both serial and parallel mechanisms were explained and were critical in development of the positional analysis in this project. The application of forces and wrenches to bodies and platforms, as discussed in this text, were needed for the proper development of the controllers.

Dynamic Modeling of Parallel Manipulators investigated the creation of a generic, compliant leg model and analysis of its dynamic behavior. The paper laid out several simplifications and adjustments to the model that were applicable to the compliant elements in this project, and enabled development of the decoupling controllers, as discussed later.

Lee et al. [4], Zhang et al. [5, 6], Lee [7], and Zhang [8] investigated the quality indices and inverses of the line matrices of various parallel mechanisms. They showed that behavior of parallel systems can only be effectively predicted and controlled within a limited workspace due to the development of linear dependence amongst the legs. Beyond a certain range, these mechanisms become "ill-conditioned", making them very difficult to control as they approach singularity configurations.

Geometric Analysis of Parallel Mechanisms investigated several parallel mechanisms with varying degrees of freedom. It discussed limitations of the systems and performed singularity analyses, once again showing restrictions on mobility of this class of devices. Tyler [10] and Abbasi et al. [11] looked at force control applications for parallel mechanisms and investigated the use of compliance to facilitate the process. They applied PID controllers to the systems and showed that even the simple control scheme could be used effectively. This was part of the inspiration for the control strategy implemented in this project.

This thesis is divided into several key sections that progress from general approaches to specific applications. While several systems were analyzed, most served to test specific aspects of the overall controls approach that was applied to the primary spatial mechanism. Chapter 2 discusses the general approach to designing the controllers

and critical assumptions and considerations for the systems that were analyzed. In Chapter 3, the numerical forward analysis, critical for positioning information, is developed and tested. Chapter 4 goes into the specifics of designing the controllers for a generic parallel mechanism without compliance. Chapter 5 shows a specific application of this controls approach to a planar mechanism as a proof of concept. Chapter 6 introduces the full spatial mechanism, along with compliant elements and discusses additional considerations for controller development. Chapter 7 develops a method for decoupling the compliant elements from the system so that the mechanism will respond similarly to the non-compliant systems. Chapter 8 shows how the obstacle of over-constraint was solved to enable proper application of the control forces. Chapter 9 discusses the results of testing on the spatial mechanism, as well as a slight modification of the system configuration. Finally, Chapter 10 presents final thoughts and areas with future research potential.

CHAPTER 2
GENERAL CONTROL STRATEGY

Control of a parallel mechanism is a fairly complicated task since the behaviors of the legs are highly cross-coupled by nature. This necessitates certain simplifications to the analysis of the kinematics and dynamics of the system. This chapter discusses the general approach and assumptions made in the design of the controllers for the mechanisms analyzed. Some of the potential errors that could be introduced into the systems due to those simplifications will also be examined.

Generic Controller Design

Any feedback controller follows a basic format that consists of several parts:

- Gather information on the current behavior of the system.
- Compare this to the desired behavior.
- Calculate and apply corrections to the system.

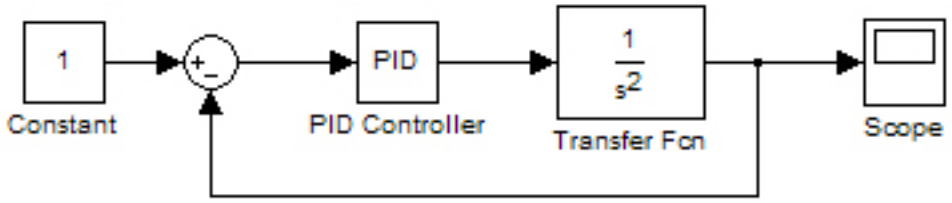


Figure 2-1. Block diagram representation of a classical feedback controller.

A generic controller is depicted in Figure 2-1. While this is a very simple concept that is familiar to all engineers, it highlights a key consideration when designing any controller: What information do you need to record and how do you obtain it? In the case of the mechanisms analyzed in this project, information on the current positioning of the system is critical. However, gathering this data is not trivial in many circumstances.

The raw data that could be most easily gathered for the parallel mechanisms was the lengths of the legs. One might remember the discussion in Chapter 1 about the limitations of parallel mechanisms, more specifically regarding the nature of forward analyses. Parallel manipulators can be especially bothersome when attempting to perform a forward analysis due to the mathematical complexity and the potentially large number of solutions. Since one of the objectives of this research was to develop a flexible control strategy that could be easily adapted to many devices, an analytical forward analysis became unreasonable. As a result, an extremely powerful, highly adaptable numerical approach was created to overcome this obstacle. The numerical analysis strategy is discussed in Chapter 3.

Decoupling the Legs

One of the biggest hurdles when designing the controllers was effectively decoupling the legs from each other. A simplification that is often used with parallel devices is the assumption that the masses and inertias of the legs and actuators are extremely small as compared with the mass and inertia of the end effector. The primary advantage of this technique is the reduction of the system to a single object, making the behavioral analysis far easier. However, as the masses of the other components increase with respect to the end effector, errors in the analysis become more apparent. This naturally brings up the question of, "What is small?" Unfortunately, the answer depends on the system in question as well as the application. To minimize errors, the models used leg masses that were five to seven orders of magnitude less than that of the end effector.

Other Assumptions

Several other assumptions were made during the course of the project:

- The first and probably most important was that the effects of the cross coupling would be essentially eliminated by setting the leg masses to low values. The geometric cross-coupling was considered to be negligible during controller design.

- The actuators were modeled to have "perfect" behavior and therefore have no contribution to errors in the system. Perfect behavior assumes instantaneous response to control inputs with no error in the response of the component.

- Springs and dampers were assumed to be ideal, linear components with no degradation in performance regardless of conditions.

- All bodies in the system were considered rigid. Any flexible elements were modeled as multi-part components. This is discussed in detail in Chapter 7

CHAPTER 3
NUMERICAL FORWARD ANALYSIS

Developing a generalized, numerical forward analysis for parallel mechanisms was a challenging, but highly rewarding experience. What developed was a novel approach that was extremely flexible and easily adaptable to the analysis of virtually any parallel system, regardless of configuration or number of legs.

Numerical vs. Analytical: Similarities in Approach

As discussed in Chapter 1, any analytical, closed-form solution to the forward positioning problem of a parallel mechanism is a tedious and rather complicated process. The more flexible the solution, the more complicated the mathematics behind it must be. However, the numerical approach closely follows the logic of the analytical solution in a couple of respects.

Geometric Constraints

The first considerations in any forward analysis are the physical constraints and limitations of the system. The relative positions and sizes of the components as well as their possible motions all must be satisfied in the solution. This is the most important restriction that inherently limits the solutions, both analytical and numerical.

Previous Positioning

Since the goal of the forward position analysis is to determine the current positioning of the system to feed into the controller, only one valid answer exists to the question of "Where am I?" These platforms generally do not move at extreme speeds so it is safe to assume that at a high enough sampling rate, the current positioning of the end

effector will be relatively close to its configuration at the previous sample. This means that we can use the previously known position information to determine which solution is correct. In the case of analytical approaches, this simply selects the proper solution from all possibilities. The numerical method instead used the previous positioning as a guess at the solution and then homes in on the correct answer.

General Numerical Solution

Any typical numerical solution to a problem involves minimization of some error function that describes how close the solver is to the desired answer. Corrections to the guess are made based on some relevant algorithm or set of equations, eventually leading to an answer that is within a specified range of the true results.

For the numerical forward analysis of any parallel platform, three things are needed: a guess at the current positioning, an error function based on the actual positioning, and a correctional function that will allow the solver to home in on the proper solution. As mentioned earlier, the initial guess at the current positioning is set as the known positioning at the previous controller update. The error function is also relatively simple in form and application and is discussed below. The correctional function is the most important piece of this approach and requires a certain amount of background in the use and formulation of Jacobian matrices. As a result, an entire section is devoted to the explanation of the correctional function.

Error Function

To formulate the error function, one must determine what information is available. In the case of the numerical forward analysis, this data consists of the measured leg lengths, the current guess at positioning, and the geometry of the platform. The error in the system is defined as the error in leg lengths between the current guess and the actual

solution. The positioning information consists of a six-element vector that describes the location of the end effector and its orientation relative to the global coordinate system. The lengths of the legs at the guess can be readily determined using a reverse positioning analysis. The error in the solution can then be calculated as shown in equation 3.1 for a system with **n** legs.

$$\underline{P_{guess}} = \begin{Bmatrix} x \\ y \\ z \\ \theta_x \\ \theta_y \\ \theta_z \end{Bmatrix} \Rightarrow \underline{L_{guess}} = \begin{Bmatrix} L_{1,guess} \\ L_{2,guess} \\ \vdots \\ L_{n,guess} \end{Bmatrix} \qquad \textbf{(3.1)}$$

$$\underline{Error} = \underline{L_{actual}} - \underline{L_{guess}}$$

One will notice that the error is not simply defined as the sum of the squares of the individual leg errors, but rather it is a vector that contains as many elements as there are legs in the system. As a result, use of this error in correctional calculations will inherently consider the geometric constraints of the entire system, as opposed to just the primary axes. This has the advantage of making the analysis of over constrained mechanisms relatively straightforward.

Another advantage of the numerical approach is that it tends to correct for the slight errors in measurements that will be present in any system. Since the solver is trying to get "close" to the correct solution, the input data does not have to be perfect to get a good guess. This especially holds true in over-constrained systems, where more measurements are taken than are needed to find the right answer. A small error in measurement of one or two elements will not ruin the calculations, inherently making this approach far more

robust than a closed-form solution. The next section looks at the Jacobian matrix and how it can be used in conjunction with the error to form a correctional term.

Jacobian Matrix and Its Applications

This section discusses the nature of the Jacobian matrix and its applications in both force-torque calculations, as well as positioning analyses. Before the actual use of the Jacobian in the forward analyses can be shown, a brief description of the derivation and meaning of the Jacobian matrix is necessary. This section also has relevance in later chapters regarding controller design and calculations.

Wrenches

When performing force-torque calculations for a parallel manipulator, one must know the forces applied in each leg actuator, as well as how these forces will contribute to the net wrench applied by the end effector.[1] Each leg actuator applies its force along a line that runs along the axis of its leg as shown in Figure 3-1.

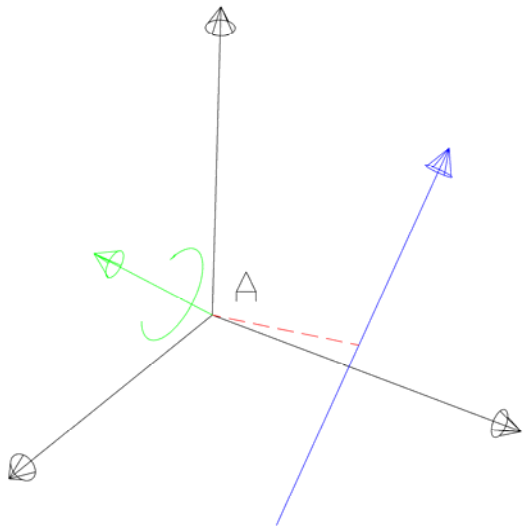


Figure 3-1. Illustration of line of action and applied wrench relative to point A.

[1] An excellent reference on position and force analyses for robots is *Kinematic Analysis of Robot Manipulators* [2].

The force that each actuator applies to the top platform can be described in terms of the line of action and the force acting along that line. The net summation of all leg forces acting on the top platform can be represented by a dyname (a force applied along a particular line of action together with a moment) or by a wrench (a force acting along a specific line of action whereby the direction of the accompanying moment is parallel to the direction of the force). The line of action is a 6×1 vector consisting of two parts:

- A 3-element unit vector indicating the direction of the line.
- A 3-element vector that is based on the relative positioning of the line to some reference point. This is obtained by taking the cross product of the moment arm and the direction vector as

$$\underline{\$} = \begin{Bmatrix} \underline{s} \\ \underline{r} \times \underline{s} \end{Bmatrix} \tag{3.2}$$

where $\underline{r}$ is the vector from the origin of the reference coordinate system to any point on the line.

Jacobian matrix formulation

Calculating the net wrench applied to a body by a group of forces is a relatively simple process. The overall wrench applied by a collection of forces can be described as the summation of these forces as

$$F_{net}\underline{\$}_{net} = F_1\underline{\$}_1 + F_2\underline{\$}_2 + \ldots + F_n\underline{\$}_n = \begin{bmatrix} \underline{\$}_1 & \underline{\$}_2 & \cdots & \underline{\$}_n \end{bmatrix} \begin{Bmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{Bmatrix} = [J] \begin{Bmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{Bmatrix} \tag{3.3}$$

$$[J] = \begin{bmatrix} \underline{\$}_1 & \underline{\$}_2 & \cdots & \underline{\$}_n \end{bmatrix} = \begin{bmatrix} \underline{s}_1 & \underline{s}_2 & \cdots & \underline{s}_n \\ \underline{r}_1 \times \underline{s}_1 & \underline{r}_2 \times \underline{s}_2 & \cdots & \underline{r}_n \times \underline{s}_n \end{bmatrix}$$

for a system with n legs. Here the notation F_i is used to represent the magnitude of the force along leg i and $\mathbf{\$}_i$ is used to represent the length 6 vector that represent the coordinates of the line of action of the force.

One can see that the Jacobian, **J**, is a 6×n matrix whose columns are the coordinates of the lines of action of the forces applied to the top platform. This is a clear indicator that the Jacobian matrix contains all of the information about the configuration of the system since the relative orientations and positions of the legs are incorporated. Logically, the Jacobian matrix, or some derivation of it, could be used in the numerical forward analysis.

Correctional Function

Now that the physical meaning of the Jacobian matrix is understood, the correctional function can be written. A slight modification of the Jacobian is introduced and will be explained shortly. This modified Jacobian will be multiplied by the error term explained earlier to form an adjustment to the guess at the current positioning. Equation 4 shows the modified Jacobian in a generic format that incorporates **n** legs.

$$[J]_{Modified} = \begin{bmatrix} s_1 & s_2 & \cdots & s_n \\ \dfrac{r_1 \times s_1}{L_{1,guess}} & \dfrac{r_2 \times s_2}{L_{2,guess}} & \cdots & \dfrac{r_n \times s_n}{L_{n,guess}} \end{bmatrix} \quad \textbf{(3.4)}$$

One will notice that the modified Jacobian is very similar in form to the original except that the bottom rows, representing the torsional contributions of the legs, have been scaled by the lengths of those legs. Visualizing how the correctional guess actually works is critical in understanding why the scaling is necessary. Let us consider a single leg and how modifying its length will affect the system state for a planar mechanism. Figure 3-2 shows how changing the length of a single leg will impact the positioning of the platform by causing a displacement along that leg's line of action. Likewise, Figure 3-3 shows the impact that a change in leg length will have on the rotation of the platform. The platform will rotate about an axis perpendicular to the plane formed by the line of

action of the leg and the moment arm of that leg. The amount of rotation is dependent on the magnitude of the change and the relative orientation of the leg with respect to the platform.

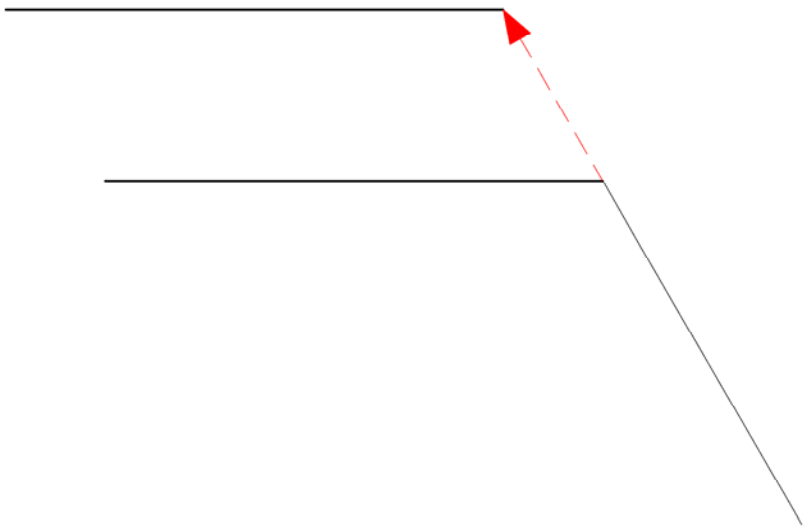


Figure 3-2. Effect of change in leg length on positioning of the end effector.

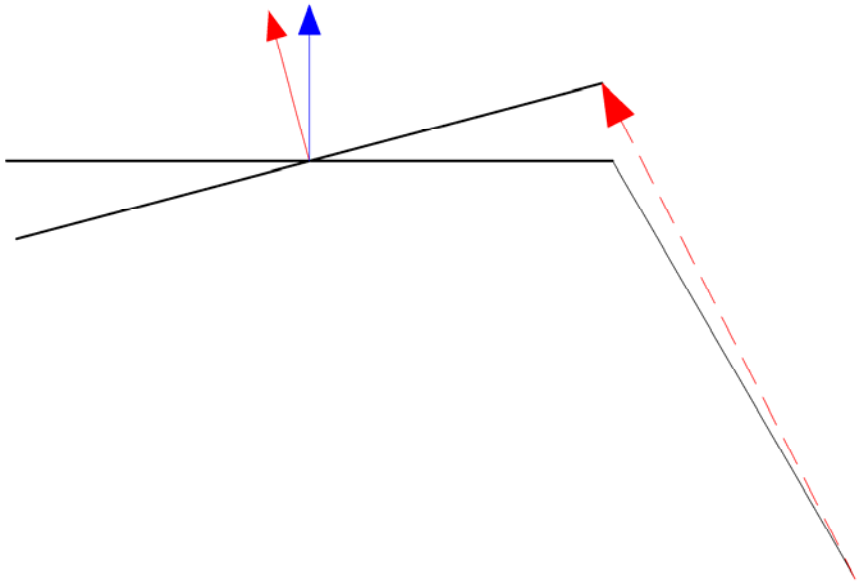


Figure 3-3. Effect of change in leg length on orientation of the end effector.

Changing the length of any single leg attempts to move the platform along the line of action of that leg. When all of the legs act together, some of the motions will cancel out while others work together, leading to a net movement in a particular direction. The orientation of the platform is affected similarly by the changes in length. Each leg will attempt to rotate the end effector about an axis perpendicular to the plane formed by itself and its moment arm. Scaling the torsional component by the change in length of the leg is necessary since this process looks at the *rotational* changes in the system. Essentially, the required change in length of the leg can be considered as a change in the rotational strain of the leg when scaled by the overall length.

$$\left\{\frac{\underline{r}\times\underline{s}}{L_{guess}}\right\}\left(L-L_{guess}\right)=\left\{\underline{r}\times\underline{s}\right\}\frac{\left(L-L_{guess}\right)}{L_{guess}}=\left\{\underline{r}\times\underline{s}\right\}\varepsilon \qquad \textbf{(3.5)}$$

$$\varepsilon \equiv Leg\_Strain$$

Another way to look at the rotational components is to consider the convergence of a change in leg length as that change becomes smaller. The change in length is very similar to $\sin(\theta)$ in a small angle approximation, implying a predictable correlation between length and rotational effects. However, the change in length cannot be used directly since it does not take into account the relative effect on the leg in question. Any particular change will have a noticeably greater impact on a system with a very short leg as compared with a longer leg. By scaling each component of the Jacobian, the impact of each error correction is scaled to compensate for the relative effects of the individual legs. The only reason that the bottom rows in the Jacobian are scaled instead of scaling the error vector is for efficiency of calculation. It avoids the issue of two separate calculations that would then require the results to be recombined into a single vector. The next step is to calculate the correction to the guess at positioning using Equation 6.

$$\underline{\Delta P_{guess}} = \xi [J]_{\mathrm{mod}\,ified} \left(L - L_{guess} \right) \qquad \textbf{(3.6)}$$

The scaling term ξ was included in the $\Delta \mathbf{P}_{\mathbf{guess}}$ calculation to improve the efficiency of the solver. Depending on the system, the solver may take extremely small or large steps when trying to find the proper solution. This can either cause the program to slow down considerably, or possibly even overshoot the solution and never converge. The scaling term allows the program to adjust the size of the step to ensure rapid convergence on a solution. This term may need to be found empirically, but at times it is quite necessary, such as when dealing with over constrained mechanisms. The next section shows application of this solver to three different systems that were analyzed in this project. Only the basic geometry of the mechanisms will be described since a more detailed explanation of the particular systems will be given in their respective chapters.

Application and Results for the Numerical Forward Analysis

The application of the solver to three systems will be discussed in this section. The first is a planar mechanism that was developed as a simplified test of the control strategy laid out in Chapter 2. The second system is the over constrained spatial mechanism that was a primary focus of this project. The third mechanism is similar to the second, except that the connection point for one of the legs was moved to test the conditioning of the second mechanism. Also note that the MATLAB code for the analyses can be found in Appendix B, along with other code blocks necessary for running the simulations.

Planar 2-2 + 1 Truss Platform

This mechanism followed a basic 2-2 configuration with the third leg bridging two of the connection points. Figure 3-4 shows the connection setup for the planar mechanism and the dimensioning in centimeters.

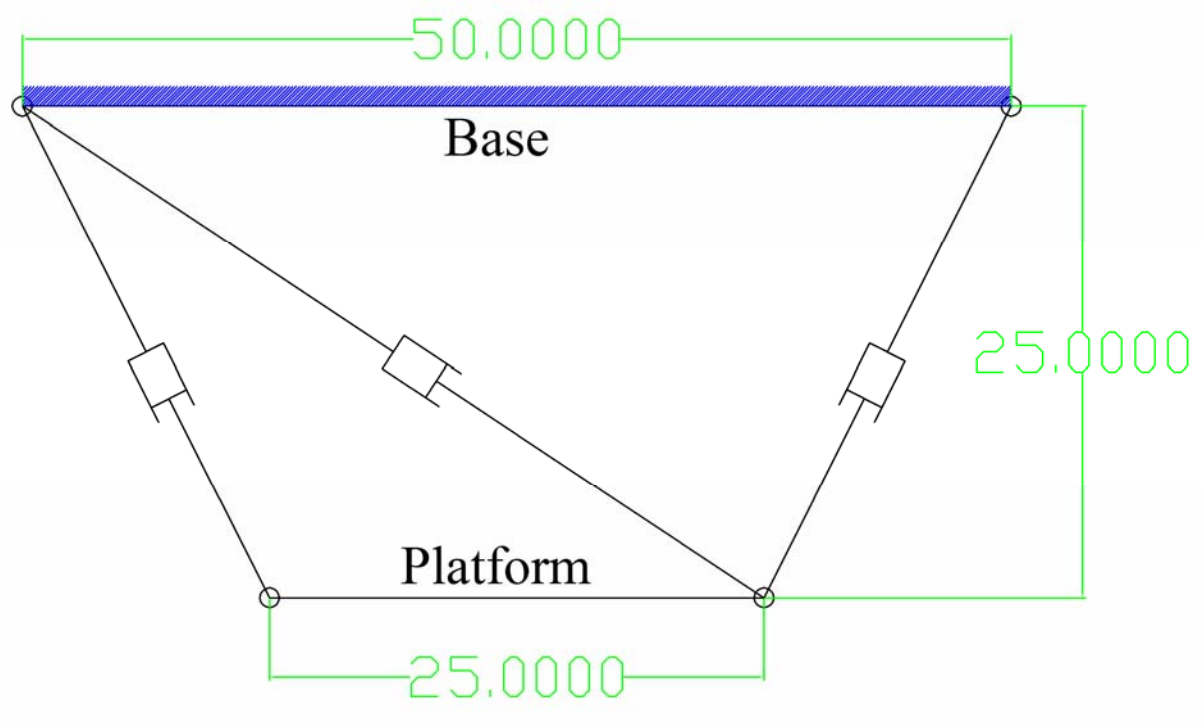


Figure 3-4. Planar truss mechanism.

The Jacobian matrix for this planar device was only a 3×3 matrix since only three degrees of freedom were present in the system. This is not a problem since the fundamental concepts behind the analysis remain intact. The rotational contribution of the legs is still directly related to their effective moment arms about the center of the end effector. The new Jacobian will take the form of

$$J_{3x3} = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ \frac{\mu_{z,1}}{L_1} & \frac{\mu_{z,2}}{L_2} & \frac{\mu_{z,3}}{L_3} \end{bmatrix} \quad \textbf{(3.7)}$$

where x_i and y_i represent the direction of leg i, $\mu_{z,i}$ represents the moment of leg i about the z axis, and L_i represents the length of leg i.

The X and Y components of the line of action for each leg are needed in the matrix, as is the rotational contribution about the Z-axis, μ_Z. The variable μ_Z is the Z component of the vector $r \times s$, which only has a Z-component for the planar case as

$$r \times s = \begin{Bmatrix} \mu_x \\ \mu_y \\ \mu_z \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ \mu_z \end{Bmatrix} \quad \textbf{(3.8)}$$

Testing of the numerical forward analysis was performed with a handful of scenarios, including situations in which the initial guess was relatively close to the solution and other cases where the guess was significantly farther away. The most relevant examples are discussed here.

Guess close to solution

In this series of tests the guess was extremely close to the solution. The idea was to test a realistic scenario as would be used in a real controls application. Under normal controls circumstances, the controller is updated and a fairly regular interval, meaning that the system has not moved much from its previous position. Since the previous position will be used for the initial guess, it is only logical to test situations in which one is very close to the solution.

In each case the solution was such that the platform was centered side to side relative to the base and was located 25 units below the base, i.e. coordinates [0, -25; 0]. The guesses were offset from the solution by up to ten units along the primary axes or 0.1 radian rotation about the Z-axis. The maximum error allowed was 0.0001 unit in leg length and the maximum number of iterations was capped at 4000 to avoid potential infinite loops. Table 3-1 shows the results of these tests. X and Y are the positioning of

the platform relative to the center of the base. Θ is the rotation of the platform about the Z-Axis. At a value of zero, the platform is parallel to the base.

Table 3-1. Results of numerical forward analysis for planar mechanism with the guess close the solution.

Initial Guess			Calculated Solution			
X (cm)	Y (cm)	Θ (rad)	X (cm)	Y (cm)	Θ (rad)	# of Iterations
0	-35	0	-0.0029	-24.9999	-0.0753	4000
0	-30	0	-0.0006	-25	-0.053	1254
10	-25	0	-0.0094	-24.9998	0.1407	4000
5	-25	0	0.001	-25	0.0212	1961
0	-25	0.1	-0.0092	-24.9998	0.1395	4000

One will notice that the calculated solutions are very close to the actual solutions. The main error present in the system is in the rotation. This is not surprising because the system is asymmetric due to the presence of the cross third leg. It inherently skews the modified Jacobian resulting in over correction in both the X direction and rotation about the Z-axis. This primarily manifests itself in rotational error, which unfortunately cannot be corrected with the current approach.

Guess far from solution

The second series of tests placed the initial guess relatively far from the actual solution. This was a given idea of the limitations of the solver in extreme circumstances, although they are not very likely to occur. Table 3-2 shows the results of the testing. One will notice the difficulty the solver has in these situations.

Table 3-2. Forward analysis for planar mechanism with the guess far from the solution.

Initial Guess			Calculated Solution			
X (cm)	Y (cm)	Θ (rad)	X (cm)	Y (cm)	Θ (rad)	# of Iterations
0	-55	0	-0.0076	-24.9998	-0.1213	4000
0	-75	0	-0.01	-24.9997	-0.1382	4000
20	-25	0	-0.2009	-24.9939	0.7429	4000
-20	-25	0	-0.4992	-24.975	1.5699	4000

The primary cause of the solver difficulties is that the error is so large that the corrections could potentially exceed the workspace. Another cause may be that several possible solutions could exist for any given set of leg lengths. This is not inherently a problem because in real controls applications, the guesses will never be this far off. The next sections discuss the results from testing with the spatial mechanisms, where it becomes clear that having extra legs can dramatically improve the response of the solver.

Over Constrained 3-3 + 1 Spatial Mechanism

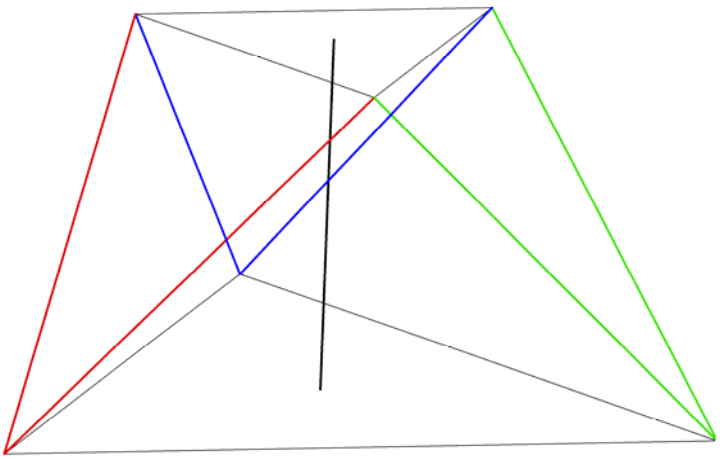


Figure 3-5. Spatial 3-3 + 1. Note that the upper triangle represents the end effector.

The spatial mechanism tested followed a 3-3+1 configuration, which is identical to a 3-3 mechanism, except that it contains an additional central leg. Figure 3-5 shows the relative placement of the central leg with respect to the rest of the mechanism. This leg ran from the center of the base platform to the center of the end effector, producing symmetry in the device. The benefits of having this leg present in the system became readily apparent once testing began. Whereas the planar mechanism would sometimes require several thousand iterations to find a solution, the spatial mechanism would often

only require 50-100 iterations, which could be calculated extremely quickly. For each test, the actual positioning of the platform was centered above the base at a height of 1 meter along the Z-Axis with no rotations about any axes. This is represented by the coordinates [0, 0, 1; 0, 0, 0]. Both the end effector and the base were configured as equilateral triangles. The corners of the end effector were located 0.5 meter from its center while the base triangle was twice the size.

Table 3-3. Results of forward analysis for the redundant spatial mechanism.

Initial Guess						Calculated Solution						
X (m)	Y (m)	Z (m)	Θ_X (rad)	Θ_Y (rad)	Θ_Z (rad)	X (m)	Y (m)	Z (m)	Θ_X (rad)	Θ_Y (rad)	Θ_Z (rad)	# of Iter
0	0	1.5	0	0	0	0	0	1	0	0	0	4
0	0	5.5	0	0	0	0	0	1.0001	0	0	0	6
0	5	1	0	0	0	0	0	1	0.0003	0	0	93
10	5	1	0	0	0	0	0	1	0.0001	-0.0002	0	86
0	0	1	1.5	0	0	0	0	1	0.0003	0	0	113
0	0	1	0	1	0	0	0	1	0.0001	0.0002	0	111
0	0	1	0	0	1	0	0	1	0	0	0.0003	69
2.5	3	5	0.5	0.5	0.5	0	0	1	0.0002	0.0001	0	107

One will notice that the error in estimated positioning is extremely small. This comes as a welcome surprise and clearly demonstrates the potential for speed and accuracy with this approach. The errors could be reduced even further by reducing the maximum allowable error in the calculations. The extra leg helped to dramatically limit the number of solutions, expanding the effective workspace of the solver, resulting in a huge increase in performance. The final test for the solver was with a modification of the spatial mechanism in which one of the connection points for the central leg was moved to produce a non-symmetric system.

Modified 3-3 + 1 Spatial Mechanism

The modified spatial mechanism moved the top connection of the central leg to one of the corners of the end-effector, so that it lay coincident with two other joints. The

location selected was coincident with the joints located 0.5 m along the Y-Axis of the platform. Figure 3-6 shows the change in configuration of the spatial mechanism.

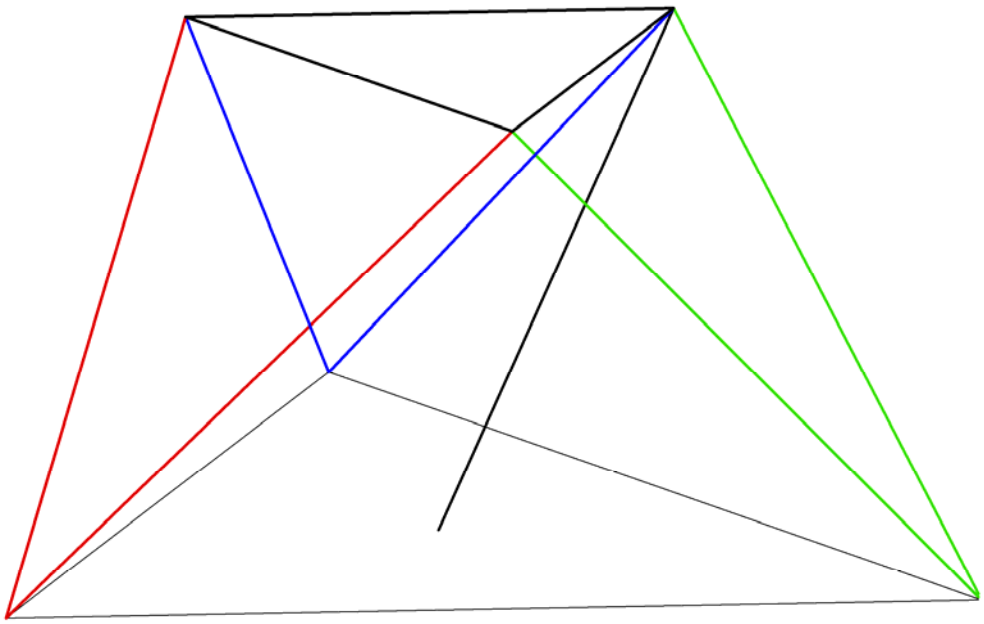


Figure 3-6. Modified Spatial 3-3 + 1.

This was an ideal test since the system was no longer symmetric. One may recall the ill-effects of asymmetry had on the solver for the planar mechanism. However, these issues never arose during testing of the solver with the modified spatial device. In each test, the modified mechanism was positioned at the same location as the 3-3+1 system, with the same dimensions. The actual position was at [0, 0, 1; 0, 0, 0]. Table 3-4 shows the results and clearly indicates that the solver is still working well. The slight increase in the number of iterations necessary to find solutions is a minor issue that could most likely be resolved by adjusting the scaling factor ξ for the error.

Table 3-4. Results for testing of numerical forward analysis for the modified spatial mechanism.

Guess						Solution						
X (m)	Y (m)	Z (m)	Θ_X (rad)	Θ_Y (rad)	Θ_Z (rad)	X (m)	Y (m)	Z (m)	Θ_X (rad)	Θ_Y (rad)	Θ_Z (rad)	# of Iter
0	0	1.5	0	0	0	0	-0.0001	1	-0.0001	0	0	12
0	0	2.5	0	0	0	0	0	1	-0.0003	0	0	40
0	2	1	0	0	0	0	0	1	-0.0003	0	0	68
10	2	1	0	0	0	0	0	1	0.0002	-0.0001	0	114
0	0	1	1.5	0	0	0	0	1	0.0003	0	0	113
0	0	1	0.5	0.5	0.5	0	0	1	0.0001	0.0002	0	101

Final Comments and Conclusions

The numerical forward analysis works exceptionally well in a wide array of situations and looks to be a highly flexible and powerful tool. It is easy to implement and can be very efficient at finding positioning, even with minimal tuning. Changing the geometry of the mechanisms presents no difficulty as the modified Jacobian matrix also changes. This approach also allows users to analyze and control mechanisms that are over-constrained or even ill-conditioned since it is never necessary to find the inverse of a matrix.

While the analysis laid out in this chapter is a powerful tool, a possible refinement might include a more precise method of tuning the solver. Currently, the only adjustment to the solver is a scaling of the step that is taken to avoid overshooting the solution. A more thorough analysis later could reveal better tuning methods that might parallel classical PID controls for other systems. However, that analysis is best left for another time and project as the current approach works extremely well.

CHAPTER 4
CONTROLLER DESIGN AND IMPLEMENTATION

As discussed in Chapter 2, a number of elements are involved in designing a controller for any system. For the specific application to parallel mechanisms, the most important and difficult part of the problem is accurate measurement of the positioning of the system. This problem was solved in Chapter 3 using a flexible numerical methods approach. The next step in the process is the design of the controllers that will actually affect the behavior of the system.

Separating the DOF

Any parallel mechanism is highly cross-coupled by nature, making control strategy potentially difficult. Chapter 2 talked about the advantages of assuming that the leg masses were negligible, and this is an important first step towards developing the controllers. One must remember that the spatial systems analyzed in this project had six degrees of freedom, necessitating at least six controllers.

Since the leg masses were extremely small, the only significant object in any particular system was the end-effector platform itself. The system model could now be developed that consisted of a lumped mass with a collection of forces acting upon it, as shown in Figure 4-1. These forces would produce an instantaneous net-wrench that would affect the motion and behavior of the end effector.

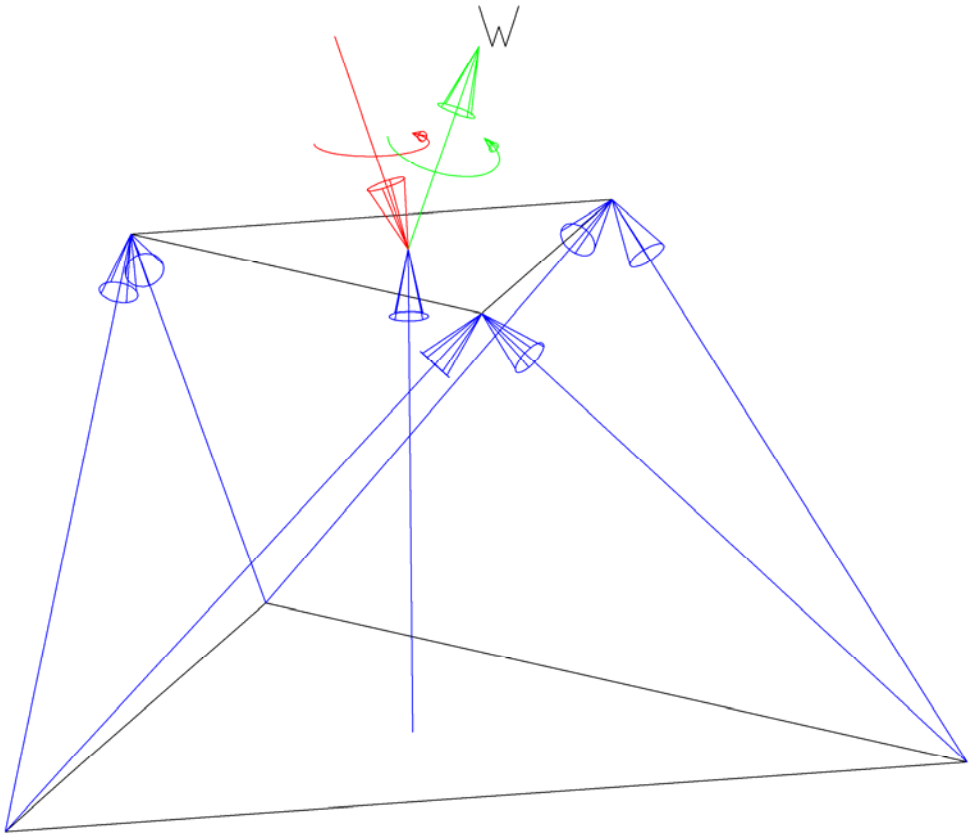


Figure 4-1. End effector and forces acting upon it. The net-wrench is shown as **W**.

Recall that a wrench consists of six elements which represent the force and torque acting upon a particular body. Theoretically, applying a wrench that only acts along a single DOF should affect the object only in that DOF. This implies that the problem can now be separated into six individual models, each corresponding to one DOF. As a result, the six resulting equations of motion take the form of

$$F_{ext} = M_{eq}\ddot{q} \qquad \textbf{(4.1)}$$

M_{eq} is a generalized representation of the equivalent mass or inertia of the system in the particular degree of freedom that is being considered. By the same token, $\ddot{q}$ is the acceleration of the system in that degree of freedom. F_{ext} is simply the external force or torque that is applied to the system during its motion.

Simplified System Model and Controller Implementation

Now that the various DOF's have been separated it is necessary to determine the behavior of the individual systems. As discussed in the previous section, each section of the model can be represented as a single mass upon which an external loading acts. Designing a controller for such a system is relatively straight forward as the motions of these systems are well known. A standard PID controller should be sufficient to produce "good" behavior. The system block diagram for each DOF now takes the form of

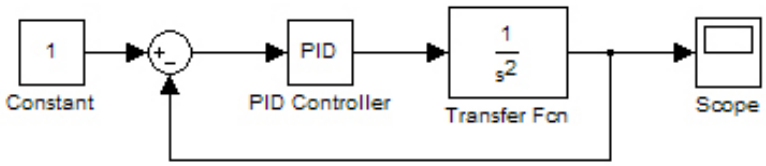


Figure 4-2. Block diagram for a single mass system.

Now that the control layout has been formulated, the next step is to determine the transfer function that describes the behavior of the system. From basic control theory, the closed-loop transfer function can be written as

$$TF = \frac{T}{1 + HT} \tag{4.2}$$

Where:

T is the controller times the representation of the system mass.
H is the value of the feedback multiplier.

When the value for T is substituted into equation (4.2), the numerator and denominator of the equation can be expanded, resulting in the third-order function

$$T = PID\frac{1}{ms^2} = \left(P + \frac{I}{s} + Ds\right)\frac{1}{ms^2} = \frac{Ds^2 + Ps + I}{ms^3}$$

$$1 + HT = 1 + H\left(\frac{Ds^2 + Ps + I}{ms^3}\right) = \frac{ms^3 + HDs^2 + HPs + HI}{ms^3} \tag{4.3}$$

$$TF = \frac{Ds^2 + Ps + I}{ms^3 + HDs^2 + HPs + HI} = \frac{{}^{D}\!/_{m}\, s^2 + {}^{P}\!/_{m}\, s + {}^{I}\!/_{m}}{s^3 + {}^{HD}\!/_{m}\, s^2 + {}^{HP}\!/_{m}\, s + {}^{HI}\!/_{m}}$$

where P, I, and D are the values of the controller.

Determining the ideal values for the controllers can be a rather tedious task, especially with a third-order system. However, certain simplifications can be made to make the problem more manageable. By inspection, one can see that setting the integral portion of the controller to zero will reduce the transfer function to a more classical second order system.

$$TF = \frac{{}^{D}/_{m}\, s^2 + {}^{P}/_{m}\, s}{s^3 + {}^{HD}/_{m}\, s^2 + {}^{HP}/_{m}\, s} = \frac{{}^{D}/_{m}\, s + {}^{P}/_{m}}{s^2 + {}^{HD}/_{m}\, s + {}^{HP}/_{m}} \qquad \textbf{(4.4)}$$

Design of controllers for second-order behavior has been well-established, enabling one to not only easily design the controllers, but also to automate the process during controller implementation. One will notice that this approach is the application of a PD controller to a free mass system. While this is a simpler approach, there may be situations where application of a PID controller might be necessary to compensate for steady-state error or unforeseen effects of the system being analyzed, as was the case for the planar mechanism discussed in Chapter 5.

Having specified the form of the controllers, one can easily determine the values to generate desired system behavior. From classical second-order analysis, the equations describing the response of the system take on the following form

$$\begin{aligned}
\omega_n^{\,2} &= \frac{HP}{m} \\
2\zeta\omega_n &= \frac{HD}{m} \\
t_s &= \frac{4}{\zeta\omega_n} \qquad \textbf{(4.5)} \\
M_p &= e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}} \\
\zeta &= \sqrt{\frac{\left({}^{\ln(M_p)}/_{\pi}\right)^2}{1+\left({}^{\ln(M_p)}/_{\pi}\right)^2}}
\end{aligned}$$

Where:

ω_n is the natural frequency of the system.

ζ is the damping ratio.

t_s is the settling time.

M_p is the ratio of maximum overshoot to steady-state value.

Final Thoughts

While the controllers used in this project are fairly simple in form, they were meant to test two things. The first was the feasibility of decoupling the legs with basic assumptions and restrictions on the system. The second was the ease with which a controller could be incorporated into the system, given the overall complexity of the problem. The highly flexible numerical forward analysis discussed in Chapter 3 was a primary focus of development in this project and was considered to be a more critical aspect of the overall controller design. More advanced control techniques could easily be implemented, but the structure of the controls would remain essentially intact, regardless of the approach.

CHAPTER 5
APPLICATION TO A PLANAR MECHANISM

Once the control methodology has been determined, the next logical step is testing on a simplified system to analyze basic performance and ascertain whether the steps to attain decoupling actually worked. The mechanism first analyzed in this project was a planar, three-leg platform that was set in a truss configuration, as described in Chapter 3. Not only did this mechanism serve as a simple test bed, but it also helped to highlight potential issues related to asymmetry.

System Description and Simulation Model

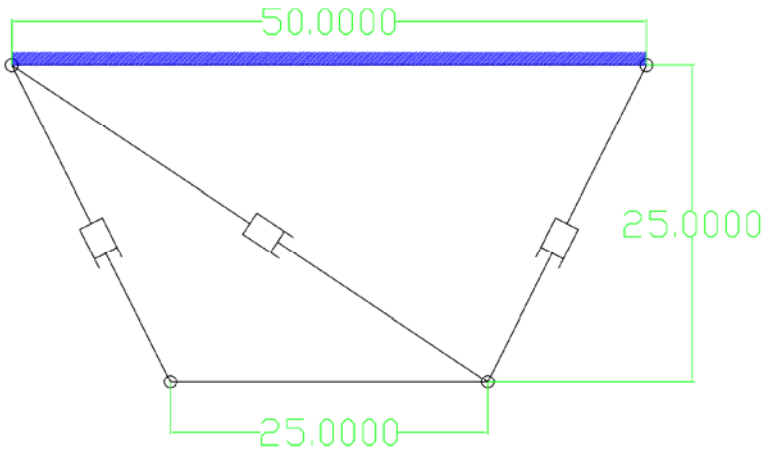


Figure 5-1. Planar truss mechanism with asymmetry.

The planar truss mechanism consisted of three legs arranged in the configuration shown in Figure 5-1. One will notice that the platform differs from the other mechanisms analyzed in this project in that the end effector hangs below the base connection, rather than being supported above it. This has no effect on the control strategy or system analysis since the wrench applied by the weight of the platform is simply reversed. The configuration shown was chosen since it was naturally stable in simulations, allowing the testing to focus purely on controls and system behavior, rather than trying to "debug" the mechanism. The dimensions of the platform were selected to mimic the configuration of the spatial mechanism that would be analyzed later. The end effector was half the length of the base, with the connection points located at both ends.

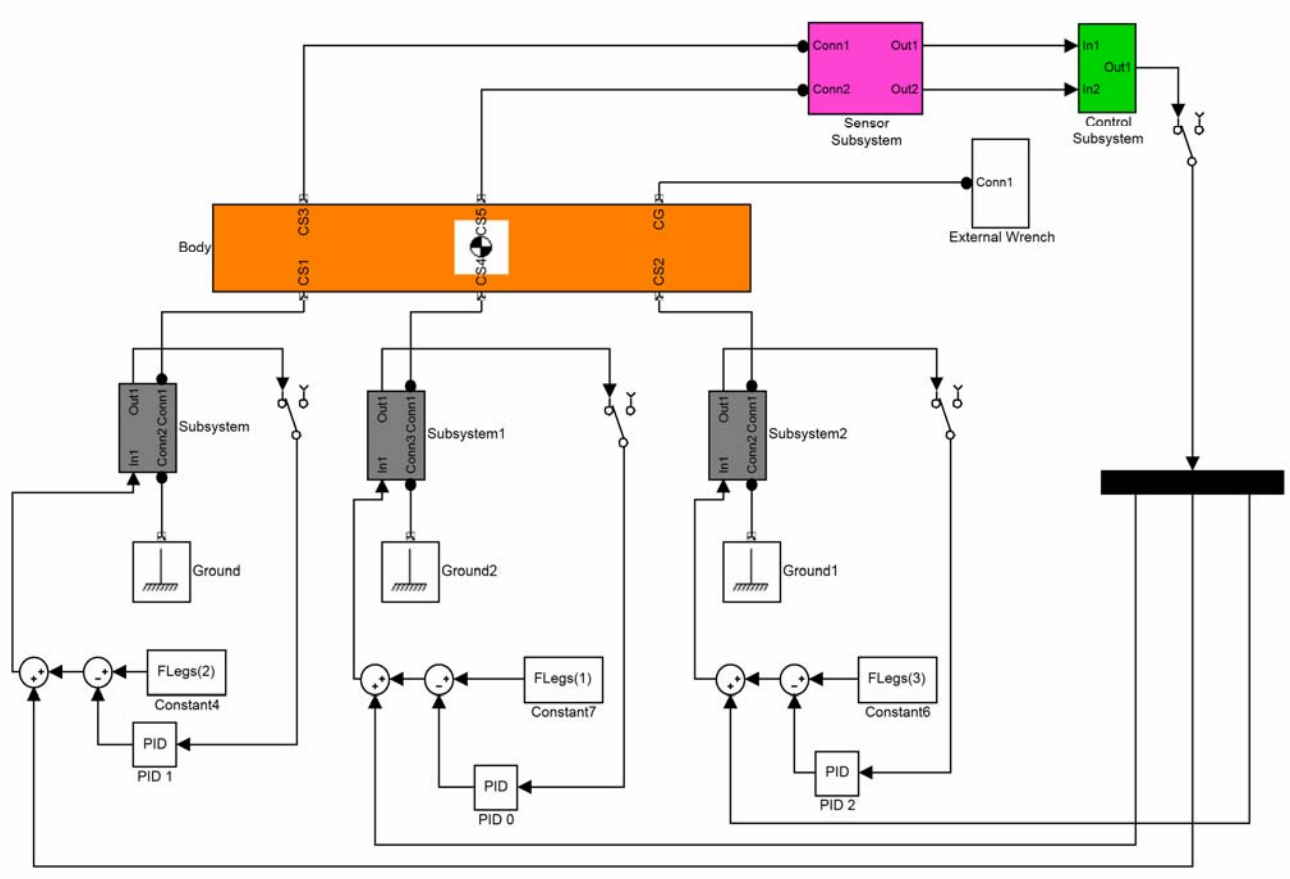


Figure 5-2. System model created in Simulink, representing the planar truss mechanism. Note that the controller is marked in green and the end effector is colored orange. This system in discussed in Appendix A.

The entire system was modeled in Simulink using MATLAB 6.5. This provided the most flexible approach for modeling as the system parameters could be easily changed. Figure 5-2 shows the model with various components highlighted, though a larger version of the model can be found in Appendix A. Since the planar mechanism was meant as a test bed for the control strategy discussed in Chapter 4, the position information was measured, rather than calculated using the numerical forward analysis. The leg elements were given extremely small, but still finite masses because the software required that they have at least some mass to avoid singularities in calculations.

System Parameters and Performance Criteria

Recall that the legs and actuators were reasoned to have negligible mass, leaving the only source of mass and inertia as that of the end effector. Since the values of the controller gains were directly proportional to the mass of the system, the mass was simply selected as 1 kg, though any value would be perfectly acceptable. The length of the end effector was 25 cm while the base length of the platform was 50 cm. The inertia contribution of the platform was calculated by modeling the end effector as a thin bar using the equation

$$I = \frac{1}{12} mL^2 \qquad \textbf{(5.1)}$$

Where:

m is the mass of the end effector.
L is the length of the end effector.

The system was specified to have a desired maximum of 25% overshoot and a settling time less than 1 second. However, preliminary testing revealed that a PD controller was insufficient to compensate for the coupling effects caused by the third leg, as mentioned in Chapter 4. This necessitated a revision to the controller that used a full

PID controller rather than the simpler PD design. While this was an unexpected hurdle during testing, it served to highlight a potential problem that asymmetry can present in these mechanisms. Apparently a high amount of asymmetry tends to exacerbate the problem of cross-coupling, possibly requiring that certain additional steps be taken to correct the issues.

Redesign of the controller assumed that a full PID controller would be necessary, which also meant that the system model was third order, rather than second order. The characteristic equation of the system now becomes

$$s^3 + {}^{HD}\!/_{m}\, s^2 + {}^{HP}\!/_{m}\, s + {}^{HI}\!/_{m} \qquad \textbf{(5.2)}$$

This formula can be split into two portions, a second order and a first order formula

$$\left(s^2 + 2\zeta\omega_n s + \omega_n{}^2\right)(s + R) \qquad \textbf{(5.3)}$$

The real pole will cause a slight change in the behavior of the system, which is the ultimate goal of this process. The complex poles for the second order response were selected to be -4.5 ± 6.75j to give an approximate response of 20% overshoot and a settling time less than 1 second. The real pole was picked as -10 since testing of a generic third order system showed that it gave good results for the design parameters. While it was a somewhat longer process designing the controllers, the solution was relatively straightforward. Then main lesson from this was that asymmetry is highly undesirable in any system, causing unpredictable behavior. However, the steps taken for redesign the controllers provided adequate compensation.

Results of Testing

The controllers performed far better than anticipated, showing less than the maximum allowable overshoot, settling times less than one-second, and steady-state

errors most likely attributable to round-off errors. Every performance requirement was exceeded during testing.

Run 1

The first run tested the response of the system in the y-direction since that is the most stable direction. The platform started centered at (0 cm, -25 cm) with no rotation and was moved to the position (0 cm, -15 cm). Figures 5-3 through 5-5 show the system response in the x, y, and θ directions respectively. Table 5-1 shows the most pertinent numerical results from the run.

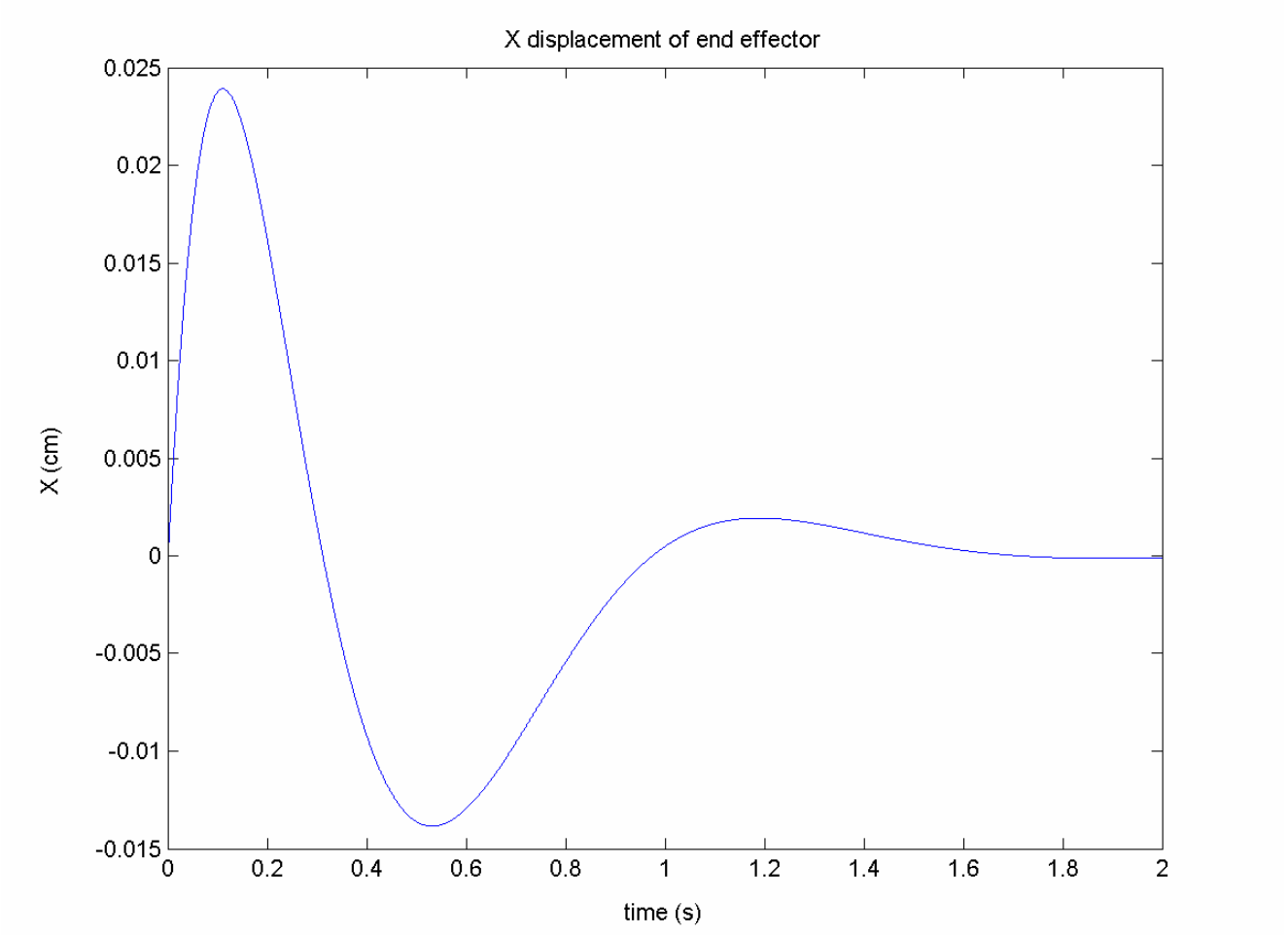


Figure 5-3. X-displacement of end-effector for first run.

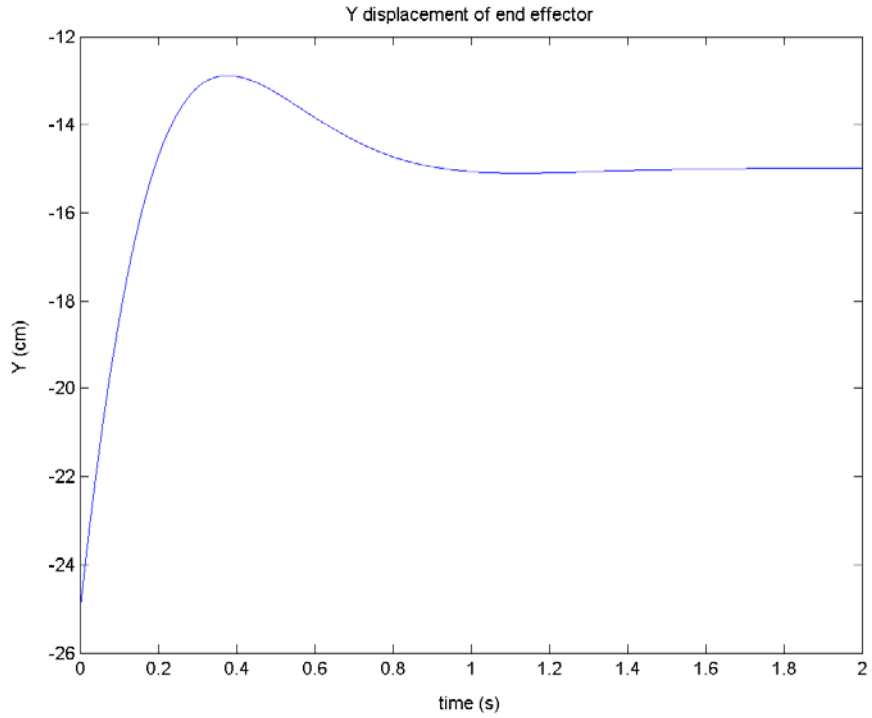


Figure 5-4. Y-displacement of end-effector for first run.

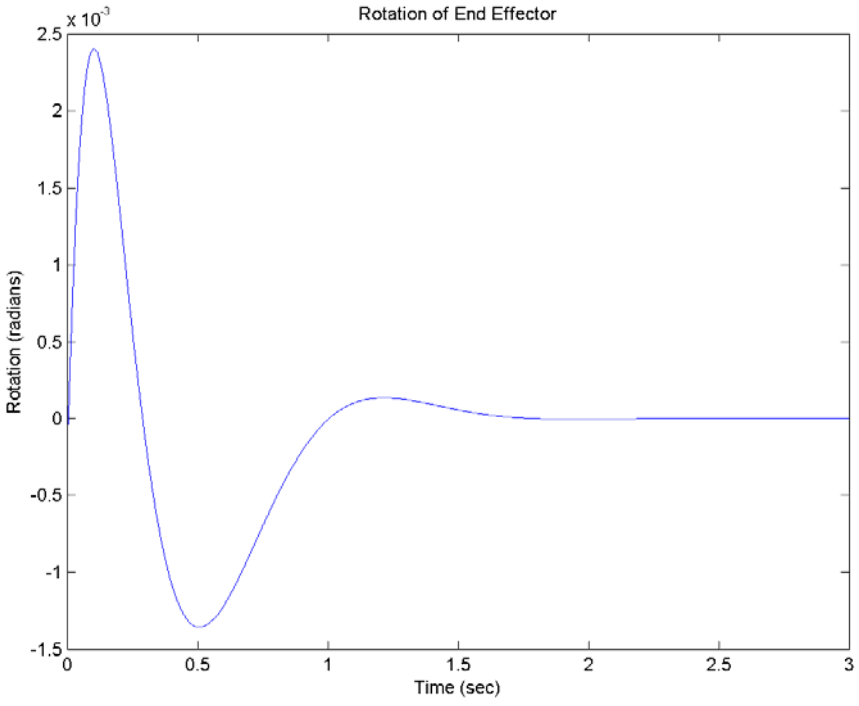


Figure 5-5. Θ-displacement of end-effector for first run.

Table 5-1. Results of Run 1.

	Desired Position	Final Position	Maximum Position	Max Overshoot
X	0 cm	-0.0001 cm	0.00240 cm	n/a
Y	-15 cm	-14.9963 cm	-12.8797 cm	21.20 %
θ	0 rad	0 rad	2.4e-5 rad	n/a

As can be seen from Table 2, the maximum steady-state error was 0.0037 cm, or 0.037%. The maximum overshoot of the y-direction was 21.20% which is less than the 25% limit. The settling time of the system is 0.82 sec, which once again is less than the specified limit of 1 sec. The extremely small steady-state errors can be attributed to round-off errors.

Run 2

The second run tested the response of the system in the x-direction. Once again, the system started at (0 cm, -25 cm), but then it was moved to (-15 cm, -25 cm). The idea of doing testing like the first three runs was to test the coupling of the system to determine how strongly one direction of motion might affect the others. The results are displayed in Figures 5-6 to 5-8.

As can be seen from Table 5-2, the maximum steady-state error was 0.0055 cm, or 0.037% once again. The maximum overshoot of the x-direction was 21.20% which is less than the 25% limit. The settling time of the system is 0.82 sec, which once again is less than the specified limit of 1 sec. The response in the x-direction is identical to the y-direction since they both run the same controllers. This also indicates that the coupling effects in the system are negligible.

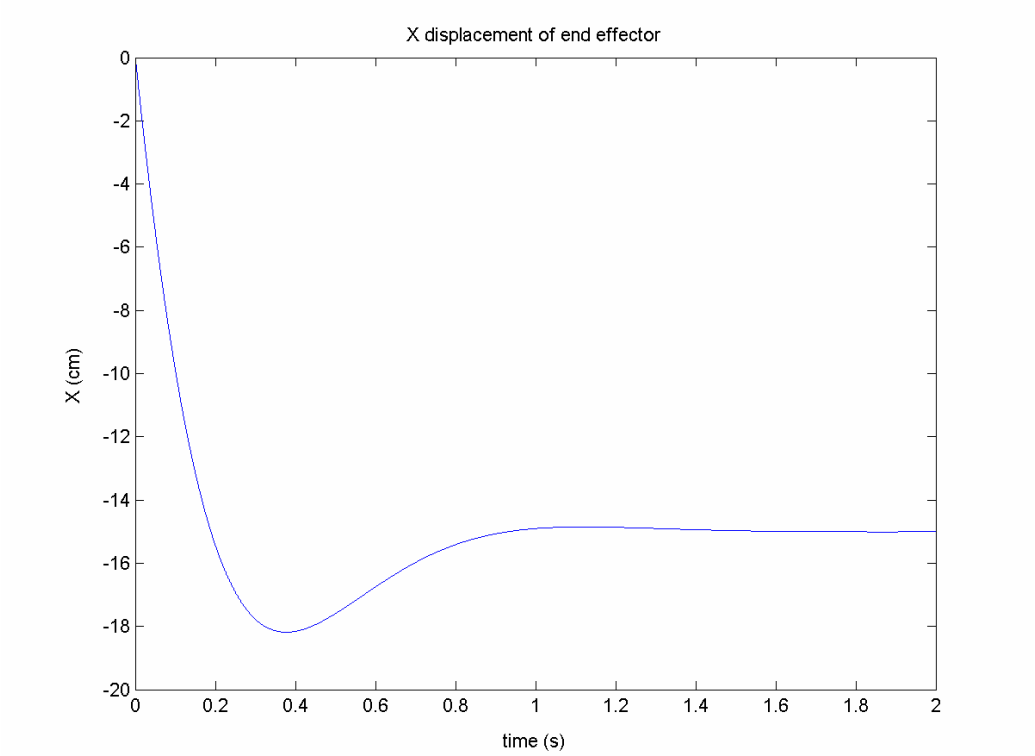


Figure 5-6. X-displacement of end-effector for second run.

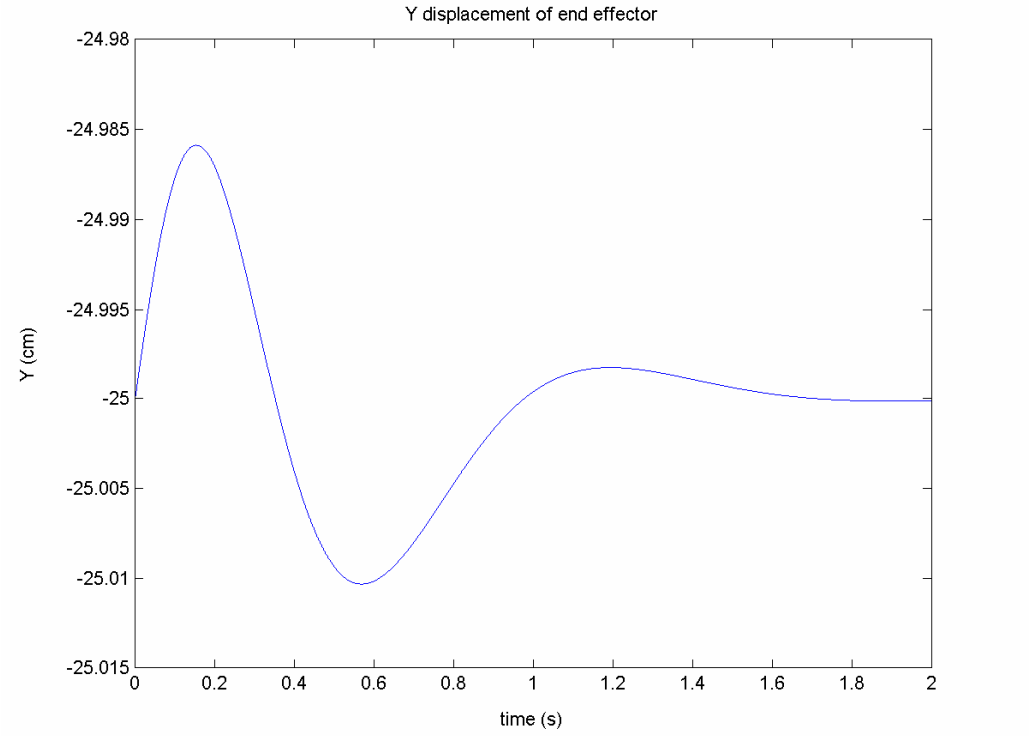


Figure 5-7. Y-displacement of end-effector for second run.

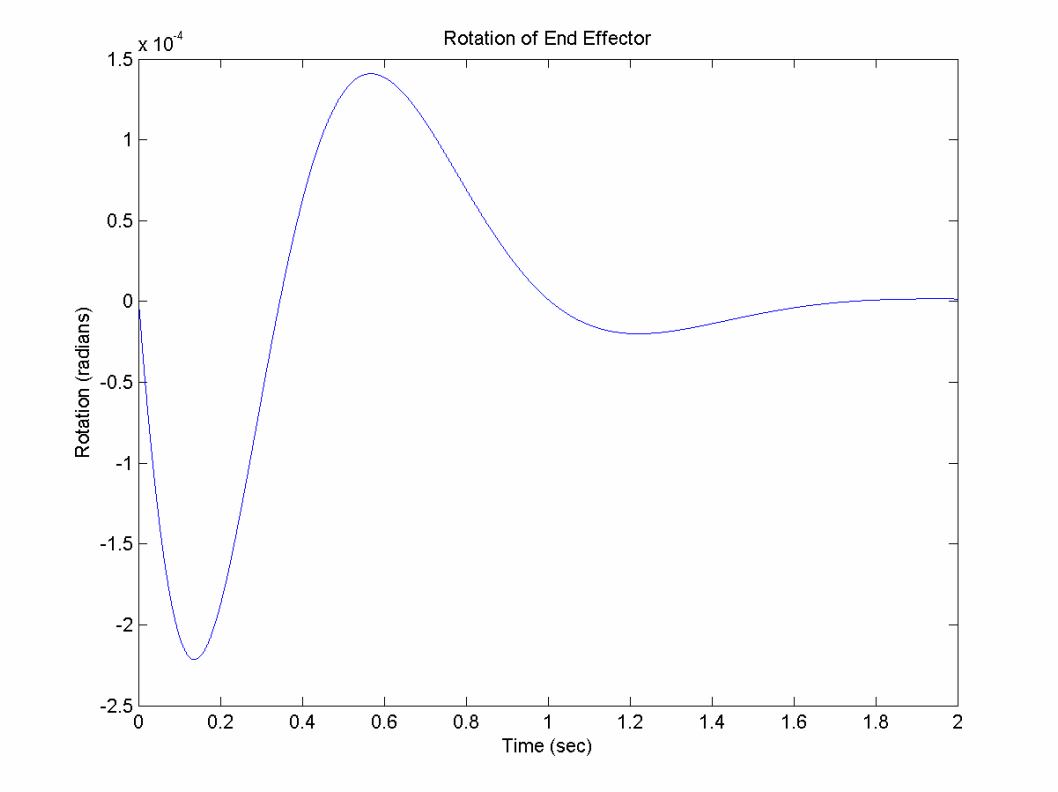


Figure 5-8. Θ-displacement of end-effector for second run.

Table 5-2. Results of Run 2.

	Desired Position	Final Position	Maximum Position	Max Overshoot
X	-15 cm	-15.0055 cm	-18.1804 cm	21.20%
Y	-25 cm	-25.0001 cm	-25.0105 cm	n/a
θ	0 rad	8.901e-5 rad	-0.0021 rad	n/a

Run 3

The third run tested the rotational response of the system. The platform was rotated by 10 degrees. The results are shown in Figures 5-9 to 5-11. As can be seen from Table 5-3, the maximum steady-state error was 0.0001 deg, which is most likely round-off error. The maximum overshoot of the θ-direction was 20.952% which is less than the 25% limit, but different from the x and y responses. The settling time of the system is approximately 0.75 sec, which once again is less than the specified limit of 1 sec.

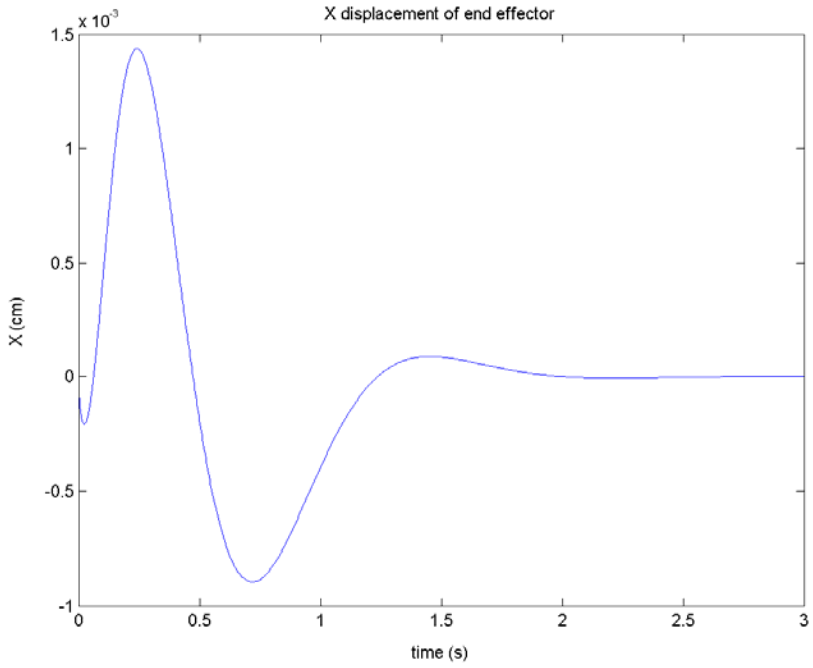


Figure 5-9. X-displacement of end-effector for third run.

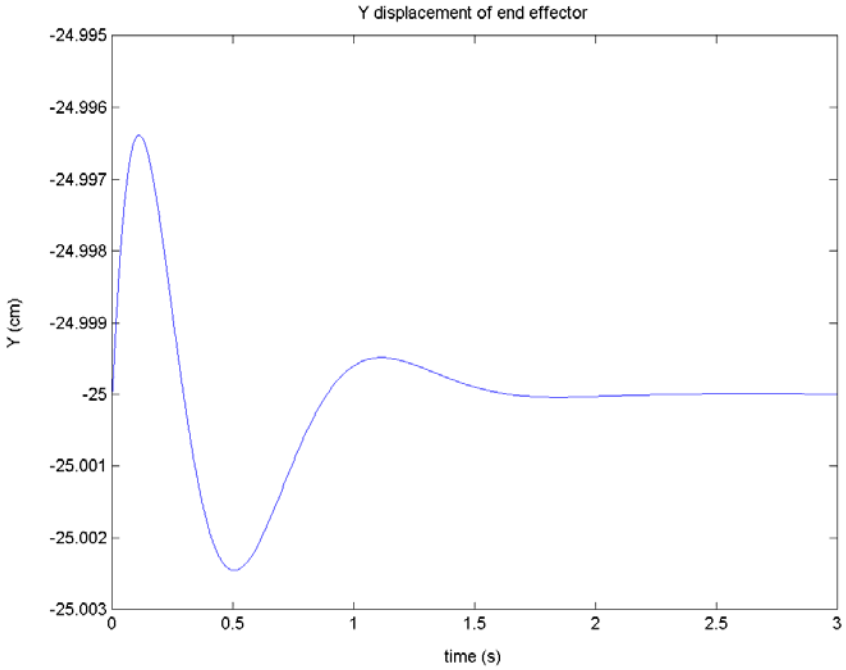


Figure 5-10. Y-displacement of end-effector for third run.

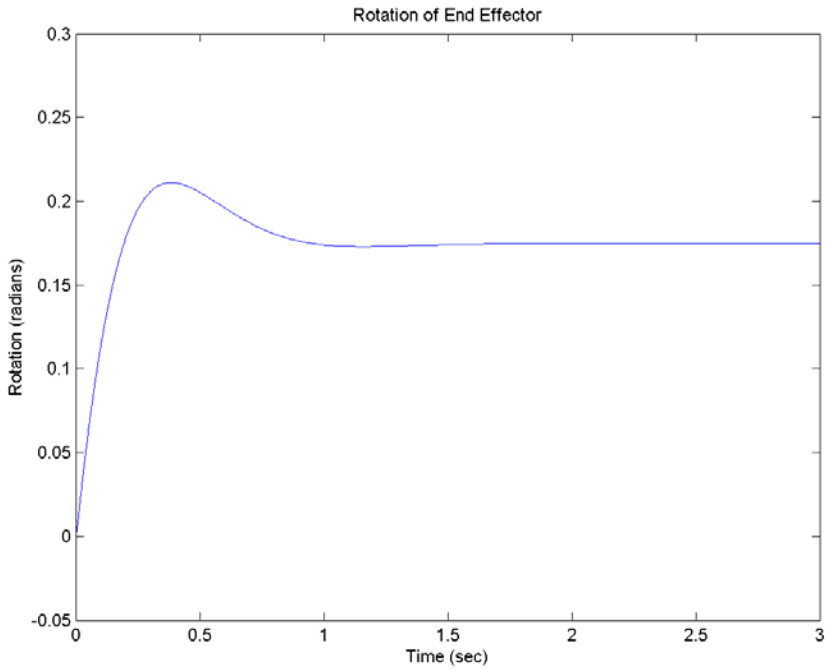


Figure 5-11. Θ-displacement of end-effector for third run.

Table 5-3. Results of Run 3.

	Desired Position	Final Position	Maximum Position	Max Overshoot
X	0 cm	0 cm	-0.0014 cm	n/a
Y	-25 cm	-25 cm	-25.0025 cm	n/a
θ	0.1745 rad	0.1745 rad	0.2111 rad	21%

Run 4

The final run tested the response of the system when the platform was commanded to move in all three directions from its initial position of (0 cm, -25 cm) to a final position of (-15 cm , -35 cm) with a rotation of 20 degrees. The results are shown in Figures 5-11 through 5-13.

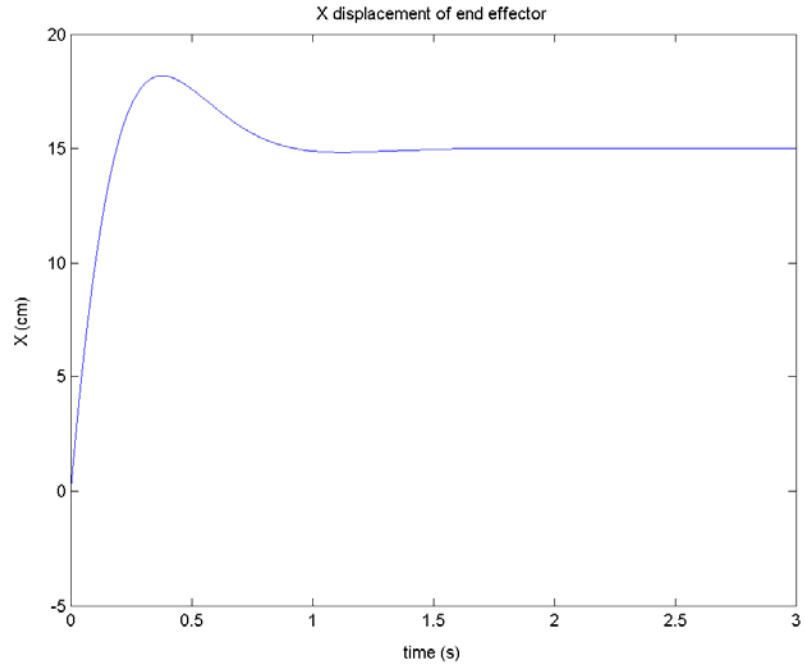


Figure 5-11. X-displacement of end-effector for fourth run.

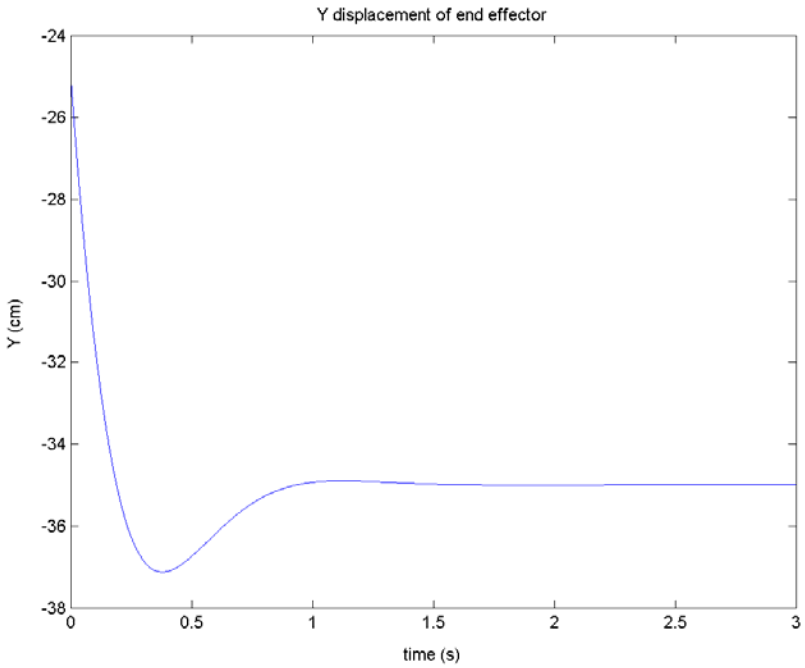


Figure 5-12. Y-displacement of end-effector for fourth run.

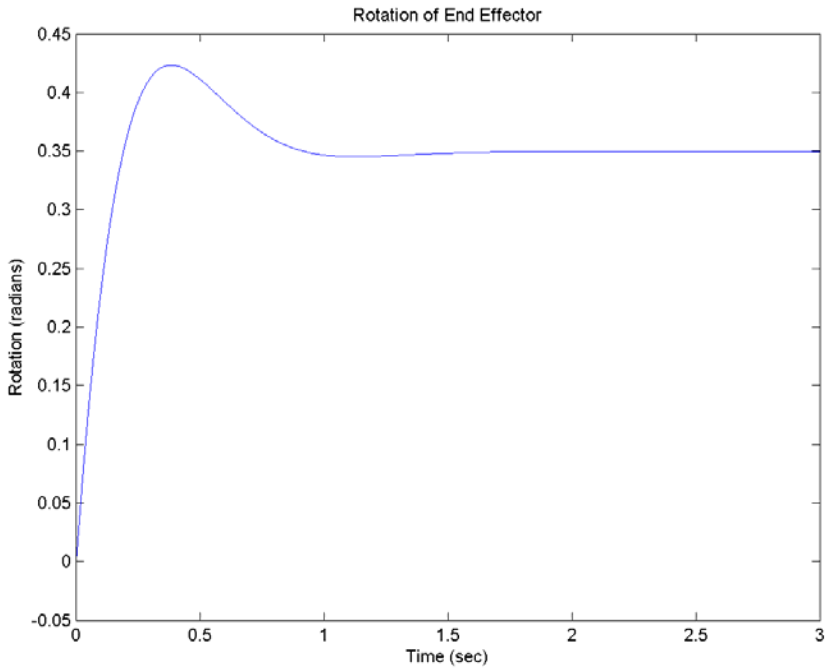


Figure 5-13. Θ-displacement of end-effector for fourth run.

Table 5-4. Results of Run 4.

	Desired Position	Final Position	Maximum Position	Max Overshoot
X	15 cm	14.9999 cm	18.1902 cm	21.3%
Y	-35 cm	-35 cm	-37.1259 cm	21.3%
θ	0.3491 rad	0.3491 rad	0.4233 rad	21.3%

Despite being commanded to move in all three directions simultaneously, the system behaved very well, meeting all design specifications. This shows that the controller design was a success.

Final Comments

The testing with the planar mechanism served several purposes. First, it verified the generalized control strategy for these parallel mechanisms and showed that the systems can be decoupled fairly well, even with simple controllers; the decoupling is not perfect, showing perturbations in the other degrees of freedom, but these are relatively small. Secondly, the planar mechanism revealed the potential difficulties that can arise with asymmetric mechanisms. These are far more difficult to decouple due to the

imbalanced nature of force application in various regions of the system. This can result in excessive motion of parts of the system that are meant to remain stationary and can also cause significant steady-state error. However, these issues were readily resolved with a slight redesign of the controllers. Despite minor setbacks and challenges, the testing of the planar mechanism was highly successful and accomplished all of the goals that were specified.

CHAPTER 6
SPATIAL MECHANISM AND ADDITIONAL CONSIDERATIONS FOR CONTROLLER DESIGN

Once the control strategy had been verified with the planar mechanism, the next step was to consider how to handle the spatial mechanism. The spatial mechanism had two major differences from the planar system analyzed in the previous chapter. First, it was an over-constrained system, meaning that more control actuators were present than were actually needed for motion in all degrees of freedom. The second difference was that the system was meant to have compliant components in the legs. These changes necessitated that additional measures be taken when designing the controllers for the system.

General System Description

Determining the configuration of the spatial mechanism was one of the critical steps. Since the objective of this project was to design a generalized controls strategy for an over constrained parallel mechanism, a symmetric structure was selected. The advantage of having a symmetric structure was highlighted when analyzing the planar mechanism. Symmetry in structural design and motion are highly desirable because the behavior of the system becomes far more predictable as the coupling effects appear to be minimized.

The platform was selected to follow a classical 3-3 configuration with the addition of a central leg that connected the centers of the base platform and the end effector. When placed in a "neutral" position that centered the end effector above the base, the

structure was symmetric and stable. Any other placement of the central leg would have led to asymmetry in the system. Figure 6-1 shows the generalized configuration for the mechanism. Both the top platform and the base were modeled as equilateral triangles, with corners placed at 0.5 meter and 1 meter from their centers respectively. At the unloaded home position the top platform was parallel to the base platform and was located 1 meter above it. The stiffness values used in the leg connectors will be discussed in subsequent chapters.

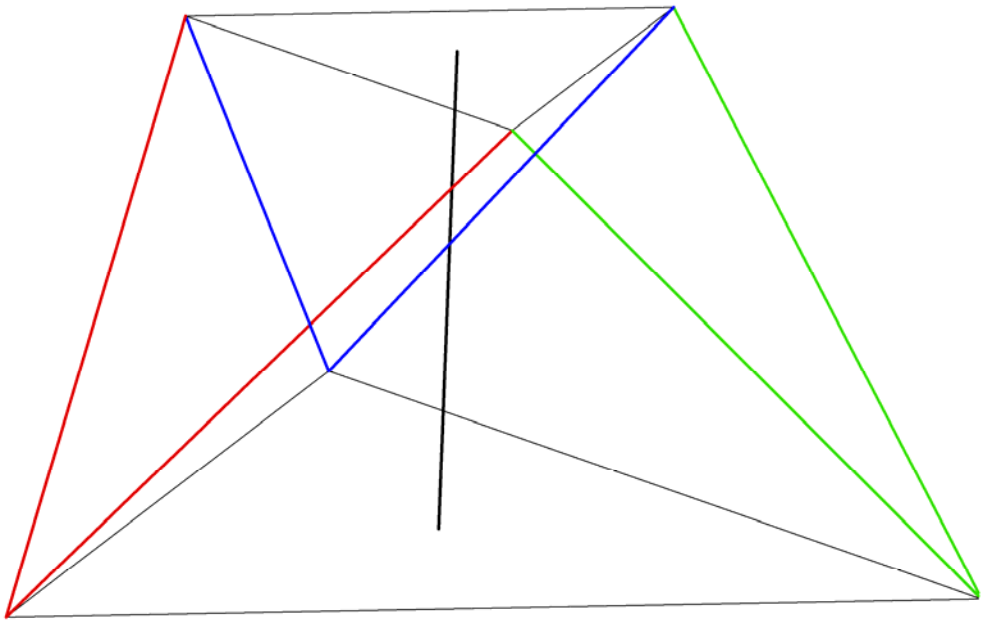


Figure 6-1. Layout of the spatial mechanism and connection points. The base is larger than the end effector, which is centered above the base triangle.

It is also important to note that the other spatial mechanism that was analyzed was a slightly modified version of the original, in that the location of the end of the central leg, relative to the end effector, was shifted to determine the efficacy of central leg at

countering rotation in certain situations. The exact modifications are described in Chapter 9 to highlight the differences in results.

Modifications to Control Strategy

The differences between the planar and spatial mechanisms analyzed in this project necessitated certain changes to the control strategy. The most important was the need to decouple the new elements so that the spatial mechanism would have similar behavior to the planar mechanism, albeit with a greater number of degrees of freedom.

Compensating for Compliance and Over-Constraint

Adding compliance to the system changed its behavior since the legs no longer responded as stiff elements. For the control strategy tested in Chapter 5 to work, the compliant elements had to be decoupled from the system with their own set of controllers so that the legs would once again behave as rigid components. Chapter 7 discusses the steps that were taken to decouple the legs and restore "normal" control to the system.

While utilizing an over-constrained system is not overly problematic, it presents a special set of challenges as the standard methods for determining the force distribution in the system break down. The issue is that the Jacobian matrix used to calculate the net wrench applied to the end-effector is no longer square, hence it is non-invertible. Fortunately, an easy solution presented itself during the decoupling of the compliant elements in the system. Since each leg has compliance, minimizing the energy in the system would allow one to determine the optimal force distribution in the system. The minimum energy optimization is presented in Chapter 8.

Once the legs had been decoupled and an effective method had been developed for finding the optimal force distribution in the system, the original control strategy could be implemented. The mechanism now had six degrees of freedom, each of which would

require a separate controller. These controllers would be designed in the same way as for the planar mechanism, under the assumption that the system was sufficiently decoupled to have good response and behavior. However, the need for one additional adjustment became readily apparent once work began on laying out the control system.

Coordinate System Conversion

The method for specifying the orientation of the platform is now laid out. A series of three rotations are needed, first about the local X-axis, then about the new local Y-axis, and finally about the new local Z-axis. Each rotation is performed about local coordinates, *not* global coordinates, meaning that each successive rotation was directly dependent on the previous one. Since the controller was meant to handle local coordinates, but the control forces were being applied in global coordinates, the torques applied by the legs had to be rotated into the global coordinate system for the minimum energy optimization to work properly.

The rotation matrix necessary for the coordinate system conversion was readily calculated using the methods laid out in reference [2] as follows

$$
\begin{aligned}
R_{global} &= R_X R_Y R_Z \\
R_X &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_X) & -\sin(\theta_X) \\ 0 & \sin(\theta_X) & \cos(\theta_X) \end{bmatrix} \\
R_Y &= \begin{bmatrix} \cos(\theta_Y) & 0 & \sin(\theta_Y) \\ 0 & 1 & 0 \\ -\sin(\theta_Y) & 0 & \cos(\theta_Y) \end{bmatrix} \\
R_Z &= \begin{bmatrix} \cos(\theta_Z) & -\sin(\theta_Z) & 0 \\ \sin(\theta_Z) & \cos(\theta_Z) & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\qquad \textbf{(6.1)}
$$

By multiplying together the rotation matrices about the respective degrees of freedom, in the order of rotation, a conversion matrix could be derived. Multiplying by the three element vector representing the local control torques, the global control torques can be calculated. The conversion is

$$\begin{Bmatrix} \tau_X \\ \tau_Y \\ \tau_Z \end{Bmatrix} = R_{global} \begin{Bmatrix} \tau_{x,local} \\ \tau_{y,local} \\ \tau_{z,local} \end{Bmatrix} \tag{6.2}$$

Combining the torque vector with the three element force vector produces the net wrench that the control legs need to apply to the end effector

$$w_{control} = \begin{Bmatrix} F_X \\ F_Y \\ F_Z \\ \tau_X \\ \tau_Y \\ \tau_Z \end{Bmatrix} \tag{6.3}$$

Finally one is able to lay out the structuring of the controller for the full spatial mechanism. It requires knowledge of both the destination and the current positioning, which are then fed into the separate controllers for the individual degrees of freedom. The control forces are then rotated into global coordinates as appropriate and then recombined so that the minimum energy optimization determine the optimal force distribution in the legs. Figure 6-2 shows the wiring of the control block.

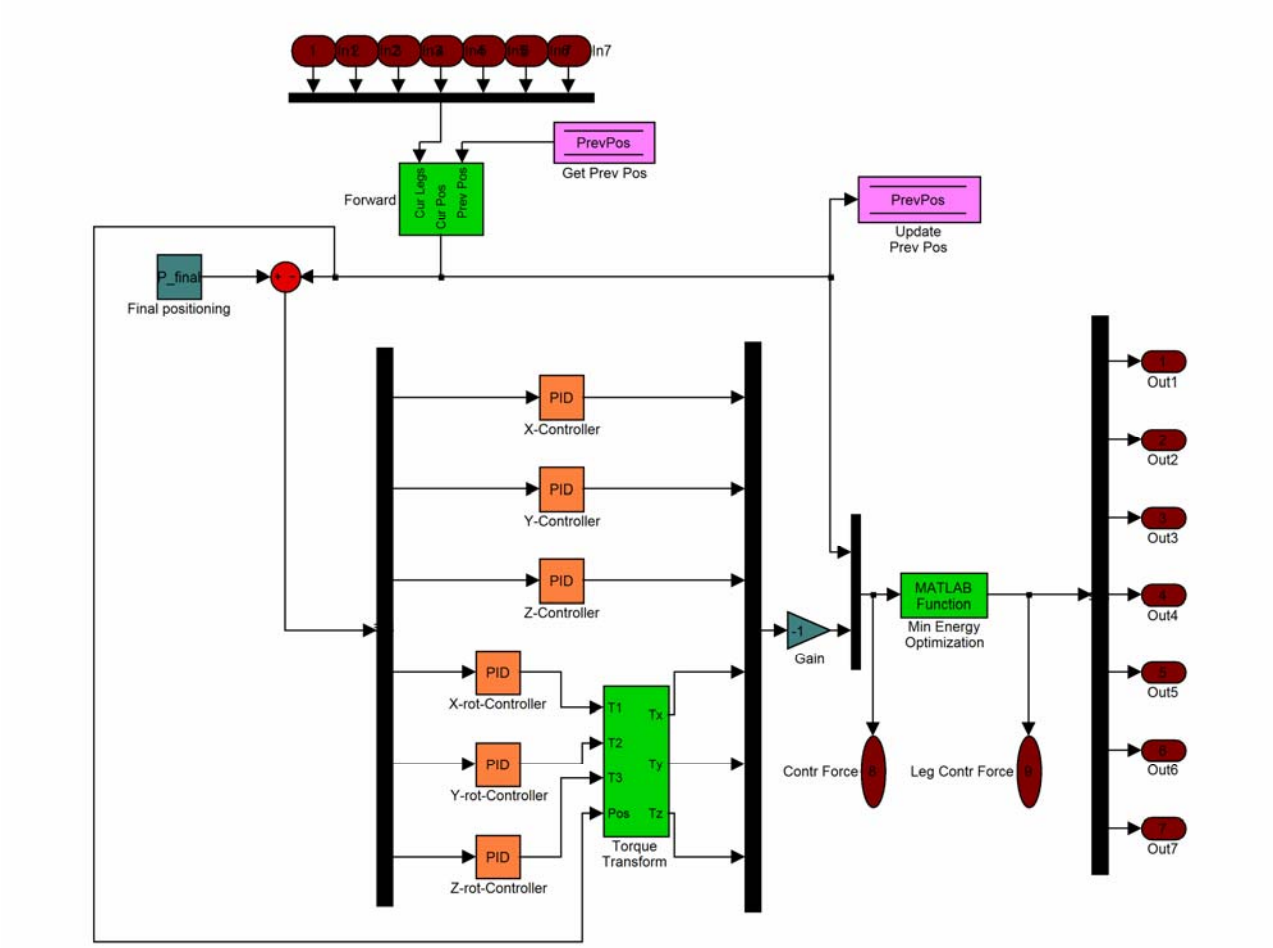


Figure 6-2. The control block used in the spatial mechanism.

In Figure 6-2, note the decoupling of the separate controllers, followed by the recombination to form a single desired wrench. The lengths of the seven legs are fed into the numerical forward analysis, along with the latest known positioning, which acts as the current guess. Once the current position has been determined, the position error is calculated and split up amongst the separate controllers. The outputs of the rotational controllers need to be rotated into global coordinates, then recombined with the corrections from the X, Y, and Z controllers. The net control wrench is then sent to the minimum energy optimization, which determines the distribution of control forces that are sent to the individual legs.

CHAPTER 7
DECOUPLING COMPLIANT ELEMENTS FROM LEGS

The compliance in the legs of the platform presented special challenges when designing the controllers. The major issue was that the legs would act as two degree of freedom systems, rather than as the single degree of freedom actuators desired. This meant that an extra set of controllers was needed to overcome this potentially crippling problem. The first step to resolving this difficulty was developing a model of one of the legs.

Leg Model

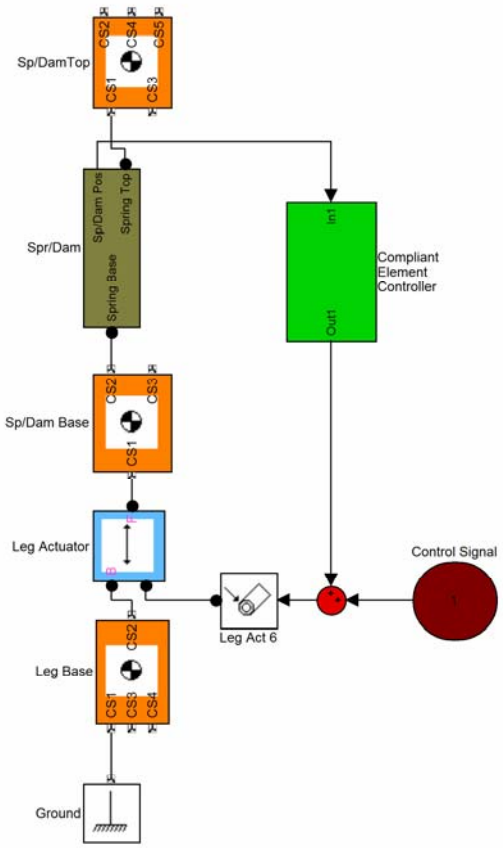


Figure 7-1. Generic model of a leg containing compliant elements.

The legs were modeled as having negligible mass, but that still did not resolve the question of how to decouple the compliant elements. At least a certain amount of mass was needed for an effective controller to be developed. One can see in Figure 7-1 that the leg model consisted of three mass elements, a spring, and a damper. The most significant mass was located at the free end, representing the end effector. The actual details of this mass were unimportant as the control forces to be applied to the platform were already known. The other mass that played a role was the actuator element (lumped with the spring-damper base mass) since it was directly connected to the spring and damper.

As previously stated, the force to be applied to the end effector had already been calculated by the primary controller, on the assumption that the compliance in the legs would be decoupled effectively. This effectively reduced the problem to controlling the motion of the actuator element. A free body diagram of the actuator could now be drawn that would allow the equations of motion for that element to be written.

Free Body Diagram and EOM

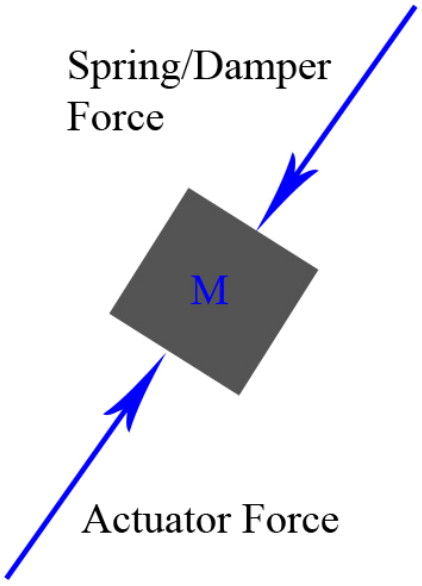


Figure 7-2. Free body diagram of the actuator mass with reactionary forces from the springs and dampers.

The forces acting on the actuator mass consist of four parts:

- Actuator force. This is what actually causes motion in the system. It also has to compensate for the reactionary forces.
- Acceleration of the actuator mass. The nature and effect of this depends heavily on the mass of this element.
- Spring force. This is simply determined by how much the spring is compressed.
- Damper force. This is purely based on how fast the actuator is moving with respect to the end effector.

The force contributions from the spring and damper can be lumped together because they *must* equal the force that the leg in question should apply to the end effector. This simplifies the problem into a very familiar form.

$$F_{act} - m_{act}\ddot{x} - F_{platform} = 0$$
$$m_{act}\ddot{x} = F_{act} - F_{platform} \quad \textbf{(7.1)}$$

A controller could be developed in exactly the same way as for the main platform. In theory, a basic PD controller should completely counteract the effects of the actuator mass, the spring, and the damper. The controller now took on the following form:

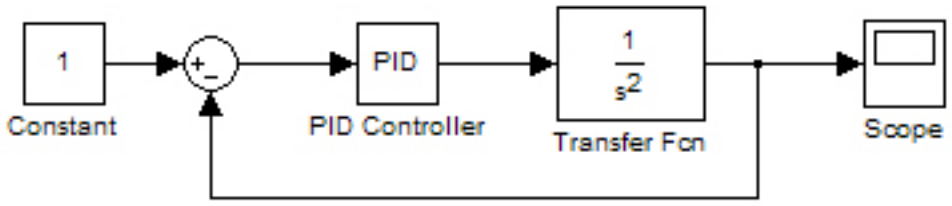


Figure 7-3. Controller for the leg actuator. Its form is virtually identical to those of the end effector.

The values for the controller were calculated in the same way that those of the main controllers were determined. Testing of the decoupling controllers will be discussed in the next section.

Performance of the Decoupling Controllers

Testing of the individual legs revealed that the controller design worked extremely well. The spring-damper had no ill effects on the motion of the end effector, though the

actuator system did tend to show some steady-state error as the mass of the actuator increased. The models used for testing are shown below and are also presented in Appendix A.

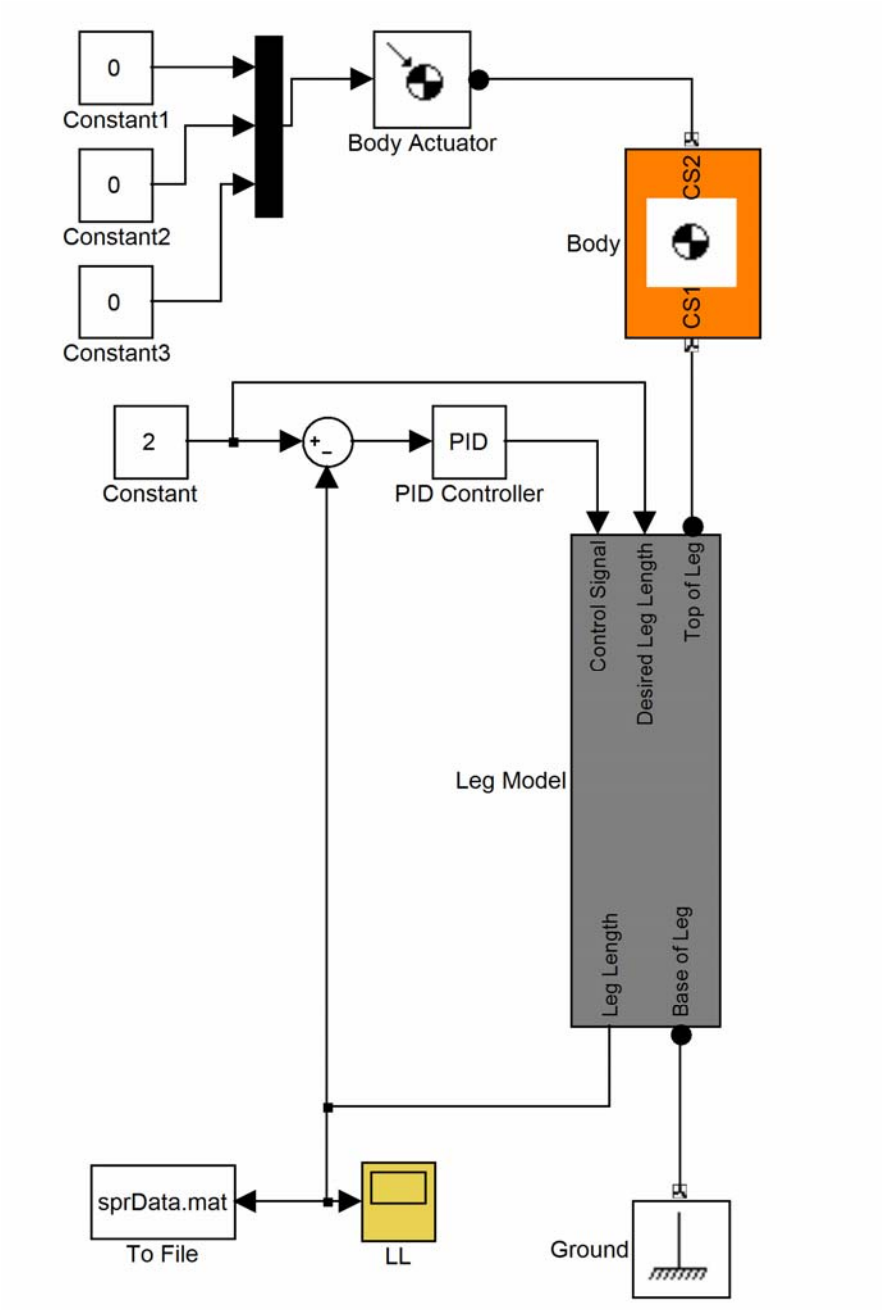


Figure 7-4. Overview of leg model. This shows the primary controller and a representation of the entire leg, including compliant elements. The constants feeding the body actuator represent the components of an external wrench that might be applied to the system.

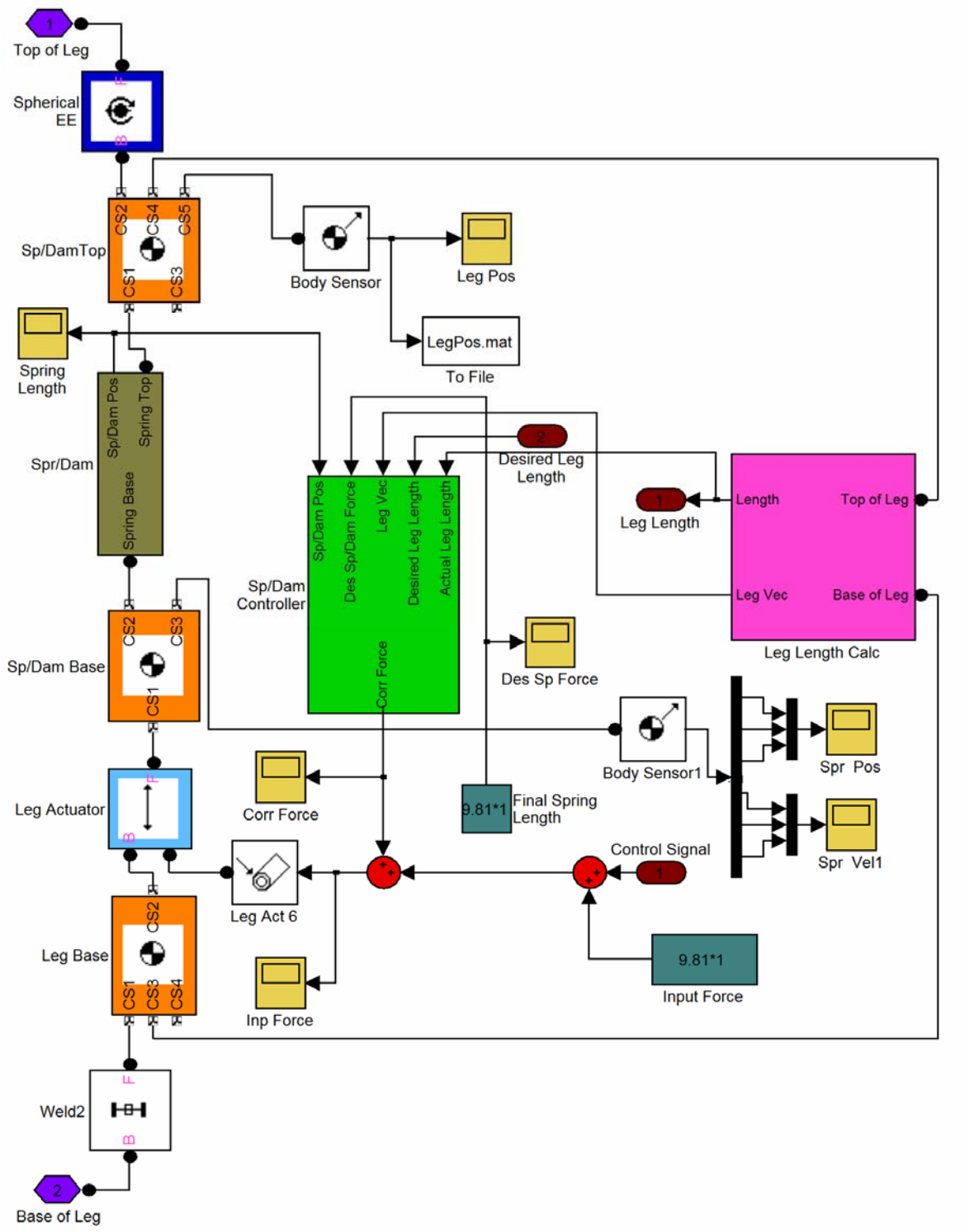


Figure 7-5. The actual leg model itself. The green block is the decoupling controller, while the brown block contains the compliant elements of the leg. One will notice the presence of the actuator and the combination of the control signal with the decoupling signal.

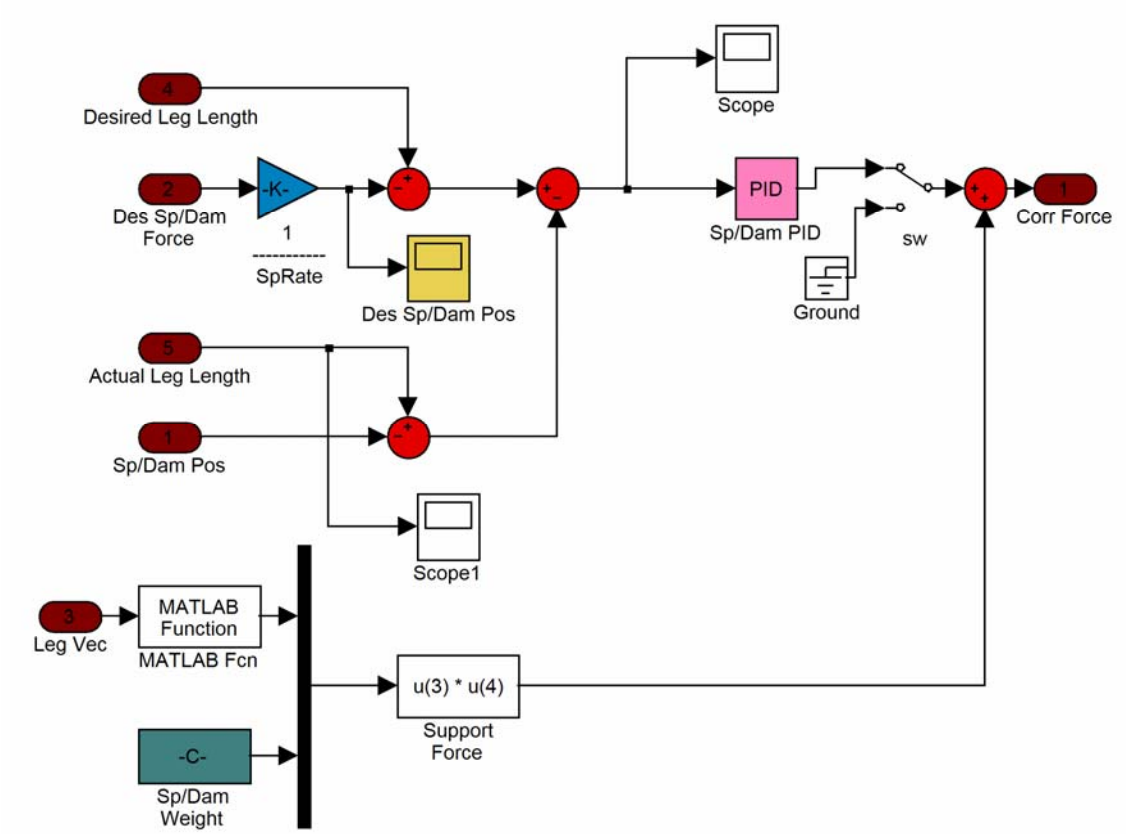


Figure 7-6. Controller for decoupling the compliant portion of the leg. The signal generated by this subsystem is combined with the general control force and serves to decouple the compliant elements.

Results of Testing

The decoupling controller showed exceptional performance, even with huge actuator/platform mass ratios. The system exhibited the good behavior of a well-tuned, classical second order system, despite the fact that it was actually a rather nasty looking two degree of freedom model. To test the effectiveness of the approach, a leg actuator mass of 1 kg was used, which was the same as the mass of the representative "end effector". The leg was commanded to move from an initial position of 1 m to a length of 2 m. Without the decoupling controller in effect, the system exhibited 33.5% overshoot and a settling time of 1.75 second. With the controller activated, the overshoot dropped

to 18.7% and the settling time to 0.92 second. The results are shown in Figure 7-7 and clearly demonstrate the effectiveness of the controls strategy.

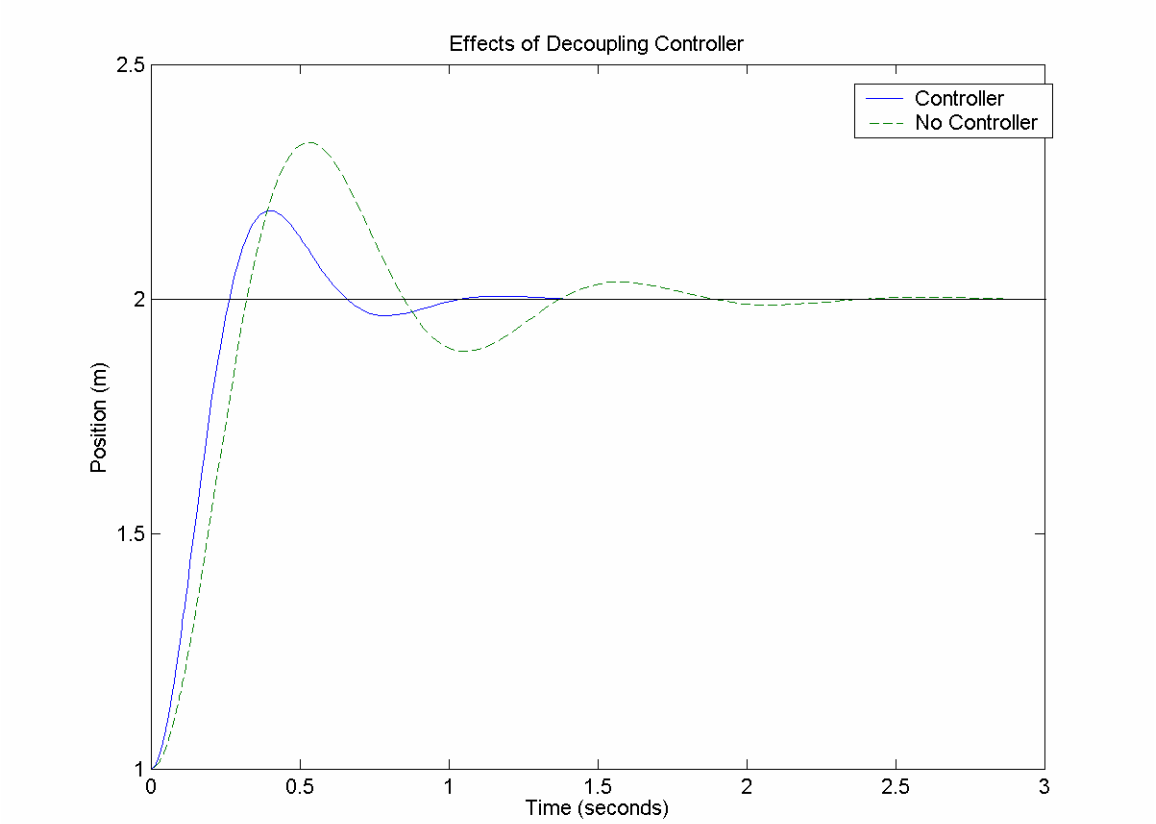


Figure 7-7. Results of testing both with and without the decoupling controller.

Limitations of Controller Usage

The only limitation on use of the decoupling controllers was not the design, but rather the simulation software used. Both MATLAB 6.5 and ADAMS had significant trouble with simulating parallel mechanisms. The solvers in both programs found difficulty when dealing with any parallel devices because of the multiple closed loops present in the systems. ADAMS simply couldn't handle the systems, while MATLAB would sometimes struggle. Even without compliant elements, the solvers would occasionally slow down to time steps on the order of 10^{-9} to 10^{-11} second. When *any* compliant elements were included, the time steps reduced to 10^{-11} to 10^{-13} second, making

completion of any simulations impossible. This was an unfortunate setback, but testing of the controllers for the leg actuators indicated that modeling the legs as stiff elements was accurate enough to verify the effectiveness of the main control strategy. In theory, if the decoupling controllers could be run with the model, any combination of leg masses could be used, regardless of operating conditions, greatly improving the flexibility of the system design.

CHAPTER 8
MINIMUM ENERGY OPTIMIZATION

One of the critical considerations when working with an over constrained system is how to distribute the force loading amongst the various leg connectors to obtain static equilibrium at a given position. For a perfectly constrained system, only one possible force distribution exists since there exist only as many control actuators as degrees of freedom. However for any over constrained mechanism, extra controls beyond those required are present. While there exist an infinite number of solutions to the question of force distribution, most of them would be unreasonable for the simple fact that the loadings in the legs would be far beyond the limits of any current actuators or materials. Since the spatial mechanism described in Chapter 6 contains compliant elements, a method exists to find the optimal force distribution for the mechanism, minimizing the potential energy stored in the springs.

Potential Energy: General Form and Reasoning

Even though the various components in the mechanism will be moving as the platform travels towards its objective, the ultimate goal is to have the system at static equilibrium at that final positioning. This automatically means that the dampers in the legs will not be moving during this final state, requiring all supporting forces to be supplied by the springs. During the actual motion of the platform, the force loading in each leg will be distributed between the acceleration of the actuator mass, the velocity of the damper, and the compression of the spring. However, these do not need to be considered in this analysis because the compliant elements in the legs have been

decoupled as demonstrated in Chapter 7. Another use for the minimum energy optimization is that it allows the controller to very easily determine how to distribute the control forces amongst the various legs.

The potential energy equation follows the classical form. The spring rates in each leg are assumed to be constant and follow a linear relation to the compression. If $\mathbf{k_n}$ and $\Delta\mathbf{L_n}$ represent the spring rate and compression of leg **n**, then the total potential energy equation can be written as follows:

$$U = \sum_{n=1}^{7} \frac{1}{2} k_n \left(\Delta L_n\right)^2 \qquad \textbf{(8.1)}$$

While this equation comes as no big surprise, more information is needed to be able to solve the problem. Namely, the wrench that the legs must exert on the end effector is the key restriction on the system.

Force-Torque State and Equations

Since the platform is supposed to remain at static equilibrium at its final positioning, the wrench exerted by the legs must counteract any external wrench on the system, including gravitational effects and reactionary forces applied to the end effector. The wrench exerted by the legs can be calculated using the 6×7 Jacobian of the system as discussed in Chapter 3.

$$\begin{aligned} &\underline{w_{Legs}} + \underline{w_{ext}} = 0 \\ &\underline{w_{Legs}} = \left[J\right]_{6x7} \underline{F_{Legs}} \\ &\underline{F_{Legs}} = \begin{Bmatrix} F_1 \\ F_2 \\ \vdots \\ F_7 \end{Bmatrix} \Rightarrow F_n = k_n \left(\Delta L_n\right) \end{aligned} \qquad \textbf{(8.2)}$$

Determining the forces in the legs is not so straightforward because no true inverse exists for a non-square matrix. The form of the equations must be modified slightly to place everything in terms of one variable to optimize. The equation for the wrench exerted by the legs can be rewritten as follows:

$$\underline{w_{legs}} = [J]_{6x6}\,\underline{F_{1-6}} + \underline{w_7} = [J]_{6x6}\begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \end{Bmatrix} + F_7\,\underline{\$_7} \tag{8.3}$$

Since $\underline{w_{Legs}} + \underline{w_{ext}} = 0$, everything can be rewritten in terms of $\mathbf{F_7}$ and ultimately $\Delta\mathbf{L_7}$. This will then allow one to rewrite the potential energy equation in terms of one variable for minimization.

$$\begin{aligned} \underline{F_{1-6}} &= -\left([J]_{6x6}\right)^{-1}\left(\underline{w_{ext}} + \underline{w_7}\right) = -\left([J]_{6x6}\right)^{-1}\left(\underline{w_{ext}} + k_7\Delta L_7\,\underline{\$_7}\right) \\ \underline{F_{1-6}} &= \underline{A} + \underline{B}\left(k_7\Delta L_7\right) \\ \underline{A} &= -\left([J]_{6x6}\right)^{-1}\underline{w_{ext}} \\ \underline{B} &= -\left([J]_{6x6}\right)^{-1}\$_7 \end{aligned} \tag{8.4}$$

$$\begin{aligned} F_n &= k_n\Delta L_n = A_n + B_n\left(k_7\Delta L_7\right) \\ \Delta L_n &= \frac{A_n}{k_n} + \frac{B_n}{k_n}\left(k_7\Delta L_7\right) \end{aligned} \tag{8.5}$$

Now that the displacements of the various springs have been written in terms of the compression of the seventh leg, the potential energy equation can be rewritten, then minimized with respect to $\Delta\mathbf{L_7}$. One may notice that the displacements take on a very nice form that will be easy to work with.

$$U = \sum_{n=1}^{7} \frac{1}{2} k_n (\Delta L_n)^2$$

$$(\Delta L_n)^2 = \frac{1}{k_n{}^2}\left(A_n{}^2 + B_n{}^2 (k_7 \Delta L_7)^2 + 2A_n B_n (k_7 \Delta L_7)\right)$$

$$\frac{dU}{d(\Delta L_7)} = 0 = \frac{d}{d(\Delta L_7)} \sum_{n=1}^{7} \frac{1}{2k_n}\left(A_n{}^2 + B_n{}^2 k_7{}^2 \Delta L_7{}^2 + 2A_n B_n (k_7 \Delta L_7)\right)$$

$$\frac{dU}{d(\Delta L_7)} = \sum_{n=1}^{7} \frac{1}{2k_n}\left(2A_n B_n k_7 + 2B_n{}^2 k_7{}^2 \Delta L_7\right) = A_u + B_u \Delta L_7$$

$$A_u = \sum_{n=1}^{7} \frac{k_7}{k_n} A_n B_n$$

$$B_u = \sum_{n=1}^{7} \frac{k_7}{k_n} k_7 B_n{}^2$$

$$\Delta L_7 = -\frac{A_u}{B_u} = \frac{1}{k_7} \frac{\sum_{n=1}^{7} \frac{A_n B_n}{k_n}}{\sum_{n=1}^{7} \frac{B_n{}^2}{k_n}} \qquad \textbf{(8.6)}$$

The solution for the ideal compression of the seventh leg is fairly simple in form and subsequent calculations for the compressions (and ultimately forces) of the other legs follow easily. Any leg can be set as the "seventh" leg depending on the conditioning of the other six. There may be situations where one group of six legs are at a singularity while the others are not. A basic check for conditioning is required. However, one must note that this method still cannot handle a situation when the *entire* system is ill-conditioned. Remember that one of the major limitations of parallel mechanisms is their limited workspaces. The best way to avoid problems with singularities is to limit the range of motion of the device to a predetermined "safe" environment.

Final Comments

This chapter has shown a way to quickly and easily determine the ideal force distribution for an over constrained parallel mechanism. The process is relatively simple and can be programmed to handle certain contingencies. The process used to derive the

equations could be used to handle more complicated systems, though certain additional considerations would be necessary. Ill-conditioned systems are unavoidable with parallel mechanisms, but the use of over constrained devices can help to alleviate the problem.

CHAPTER 9
RESULTS OF TESTING ON THE SPATIAL MECHANISMS

This chapter focuses on discussing the performance of the controllers when applied to the spatial mechanisms, with primary attention paid to the main system. The testing done with the modified mechanism was purely to investigate the effects on rotational performance if the leg configuration was changed. A number of simulations were run to determine the behavior of the primary system when moving in one or more degrees of freedom. Six of the simulations focused solely on coupling effects by moving in the six degrees of freedom separately. Further testing combined motions in two or more degrees of freedom as these represented more realistic utilization of the system.

Motions in Single DOF

This was the most critical set of experiments since it tested the basic viability of the control strategy. Proving that controlled motion in a single DOF is possible will demonstrate at least simple functionality. The following tests showed that not only does the control strategy work well, but the coupling effects are relatively small. Each time, the mechanism started at a symmetric position of [0 m, 0 m, 1 m; 0 rad, 0 rad, 0 rad], which represents the X, Y, Z positions and the rotations about the X, Y, and Z axes respectively. This had the advantage of being a naturally stable configuration in simulation, helping to minimize the calculational errors. The importance of symmetry has already been demonstrated with the planar mechanism, and this is no exception. Unless stated otherwise, the specified performance for each test was less than 20% overshoot with a settling time of less than 1 second. More information of initialization

and configuration of the system can be found in the initialization file located in Appendix B, along with important code that is necessary for the simulation to run. The dimensions of the upper platform and the base were discussed in Chapter 6. The legs were essentially split into two sections, a lower connection that comprised half of the initial length, and the upper portion that was attached to the end effector, making up the other half of its length.

Z-Axis Displacement

The first test was displacement along the Z-Axis to a final height of 1.5 m. This motion causes a fairly sizeable change in positioning and platform configuration, but maintains the symmetry, which helps to minimize the coupling effects. As one can see in Figures,9-1 and 9-2, the system responded quite well, with essentially negligible coupling effects.

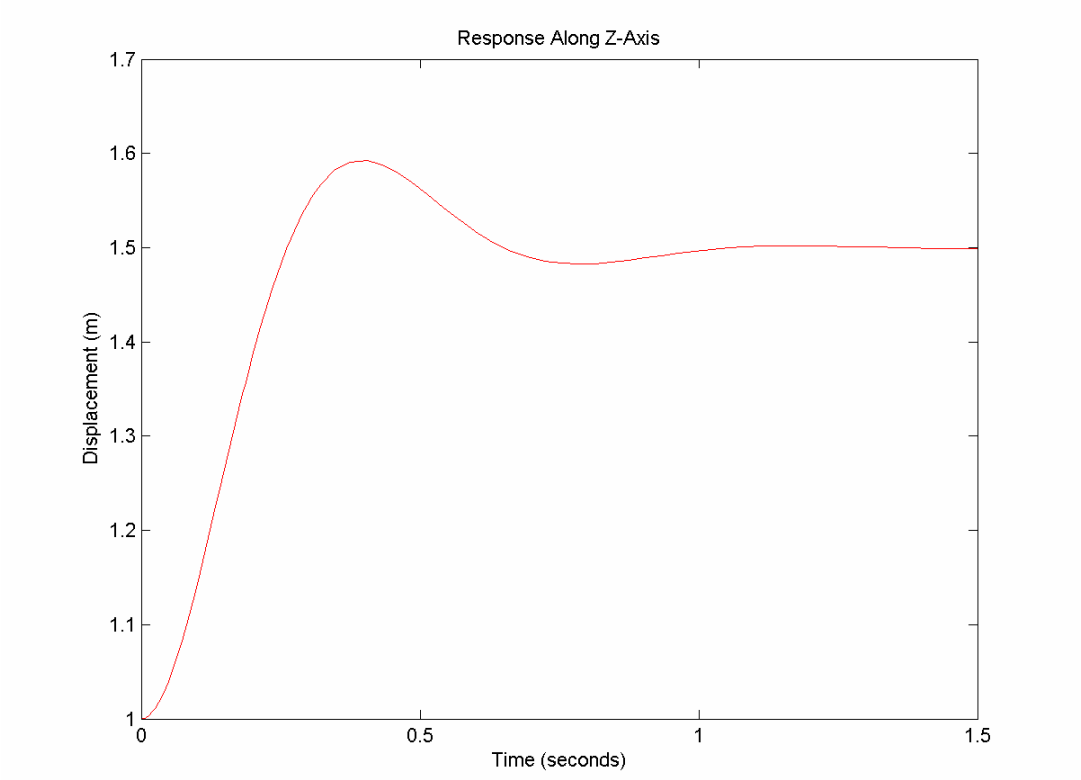
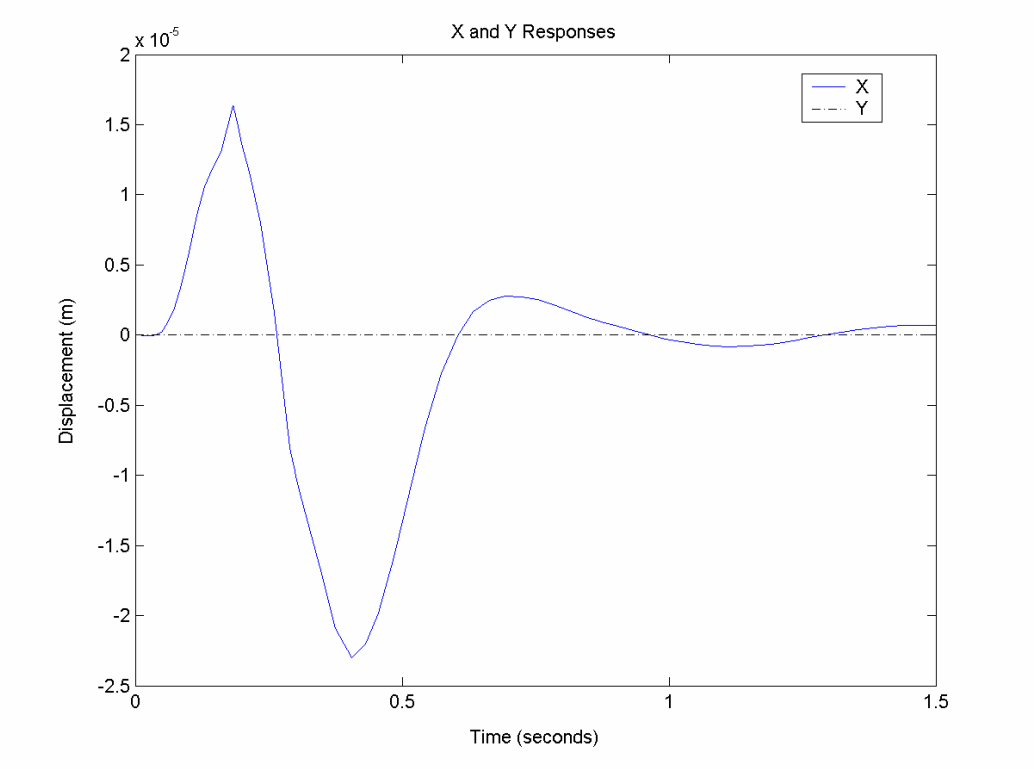


Figure 9-1. Motion of the mechanism along the Z-Axis.

(a)



(b)

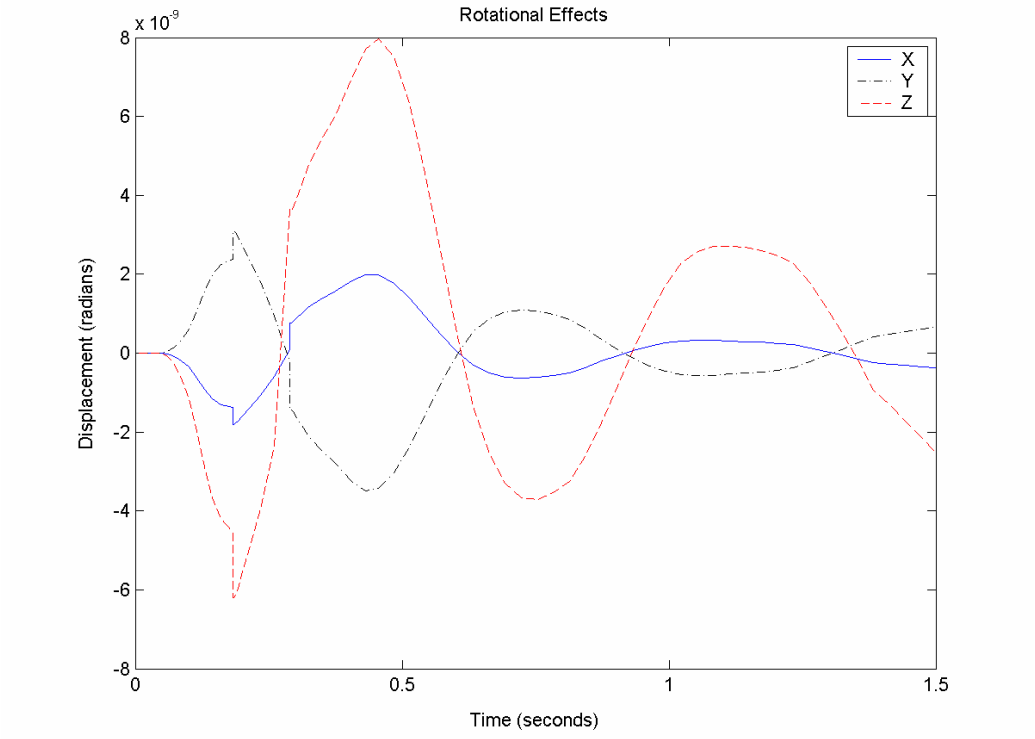


Figure 9-2. Motions in the other DOFs. (a) Y and X axes. (b) Rotations about the primary axes.

The system reached its final positioning with no steady-state error and still met the performance requirements. System overshoot was 18.4% and settling time was 0.915 seconds. The maximum fluctuations in positioning of the other DOFs was extraordinarily small, with 1.7×10^{-4} m along the primary axes, and 8×10^{-9} radian in rotations.

Z-Axis Rotation

Considering the performance of the system when moving purely along the Z-Axis, one would naturally expect that similar performance would be possible in pure rotation. This still maintains the symmetry of the system, albeit within a more limited workspace; the platform was rotated 0.5 radian about the Z-Axis.

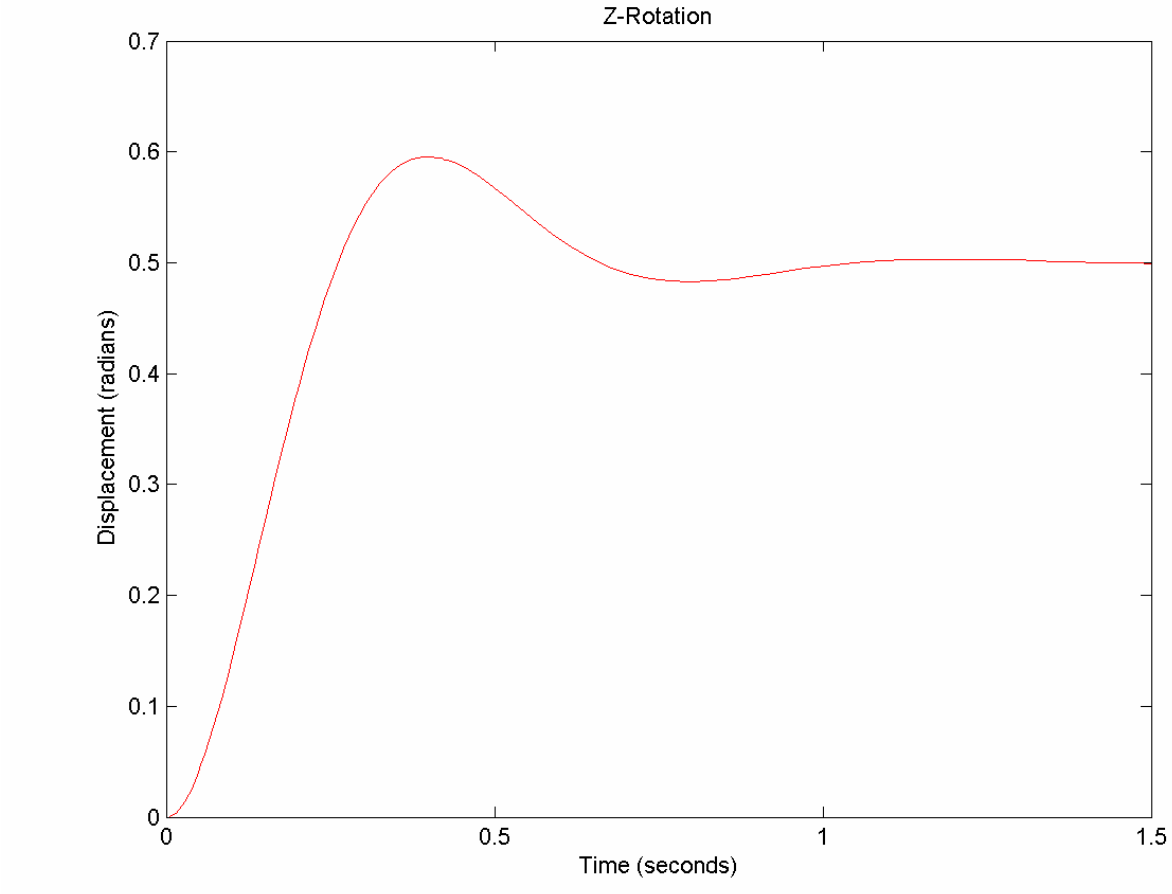
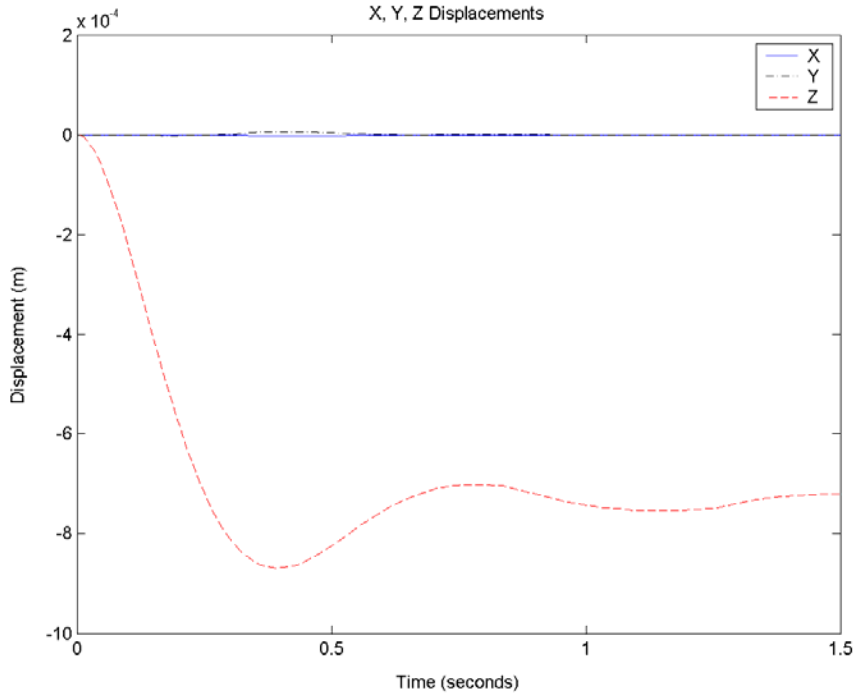


Figure 9-3. Rotation of the mechanism about the Z-Axis.

(a)



(b)

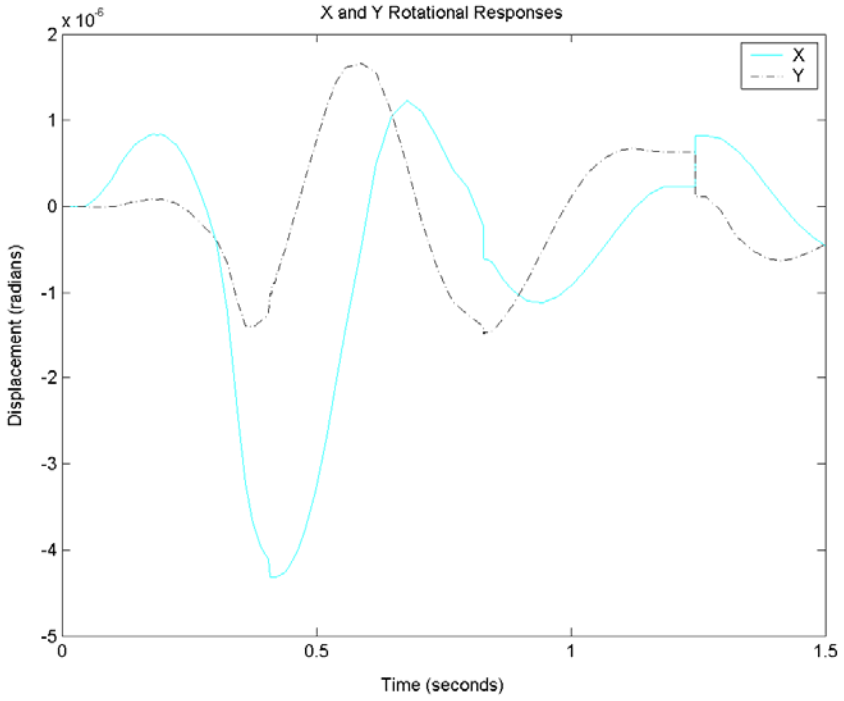


Figure 9-4. Motions in the other DOFs. (a) Disturbance of Z, Y, and X axes. (b) Rotations about the Y and X axes.

Once again, the system showed excellent performance as expected, exhibiting 19.1% overshoot and a settling time of 0.917 second. The maximum motion error in the other DOFs was $7x10^{-4}$ m along the primary axes and $4.3x10^{-6}$ radian in rotation.

Y-Axis Displacement

Motion along the Y-Axis is still pseudo-symmetric, though it is far less ideal than motions with the Z-Axis. The behavior of the system was expected to be good, but not as good as the previous tests since even slight asymmetry in a system can have enormous influence on response. The end-effector was commanded to move 1 m along the Y-Axis, with 10% overshoot and a settling time less than 1 second.

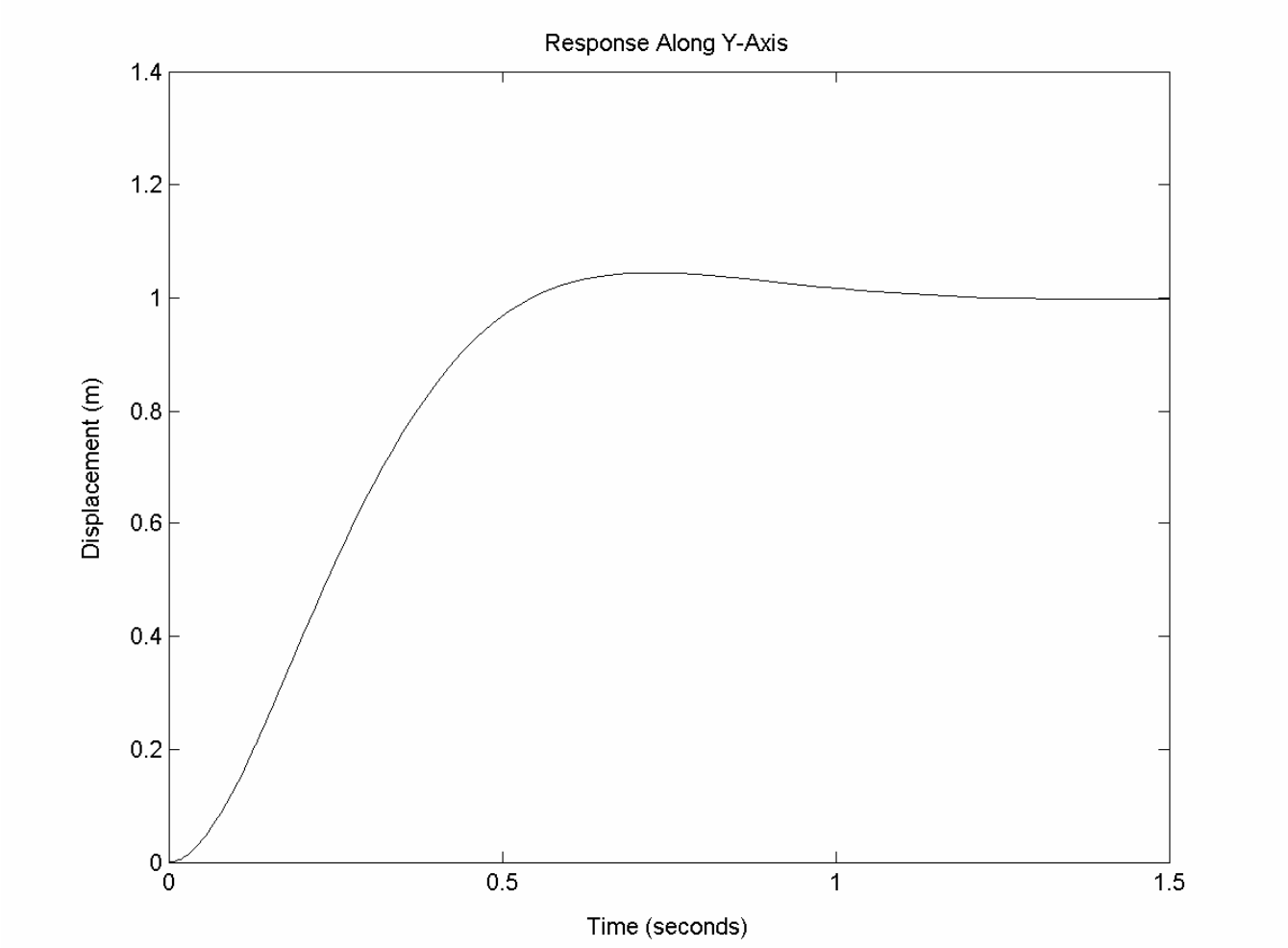
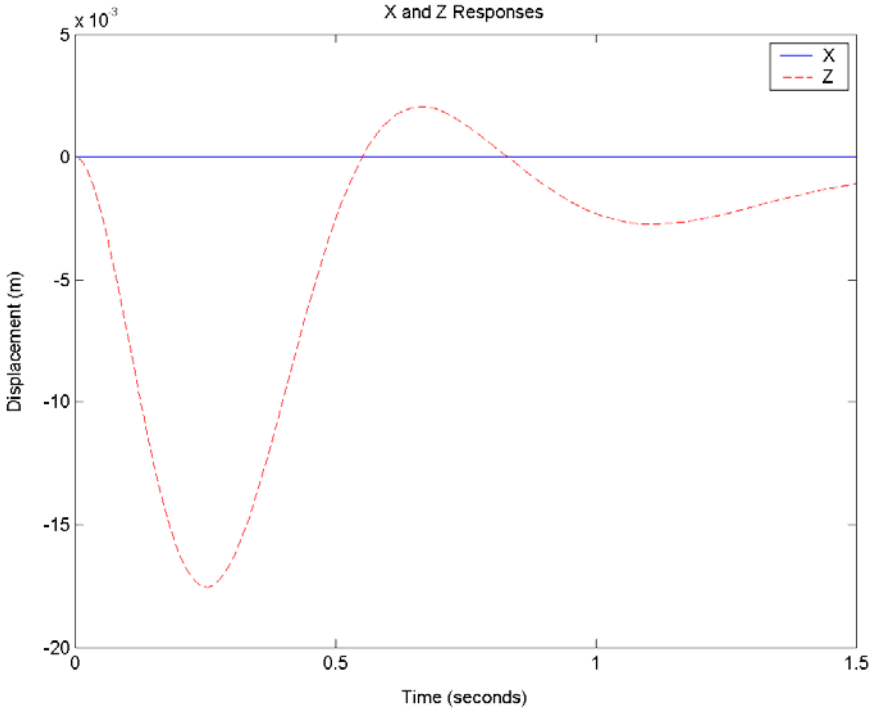


Figure 9-5. Motion of the mechanism along the Y-Axis.

(a)



(b)

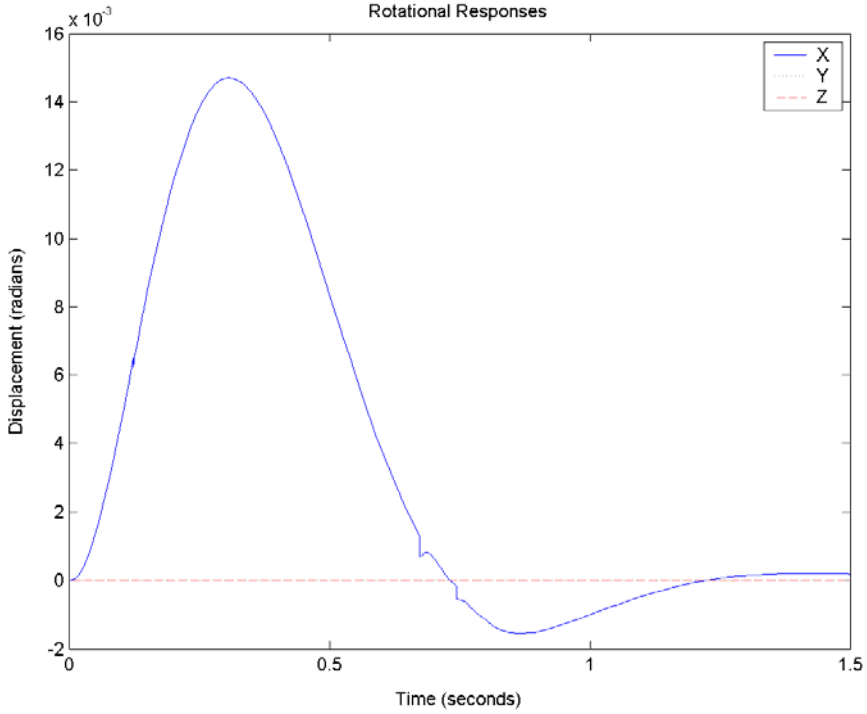


Figure 9-6. Motions in the other DOFs. (a) Disturbance of Z and X axes. (b) Rotations about the Z, Y, and X axes.

The system still responded extremely well, only exhibiting 9% overshoot and a settling time of 0.977 second. The maximum motion error in the other DOFs was $7.5x10^{-3}$ m along the primary axes and $1.5x10^{-2}$ radian in rotation. The largest errors were generated in the Z-Axis motion and the X-Axis rotation, both of which are perpendicular to the Y-Axis motion. The drop along the Z-Axis is attributable to the fact that the support legs are only located underneath the end effector. Therefore, trying to move along the Y-Axis will cause a localized drop in the height of the platform, until such time as the controllers can correct. Since the "top" corner of the end effector has only two support legs, but the "bottom" edge has four, an imbalance occurs when trying to control rotation about the local X-Axis. Since the end-effector drops slightly, one would also expect that one section will drop more than the others, due to asymmetry in support distribution, leading to unwanted rotations.

Y-Axis Rotation

Rotation about the Y-Axis is similar to the previous test in that the system is being actuated about a pseudo-symmetric lobe. Once again, good performance would be expected, though errors in other degrees of freedom should be more prevalent than the Z-Axis testing. The platform was commanded to rotate by 0.5 radian with no motion in the other degrees of freedom. The system showed good behavior in rotation with 8.5% in overshoot and a 0.681 second settling time. One may notice a slight steady state error of $7x10^{-4}$ m present in the positioning of the Z-Axis, which is relatively small considering the size of the mechanism at 1 m height. This may be partially attributable to the numerical forward analysis as that can result in a slight error in positioning information. Addition of an integral portion to the controller might also help to correct the error.

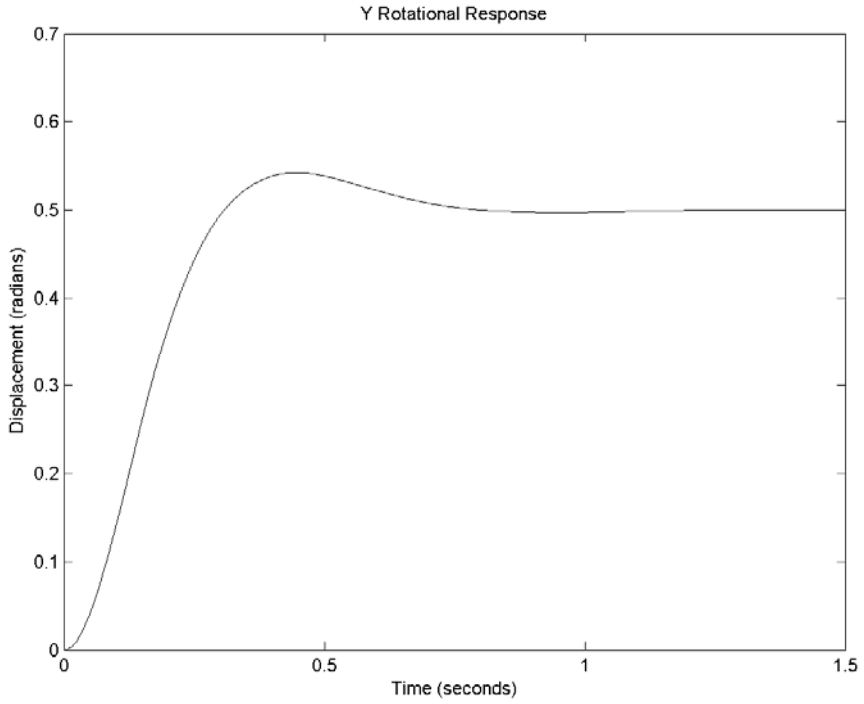
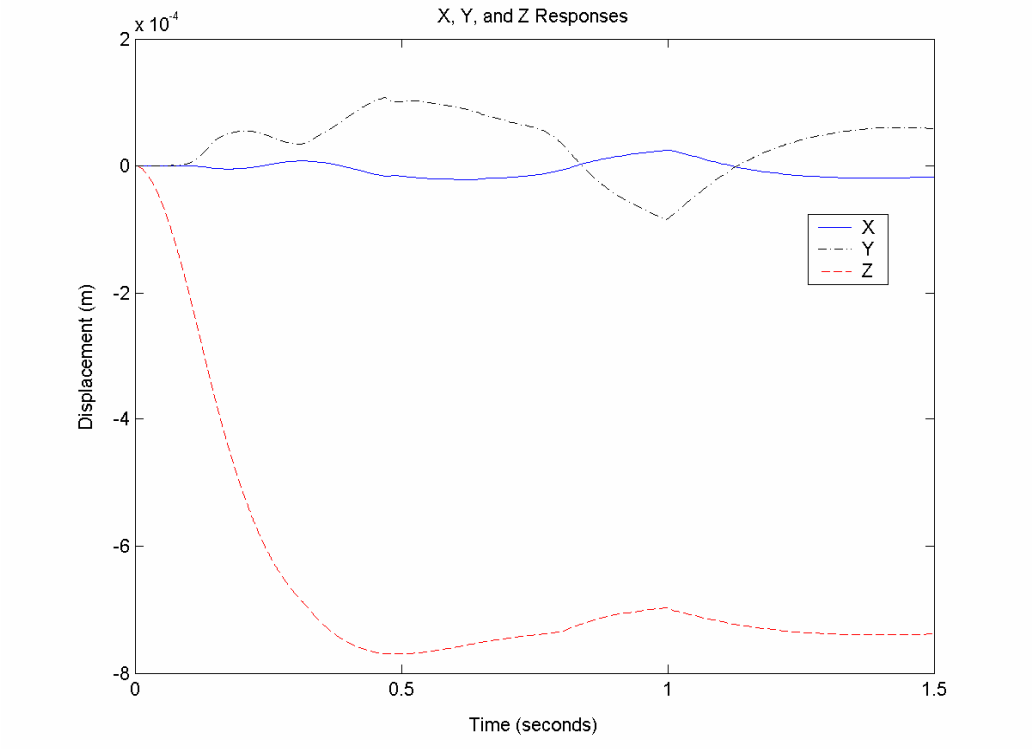


Figure 9-7. Rotation of the mechanism about the Y-Axis.

(a)



(b)

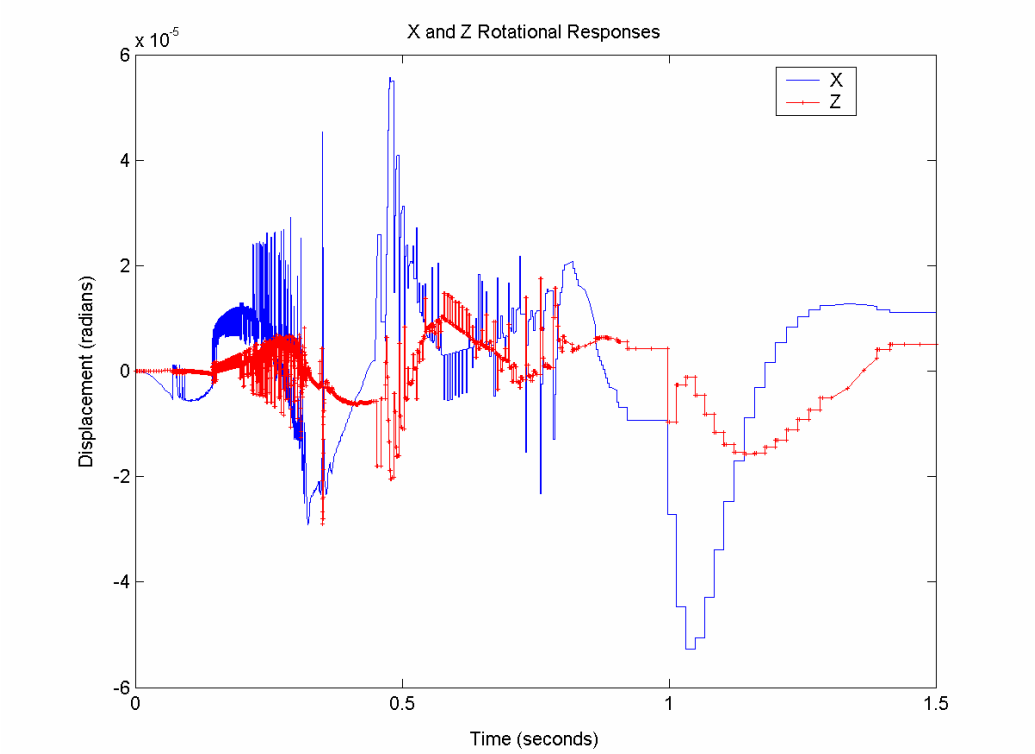


Figure 9-8. Motions in the other DOFs. (a) Disturbance of Z, Y, and X axes. (b) Rotations about the Z and X axes.

The apparent noise present in the response during this test is most likely due to numerical errors in the simulation. During the early stages of the tests, the solver would require extremely small time steps during convergence. The apparent stair-stepping later is a result of large steps as the simulation reached easier configurations for the solver.

X-Axis Displacement

Motion of the platform along the X-Axis showed similar performance to the tests commanding motion along the Y-Axis, with primary errors showing up in the vertical positioning of the platform, i.e. the Z-Axis displacement. This is hardly surprising considering that the system is moving along an asymmetric direction, which would tend to result in similar errors. The other aspects of the mechanism performed remarkably well, with the largest rotational displacement being 1.9×10^{-4} radian.

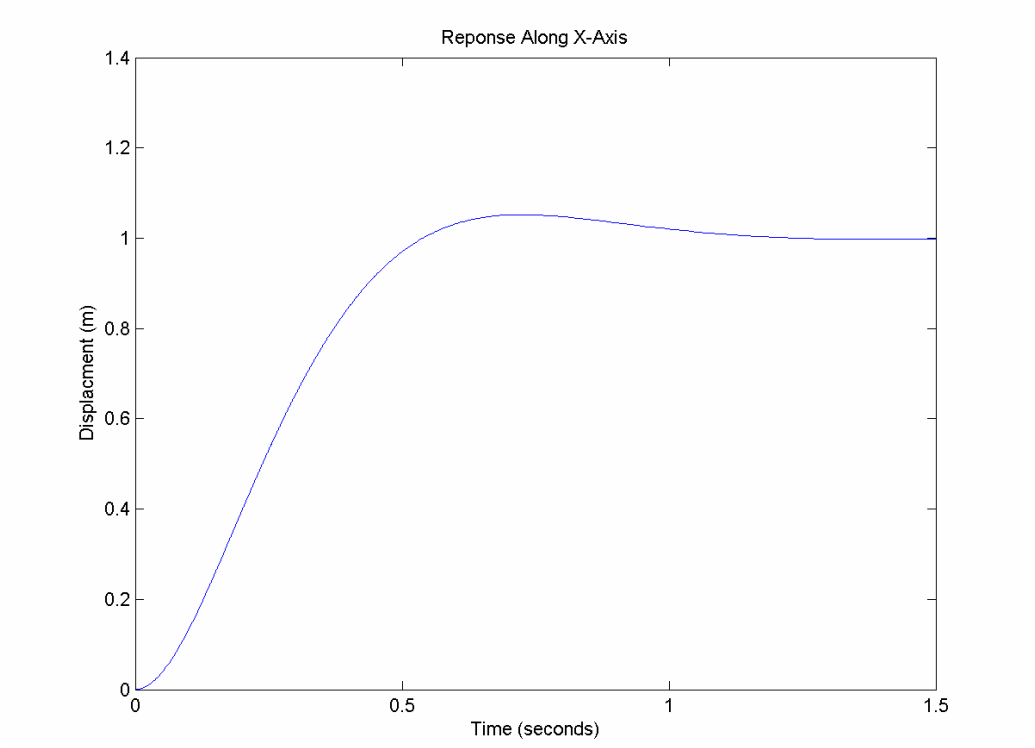
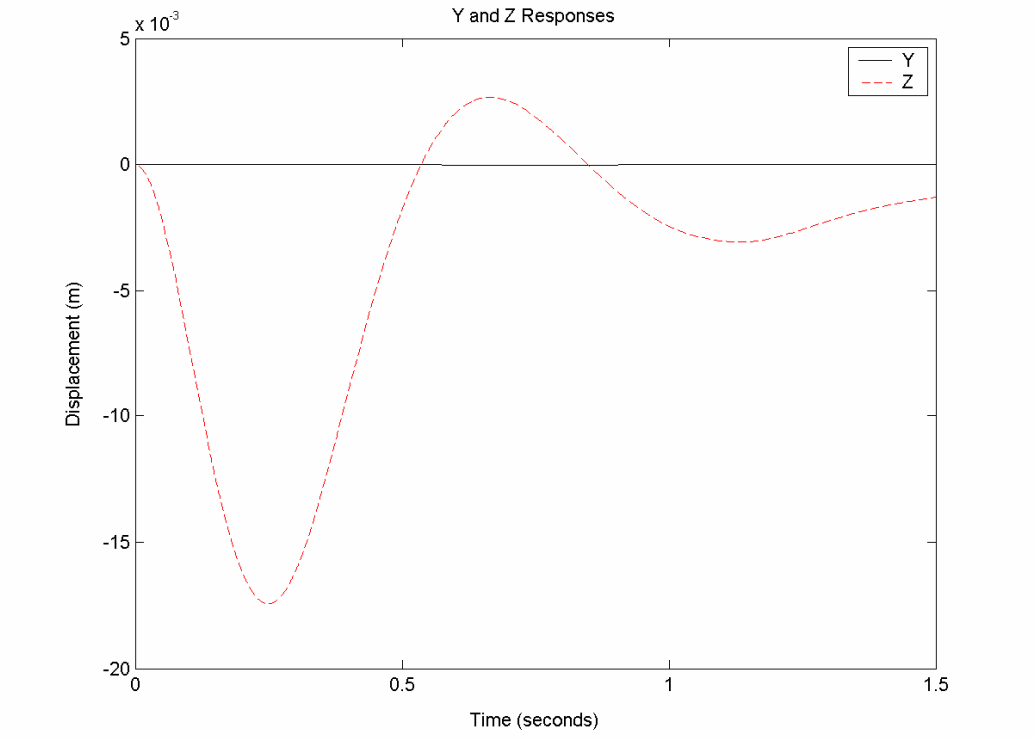


Figure 9-9. Motion of the mechanism along the X-Axis.

(a)



(b)

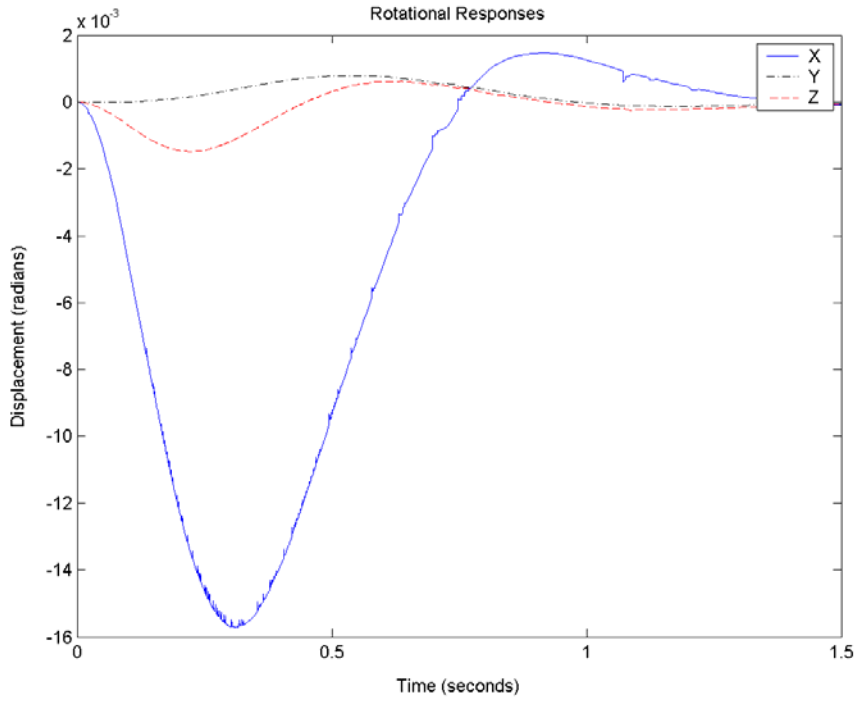


Figure 9-10. Motions in the other DOFs. (a) Disturbance of Z and Y axes. (b) Rotations about the Z, Y, and X axes.

X-Axis Rotation

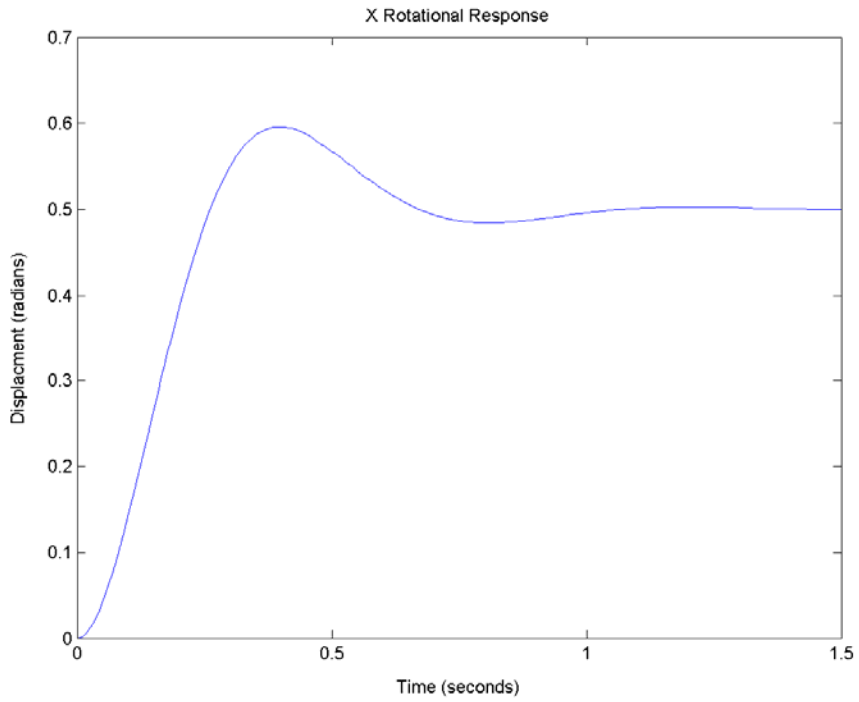
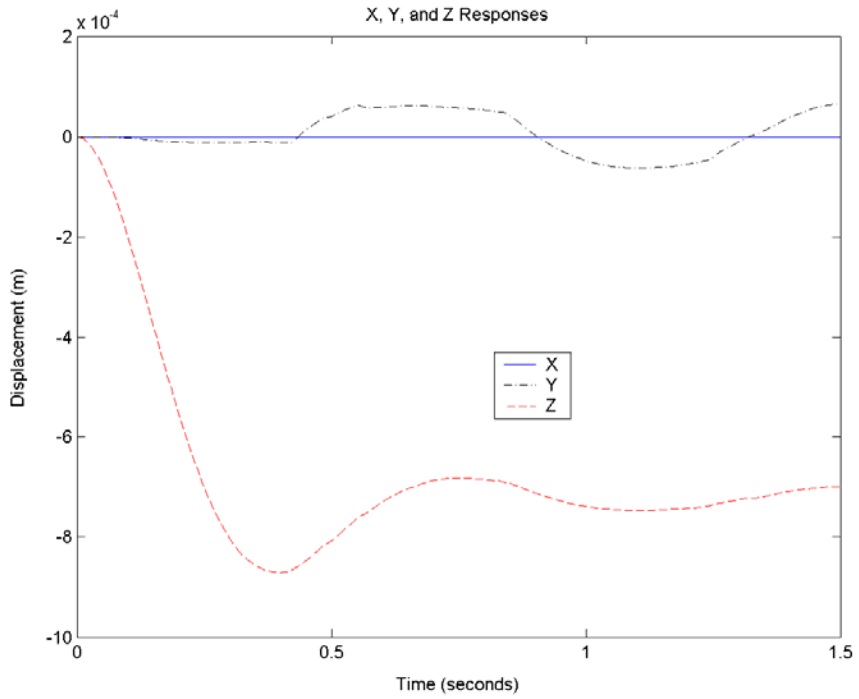


Figure 9-11. Rotation of the mechanism about the X-Axis.

(a)



(b)

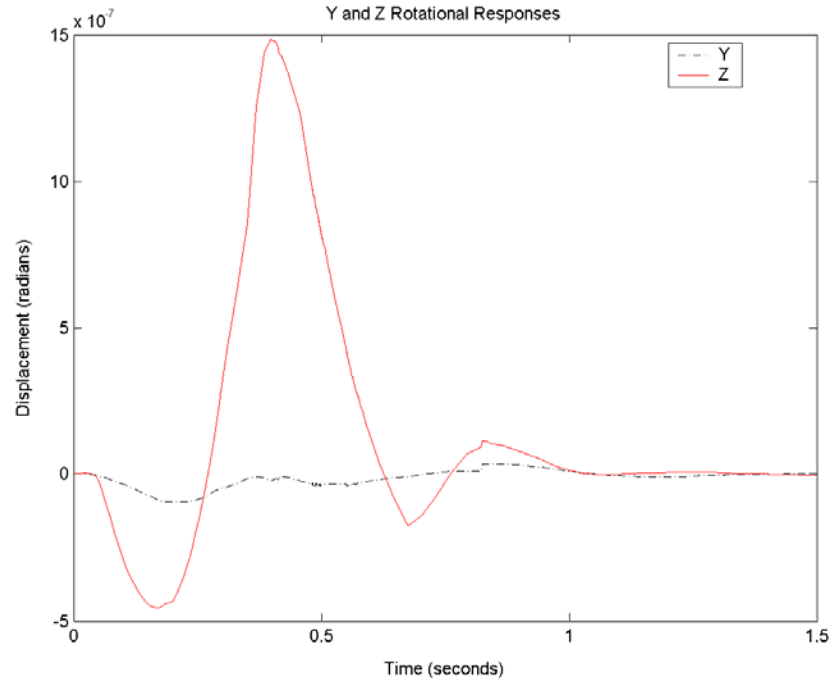


Figure 9-12. Motions in the other DOFs. (a) Z, Y, and X axes. (b) Rotations about the Z and X axes.

Testing of the response to rotations about the X-Axis simply confirmed the previous five tests and verified that the controllers were performing properly. Once again, some slight steady-state error was present in the positioning along the Z-Axis, but as previously stated, this could be attributable to the resolution specified for the forward analysis. Otherwise, the mechanism behaved impeccably.

Motions in Multiple DOF

Once the performance of the mechanism in single DOFs had been verified, the behavior of the system when moved in multiple DOFs needed to be evaluated. Since each separate DOF responded as designed with relatively small effects on the other motions, commanding motion in several DOFs should all show good behavior. The coupling effects will never be completely eliminated, but they have a lessened impact on system performance. The tests that show the most significant effects and results are presented here.

X-Y Displacement

Considering that individual motions in either the global X or Y directions showed greatest impact on the vertical positioning of the system, it was only logical to test the effects when the mechanism was moved in both DOFs. As expected, the largest error was present in the Z-Axis motion, with a maximum disturbance of 0.0046 m. This very rapidly decayed to a final positioning of approximately 0.9992 m, or 0.08% steady state error. This was a clear indication that the coupling effects had been essentially eliminated, and the numerical forward analysis was functioning properly. The small error of 0.0008 m was well within the specified resolution for the forward analysis.

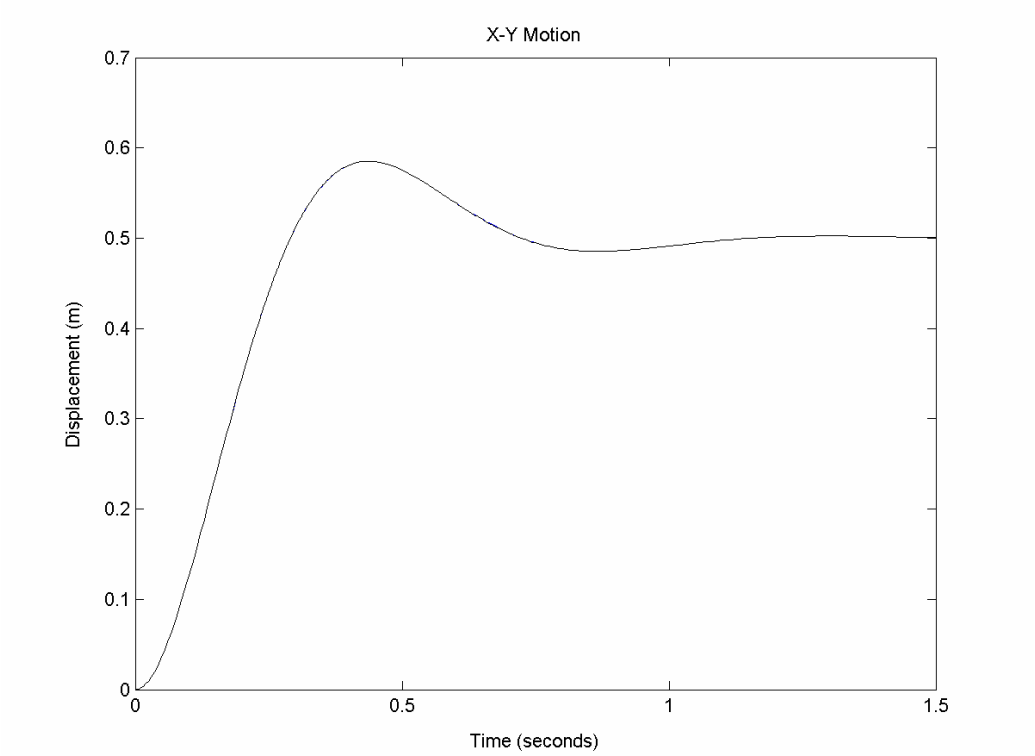
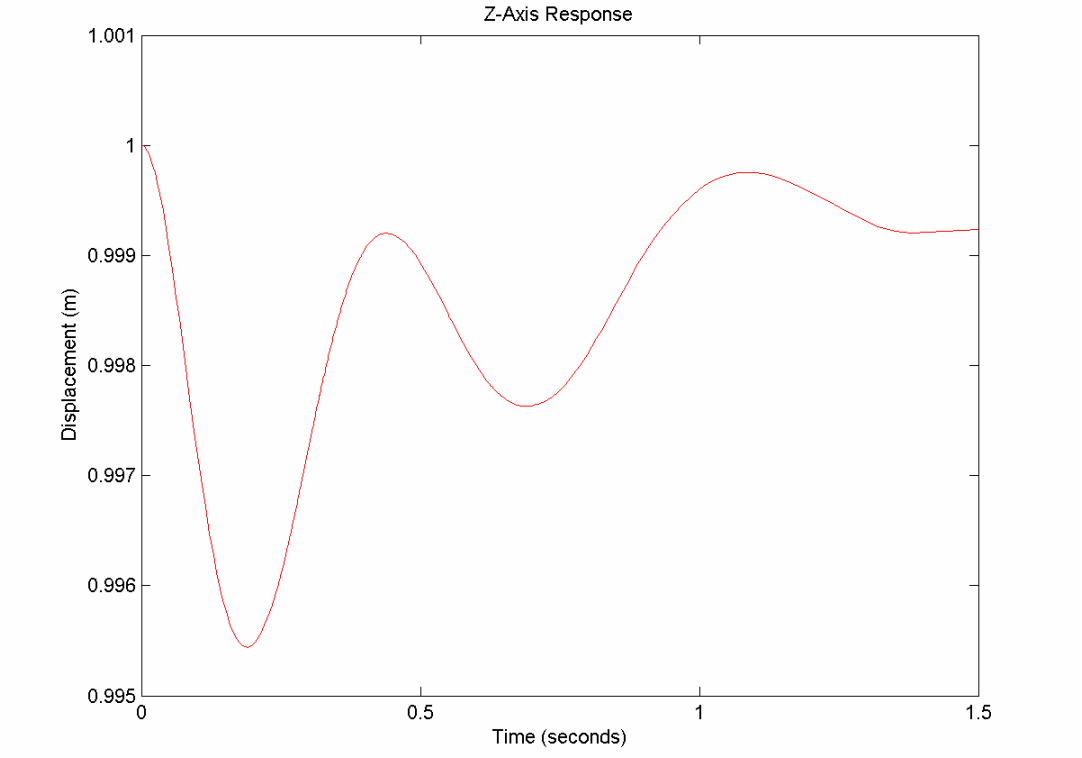


Figure 9-13. Motion of the mechanism along the X and Y axes.

(a)



(b)

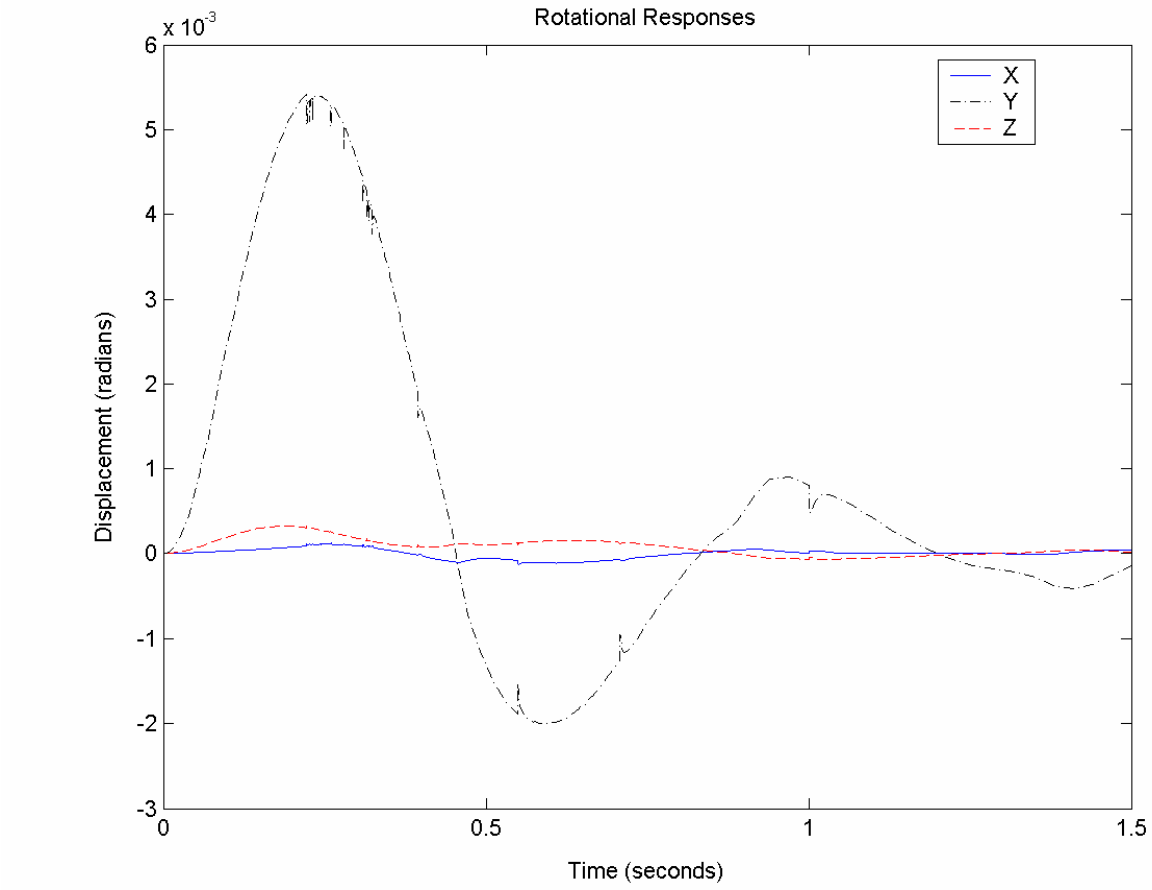


Figure 9-14. Motions in the other DOFs. (a) Z and Y axes. (b) Rotations about the Z, Y, and X axes.

The motions in the X and Y directions followed classical second order response and both exhibited only 17% overshoot, which is less than the design specification. The settling time was approximately 0.98 second, which also matched the performance requirements.

X-Y-Z Displacement

The results for testing motion in three axes should come as little surprise considering the behavior of the system in previous tests. The mechanism was commanded to move 0.5 m along each axis relative to its initial position.

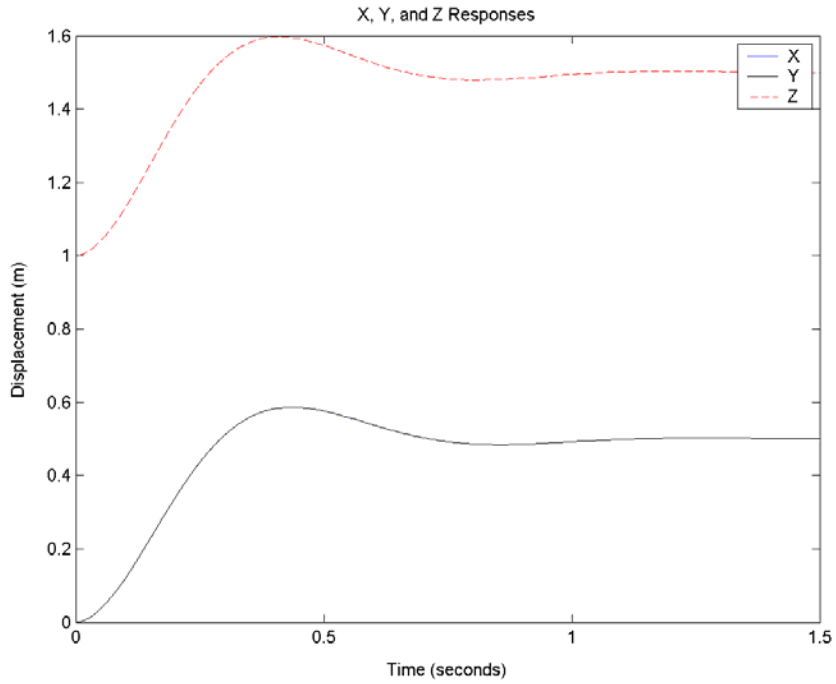


Figure 9-15. Motion of the mechanism along all three axes.

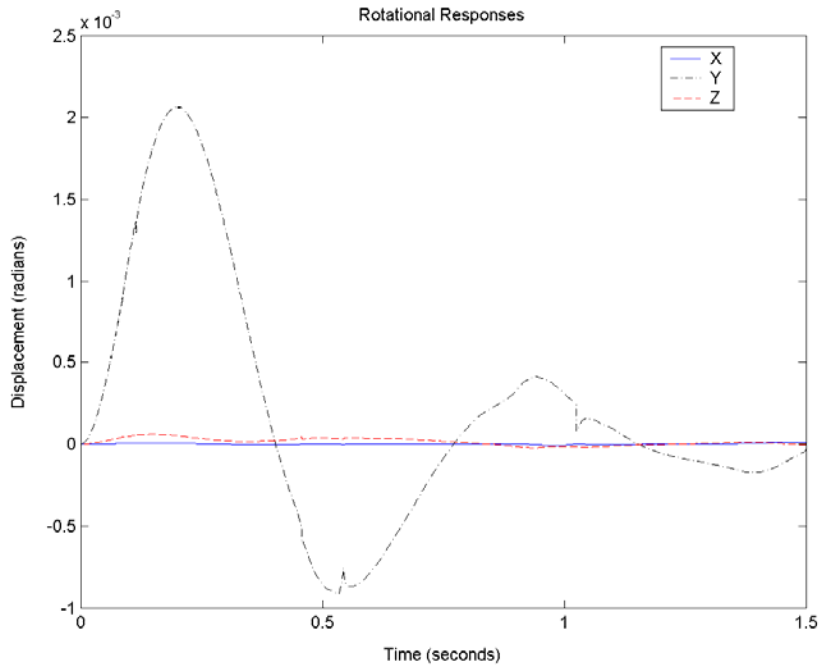


Figure 9-16. Rotations about the Z, Y, and X axes.

The motions in the primary axes followed classical second order responses as predicted. The X and Y motions were virtually identical with 17.3% overshoot and a

settling time of 0.975 second. The Z-Axis motion showed 19.4% overshoot and a settling time of 0.955 second. While the overshoot along the Z-Axis is larger than the X and Y motions, it still meets the design specifications. It is hardly surprising that the Z-Axis motion would have larger overshoot when one remembers that the testing of the X and Y axes tended to generate fluctuations in the vertical positioning of the platform.

X-Y-Z Rotation

Testing of the rotational performance of the system in three axes simultaneously was a critical step in evaluating the performance capabilities of the system. Up to this point, the platform and controllers had responded admirably. While it was expected that the system would respond just as well as in previous tests, the question was over what range of rotational motion the system would remain stable.

The first test commanded rotation of 0.5 radian about each axis since previous testing had used this extent of motion. However, the system exhibited excessive overshoot in all three DOFs being commanded at 29.8%, 23%, and 25.8% overshoot for the X, Y, and Z rotations respectively.

While the excessive overshoot was unsettling at first, the cause very quickly became clear with an animation of the simulation. Apparently the mechanism was becoming ill-conditioned when moving such a great amount, resulting in less than optimal control. Essentially, the plane of the end-effector was nearly coplanar with a pair of the control legs, causing a loss of control. This can be seen below in the simplified model of the system at final positioning and was the inspiration to change the configuration of the system in the modified spatial mechanism. Since the platform was being moved to the edges of its workspace, its performance would naturally degrade. Therefore, the amount of rotation was reduced to 0.1 radian for each DOF.

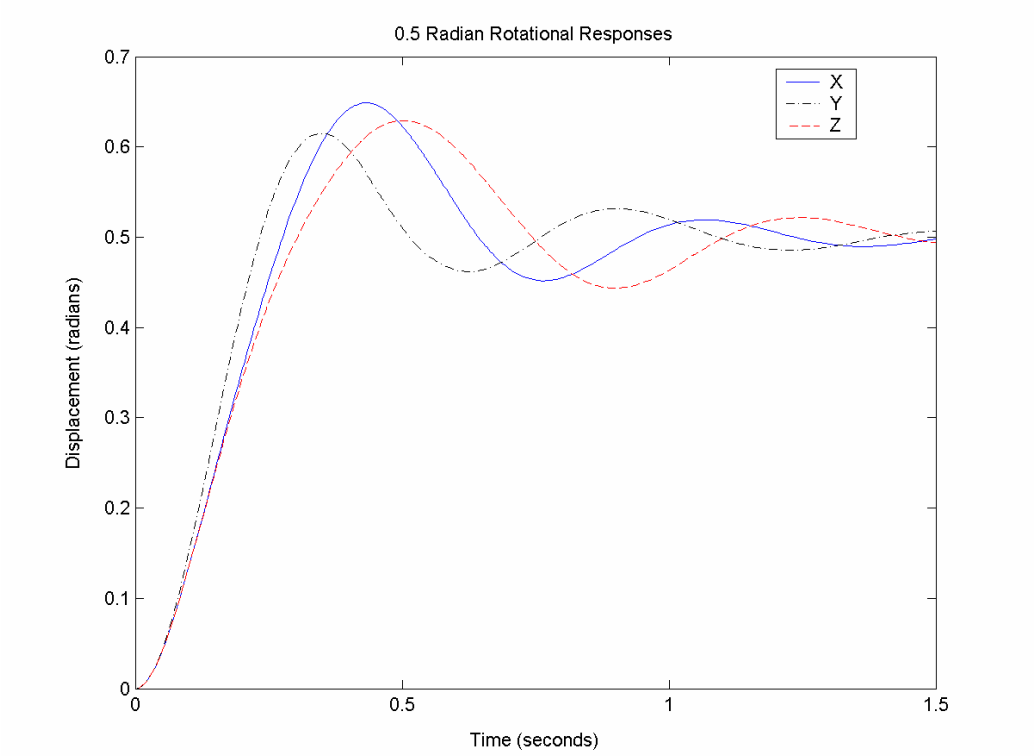


Figure 9-17. Rotation of the mechanism about all axes.

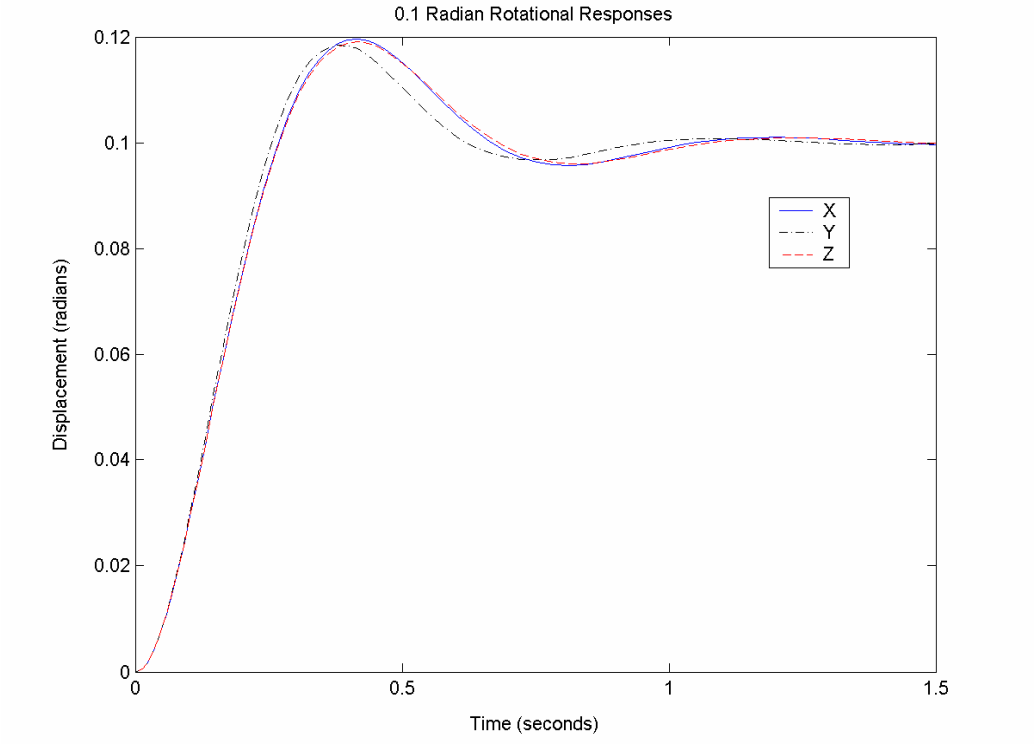


Figure 9-18. Rotation of the mechanism about all axes, albeit with a smaller commanded motion.

The system responded flawlessly in this second test because it was well within its workspace. The overshoots were 19.6%, 18.4%, and 19.1% for the X, Y, and Z rotations respectively while their settling times were 0.945, 0.848, and 0.972 second. Once again, the Z-Axis motion showed a bit of steady-state error, while movement along the X and Y axes only fluctuated by about 10^{-5} meter. Evidently, the controllers worked extremely well within the physical limitations of the mechanism.

Full 6 DOF Motion

The final test for the full spatial mechanism was to command motion in all six DOFs simultaneously. Recalling the lessons learned from the rotational experiments, the rotations were specified to be 0.1 radian, and the motions along the primary axes were commanded to 0.5 m from the starting point. The results can be seen below and clearly show that all DOFs met the 20% overshoot and 1 second settling time requirements.

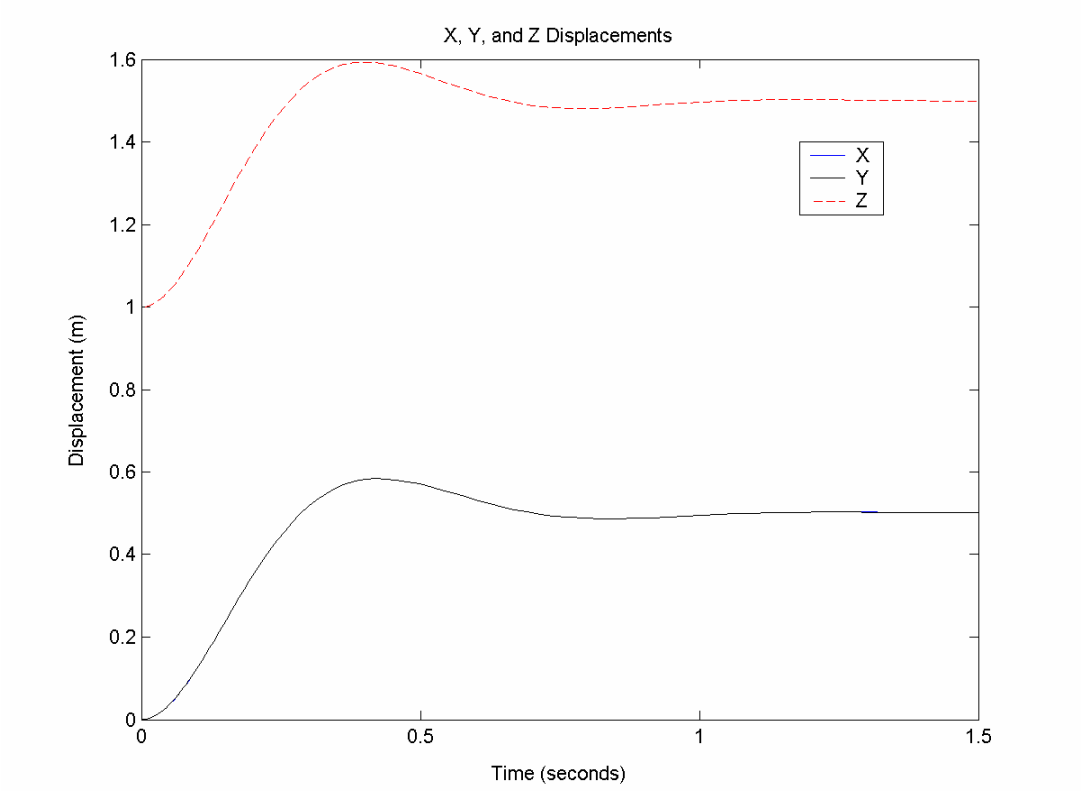


Figure 9-19. Motion of the mechanism along the primary axes.

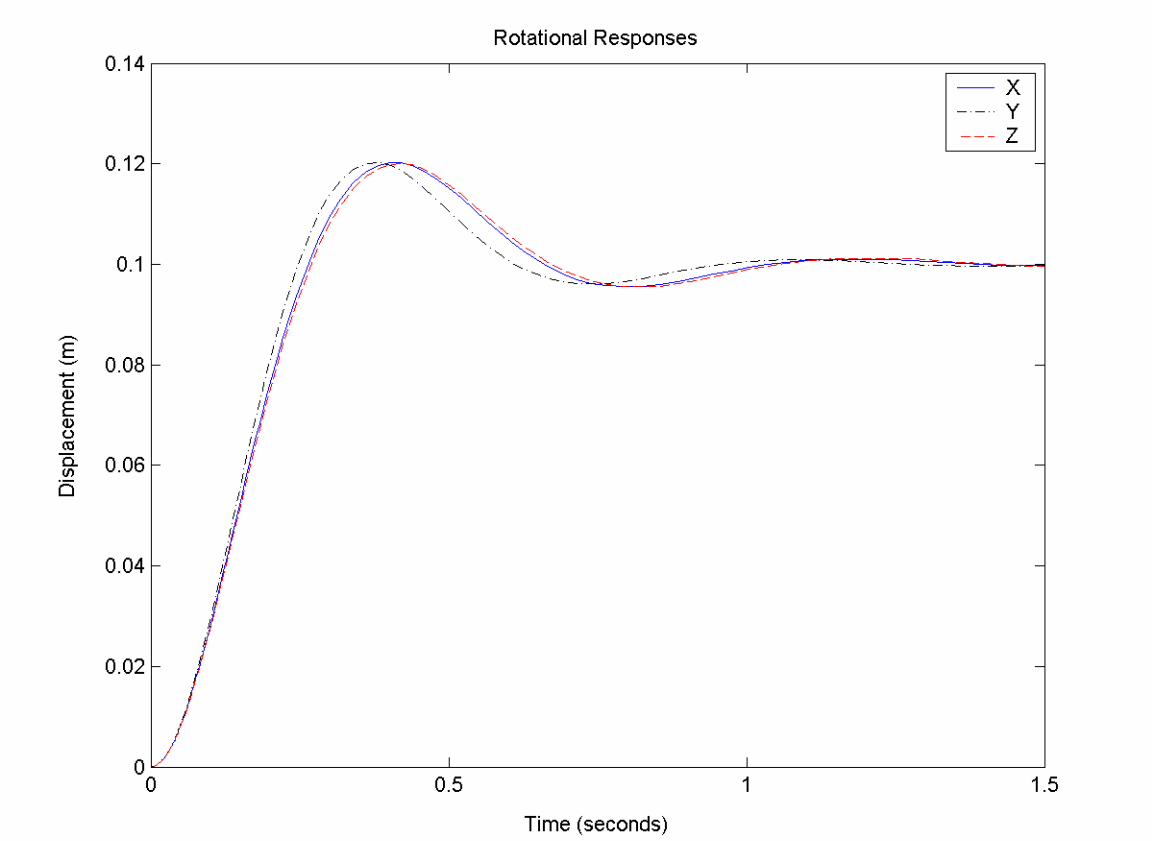


Figure 9-20. Rotations about the primary axes.

While the motions along the axes exceeded the performance requirements, the rotational behavior just barely satisfied the overshoot limitations, coming in at almost exactly 20%. This is hardly surprising when one considers that the coupling effects tend to compound errors in the system and cause fluctuations in the other controllers. However, the fact that the system did meet the performance requirements in all aspects is a testament to the effectiveness of the decoupling, even if it is a simple approach.

Modified Spatial Mechanism

One will recall that testing of the original 3-3 + 1 mechanism revealed that rotations of 0.5 radian about each axis led to ill-conditioning. The question was posed whether it was possible to improve the response in these extreme conditions by changing the geometry of the platform simply by moving the central leg. This test moved the

connection point of the central leg out of the plane of the end effector by 0.2 m along the local Z-Axis to test the effects that it would have on the rotational response of the system in extreme conditions. The figures below illustrate the new configuration of the platform and the results of testing. The platform was commanded to rotate 0.5 radian about each axis. One will notice that the overshoot of the X and Y rotations were reduced to 16 and 17.4% respectively, which is a huge improvement over the 29.8% and 23% of the non-modified mechanism. The Z rotation showed approximately the same overshoot with 26% as compared with the previous 25.8% of earlier testing. However, the settling times were not improved dramatically, clocking in at 1.28, 1.21, and 1.47 seconds for the X, Y, and Z rotations.

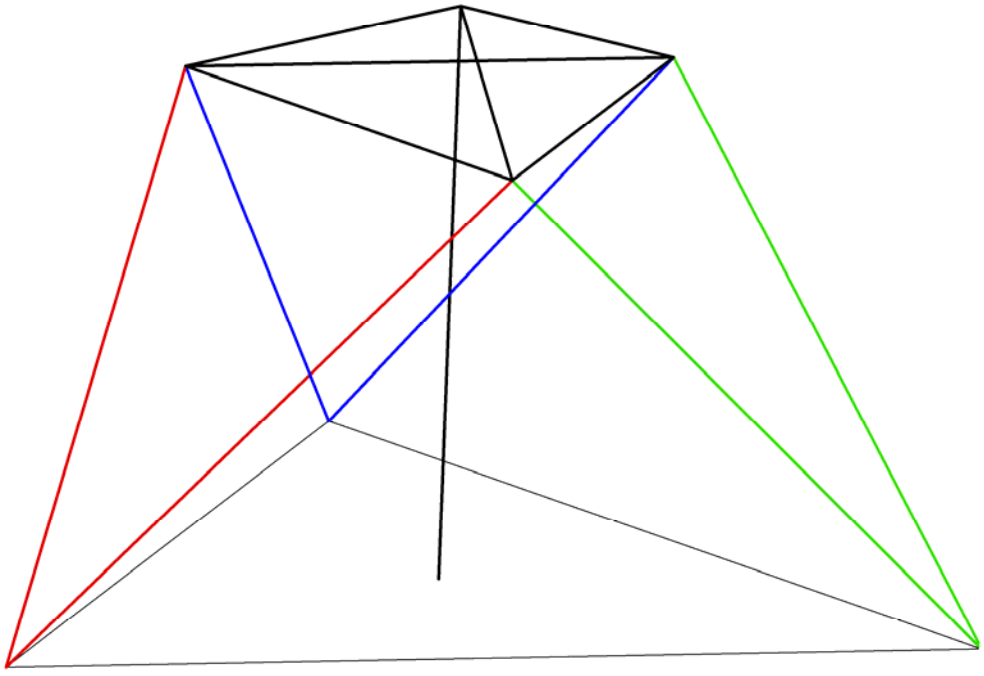


Figure 9-21. New platform configuration with modified connection point.

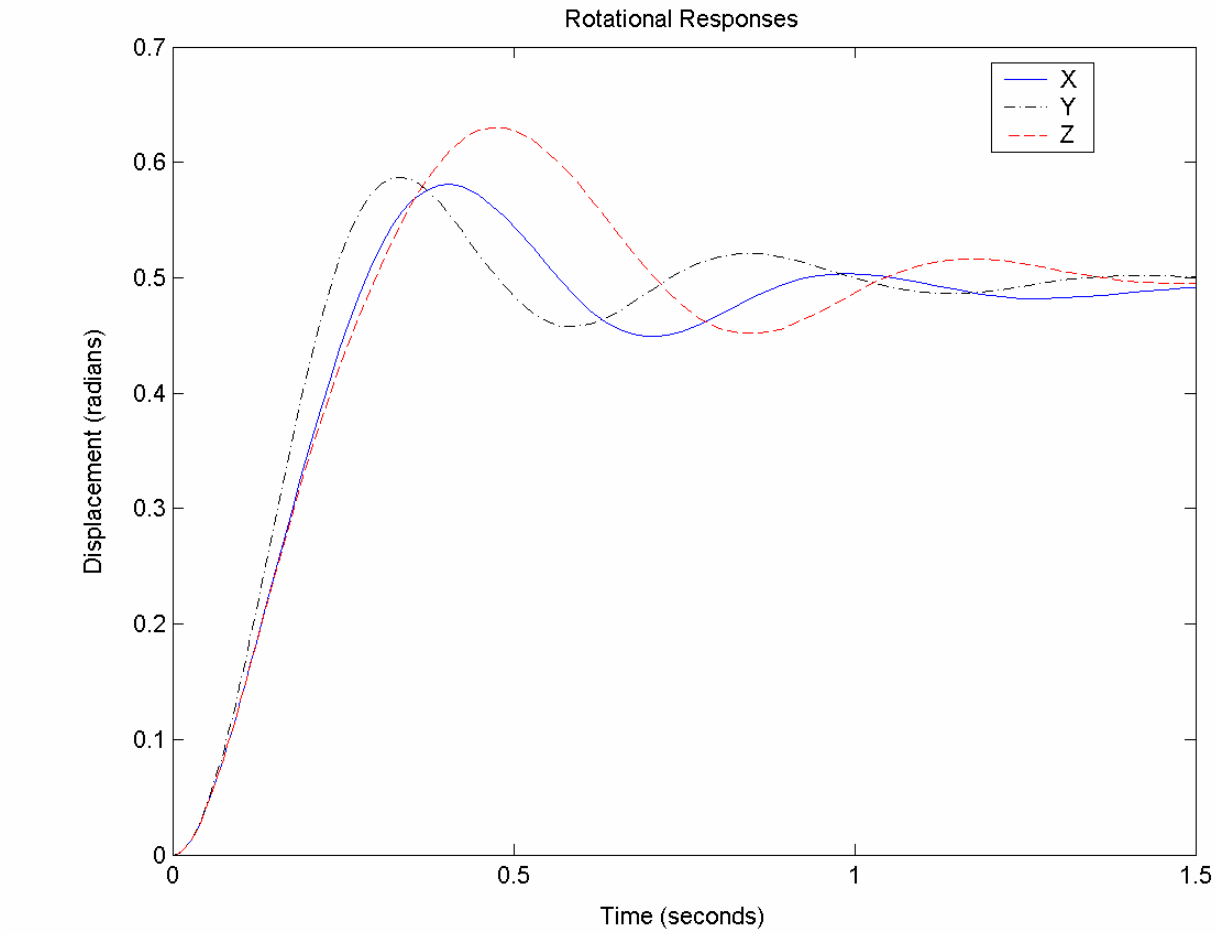


Figure 9-22. Rotations about the primary axes.

While the change in configuration did have a positive effect on the response of the system in extreme circumstances, it did not have enough of an impact due to physical limitations. Enough of the mechanism was still ill conditioned that the central leg could not effectively compensate. This drives home the point that all parallel mechanisms have limited workspaces, especially in the aspect of rotations, so knowledge of these limitations is critical for proper control. Possibly reconfiguring the outer legs or changing relative dimensions in the mechanism could help to extend its workspace, but that topic would be best left for future research.

CHAPTER 10
FINAL COMMENTS AND FUTURE WORK

This project proved to be a challenging and immensely gratifying experience. The problem required some interesting approaches to create a relatively simple, yet functional solution. The generalized control strategy that was developed is flexible and applicable to many other systems. It performed well in decoupling the system despite the difficult nature of the mechanisms. The numerical forward analysis has great potential for future applications as it is a highly flexible, robust, and efficient approach to a potentially very complicated issue. The decoupling controller, while it could not be fully tested due to simulation and software limitations, still looks to be an effective method of controlling compliant elements, even in unfavorable conditions.

While this project accomplished all that was intended, plenty of room remains for future research into related areas. For example, the numerical forward analysis still has great potential to become even more efficient and powerful if it can be analyzed as a controls system, rather than just an iterative approach. In fact, it may even be possible to derive analytical solutions to forward analyses using mathematical derivations based on the methodology. The effective decoupling of compliant elements in a parallel mechanism has great potential to allow the design of responsive, adaptive systems for a wide range of applications. Current research is investigating the responses of compliant, static platforms, which suggests that it may be possible to design the static and active behaviors separately and then easily connect them through some decoupling interface. The possibilities are endless and lend themselves to imagination.

APPENDIX A
SYSTEM DIAGRAMS

This section presents the systems that were built in SimuLink to simulate the mechanisms. They often contain several subsystems due to the complexity of the models. Where necessary, descriptions of the components are supplied.

Planar Mechanism

This section shows the systems that were constructed to analyze the planar mechanism. This was the testing model for both the control methodology and the proper way to construct models in MATLAB.

System Overview

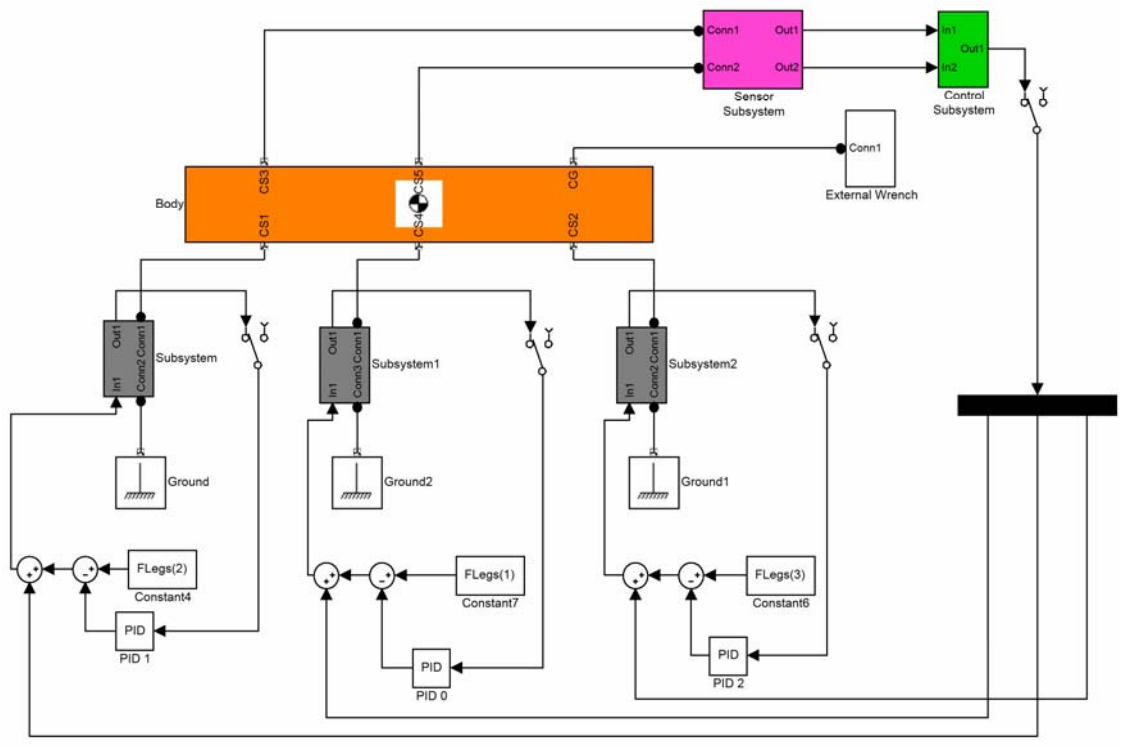


Figure A-1. Top level of planar mechanism. The orange block is the end effector, the pink block is the position sensor, the green block is the main controller, and the gray blocks are the legs.

The additional controllers were incorporated for static stability in the planar mechanism. One will notice the constants that are being fed to the leg actuators. These are the forces required to hold the system in static equilibrium at its final positioning, meaning that the controller subsystem simply makes corrections to the motion once the simulation starts.

Leg Subsystem

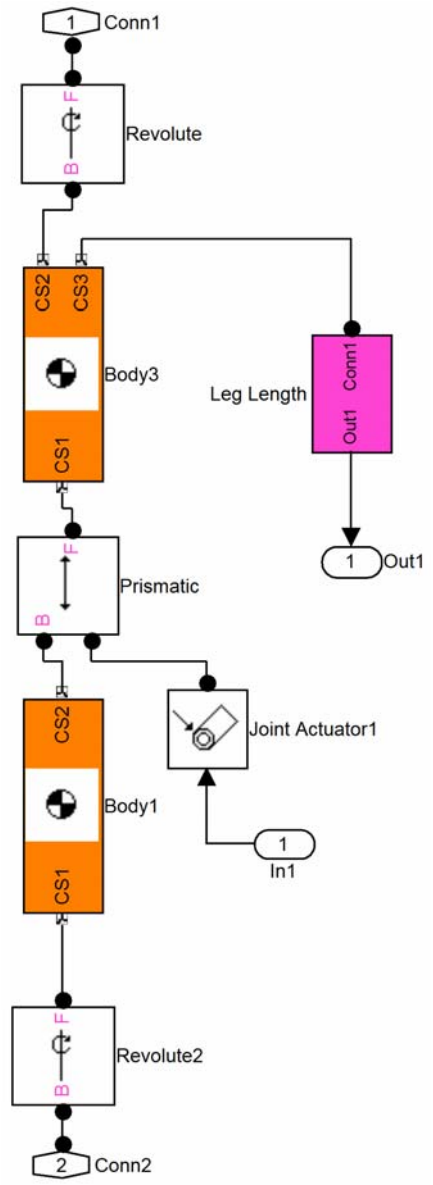


Figure A-2. Inside the leg subsystem for the planar mechanism. Orange blocks are leg pieces. This is the simplest possible leg model for accurate simulations.

Controller Subsystem

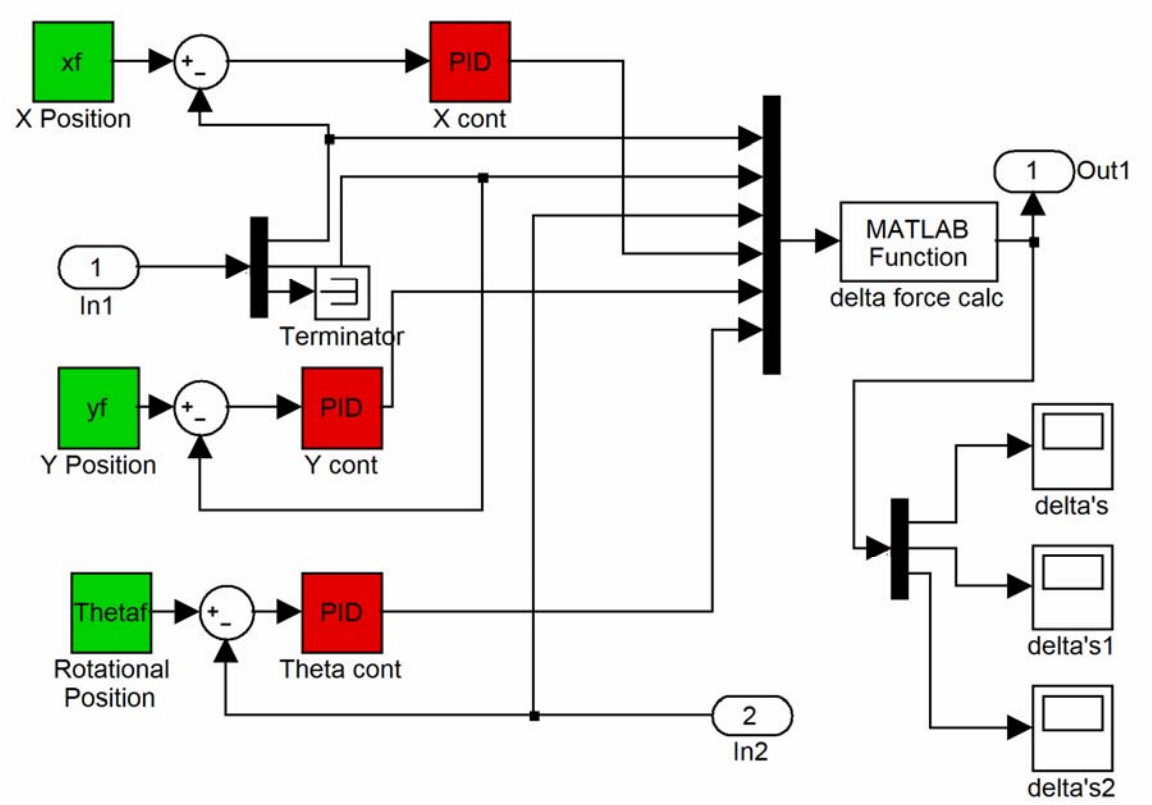


Figure A-3. Inside the control block for the planar mechanism. The three PID controllers are clearly visible and act independently of one another.

Spatial Mechanism

This section presents the models built to analyze the spatial mechanism. The same model was used for both spatial systems. Two versions of the leg subsystems are presented. The non-compliant variant was used in simulations and in the main model. The compliant version was intended for insertion into the main system, but proved to be unusable due to software limitations, as discussed in Chapter 7.

Main System Overview

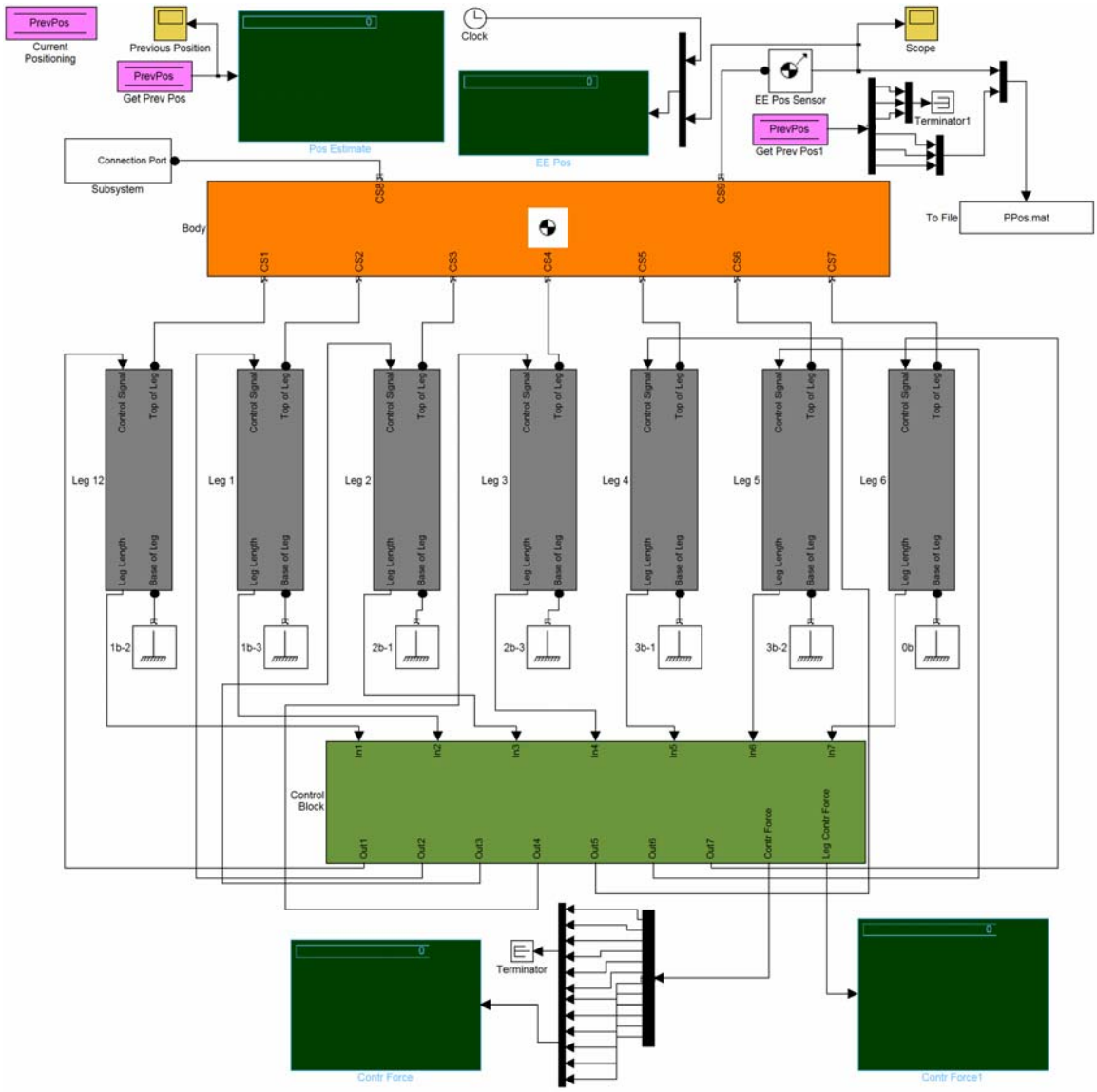


Figure A-4. Top level of the spatial mechanism model.

As with the planar mechanism, the end effector is connected to the leg subsystems and the control block. A slight change is the apparent lack of a sensor block. This is incorporated into the control block as the numerical forward analysis.

Non-Compliant Leg Model

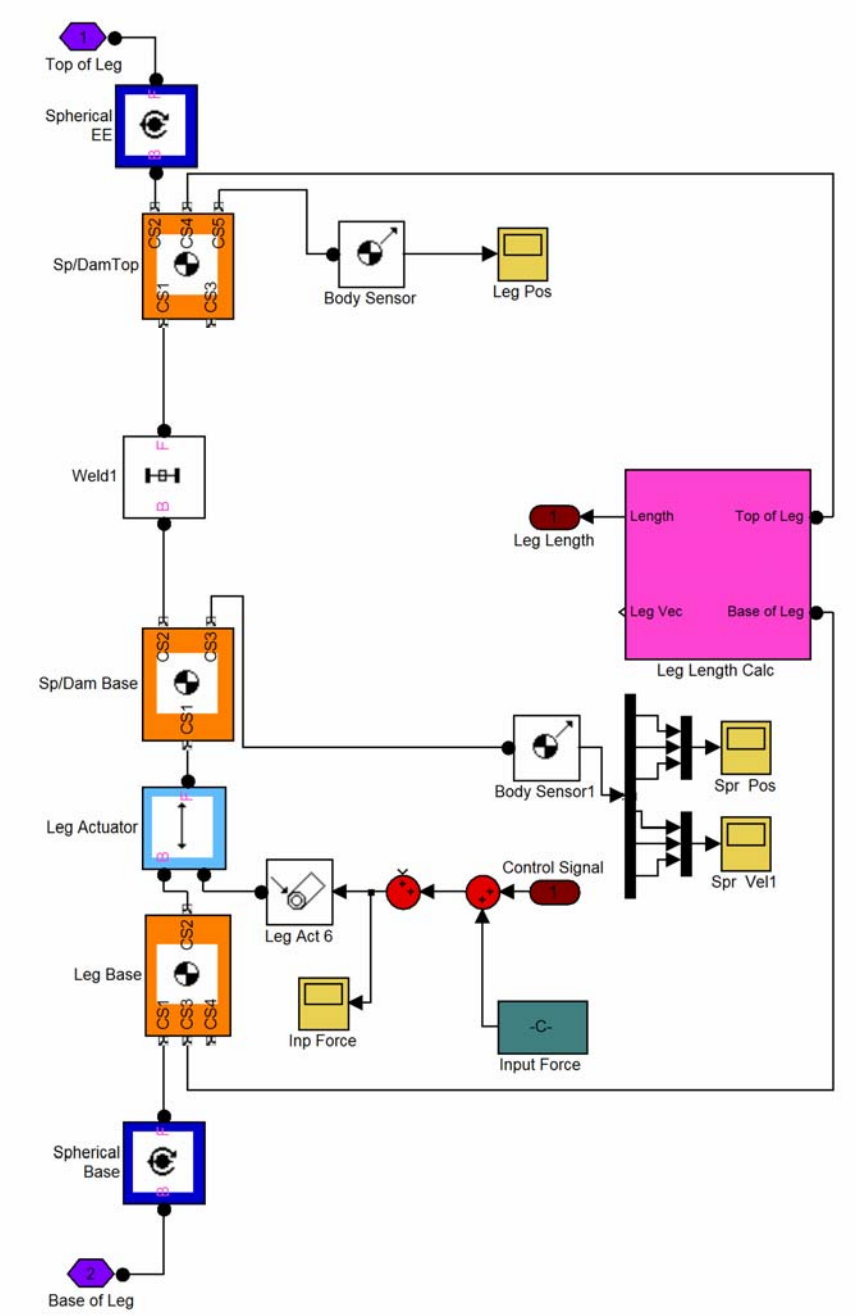


Figure A-5. Inside the leg subsystem of the spatial mechanism. This is virtually identical to the planar leg model.

Control Block

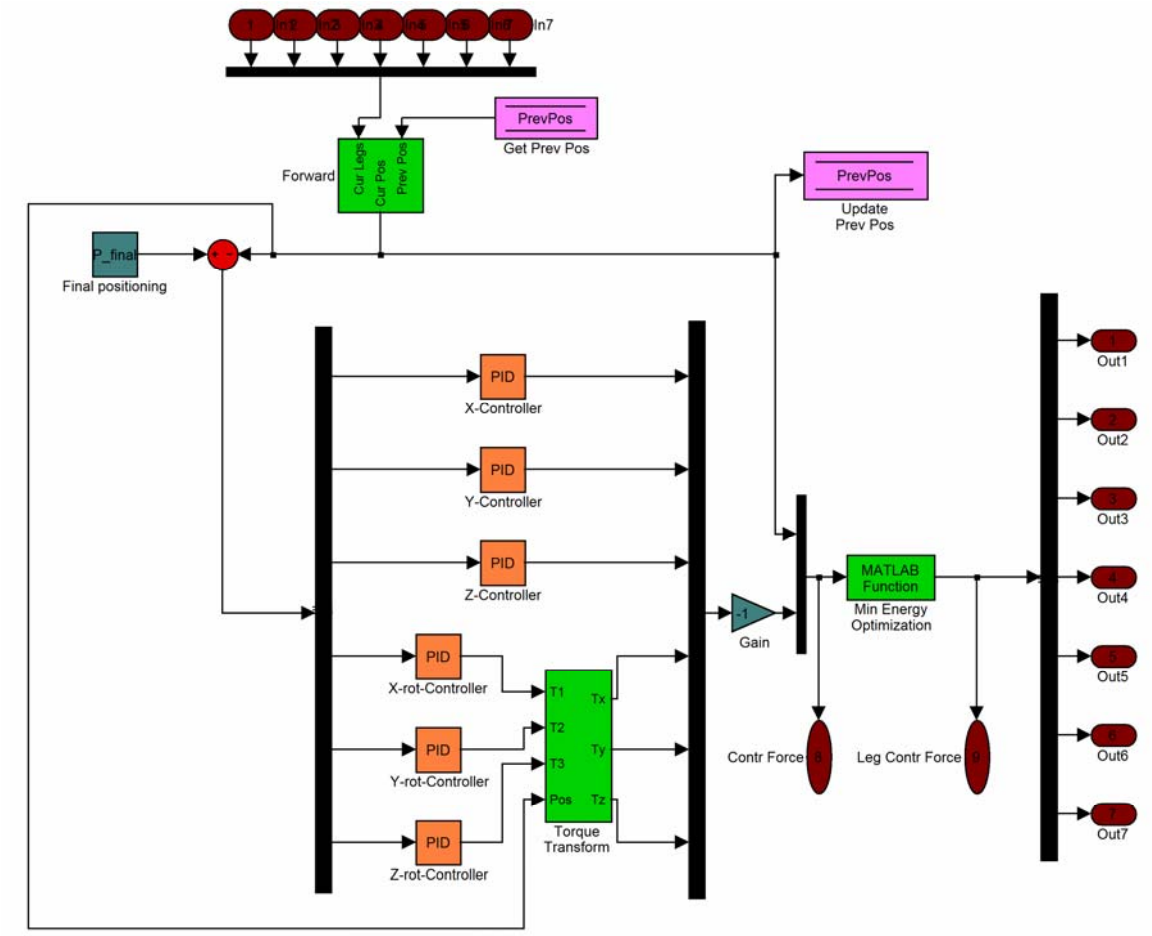


Figure A-6. The control block for the spatial mechanism.

In the control block, the forward analysis receives leg length and position estimate information, which is then fed to the 6 separate controllers. The correctional forces are converted in the minimum energy optimization block, which then sends the control signals to the appropriate legs.

Compliant Leg Model Top Level

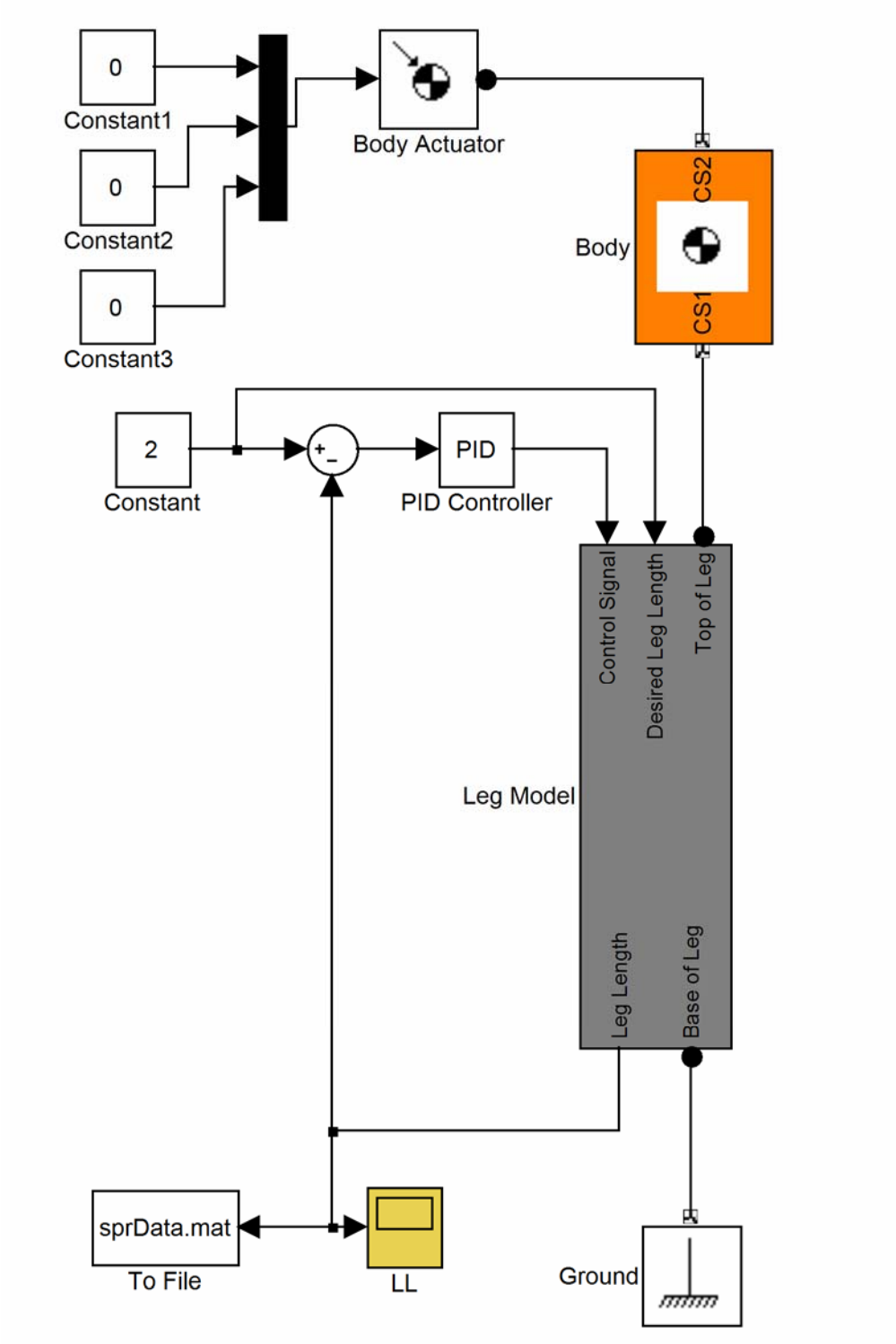


Figure A-7. Top level of compliant leg model. The three constants feeding into the body actuator are meant to represent the components of any external wrench that might be applied to the body.

Compliant Leg Model

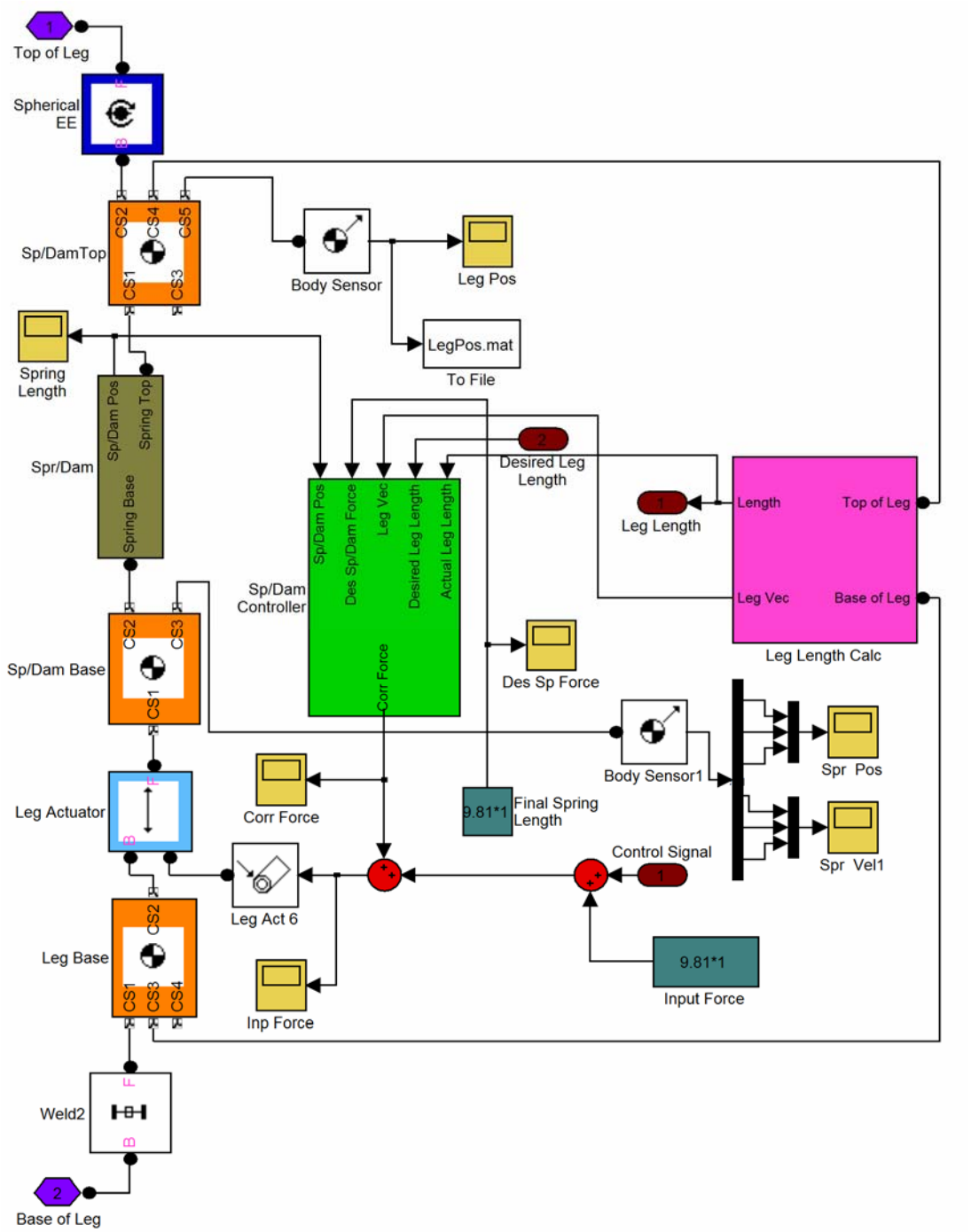


Figure A-8. Inside the compliant leg model. The brown block simulates the spring and damper, while the green block acts as the decoupling controller.

Decoupling Controller

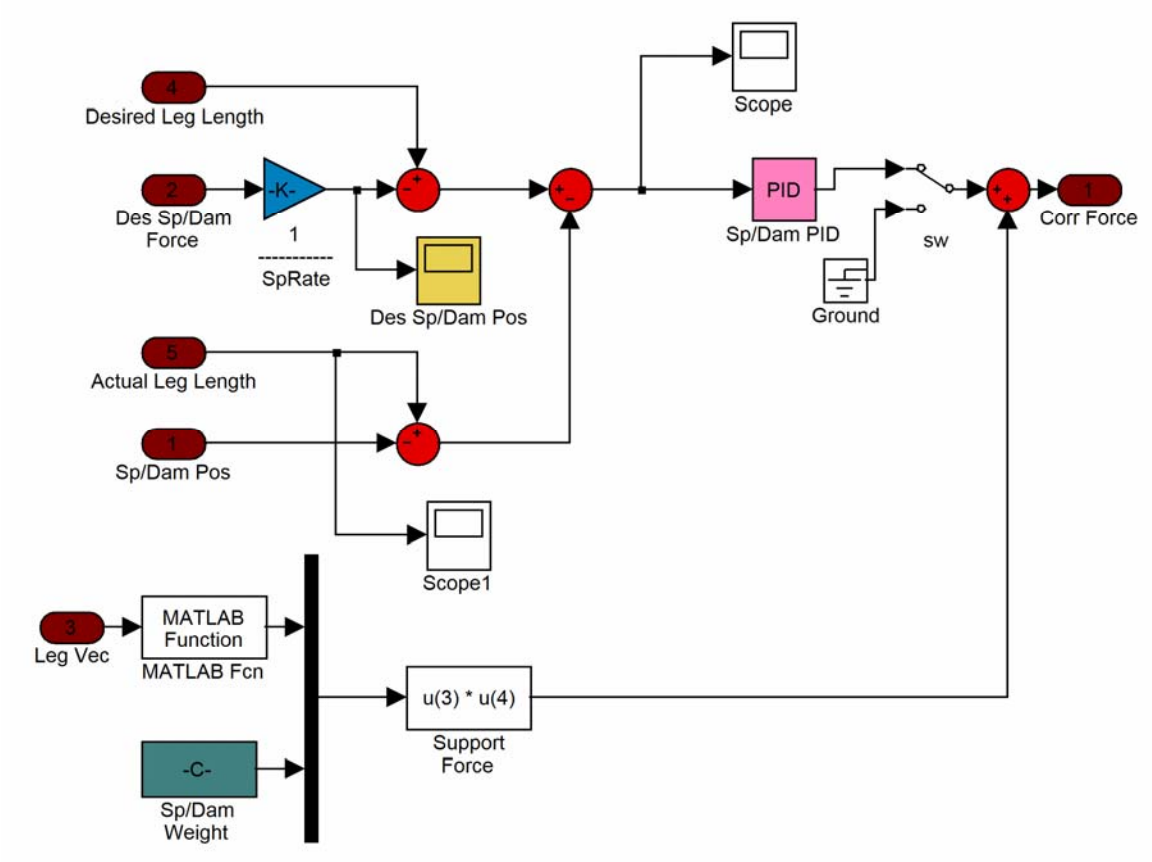


Figure A-9. The decoupling controller for the compliant leg model.

APPENDIX B
MATLAB CODE NEEDED FOR SIMULATIONS

The following sections contain the MATLAB code that enabled the simulations to run. This includes initialization files as well as functions used in the controllers and sensors. Comments can be found within the code itself. Note that initialization files have to be run before the simulation is started, otherwise the model will not assemble itself properly.

Planar Mechanism

Initialization File

```
% This file initializes the system conditions.
% all distances are in cm
% all angles are in deg

% platform dimensions
d = 25;
L = 50;

% Initial position
x = 0;
y = 25;
[x; y]
Theta = 0 * pi/180;

% Calculating initial values
p0 = [(-cos(Theta)*d + x - L) (-sin(Theta)*d + y)];
L0 = sqrt(p0 * transpose(p0));
Theta0 = atan2(p0(2), p0(1)) * 180/pi;

p1 = [(-cos(Theta)*d + x + L) (-sin(Theta)*d + y)];
L1 = sqrt(p1 * transpose(p1));
Theta1 = atan2(p1(2), p1(1)) * 180/pi;

p2 = [(cos(Theta)*d + x - L) (sin(Theta)*d + y)];
L2 = sqrt(p2 * transpose(p2));
```

```
Theta2 = atan2(p2(2), p2(1)) * 180/pi;

%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Final Position
xf = 20;
yf = 50;
[xf; yf]
Thetafd = -14;
Thetaf = Thetafd * pi/180;

% External Wrench
w = [0; 0; 0];

% part masses (kg)
mEE = 1;
IEE = [0 0 0; 0 1/12*mEE*(2*d/100)^2 0; 0 0 1/12*mEE*(2*d/100)^2]
% IEE = [0 0 0; 0 1/12*mEE*(2*d)^2 0; 0 0 1/12*mEE*(2*d)^2];
mAct = 0.01;
mBase = mAct;
IA = [0 0 0; 0 1/12*mAct*(L0/100/2)^2 0; 0 0 1/12*mAct*(L0/100/2)^2];

g = 9.81;
weight = g * [0; (mEE + 3*mAct); 0];

%%% Force Calculations
p0 = [(-cos(Thetaf)*d + xf - L) (-sin(Thetaf)*d + yf)];
L0f = sqrt(p0 * transpose(p0));

p1 = [(-cos(Thetaf)*d + xf + L) (-sin(Thetaf)*d + yf)];
L1f = sqrt(p1 * transpose(p1));

p2 = [(cos(Thetaf)*d + xf - L) (sin(Thetaf)*d + yf)];
L2f = sqrt(p2 * transpose(p2));

% Force Matrix
K11 = (-cos(Thetaf)*d + xf - L) / L0f;
K21 = (-sin(Thetaf)*d + yf) / L0f;
K31 = (-cos(Thetaf)*d*yf + sin(Thetaf)*d*xf - sin(Thetaf)*d*L) / L0f;

K12 = p1(1) / L1f;
K22 = p1(2) / L1f;
K32 = (-cos(Thetaf)*d*yf + sin(Thetaf)*d*(xf + L)) / L1f;

K13 = p2(1) / L2f;
K23 = p2(2) / L2f;
K33 = (cos(Thetaf)*d*yf + sin(Thetaf)*d*(-xf + L)) / L2f;
```

```
K = [K11 K12 K13; K21 K22 K23; K31 K32 K33];

Kdet = det(K);

if Kdet == 0
   'Singular K Matrix: shifting xf by 10^-7'
   xf = xf + 10^-10;
   p0 = [(-cos(Thetaf)*d + xf - L) (-sin(Theta)*d + yf)];
   L0f = sqrt(p0 * transpose(p0));

   p1 = [(-cos(Thetaf)*d + xf + L) (-sin(Thetaf)*d + yf)];
   L1f = sqrt(p1 * transpose(p1));

   p2 = [(cos(Thetaf)*d + xf - L) (sin(Thetaf)*d + yf)];
   L2f = sqrt(p2 * transpose(p2));

   % Force Matrix
   K11 = xf / L0f;
   K21 = yf / L0f;
   K31 = (-cos(Thetaf)*d*yf + sin(Thetaf)*d*xf - sin(Thetaf)*d*L) / L0f;

   K12 = p1(1) / L1f;
   K22 = p1(2) / L1f;
   K32 = 0.5 * ((-cos(Thetaf)*d + xf)*p1(2) - (-sin(Thetaf)*d + yf)*p1(1)) / L1f^2;

   K13 = p2(1) / L2f;
   K23 = p2(2) / L2f;
   K33 = 0.5 * ((cos(Thetaf)*d + xf)*p2(2) - (sin(Thetaf)*d + yf)*p2(1)) / L2f^2;

   K = [K11 K12 K13; K21 K22 K23; K31 K32 K33];
end

K
FLegs = -inv(K) * (w + weight)
```

Force Conversion Function

```
% This function determines the change in leg forces based on control
% forces.
function [delta_force] = delta_force_calc(pc)
d = 25;
L = 50;

xc = pc(1);
yc = pc(2);
Thetac = pc(3);
```

```
% Length
p0 = [(-cos(Thetac)*d + xc - L) (-sin(Thetac)*d + yc)];
L0c = sqrt(p0 * transpose(p0));

p1 = [(-cos(Thetac)*d + xc + L) (-sin(Thetac)*d + yc)];
L1c = sqrt(p1 * transpose(p1));

p2 = [(cos(Thetac)*d + xc - L) (sin(Thetac)*d + yc)];
L2c = sqrt(p2 * transpose(p2));

delta_contr = [pc(4); pc(5); pc(6)];

K11 = (-cos(Thetac)*d + xc - L) / L0c;
K21 = (-sin(Thetac)*d + yc) / L0c;
K31 = (-cos(Thetac)*d*yc + sin(Thetac)*d*xc - sin(Thetac)*d*L) / L0c;

K12 = p1(1) / L1c;
K22 = p1(2) / L1c;
K32 = (-cos(Thetac)*d*yc + sin(Thetac)*d*(xc + L)) / L1c;

K13 = p2(1) / L2c;
K23 = p2(2) / L2c;
K33 = (cos(Thetac)*d*yc + sin(Thetac)*d*(-xc + L)) / L2c;

K = [K11 K12 K13; K21 K22 K23; K31 K32 K33];

delta_force = inv(K) * delta_contr;
```

Spatial Mechanism and Compliant Leg Model

The spatial and modified spatial mechanisms shared many of the same functions. The main difference was that the modified mechanism used a slightly different initialization file that allowed repositioning of the central leg. The initialization file for the spatial mechanism also initialized the compliant leg model.

Initialization File: Main Mechanism

```
% This is the initialization file that will set the parameters used in the
% simulink model.  This MUST be run before the simulation will work.  Also
% note that all variables created in this file are located in the Workspace
% and should not be modified directly.

% Several steps need to be done in the initialization:
```

```
%-----------------------------------------------------%
% 0) Global variable declarations.  Workspace cleanup, etc.
% 1) Set the locations of the connection points.
% 2) Set leg compliances and System mass/inertia parameters.
% 3) Specify final positioning and force-state.
% 4) Set lengths of leg sections.  Have system in equilibrium at beginning.
%    Also determine the orientations of the legs for proper closed system.
% 5) Determine the leg forces needed to reach equilibrium at the final
% state.  Note: You must mux the positioning and force state information.
% 6) Calculate the values for the PID controllers.
% 7) Additional initializations.
%-----------------------------------------------------%

%-----------------------------------------------------------%
% 0) Global variable declarations.  Workspace cleanup, etc.
%-----------------------------------------------------------%
clc;
clear;
close all;

global J;
global R;
global MaxErr;
global MaxIt;
global k_val;
global c_val;
global p1ee;
global p2ee;
global p3ee;

%-----------------------------------------------%
% 1) Set the locations of the connection points.
%-----------------------------------------------%
p1b = [0, -1, 0];
p2b = [sqrt(3)/2, 1/2, 0];
p3b = [-sqrt(3)/2; 1/2; 0];

% p1ee = [0; 1; 0];
% p2ee = [-sqrt(3)/2; -1/2; 0];
% p3ee = [sqrt(3)/2; -1/2; 0];
p1ee = [0; 1; 0] / 2;
p2ee = [-sqrt(3)/2; -1/2; 0] / 2;
p3ee = [sqrt(3)/2; -1/2; 0] / 2;

%-----------------------------------------------------------%
% 2) Set leg compliances and System mass/inertia parameters.
```

```
%-------------------------------------------------------------%
k_val = [100; 100; 100; 100; 100; 100; 100];
c_val = 100*[100; 100; 100; 100; 100; 100; 100];

% Specify mass and inertia properties of the system.
% Mass -> kg
% Inertia -> kg-m^2
Mass = 1;
Inertia = [1 0 0; 0 1 0; 0 0 1];
% Leg masses are presented in 3 columns: leg base, actuator base, and
% actuator end.  All of these will be essentially zero for primary testing.
LegMasses = [1e-7 1e-4 1e-7;
          1e-7 1e-4 1e-7;
          1e-7 1e-4 1e-7;
          1e-7 1e-4 1e-7;
          1e-7 1e-4 1e-7;
          1e-7 1e-4 1e-7;
          1e-7 1e-4 1e-7];

GenInertia = [1e-7 0 0;
           0 1e-7 0;
           0 0 1e-7];

%-----------------------------------------------%
% 3) Specify final positioning and force-state.
%-----------------------------------------------%
P_final = [1; 0; 1; 0; 0; 0];
w_app = [0; 0; 0; 0; 0; 0]
w_ext = Mass*[0; 0; -9.81; 0; 0; 0] + w_app; % Combination of weight and external
forces.

%-------------------------------------------------------------------------%
% 4) Set lengths of leg sections.  Have system in equilibrium at beginning.
%    Also determine the orientations of the legs for proper closed system.
%-------------------------------------------------------------------------%
% Set the lengths of the actuator bases to be 0.
% Initial conditions before motion.  Purely to determine initial spring
% lengths/loads.
P_init = [0; 0; 1; 0; 0; 0];
w_init = w_ext;
Info_init = [P_init; w_init];
F_init = MinEnergy(Info_init);

% Correction of the intial rotation matrix.
RxI = [1 0 0; 0 cos(P_init(4)) -sin(P_init(4)); 0 sin(P_init(4)) cos(P_init(4))];
RyI = [cos(P_init(5)) 0 sin(P_init(5)); 0 1 0; -sin(P_init(5)) 0 cos(P_init(5))];
```

```
RzI = [cos(P_init(6)) -sin(P_init(6)) 0; sin(P_init(6)) cos(P_init(6)) 0; 0 0 1];
RI = RxI * RyI * RzI;

% Calculate the leg lengths and extract orientation vectors.
Legs = Sp_LL_J(P_init, true, false) / 3;
Orient = J(1:3, :);

% Initial spring lengths.  This is the amount the springs are COMPRESSED.
L_init = zeros(7,1);
for i = 1:7
   L_init(i) = F_init(i) / k_val(i);
end

%-------------------------------------------------------------------------%
% 5) Determine the leg forces needed to reach equilibrium at the final
% state.  Note: You must mux the positioning and force-state information.
%-------------------------------------------------------------------------%
P_input = [P_final; w_ext];
F_legs = MinEnergy(P_input);

%-------------------------------------------------%
% 6) Calculate the values for the PID controllers.
%-------------------------------------------------%
% Specify the system performance.
MaxOvershoot = 0.20; % This is the fraction of distance moved.
SettlingTime = 1;   % Time in seconds.

% Determine the controllers based on mass and inertia in the system.  The
% three controllers for the x, y, z directions will be identical.  However,
% the rotational controllers will be different based on the inertia of the
% system.  Therefore, 4 sets of calculations are needed in addition to the
% basic performance criteria.

% Determine poles.
zeta = sqrt((log(MaxOvershoot)/pi)^2 / (1 + (log(MaxOvershoot)/pi)^2));
omega_n = 4 / (zeta * SettlingTime);

ClosedLoopPoles = [-zeta * omega_n; omega_n * sqrt(1 - zeta^2)];
RealPole = 0 * ClosedLoopPoles(1);
CharacteristicEq = [1; (-RealPole - 2*ClosedLoopPoles(1));
                (2*RealPole*ClosedLoopPoles(1) + ClosedLoopPoles(1)^2 +
ClosedLoopPoles(2)^2);
                -RealPole*(ClosedLoopPoles(1)^2 + ClosedLoopPoles(2)^2)];

% Design x, y, z controllers.
Kp = CharacteristicEq(3) * Mass;
```

```
Ki = CharacteristicEq(4) * Mass;
Kd = CharacteristicEq(2) * Mass;

% Design x-rotational controller.
Kpx = CharacteristicEq(3) * Inertia(1,1);
Kix = CharacteristicEq(4) * Inertia(1,1);
Kdx = CharacteristicEq(2) * Inertia(1,1);

% Design x-rotational controller.
Kpy = CharacteristicEq(3) * Inertia(2,2);
Kiy = CharacteristicEq(4) * Inertia(2,2);
Kdy = CharacteristicEq(2) * Inertia(2,2);

% Design x-rotational controller.
Kpz = CharacteristicEq(3) * Inertia(3,3);
Kiz = CharacteristicEq(4) * Inertia(3,3);
Kdz = CharacteristicEq(2) * Inertia(3,3);

% Design Controllers for the leg springs/dampers.
MaxOvershoot = 0.2; % This is the fraction of distance moved.
SettlingTime = 1;   % Time in seconds.
zeta = sqrt((log(MaxOvershoot)/pi)^2 / (1 + (log(MaxOvershoot)/pi)^2));
omega_n = 4 / (zeta * SettlingTime);

ClosedLoopPoles = [-zeta * omega_n; omega_n * sqrt(1 - zeta^2)];
RealPole = 0 * ClosedLoopPoles(1);
CEq = [1; (-RealPole - 2*ClosedLoopPoles(1));
            (2*RealPole*ClosedLoopPoles(1) + ClosedLoopPoles(1)^2 +
ClosedLoopPoles(2)^2);
            -RealPole*(ClosedLoopPoles(1)^2 + ClosedLoopPoles(2)^2)];
LegMasses = [1e-7 1e-4 1e-7;
          1e-7 1e-4 1e-7;
          1e-7 1e-4 1e-7;
          1e-7 1e-4 1e-7;
          1e-7 1e-4 1e-7;
          1e-7 1e-4 1e-7;
          1e-7 1e-0 1e-7];

KpS = zeros(7,1);
KiS = zeros(7,1);
KdS = zeros(7,1);
for LegNum = 1:7
   KpS(LegNum) = CEq(3) * LegMasses(LegNum,2);
   KiS(LegNum) = CEq(4) * LegMasses(LegNum,2);
   KdS(LegNum) = CEq(2) * LegMasses(LegNum,2);
end
```

```
%-----------------------------%
% 7) Additional initializations.
%-----------------------------%
MaxErr = 1e-4;
MaxIt = 3000;
```

Initialization File: Modified Mechanism

```
% This is the initialization file that will set the parameters used in the
% simulink model.  This MUST be run before the simulation will work.  Also
% note that all variables created in this file are located in the Workspace
% and should not be modified directly.

% Several steps need to be done in the initialization:
%-----------------------------------------------------%
% 0) Global variable declarations.  Workspace cleanup, etc.
% 1) Set the locations of the connection points.
% 2) Set leg compliances and System mass/inertia parameters.
% 3) Specify final positioning and force-state.
% 4) Set lengths of leg sections.  Have system in equilibrium at beginning.
%    Also determine the orientations of the legs for proper closed system.
% 5) Determine the leg forces needed to reach equilibrium at the final
% state.  Note: You must mux the positioning and force state information.
% 6) Calculate the values for the PID controllers.
% 7) Additional initializations.
%-----------------------------------------------------%

%-------------------------------------------------------%
% 0) Global variable declarations.  Workspace cleanup, etc.
%-------------------------------------------------------%
clc;
clear;
close all;

global J;
global R;
global MaxErr;
global MaxIt;
global k_val;
global c_val;
global p1ee;
global p2ee;
global p3ee;
global p0ee;

%---------------------------------------------%
```

```
% 1) Set the locations of the connection points.
%-----------------------------------------------%
p1b = [0, -1, 0];
p2b = [sqrt(3)/2, 1/2, 0];
p3b = [-sqrt(3)/2; 1/2; 0];

% p1ee = [0; 1; 0];
% p2ee = [-sqrt(3)/2; -1/2; 0];
% p3ee = [sqrt(3)/2; -1/2; 0];
p1ee = [0; 1; 0] / 2;
p2ee = [-sqrt(3)/2; -1/2; 0] / 2;
p3ee = [sqrt(3)/2; -1/2; 0] / 2;
p0ee = [0; 0; 0.2];

%-------------------------------------------------------------%
% 2) Set leg compliances and System mass/inertia parameters.
%-------------------------------------------------------------%
k_val = [100; 100; 100; 100; 100; 100; 100];
c_val = 100*[100; 100; 100; 100; 100; 100; 100];

% Specify mass and inertia properties of the system.
% Mass -> kg
% Inertia -> kg-m^2
Mass = 1;
Inertia = [1 0 0; 0 1 0; 0 0 1];
% Leg masses are presented in 3 columns: leg base, actuator base, and
% actuator end.  All of these will be essentially zero for primary testing.
LegMasses = [1e-7 1e-3 1e-7;
             1e-7 1e-3 1e-7;
             1e-7 1e-3 1e-7;
             1e-7 1e-3 1e-7;
             1e-7 1e-3 1e-7;
             1e-7 1e-3 1e-7;
             1e-7 1e-3 1e-7];

GenInertia = [1e-7 0 0;
              0 1e-7 0;
              0 0 1e-7];

%-----------------------------------------------%
% 3) Specify final positioning and force-state.
%-----------------------------------------------%
P_final = [0; 0; 1; 0.5; 0.5; 0.5];
w_app = [0; 0; 0; 0; 0; 0]
w_ext = Mass*[0; 0; -9.81; 0; 0; 0] + w_app; % Combination of weight and external
forces.
```

```
%-------------------------------------------------------------------------%
% 4) Set lengths of leg sections.  Have system in equilibrium at beginning.
%    Also determine the orientations of the legs for proper closed system.
%-------------------------------------------------------------------------%
% Set the lengths of the actuator bases to be 0.
% Initial conditions before motion.  Purely to determine initial spring
% lengths/loads.
P_init = [0; 0; 1; 0; 0; 0];
w_init = w_ext;
Info_init = [P_init; w_init];
F_init = MinEnergy(Info_init);

% Correction of the intial rotation matrix.
RxI = [1 0 0; 0 cos(P_init(4)) -sin(P_init(4)); 0 sin(P_init(4)) cos(P_init(4))];
RyI = [cos(P_init(5)) 0 sin(P_init(5)); 0 1 0; -sin(P_init(5)) 0 cos(P_init(5))];
RzI = [cos(P_init(6)) -sin(P_init(6)) 0; sin(P_init(6)) cos(P_init(6)) 0; 0 0 1];
RI = RxI * RyI * RzI;

% Calculate the leg lengths and extract orientation vectors.
Legs = Sp_LL_J(P_init, true, false) / 3;
Orient = J(1:3, :);

% Initial spring lengths.  This is the amount the springs are COMPRESSED.
L_init = zeros(7,1);
for i = 1:7
   L_init(i) = F_init(i) / k_val(i);
end

%-------------------------------------------------------------------------%
% 5) Determine the leg forces needed to reach equilibrium at the final
% state.  Note: You must mux the positioning and force-state information.
%-------------------------------------------------------------------------%
P_input = [P_final; w_ext];
F_legs = MinEnergy(P_input);

%-------------------------------------------------%
% 6) Calculate the values for the PID controllers.
%-------------------------------------------------%
% Specify the system performance.
MaxOvershoot = 0.2; % This is the fraction of distance moved.
SettlingTime = 1;   % Time in seconds.

% Determine the controllers based on mass and inertia in the system.  The
% three controllers for the x, y, z directions will be identical.  However,
% the rotational controllers will be different based on the inertia of the
```

```
% system.  Therefore, 4 sets of calculations are needed in addition to the
% basic performance criteria.

% Determine poles.
zeta = sqrt((log(MaxOvershoot)/pi)^2 / (1 + (log(MaxOvershoot)/pi)^2));
omega_n = 4 / (zeta * SettlingTime);

ClosedLoopPoles = [-zeta * omega_n; omega_n * sqrt(1 - zeta^2)];
RealPole = 0 * ClosedLoopPoles(1);
CharacteristicEq = [1; (-RealPole - 2*ClosedLoopPoles(1));
                (2*RealPole*ClosedLoopPoles(1) + ClosedLoopPoles(1)^2 +
ClosedLoopPoles(2)^2);
                -RealPole*(ClosedLoopPoles(1)^2 + ClosedLoopPoles(2)^2)];

% Design x, y, z controllers.
Kp = CharacteristicEq(3) * Mass;
Ki = CharacteristicEq(4) * Mass;
Kd = CharacteristicEq(2) * Mass;

% Design x-rotational controller.
Kpx = CharacteristicEq(3) * Inertia(1,1);
Kix = CharacteristicEq(4) * Inertia(1,1);
Kdx = CharacteristicEq(2) * Inertia(1,1);

% Design x-rotational controller.
Kpy = CharacteristicEq(3) * Inertia(2,2);
Kiy = CharacteristicEq(4) * Inertia(2,2);
Kdy = CharacteristicEq(2) * Inertia(2,2);

% Design x-rotational controller.
Kpz = CharacteristicEq(3) * Inertia(3,3);
Kiz = CharacteristicEq(4) * Inertia(3,3);
Kdz = CharacteristicEq(2) * Inertia(3,3);

% Design Controllers for the leg springs/dampers.
KpS = zeros(7,1);
KpS = zeros(7,1);
KpS = zeros(7,1);
for LegNum = 1:7
   KpS(LegNum) = CharacteristicEq(3) * LegMasses(LegNum,2);
   KiS(LegNum) = CharacteristicEq(4) * LegMasses(LegNum,2);
   KdS(LegNum) = CharacteristicEq(2) * LegMasses(LegNum,2);
end

%-----------------------------%
% 7) Additional initializations.
```

```
%------------------------------%
MaxErr = 1e-4;
MaxIt = 3000;
```

Minimum Energy Optimization

```
% This function will determine the force that needs to be applied in each
% leg to satisfy the force state.  This will be determined using the
% minimum energy configuration of the system.  The optimal spring lengths
% will be determined, allowing us to easily calculate the force in each
% leg.
% This also has the advantage of allowing us to calculate small changes in
% the forces for control applications.
% The equation used for optimization is d/dL{sum((A[n] + B[n]*k7*d7)^2)} = 0.

% k_val is the vector of spring constants.
function [MinEnergy] = MinEnergy(Pc)

global J;
global k_val;

% extract the positioning and correctional data.
Pos = zeros(6,1);
w_ext = zeros(6,1);
for i=1:6
   Pos(i) = Pc(i);
   w_ext(i) = Pc(i + 6);
end

Sp_LL_J(Pos, true, false);

J_opt = zeros(6, 6);

cont = true;
n = 7;
while cont
   m = 1;
   for i = 1:6
      if m ~= n
         J_opt(:, i) = J(:, m);
         m = m + 1;
      else
         J_opt(:, i) = J(:, m+1);
         m = m + 2;
      end
   end
```

```
   % This checks for closeness to singularity.
   if abs(det(J_opt)) >= 0.1
      cont = false;
   else
      n = n - 1;
   end
end

% Once we have a non-singular reduced Jacobian, we can calculate the leg
% forces using the minimum energy principle.

% These are the values for the constants of the other legs.
An = -J_opt^-1 * w_ext;
Bn = -J_opt^-1 * J(:, n);

kv = zeros(6, 1);
m = 1;
for i = 1:7
   if i ~= n
      kv(m) = k_val(i);
      m = m + 1;
   end
end

% k_val(n) ==> k7
% k_v(i) ====> kn
Au = 0;
Bu = 0;
for i = 1:6
   Au = Au + k_val(n)/kv(i) * An(i)*Bn(i);
   Bu = Bu + k_val(n)^2/kv(i) * Bn(i)^2;
end

% Calculate the force in the optimized leg.
% Positive delta L7 is DECREASE in spring length.
delta7 = -Au / Bu;
F7 = k_val(n) * delta7;

% Determine the forces in the other legs.
Fo6 = zeros(6, 1);
for i = 1:6
   Fo6(i) = An(i) + Bn(i) * F7;
end

% Collect all force terms into one vector.
F_opt = zeros(7, 1);
```

```
m = 1;
for i = 1:7
   if i ~= n
      F_opt(i) = Fo6(m);
      m = m + 1;
   else
      F_opt(i) = F7;
   end
end

MinEnergy = F_opt;
```

Numerical Forward Analysis

This consists of two pieces, the iterative process and the reverse analysis that generates the modified Jacobian and performs leg length calculations. The iterative process is relatively simple in form and is easily adapted to many systems.

Iterative Process

```
% This is the numerical forward analysis for the Spatial mechanism.
function [Sp_Forward_Leg] = Sp_Forward_Leg(Inputs)

global J;
global MaxErr;
global MaxIt;

DesLeg = Inputs(1:7);
InitGuess = Inputs(8:13);

CurGuess = InitGuess;
CurLeg = Sp_LL_J(CurGuess, true, true);
Err = DesLeg - CurLeg;

NumIt = 0;
while max(abs(Err)) >= MaxErr && NumIt < MaxIt,
   delta = J * Err;
   CurGuess = CurGuess + 0.25*delta;
   NumIt = NumIt + 1;
   CurLeg = Sp_LL_J(CurGuess, true, true);
   Err = DesLeg - CurLeg;
end

NumIt
Sp_Forward_Leg = CurGuess;
```

Reverse Analysis

```
% This function will calculate leg lengths and the Jacobian for a
% particular configuration
function [Sp_LL_J] = Sp_LL_J(Pos, both, it_J)

global R;
global p1ee;
global p2ee;
global p3ee;
global p0ee;

% Define Rotation matrix
Rx = [1 0 0; 0 cos(Pos(4)) -sin(Pos(4)); 0 sin(Pos(4)) cos(Pos(4))];
Ry = [cos(Pos(5)) 0 sin(Pos(5)); 0 1 0; -sin(Pos(5)) 0 cos(Pos(5))];
Rz = [cos(Pos(6)) -sin(Pos(6)) 0; sin(Pos(6)) cos(Pos(6)) 0; 0 0 1];
R = Rx * Ry * Rz;

% Central point
ss00 = [Pos(1); Pos(2); Pos(3)];

% Define base points
p1b = [0; -1; 0];
p2b = [sqrt(3)/2; 1/2; 0];
p3b = [-sqrt(3)/2; 1/2; 0];

% Define ee points
% p1ee = [0; 1; 0];
% p2ee = [-sqrt(3)/2; -1/2; 0] / 2;
% p3ee = [sqrt(3)/2; -1/2; 0] / 2;
p1e = ss00 + R*p1ee;
p2e = ss00 + R*p2ee;
p3e = ss00 + R*p3ee;
p0e = ss00 + R*p0ee;

% Define leg orientation vectors.
ss12 = p2e - p1b;
ss13 = p3e - p1b;
ss21 = p1e - p2b;
ss23 = p3e - p2b;
ss31 = p1e - p3b;
ss32 = p2e - p3b;
ss00 = p0e; % Redefine ss00 since the seventh leg is no longer in the plane of the EE.

LegLengths = zeros(7,1);
LegLengths(1) = sqrt(dot(ss12, ss12));
```

```
LegLengths(2) = sqrt(dot(ss13, ss13));
LegLengths(3) = sqrt(dot(ss21, ss21));
LegLengths(4) = sqrt(dot(ss23, ss23));
LegLengths(5) = sqrt(dot(ss31, ss31));
LegLengths(6) = sqrt(dot(ss32, ss32));
LegLengths(7) = sqrt(dot(ss00, ss00));

if(both)
   s12 = ss12 / LegLengths(1);
   s13 = ss13 / LegLengths(2);
   s21 = ss21 / LegLengths(3);
   s23 = ss23 / LegLengths(4);
   s31 = ss31 / LegLengths(5);
   s32 = ss32 / LegLengths(6);
   s00 = ss00 / LegLengths(7);

   r1 = R * p1ee;
   r2 = R * p2ee;
   r3 = R * p3ee;

   s12L = cross(r2, s12);
   s13L = cross(r3, s13);
   s21L = cross(r1, s21);
   s23L = cross(r3, s23);
   s31L = cross(r1, s31);
   s32L = cross(r2, s32);
   s00L = [0; 0; 0];

   % scale the moment vectors if needed for forward analysis
   if it_J
      s12L = s12L / LegLengths(1);
      s13L = s13L / LegLengths(2);
      s21L = s21L / LegLengths(3);
      s23L = s23L / LegLengths(4);
      s31L = s31L / LegLengths(5);
      s32L = s32L / LegLengths(6);
      s00L = s00L / LegLengths(7);
   end

   global J;
   J = [s12 s13 s21 s23 s31 s32 s00; s12L s13L s21L s23L s31L s32L s00L];
end

Sp_LL_J = LegLengths;
```

Torque Transformation

```
% It is necessary to determine the control torques in terms of the global
% coordinates.
function [TorqueTransform] = TorqueTransform(Pos)

% Define Rotation matrix
Rx = [1 0 0; 0 cos(Pos(4)) -sin(Pos(4)); 0 sin(Pos(4)) cos(Pos(4))];
Ry = [cos(Pos(5)) 0 sin(Pos(5)); 0 1 0; -sin(Pos(5)) 0 cos(Pos(5))];
Rz = [cos(Pos(6)) -sin(Pos(6)) 0; sin(Pos(6)) cos(Pos(6)) 0; 0 0 1];
R = Rx * Ry * Rz;

TorqueTransform = R * [Pos(7); Pos(8); Pos(9)];
```

LIST OF REFERENCES

1. Duffy, J., *Statics and Kinematics with Applications to Robotics*, Cambridge University Press, 1996, New York, NY.

2. Crane, C., Duffy, J., *Kinematic Analysis of Robot Manipulators*, Cambridge University Press, 1996, New York, NY.

3. Baiges-Valentin, I., *Dynamic Modeling of Parallel Manipulators*, Ph.D. dissertation, 1995, University of Florida, Gainesville, Fl.

4. Lee, J., Duffy, J., and Rooney, J., *An Initial Investigation into the Geometrical Meaning of the (Pseudo-) Inverses of the Line Matrices for the Edges of Platonic Polyhedra*, Presentation at Sir Robert Stawell Ball 2000 Symposium, University of Cambridge, UK, July 2000.

5. Zhang, Y., Duffy, J., and Crane, C., *The Optimum Quality Index for a Spatial Redundant 4-8 In-Parallel Manipulator*, Proceedings of the Advances in Robot Kinematics Conference, Piran, Slovenia, June 2000, p. 239-248.

6. Zhang, Y, Duffy, J., and Crane C., *The Optimal Quality Index for a Spatial Redundant 8-8 In-Parallel Manipulator*, Presentation at Proceedings of the ASME Mechanisms Conference, Baltimore, Md., Sep 2000.

7. Lee, J., *Investigations of Quality Indices of In-Parallel Platform Manipulators and Development of Web Based Analysis Tool*, Ph.D. dissertation, 2000, University of Florida, Gainesville, Fl.

8. Zhang, Y., *Quality Index and Kinematic Analyisis of Spatial Redundant In-Parallel Manipulators*, Ph.D. dissertation, 2000, University of Florida, Gainesville, Fl.

9. Bonev, I., *Geometric Analysis of Parallel Mechanisms*, Ph.D. dissertation, 2002, University of Florida, Gainesville, Fl.

10. Tyler, C., *In-Parallel Passive Compliant Coupler for Robot Force Control*, MS thesis, 2000, University of Florida, Gainesville, Fl.

11. Abbasi, W., Ridgeway, S., Adsit, P., Crane, C., and Duffy, J., *Investigation of a Special 6-6 Parallel Platform for Contour Milling*, Proceedings of the ASME Manufacturing Engineering Division 1997 International M.E. Congress and Exposition (IMECE), Dallas, Nov 97, pp. 373-380.

BIOGRAPHICAL SKETCH

Jean-François Ajit Kamath received a Bachelor of Science in mechanical engineering in the fall of 2002. He will receive his Master of Science in mechanical engineering from the University of Florida in August 2005.