2014-12-01

A Variable-Stiffness Compliant Mechanism for Stiffness-Controlled Haptic Interfaces

Jeffrey C. Hawks
Brigham Young University - Provo

A Variable-Stiffness Compliant Mechanism for Stiffness-Controlled Haptic Interfaces

Jeffrey C. Hawks

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Mark B. Colton, Chair
Steven K. Charles
Larry L. Howell

Department of Mechanical Engineering

Brigham Young University

December 2014

ABSTRACT

A Variable-Stiffness Compliant Mechanism for Stiffness-Controlled Haptic Interfaces

Jeffrey C. Hawks
Department of Mechanical Engineering, BYU
Master of Science

In this research a variable-stiffness compliant mechanism was developed to generate variable force-displacement profiles at the mechanisms coupler point. The mechanism is based on a compliant Roberts straight-line mechanism, and the stiffness is varied by changing the effective length of the compliant links with an actuated slider. The variable-stiffness mechanism was used in a one-degree-of-freedom haptic interface to demonstrate the effectiveness of varying the stiffness of a compliant mechanism. Unlike traditional haptic interfaces, in which the force is controlled using motors and rigid links, the haptic interface developed in this work displays haptic stiffness via the variable-stiffness compliant mechanism. The force-deflection behavior of the mechanism was analyzed using the Pseudo-Rigid Body Model (PRBM), and two key parameters, K_Θ and γ, were optimized using finite element analysis (FEA) to match the model with the behavior of the device. One of the key features of the mechanism is that the inherent return-to-zero behavior of the compliant mechanism was used to provide the stiffness feedback felt by the user. A prototype haptic interface was developed capable of simulating the force-displacement profile of Lachmans Test performed on an injured ACL knee. The compliant haptic interface was capable of displaying stiffnesses between 4200 N/m and 7200 N/m.

Keywords: Variable Stiffness, Variable-Stiffness Compliant Mechanism, Haptic Interface, Compliant Mechanism, Straight-Line Mechanism, Master's Thesis, BYU

ACKNOWLEDGMENTS

First and foremost, I would like to thank my wife, Rachel, for being patient and supportive throughout the graduate process. She also gets credit for being the top grammar editor in my life. Thank you Rachel.

Next, I would like to thank my son, Caledon, for helping me write my thesis. Thanks Cal.

I would like to thank Dr. Colton for taking me on as a graduate student and allowing me to integrate my personal interests of bio-mechanics into my research. Thank you Dr. Colton

I would like to thank Marcus Tanner and Jack Webster for making my design come to life, spending long hours in the machine shop and Instron room. Thank you both.

I would like to thank Ariana Pederson and Jessica Morrise for their help with preliminary CAD designs and Matlab GUI development respectively. Thank you both.

I would like to thank the Mechanical Engineering Department for the TA/RA Fellowship that I was awarded. It helped me finance my first year of graduate school.

I would also like to thank Dr. Howell for his guidance in developing the PRBM for the compliant mechanism designed in this research. Thank you Dr. Howell.

Finally, I would like to thank those in Dr. Colton's lab group without whose help I would still be stuck in one of the many pot holes of research. Thank you Brandon Norton, Jacob Robinson, Dallin Swiss, and Sam McDonald. All of your contributions were priceless.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

In this research a variable-stiffness compliant mechanism was developed to control a novel one-degree-of-freedom haptic interface. Unlike traditional haptic interfaces, in which the force is controlled using motors and rigid links, the haptic interface developed in this work displays haptic stiffness via the variable-stiffness compliant mechanism. The objective of this work was to explore the use of compliant mechanisms in haptic interfaces.

The following is a brief description of our novel approach of incorporating compliant mechanisms into a haptic device. Some of the inherent benefits of using compliant members are reduced friction, weight, and potentially cost. There are also some complications that are added when using compliant mechanisms, namely the return-to-zero behavior. The haptic interface developed in this research will capitalize on the return-to-zero behavior because it is a stiffness controlled interface as opposed to a typical force controlled interface.

1.1 Background

1.1.1 Typical Haptic Interfaces

Haptic interfaces are robotic force-feedback devices that provide a sense of touch to a user that is interacting with a virtual world or teleoperated device. Haptic interfaces typically consist of rigid links driven by motors to exert forces on a user's hand to simulate the feel of an object or environment. An example of such a device, the Phantom, can be seen in Figure 1.1. Most haptic interfaces are force/torque controlled through commands to the motors.

1.1.2 Compliant Mechanisms and Variable Stiffness

The haptic interface developed in this work is based on compliant mechanisms, which transfer force or displacement to the user through the flexibility of their components. Compliant

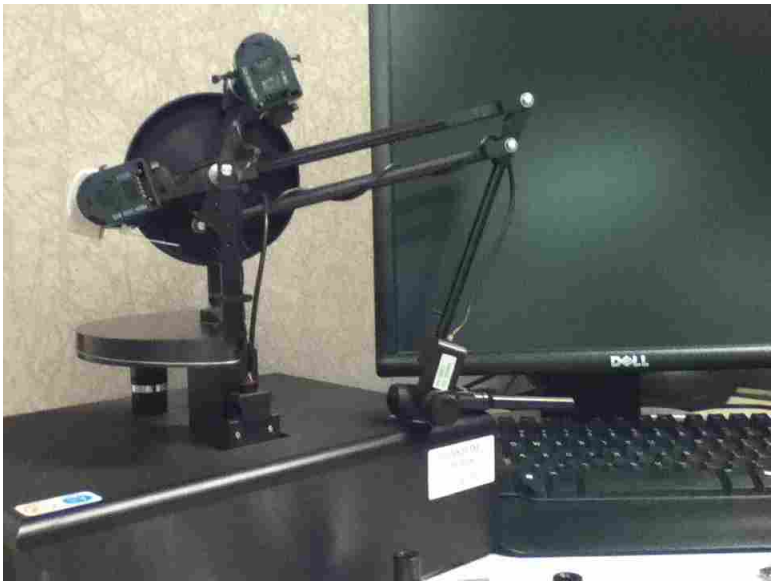


Figure 1.1: Phantom Premium. Made by Sensable

mechanisms have the potential to enable haptic interfaces to create more realistic touch sensations, and may also result in cheaper, lighter, and safer haptic interfaces.

Compliant mechanism have typically been difficult to analyze because small deflection analysis equations are not sufficient to predict their behavior. This research has relied on the use of the Pseudo-Rigid Body Model (PRBM) to predict the behavior of the compliant members [1].

The stiffness of compliant mechanisms is set by three components: the material, the geometry (i.e. length, width, etc.), and the boundary conditions. To create a variable-stiffness compliant mechanism one or more of these three components needs to be variable. In this research the length of the compliant links was used to vary the stiffness. This was accomplished by varying the effective length of the compliant member in the haptic device.

1.1.3 Application

The return-to-zero behavior of compliant mechanisms is typically seen as a drawback when using compliant mechanisms in haptic interfaces because the user feels forces not associated with the virtual (simulated) environment. In this work, the return-to-zero behavior is used beneficially because the haptic device is stiffness-controlled, rather than force-controlled. In essence, the controllable return-to-zero behavior provides the variable-stiffness haptic feedback felt by the user.

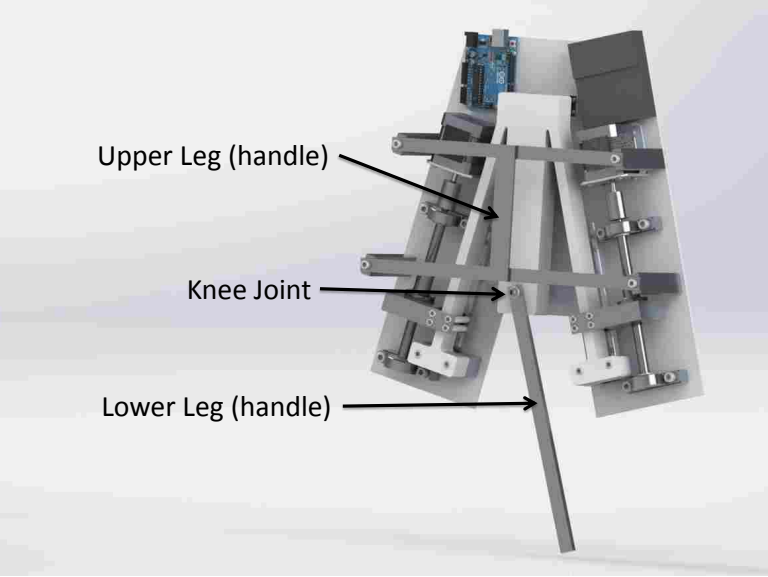


Figure 1.2: Haptic Knee Simulator

This is demonstrated in the development of a knee simulator that can be used to teach health students the proper motion and feel to expect when performing Lachman's test, a test where the lower leg is translated relative to the upper leg to measure the strength of the Anterior Cruciate Ligament (ACL) of the knee. A rendering of the prototype haptic interface is shown in Figure 1.2. In this research, the prototype knee simulator is a one-degree-of-freedom mechanism used to incorporate and investigate variable-stiffness control in haptic interfaces. Future research may expand its use to multiple knee tests with several degrees of freedom.

The haptic knee simulator was designed to match the translational force-deflection profile of the knee obtained in [2], which is the black force profile recreated in Figure 1.3. Two other knee force profiles are shown in Figure 1.3 to represent the various studies performed on the knee. In this work we did not determine which force profile best represented the knee, but several are shown to give the reader an idea of the possible knee force profiles.

The haptic knee simulator consist of three parts: an upper leg, a lower leg, and a haptic knee simulator that acts as the knee joint between the upper and lower leg. The final prototype can be seen in Figure 1.2. The mechanism that makes up the haptic knee is large, and as such consists of the knee and the upper leg. Details of the design and analysis methods, as well as the prototype device will be given in later chapters.

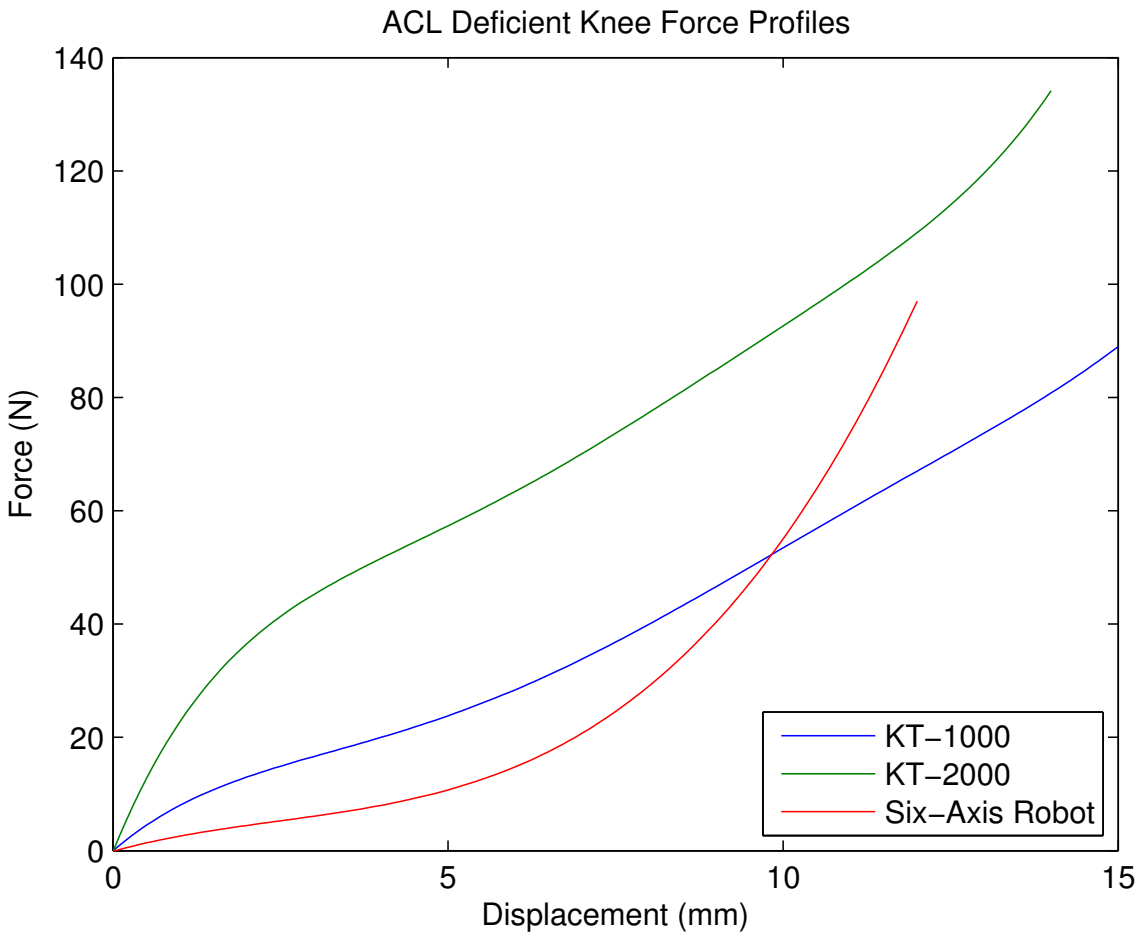


Figure 1.3: ACL Force-Deflection Profile. Force-deflection profile of an ACL injured knee as measured by three different mechanisms KT-1000 arthrometer [2], KT-2000 arthrometer [3], and a six-axis robot [4]

1.2 Literature Review

1.2.1 Variable Stiffness in Robotics and Haptics

There has been a push in robotics to make robots safe around humans. One method to make robots safer is to incorporate compliant elements (i.e. springs) in the robot arms, and it is typically preferable for the compliance to be tunable [5] [6] [7].

Variable stiffness designs in robotics range from simple cantilever beams with a mechanism that can change the effective length of the beam [8] [9] [10] [11] to traditional springs whose length or position is adjustable [12] [13] [14].

1.2.2 Compliance in Haptics

There has been considerable work on incorporating variable-stiffness elements in robot designs, but this has not transitioned as readily into the design of haptic interfaces. Research at BYU has shown that it is possible to get a prescribed force-deflection relationship with compliant mechanisms [15] [16] [17]. Elsewhere, a haptic laparoscopic (surgical scissors) device was developed that is virtually transparent to the user because of its incorporation of variable-stiffness components [18]. Prescribed force-deflection relationships and transparency are two key aspects of successful haptic devices.

Haptic devices have started to take advantage of the benefits that come from using compliant mechanisms such as reduced wear, no need for lubrication, less weight, etc. This has been most important in medical devices where clean room requirements are high, but large motion in the device is still needed [19] [20]. The advantages of compliant mechanisms have also led to their incorporation into traditional haptics, such as a compliant pantograph [21]. But this has led to an increase in the complexity of the controls. Another application has been to take advantage of the lower force requirements to flex plastic compliant mechanism to develop low force instrumentation for mechanism that interface with people [22].

Those few haptic devices that incorporate compliant mechanisms into their design still fail to take full advantage of using compliant mechanisms. In the end, the key component in compliant mechanisms is that when they are deflected there is a resultant force, and this force needs to be the key to the force feedback in haptic devices as opposed to the problem that needs to be overcome.

1.2.3 Straight-line Mechanisms

Lachman's test on the knee is based on a translational displacement of the lower leg relative to the upper leg, so straight-line mechanisms, especially haptic and/or compliant mechanisms, were studied. Typical straight-line mechanisms that have been developed throughout the years, such as Watt, Hoeken, etc. can be found in [23]. Each of the common straight-line mechanisms have strengths depending on the application. For example, in [24] the Hoeken straight-line mechanism was chosen as the best choice for a haptic device because its input-to-output torque ratio was ideal for a haptic device.

In [25] parallel guiding mechanisms were used to make simple compliant straight-line mechanisms. To ensure a straight-line two parallel guiding mechanisms were stacked on top of each other to cancel out the inherent vertical motion. Building off of the simple mechanism developed by [25] a more sophisticated compliant straight-line mechanism was developed by [26]. The mechanism was created using mirrored Robert's straight-line mechanisms. The mirroring was necessary to turn the approximate straight-line mechanism into an exact straight-line mechanism. Exact straight-line mechanisms do exist, but they are not as easy to synthesize into compliant mechanisms.

In this research, a Robert's straight-line mechanism was used to create a variable-stiffness straight-line mechanism.

1.3 Design Requirements

1.3.1 Improvements in Haptics Through Compliance

To demonstrate the use of compliance in haptic interfaces the novel mechanism was developed using the following criteria:

- The haptic device produces its force feedback through the inherent stiffness of the compliant mechanism
- The force feedback is controlled by changing the stiffness of the compliant mechanism
- The novel device demonstrates that the inherent return-to-zero behavior of compliant mechanisms can be used as the force feedback for haptic devices

It is anticipated that future compliant haptic interfaces may also have the following benefits.

- Reduced friction, weight, and cost compared to traditional haptic devices
- Increased transparency through matching the compliant mechanism's design/dynamics to the dynamics of the object being modeled.

1.3.2 Application Specific Requirements

Current Knee Simulators

The knee is one of the most studied parts of the body because of its role in mobility. Many studies have developed dynamic models and equations of motion to describe the movement of the knee [27] [28] [29] [30]. Others have performed multiple tests on cadaveric specimens and in vivo subjects with robots and other machinery to measure knee parameters [31] [32] [33] [34]. The combination of these works provides a comprehensive understanding of the knee's motion, and this information was used to develop the haptic simulator of the knee.

Other knee simulators fall into one of two categories: too expensive for use in a classroom [4] [35], or not enough degrees of freedom to fully represent the knee [36]. The first category of knee simulators involve robots, often six-axis robots, to simulate the motion of the knee, which make the knee simulator expensive for use in a teaching environment. The second category of knee simulators focuses on a specific aspect of the knee, e.g. range of motion of knee flexion, without the ability to easily expand the concept to a full knee simulator.

Knee Simulator Requirements

In this work we develop a simple knee simulator as a test application for variable-stiffness haptic interfaces. The following are a list of the requirements used to develop a successful knee simulator:

- Modular design, such that components of the knee's motion can be developed and tested one DOF at a time
  - In this research, anterior-posterior translation of the knee was developed
- Match the force-deflection curve for an injured ACL knee, which requires certain force and displacement ranges
  - Range of force 0 to 90 Newtons
  - Range of displacement 0 to 15 millimeters

1.4 Contributions

In this research, a Robert's straight-line mechanism was designed as the bases for a variable-stiffness compliant haptic device. This thesis furthered the work of incorporating compliant mechanisms into haptic devises. This thesis accomplished the following achievements.

- Demonstrated the ability to control haptic devices through variable-stiffness compliant mechanisms
- Showed that the inherent return-to-zero force in a compliant mechanism can be designed to match a simulated force feedback profile
- Developed a Pseudo-Rigid Body Model (PRBM) to predict the force output for a variable-stiffness Robert's straight-line compliant mechanism
- Demonstrated the feasibility of a variable-stiffness haptic interface by developing a prototype one degree of freedom haptic knee simulator capable of simulating the feel of Lachman's test

Along with these achievements this thesis has opened the door for the following progress to take place in the development of future haptic interfaces.

- Take advantage of the common benefits that come from using compliant mechanisms
  - Decrease the cost of the mechanism
  - Decrease the weight of the mechanism
  - Decrease the wear of the mechanism
- Improve the transparency of the haptic interface by matching the compliant mechanism's dynamics to the simulation's dynamics
- Increase the degrees of freedom and range of force capable of by compliant mechanism in haptic interfaces

CHAPTER 2. DESIGN OF COMPLIANT HAPTIC INTERFACE

2.1 Introduction

The object of this research was to design a novel one degree of freedom translational haptic interface. This chapter will discuss the design of the compliant mechanism used to provide the force feedback for the haptic interface. Our approach was to use and modify traditional synthesis and analysis tools such as optimization, the Pseudo-Rigid Body Model (PRBM) and Finite Element Analysis (FEA). This chapter will cover the following topics.

- The functional specifications used to design the mechanism
- An overview of the final design
- The kinematics of the compliant mechanism
- The design of the force output (dimensions) of the mechanism
- The final design of the mechanism

2.2 Functional Specifications

As in any good design, the needs and the functional specifications need to be laid out to guide the design. The functional specifications were developed with two main goals. First, the haptic interface was to be controlled by changing the stiffness of a compliant mechanism. This guided the method that was to be used in developing the haptic interface. Second, the haptic interface needed to be able to replicate Lachman's test for an injured knee. This provided a bound on the forces and displacements needed to be replicated by the haptic interface. Below is a list of the functional specification to which the compliant haptic interface was designed.

Since the goal was to simulate a human knee the first set of design requirements revolve around designing the haptic interface to have similar dimensions to a human leg.

- ”Upper Leg” - This is the stationary part of the haptic interface. A user grips a handle on the upper leg to steady the device while the lower leg is moved relative to the upper leg.
  - Diameter $\approx 431mm$
  - Length $\approx 431mm$
  - Weight $< 58N$
- ”Lower Leg” - This is part of the haptic interface that performs a translational movement. There is a handle for the lower leg that the user grips to manipulate with respect to the upper leg.
  - Diameter $\approx 304mm$
  - Length $\approx 431mm$
  - Weight $< 36N$
- ”Knee” - This is the compliant mechanism portion of the haptic interface, and it is the key to providing the force feedback
  - Diameter $\approx 381mm$
  - Length $\approx 76mm$
  - Weight $< 13N$

The second set of design requirements revolves around the need to produce the same translational movement that occurs when Lachman’s test is performed on an ACL injured knee.

- Movement, the translational motion of the lower leg with respect to the upper leg during the knee test
  - Ideal: -20 to $20mm$
  - Acceptable: 0 to $15mm$
- Force, the resistance to translational motion of the lower leg with respect to the upper leg
  - Ideal: -150 to $150N$

  - Acceptable: 0 to 90N

- The actuator controlling the force profile needs to cover the full range of forces in about 1 second

- Prescribed force-displacement curve has $< 10\%$ error

These specifications were used to develop the compliant haptic interface. The designs and decisions made through the development process will be described in this chapter in the following order. First, the development of the kinematics will be described. This involved selecting a kinematic chain that would produce the desired motion, and optimizing the kinematic chain to remain a straight-line mechanism as the stiffness was varied. Second, the selection of appropriate dimensions of the mechanism will be discussed, which is key to producing the desired range of forces and the desired safety factor for the mechanism. Finally, the verification and fine tuning of the design will be discussed.

2.3 Overview of the Design

The final design chosen was a Robert's straight-line mechanism with links 2 and 4 being compliant (see Figure 2.1). To provide variable force-displacement profiles two sliders are used to move along the compliant links. The length of links 2 and 4 do not change, but the sliders change the length of the compliant links that are free to flex during the motion of the mechanism. As the sliders move up along the link lengths, the effective length of the compliant links becomes shorter, thus increasing the force felt by the user at the coupler point. The reverse is also true; as the sliders move down along the compliant links the force felt by the user decreases.

The coupler point of a four bar mechanism is a point on the third link whose path is vital to the function of the mechanism. Often the third link of a four bar mechanism is drawn/visualized as a straight-line, but in practice can be any shape or size as long as it connects links 2 and 4. In this research, link 3 looks like a "T," and the coupler point is located at the base of the "T," as indicated by a black dot in Figure 2.1. In the compliant haptic interface the coupler point was placed to be the pivot point between the two grips that the user will use. For the application of the knee test the coupler point lines up with the knee joint that is being tested.

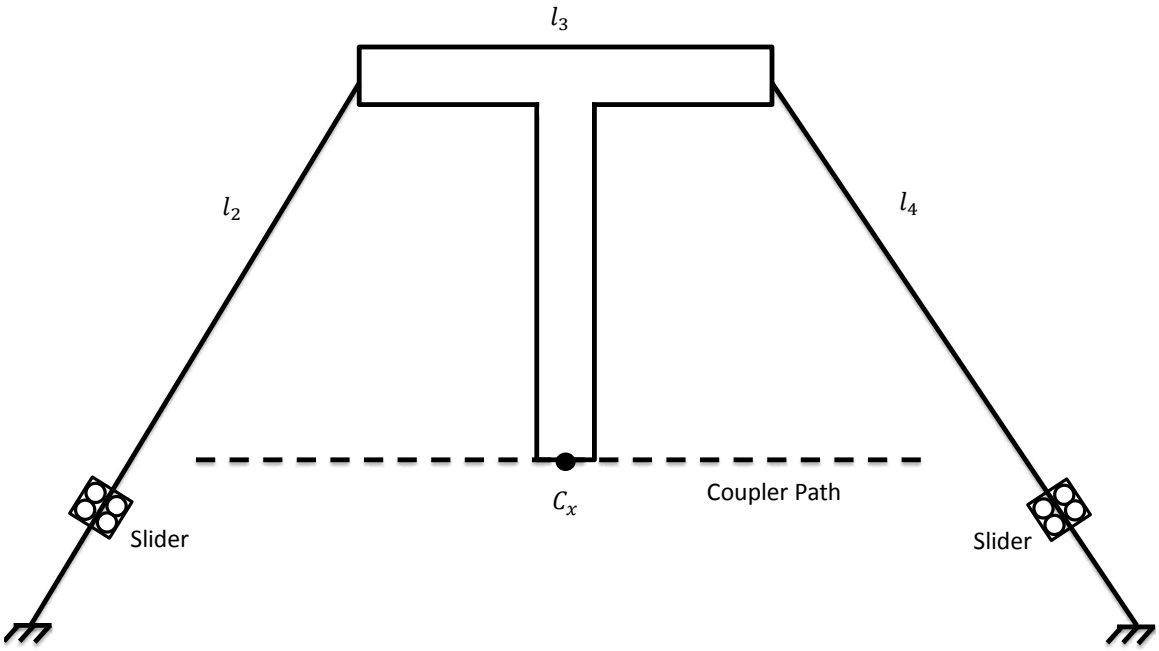


Figure 2.1: Final Compliant Mechanism Design. The final Robert's straight-line mechanism used as the center piece for the compliant haptic interface. The sliders move along the compliant links (l_2 and l_4) to change their effective length. The coupler point (C_x), the end point of the ridge link (l_3), is manipulated by the end user and follows the coupler path at all lengths of the compliant links

Polypropylene, a common compliant mechanism material, was chosen as the material for the compliant mechanism. Polypropylene has a high yield strength to Young's modulus ratio, which allows it to be very flexible, but still provides a substantial resistance. Other materials were analyzed, but none of them performed as well as polypropylene.

2.4 Kinematics Design

2.4.1 Comparison of Straight-line Mechanisms

A one degree of freedom linear motion was desired for the haptic interface for two reasons. First, this provided the simplicity needed while exploring the novel concept of providing

force feedback through changing the stiffness of a compliant mechanism. Second, the linear motion matches well with several applications in the medical field, of which we are focusing on the translational motion of the knee during Lachman's test.

Many straight-line mechanisms have been presented in literature [23], ranging from crank sliders to four-bar mechanisms. To create a successful compliant haptic device it is crucial to maintain a straight-line as the stiffness is varied. This is especially difficult when developing a fully compliant mechanism because as the stiffness changes (in this case the stiffness will be varied by changing the length of the compliant links in the mechanism) the link length ratios change. When the ratios of the link lengths change the path of the coupler is changed, and the change in the coupler path needs to be minimized to maintain an approximate straight line.

To be a successful compliant, variable-stiffness, straight-line mechanism several criteria needed to be met:

- The mechanism must maintain a straight-line as the stiffness changes. In this research it was decided to change the stiffness by changing the effective length of the compliant members in the mechanism, and this causes the coupler path to change. So care must be used to ensure that when the compliant link lengths change the straight-line path is not compromised.

- Ideally, if multiple compliant members are involved they should be the same length, thus allowing for simpler control.

- Preferably, the links should not cross over one another. Typically, fully compliant mechanisms are manufactured using a planar process (laser printer, 3 axis CNC, etc), and if there are crossing links in the mechanism a non-planar manufacturing process would need to be employed, thus, increasing the difficulty of manufacturing.

Four straight-line mechanisms were selected as possible candidates for the compliant haptic device: Robert, Watt, Chebysev, and Evan 3 as described in [23], and were compared to each other and evaluated against the criteria above. The motion of each mechanism is shown in Figure 2.2. For each mechanism, links 2 and 4 (blue lines in the figure) were assigned to be the compliant links. While links 3 (the red line(s) in the figure) and link 1 (the ground link, not featured in the figure visibly) were assigned to be rigid.

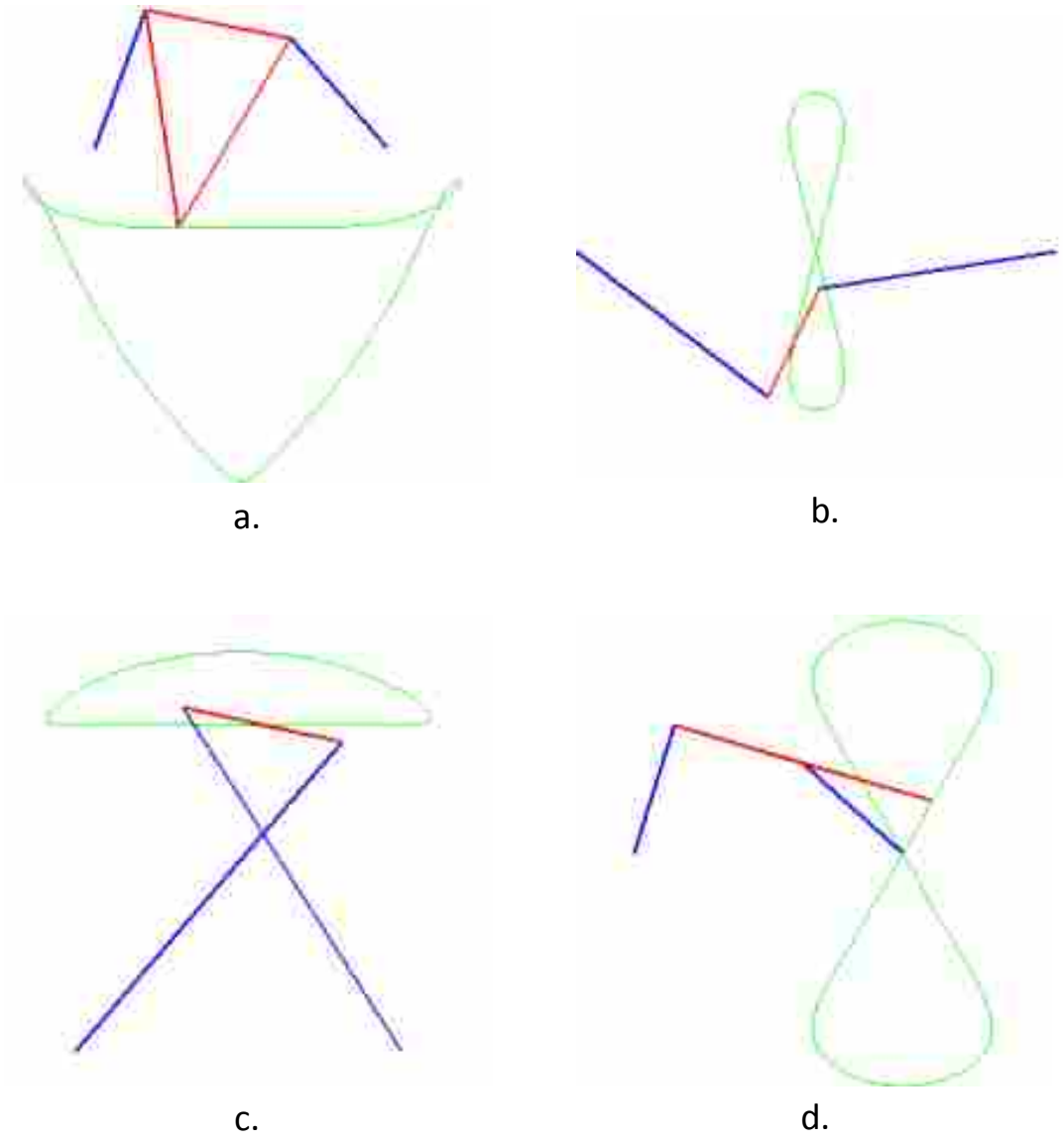


Figure 2.2: Straight-Line Mechanism Comparison. Four common straight-line mechanisms with the correct link length ratios are shown above (a. Robert, b. Watt, c. Chebysev, d. Evan 3). The blue lines represent the second and fourth links in the four bar mechanism. The red lines represent the third link and coupler of the mechanism. The green line represents the path that the coupler takes during the motion of the mechanism. The figure was generated with FourBar.m , a matlab script developed by Dr. Brian Jensen and Yanal Issac at Brigham Young University

The four candidate mechanisms, all of similar size, were evaluated in simulation using methods similar to those described in [24] to determine the linearity of each one as the length of the compliant members varied (links 2 and 4). The tests involved three parts, where each mechanism was to have a coupler path that moved in a straight-line along the horizontal direction, or the x axis. Ideally, the coupler path would move in the horizontal direction with no change in the vertical direction, or the y axis.

First, the change in the y axis over the change in the x axis ($\Delta y/\Delta x$) was measured from the coupler path. This test was important because a high enough $\Delta y/\Delta x$ will be felt and perceived as a deviation from a straight-line by a user. The coupler point was moved in simulation along the entire coupler path at the given link lengths for the mechanism, and the max $\Delta y/\Delta x$ was recorded. Then links 2 and 4 were shortened by 1 mm, and the coupler point was moved along the coupler path, once again recording the max $\Delta y/\Delta x$ for the shortened link lengths. This process was repeated, and the $\Delta y/\Delta x$ was evaluated and recorded for 10 different lengths of the compliant links. Links 2 and 4 changed length similar to how the final design would change the effective length of the compliant links. The most important part of the testing was to see which mechanisms maintained a straight-line as links 2 and 4 were varied.

Second, the change in the vertical direction of the coupler point while moving along the coupler path was measured (Δy). This test is important because the farther the mechanism deviates from a straight-line, the easier it is for a user to sense the deviation. As described above, the coupler point would move along the coupler path for each change in the length of the compliant links, 2 and 4. For each change in length of the compliant links the max Δy was recorded. It was important to find the mechanism that maintained a small Δy through all of the changes of length of the compliant links.

Finally, the total linear distance traveled by the coupler point along the coupler path was measured for each mechanism. As described above the lengths of links 2 and 4 were varied 10 times, and the test was repeated each time. This test was important because it presented a comparison of the size of the mechanism needed to cover a certain linear distance.

The results for the three tests are shown in Tables 2.1 through 2.3.

As can be seen in the tables, only the Evan's 3 mechanism was significantly worse than the other three mechanism. The Chebysev mechanism was also excluded from further consideration

Table 2.1: Max Δy/Δx. Each column underneath Max Δy/Δx represents a relative change of 1 mm in the compliant link lengths (links 2 and 4)

Mechanism	**Max Δy/Δx**										**Average**	**Rank**
Robert	0.08	0.04	0.05	0.06	0.07	0.08	0.09	0.10	0.11	0.12	0.08	3
Watt	0.03	0.02	0.02	0.03	0.03	0.04	0.05	0.07	0.08	0.10	0.05	1
Chebysev	0.01	0.03	0.04	0.05	0.06	0.07	0.08	0.10	0.11	0.12	0.07	2
Evan 3	0.04	0.04	0.05	0.07	0.09	0.11	0.13	0.15	0.16	0.18	0.10	4

Table 2.2: Max Δy. Each column underneath Max Δy represents a relative change of 1 mm in the compliant link lengths (links 2 and 4)

Mechanism	**Max Δy (mm)**										**Average**	**Rank**
Roberts	0.45	0.18	0.22	0.26	0.31	0.36	0.41	0.47	0.53	0.60	0.38	1
Watt	0.19	0.17	0.18	0.24	0.34	0.47	0.65	0.89	1.19	1.58	0.59	2
Chebysev	0.12	0.06	0.20	0.32	0.43	0.60	0.81	1.04	1.31	1.60	0.65	2
Evan's 3	0.08	0.45	0.74	1.01	1.25	1.48	1.71	1.92	2.14	2.35	1.31	4

because its crossing links would make it difficult to manufacture as a fully compliant mechanism. The final design was between the Watt and the Robert straight-line mechanisms. In the end, it was decided to use Robert's mechanism because it fits the knee application better than the Watt mechanism because it would be easily separated into an upper and lower leg. The MATLAB code and other graphics used in the full mechanism comparison can be found in Appendix A.

2.4.2 Pseudo-Rigid Body Model

The Pseudo-Rigid Body Model (PRBM) [1] was used to synthesize the best dimensions for the Robert's straight-line mechanism (refer to Section 6.8 of [1] for the derivation). Figure 2.3 is a visualization of the PRBM for a Robert's straight-line mechanism with links 2 and 4 being compliant and a force applied at the coupler point, P.

The key to the PRBM is to model the compliant links in the mechanism with rigid links connected by pin joints with torsional springs, representing the stiffness of the compliant links. The process used to perform this transformation is fully described in Chapter 8 of [1]. In this section I describe the most important aspects of the PRBM used to solve Robert's straight-line mechanism.

Table 2.3: Coupler Path Distance. Each column underneath Distance Traveled by the Coupler Point represents a relative change of 1 mm in the compliant link lengths (links 2 and 4)

Mechanism	**Distance Traveled by the Coupler Point (mm)**										**Average**	**Rank**
Roberts	37.7	35.6	35.0	34.7	34.8	35.0	35.2	35.5	35.9	36.2	35.5	1
Watt	33.2	33.9	34.5	35.1	35.8	36.4	37.0	37.4	37.8	38.0	35.9	1
Chebysev	34.2	35.0	35.6	36.2	36.8	37.4	38.0	38.5	38.8	39	37.0	1
Evan's 3	23.8	20.4	18.8	17.8	17.2	16.7	16.3	16.0	15.7	15.6	17.8	4

In this research, the Robert's straight-line mechanism was designed to be fully compliant, meaning that all of the motion would come from the flexing of compliant members. It was chosen to make links 2 and 4 compliant links that are fixed rigidly at the ends to the coupler link (link 3) and the ground link (link 1). Often this is referred to as a fixed-fixed compliant link. The next important design feature was that the user can manipulate the position of the coupler point via the application of a force, as represented in Figure 2.3 by P_x and P_y.

In the PRBM the compliant link lengths are multiplied by a variable called γ to find the location of the equivalent pin joint, and the stiffness of the spring representing the compliant beam is found using the variable K_Θ. Both of these variables are dependent upon the boundary conditions of the compliant link lengths. γ and K_Θ have been tabulated for several mechanisms with varying boundary conditions [1], [26]. To aid in the design the same γ and K_Θ as used in a parallel guiding mechanism were used for the initial design ($\gamma = 0.8517$ and $K_\Theta = 2.67617$), and then were later tuned to be specific for the boundary conditions of the mechanism designed in this research (see Section 2.6).

For example, if the second link (l_2) of the compliant mechanism is 5 cm long then the PRBM uses the equation $r_2 = \gamma l_2$ to find the equivalent rigid link length to be $r_2 = 4.25$ cm, based on $\gamma = 0.85$.

Once the link lengths have been modeled as rigid links and the springs have been added to the pin joints it can be analyzed using virtual work. Applying the principles of virtual work to the mechanism shown in Figure 2.3 yields the relationship between joint angles $\theta_1 - \theta_4$ and the x component of the coupler force, P_x:

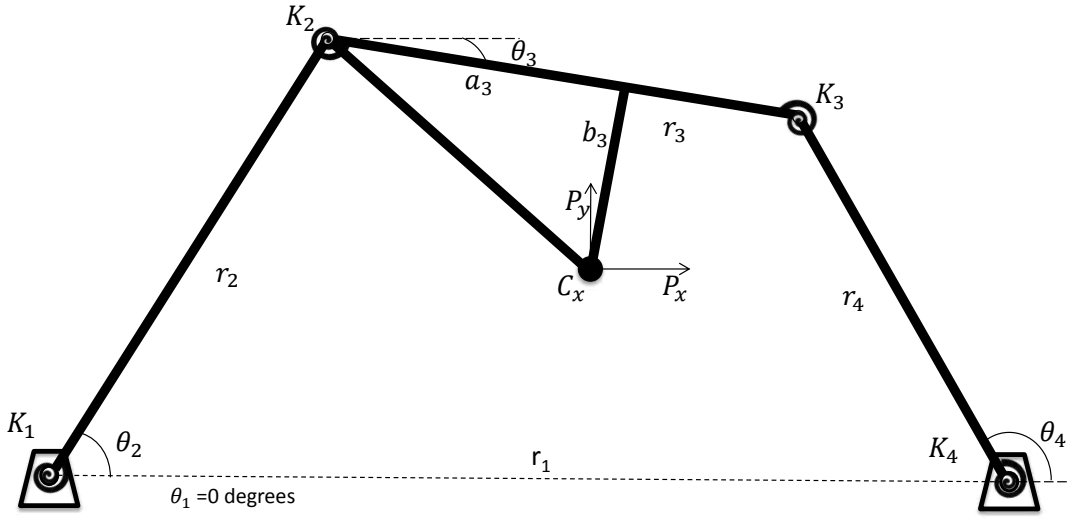


Figure 2.3: PRBM Diagram. $r_1 - r_4$ represent the lengths of the links as if they were rigid. $K_1 - K_4$ represent the stiffness of the compliant beams, and we can imagine them as springs at the pin joints. $\theta_1 - \theta_4$ represent the angle of the respective links. C_x is the coupler point, and in this paper the motion of the coupler point is key to the functionality of the mechanism. a_3 and b_3 represent the position of the coupler point relative to r_3. Finally, P_x and P_y represent the external force applied at the coupler point

$$P_x = \frac{K[(\theta_2 - \theta_{2_0})(\frac{d\theta_3}{d\theta_2} - 2) + (\theta_3 - \theta_{3_0})(1 - 2\frac{d\theta_3}{d\theta_2} + \frac{d\theta_4}{d\theta_2}) + (\theta_4 - \theta_{4_0})(-2\frac{d\theta_4}{d\theta_2} + \frac{d\theta_3}{d\theta_2})]}{r_2 sin(\theta_2) + a_3 sin(\theta 3)\frac{d\theta_3}{d\theta_2} + b_3 cos(\theta_3)\frac{d\theta_3}{d\theta_2}} \tag{2.1}$$

$$\frac{d\theta_3}{d\theta_2} = \frac{r_2 sin(\theta_4 - \theta_2)}{r_3 sin(\theta_3 - \theta_4)} \tag{2.2}$$

$$\frac{d\theta_4}{d\theta_2} = \frac{r_2 sin(\theta_3 - \theta_2)}{r_4 sin(\theta_3 - \theta_4)} \tag{2.3}$$

$$K = K_1 = K_2 = K_3 = K_4 \tag{2.4}$$

$$K = 2\frac{\gamma^2}{r_2}K_\Theta EI \tag{2.5}$$

P_x is the force applied at the coupler point in the x direction. The interface with the user is set up such that the only external force that can be applied is in the x direction at the coupler point. K is the stiffness of each of the springs at the pin joints. These equations were used in two major steps in the design of the final mechanism. First, to ensure that the mechanism had the correct force feedback. Second, equation 2.1 was rearranged to show the relationship between the PRBM link length r_2 and the desired force, P_x, which was needed to build the table used to control the force-displacement profile.

2.5 Optimizing Robert's Straight-Line Mechanism

2.5.1 Link Lengths

There are several variations of Robert's straight-line mechanism; one of the common versions uses the following link length ratios, as defined in Figure 2.3:

- $r_2 = r_4$
- $r_1 = 2.2r_2$
- $r_3 = 0.9r_2$
- $a_3 = 0.5r_3$
- $b_3 = 1.3r_2$

These ratios were used as the starting values in an optimization routine whose objective was to find the best link lengths of the mechanism to maximize the straightness of the coupler path. The optimization routine manipulated the link lengths to find the best Robert's straight-line mechanism with variable lengths of links 2 and 4 while minimizing the negative effects on coupler path (deviations from a straight-line). During the optimization, there were two important constraints needed to ensure that a Robert's straight-line mechanism was produced, $r_2 = r_4$ and $r_3 < r_1$. These constraints helped to maintain the inherent trapezoidal shape of the Robert's straight-line mechanism, which was desired because of the discussion in Section 2.4.1.

The following two objective function were used to determine the best link ratios. In the objective functions N represented the total change in length of the compliant links in 1 mm increments. In the final design N was chosen to be 50, meaning the compliant links could change length up to 50 mm during operation of the haptic interface. Along with the two objective functions, constraints of 250 mm were placed on the link lengths to prevent the link lengths from becoming too large.

- First, the sum of the square of the max change in the y direction of the coupler point at each change in the link length was minimized, $(\sum_{i=0}^{N}(max\Delta y_i)^2)$.
- Second the sum of the square of the max slope as the coupler point moves along the straight-line at each change in the link length was minimized, $(\sum_{i=0}^{N}(\frac{max\Delta y_i}{max\Delta x_i})^2)$.

2.5.2 Force Output

Once the best link lengths were found to achieve straight-line motion in the presence of changing link lengths, two competing objectives were used to find the best thickness and width needed to obtain the correct output force at the coupler point:

- Reach the highest forces possible, at least 90 N (to reach this objective the optimizer would drive the thickness and width to the maximum constraints)
- Maximize the range of force values that can be felt by the user at the coupler point (to reach this objective the optimizer would drive the thickness and width to the minimum constraints)

Along with the virtual work equation (2.1) the following stress analysis equations were modified from [1] and used in the optimization of the dimensions, where σ_{max} is the maximum stress in the compliant links (2 and 4), P_x is the applied force at the coupler point, and a and b represent the deflection of the compliant link in the "horizontal" (x) and "vertical" (y) directions.

$$\sigma_{max} = \pm\frac{F_P(a+nb)c}{2I} - \frac{nP_x}{A} \tag{2.6}$$

$$a = l_2\{1-\gamma[1-cos(\theta_{2_0}-\theta_2)]\} \tag{2.7}$$

$$b = \gamma l_2 sin(\theta_{2_0}-\theta_2) \tag{2.8}$$

The optimized PRBM results of the link lengths and force output are reported below:

$$r_2 = r_4 = 249mm \tag{2.9}$$

$$r_1 = 217.3mm \tag{2.10}$$

$$r_3 = 100mm \tag{2.11}$$

$$a_3 = 50mm \tag{2.12}$$

$$b_3 = 193.6mm \tag{2.13}$$

$$w = 25.4mm \tag{2.14}$$

$$t = 11.2mm \tag{2.15}$$

Although the optimization results were promising, there were some limitations in the optimization. Namely, the objective functions were overly simplified, the constraints led to a low safety factor, and the key PRBM parameters γ and K_Θ were not tuned. The objective functions needed to be improved by incorporating the interdependence of the kinematics and the forces optimizations. The constraints led to dimensions that were too small to have a reasonable safety factor and the desired range of force and displacement. As a result, the optimal values were taken and used to aid in an iterative process used to tune γ and K_Θ and to finalize the mechanisms link lengths and dimensions, as described in the next section.

2.6 Final Design

This section discusses the challenges presented in synthesizing a variable-stiffness compliant mechanism and how they were overcome.

The first challenge has to do with the key PRBM parameters, γ and K_Θ, which are dependent on the boundary conditions of the mechanism. When the link lengths of the mechanism are changing, so are the boundary conditions, which means that γ and K_Θ are not constant.

Second, the design of the link lengths (kinematics) and the synthesis of the forces (kinetics) that the mechanism will output are interdependent design parameters.

As stated earlier, these challenges led to an unsatisfactory optimization algorithm. To improve the synthesis of a functional mechanism several steps were taken in an iterative design process to overcome these challenges.

- Finite Element Analysis (FEA) was used to determine the best K_Θ and γ to accurately predict the force-displacement curve of the mechanism.
- The mechanism was designed to minimize the change of the boundary conditions in the two ways listed below.
  - The neutral angles of θ_2, θ_3, and θ_4 were maintained as the link lengths changed. Neutral angle means the angle of the links when there is no force applied at the coupler point.
  - The relationship of $l_2 = l_4$ was maintained as the links changed length.
- The thickness and width of the links were tuned to produce the best balance between achieving the desired range of forces and maintaining an appropriate safety factor
- The link lengths were designed to their actual length before they were converted to the equivalent PRBM link lengths (This was needed because the actual coupler link, l_3 did not change length, but PRBM link length of r_3 changed as r_2 and r_4 changed)

Using as a starting point the link lengths and dimensions found in the optimization, a FEA was used to tune the PRBM parameters of γ and K_Θ. γ has a large impact on the kinematics of PRBM, while K_Θ impacts the predicted forces of a PRBM. To get the best results in the tuning of γ and K_Θ two steps were taken. First γ was found to match the path of the coupler point in the PRBM to the FEA. Second, K_Θ was found to match the predicted force by the PRBM with the FEA. In this process, connections between changes in the mechanism (i.e. link length ratios) and the key parameters γ and K_Θ were looked for, but no significant connections were found.

To find γ, the relationship between the horizontal and vertical displacement of the mechanism was analyzed in both the FEA and in the PRBM. Since the mechanism has changing link lengths this analysis needed to be performed in such a way as to accommodate these changes. It was decided to perform the FEA on the longest and smallest link length of l_2 and l_4. Then γ was

found for both the shortest and longest l_2 and l_4 using an optimization algorithm. To simplify the analysis the γ for the longest l_2 and l_4 was chosen to be used. A plot of the vertical vs. horizontal displacements of the coupler path for both the FEA and the PRBM can be found in Figure 2.4. There is a big difference between the PRBM and the FEA for the shortest length of l_2 and l_4 with the percent relative error being 34%. This error was expected because the change in the link lengths change the boundary conditions of the mechanism. As a result, there needs to be a different γ for the shortest length of l_2 than the γ used for the longest length of l_2. As will be shown the decision to use one γ value for all of the changes in link length did contribute to the experimental error in predicting the force displacement profile, but this impact was not large.

Once γ was found, the relationship between the force and horizontal displacement of the mechanism was analyzed using both FEA and PRBM. Once again, the relationship was found at both the largest and smallest lengths of l_2 and l_4. A second optimization algorithm was then used to find the best K_Θ to match the PRBM with the FEA at the longest link length. The optimization could have been done to minimize the difference between the shortest link length for the PRBM and the FEA, but it was not necessary. Plots for the FEA and the PRBM with the optimal K_Θ can be seen in Figure 2.5. In the end, the percent relative error between the FEA and PRBM in predicting the force-displacement profile was small, 3% for the shortest length and 1% for the longest length (more details can be found in Appendix C). The error in the force-displacement profile of the shortest link lengths was mostly due to using the values of γ and K_Θ found for the longest link lengths, and there was some numerical error involved due to multiple numerical analyses. The error in the force-displacement profile of the longest link lengths was very small and likely due to some numerical error from rounding γ and K_Θ and the approximating nature of optimization algorithms.

As the parameters γ and K_Θ were optimized the predicted range of displacement and force for the chosen mechanism were changed. As a result, new link lengths and dimensions were chosen based off of a visual (see Figure 2.6) showing the mechanism's ability to cover the desired force range. These new link lengths and dimensions would slightly change the boundary conditions, which led to the need to re-tune the parameters γ and K_Θ following the process described above. These steps were taken iteratively until a suitable set of link lengths and dimensions for the mechanism were found that covered the desired range of displacement and force.

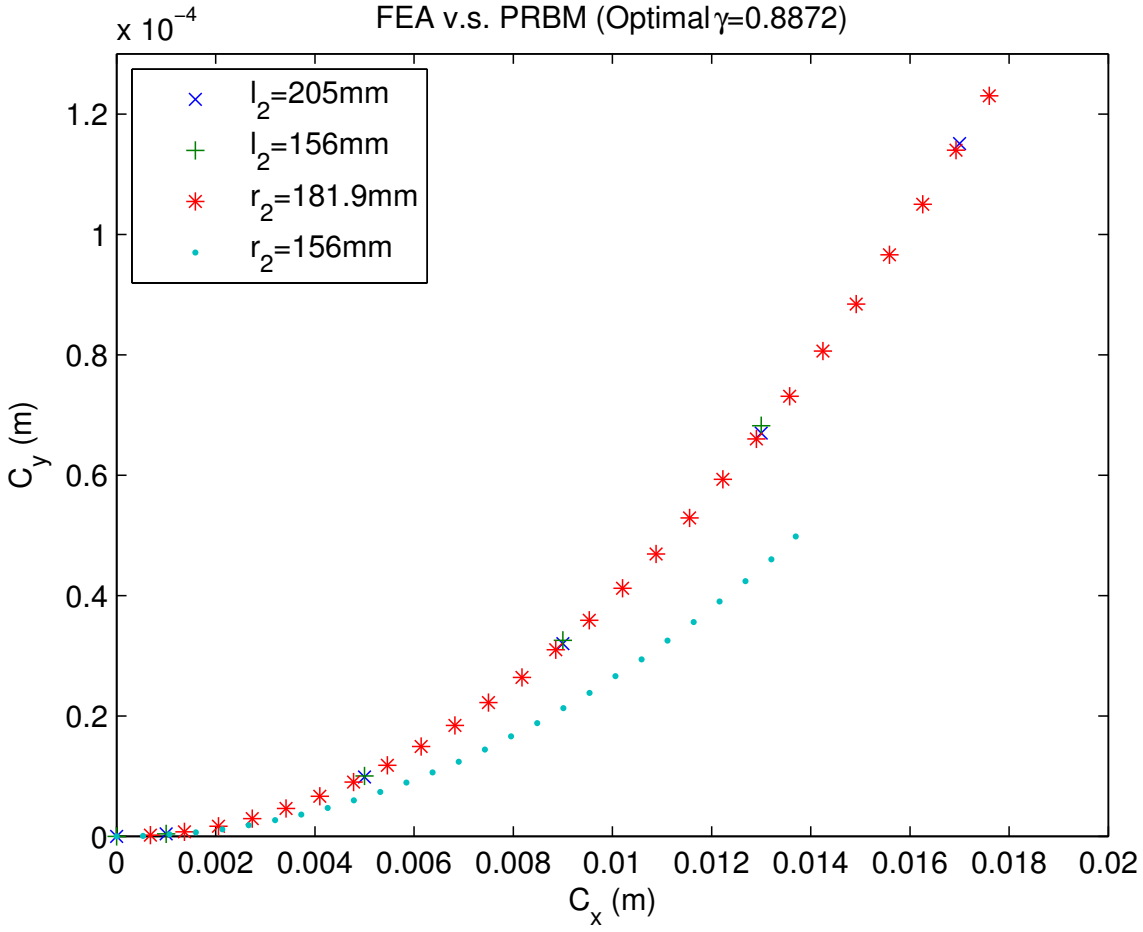


Figure 2.4: FEA vs. PRBM Optimal γ. C_x and C_y represent the horizontal and vertical displacements of the coupler point, respectively. The '*' and '.' represent the coupler path for the PRBM at the longest and shortest link lengths of the mechanism respectively. The 'x' and '+' represent the coupler path for FEA at the longest and shortest link lengths of the mechanism respectively. γ was tuned to minimize the difference between the coupler paths for the FEA and PRBM for the longest link length.

The final model of the compliant mechanism predicted a range of stiffness from 4000 N/m to 7600 N/m, which allowed for the desired range of forces and displacements. This range of stiffness results from the following final dimensions.

$$l_2 = l_4 = 205mm \tag{2.16}$$

$$l_1 = 157mm \tag{2.17}$$

$$l_3 = 54.5mm \tag{2.18}$$

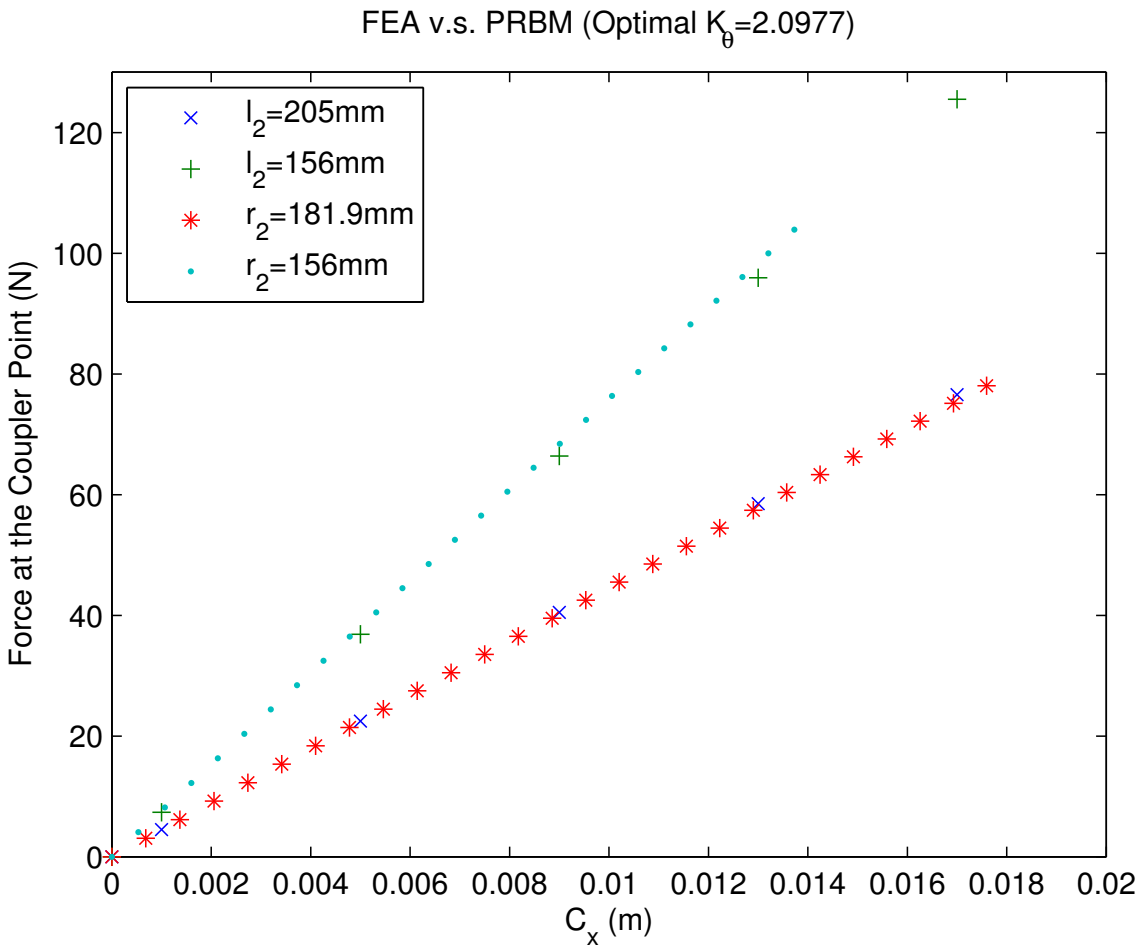


Figure 2.5: FEA vs. PRBM Optimal K_Θ. C_x represents the horizontal displacement of the coupler point. The '*' and '.' dots represent the coupler path for the PRBM at the longest and shortest link lengths of the mechanism, respectively. The 'x' and '+' represent the coupler path for FEA at the longest and shortest link lengths of the mechanism respectively. K_Θ was optimized to minimize the difference between the force displacement curve for the longest link lengths between the PRBM and the FEA

$$A_3 = 27.25mm \tag{2.19}$$

$$B_3 = 155mm \tag{2.20}$$

$$w = 25.4mm \tag{2.21}$$

$$t = 9.6mm \tag{2.22}$$

Finally, a stress analysis of the mechanism was performed to make sure the mechanism would last through testing and use. The stress was analyzed in both the PRBM and FEA. It is

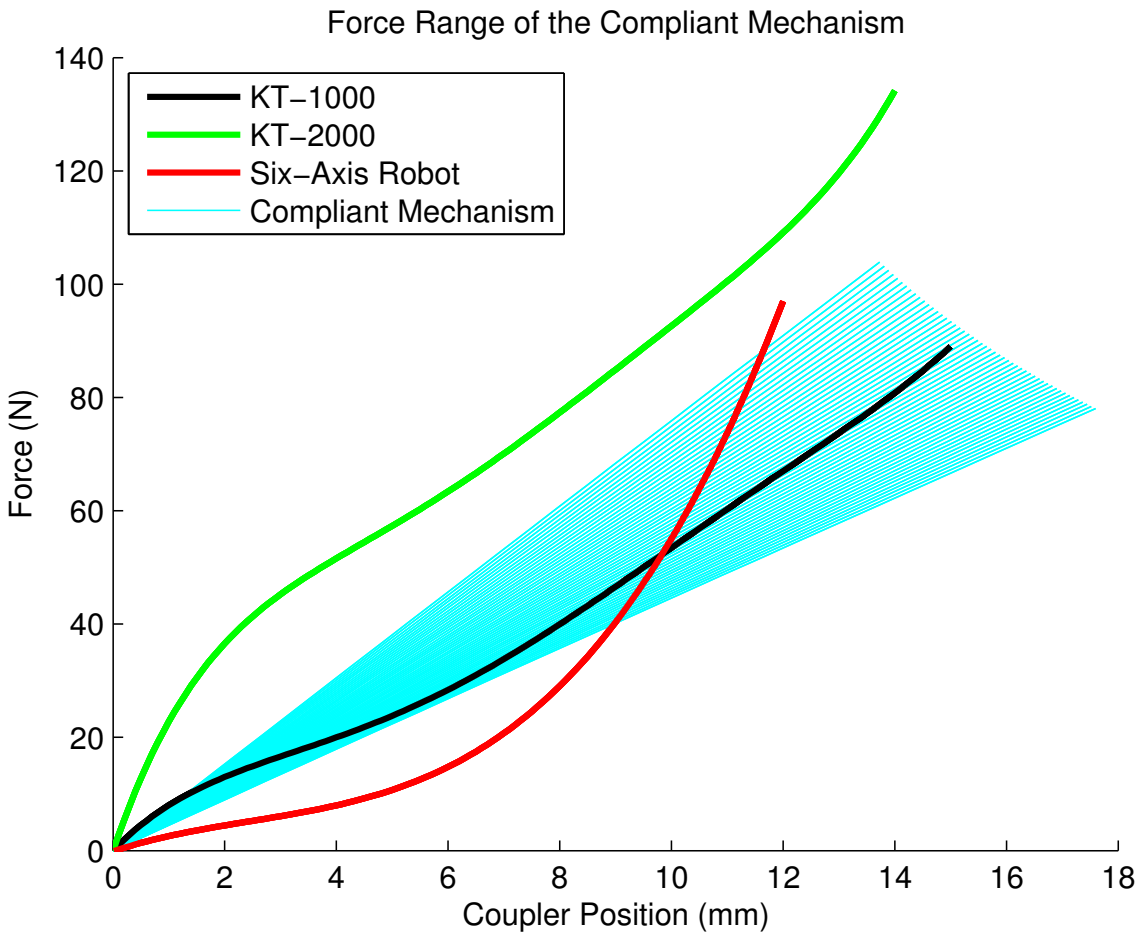


Figure 2.6: Analysis Tool. This is an interface that was used to change the dimensions of the compliant mechanism (length, width, etc) and be able to see the effect on the force output of the mechanism. The red, black and green lines represent potential force profiles of ACL injured knees. The blue lines represent the force-displacement combinations that the compliant mechanism can achieve. In this figure the blue lines surrounding the black line demonstrate that the given dimensions for the compliant mechanism can achieve the force profile given by the black line. See appendix B for the full code.

known that the PRBM is a better estimate of the force than the stress of compliant mechanisms, but it does provide a good ball park estimate. The FEA is known to give more accurate stress results depending on the accuracy of the model built, but in the simplified model used for the FEA stress concentrations (sharp corners) were not removed, while in the final design all of the sharp corners were rounded to reduce the stress concentrations. I added a stress concentration factor of $k_t = 1.285$ in my PRBM analysis, ANSYS used values between 1.24 - 1.33, and a table gave a value of 1.4, see [37]. A more sophisticated model was needed to predict an accurate estimate of

the stress, but between the two simple models a safety factor around 1.5 was determined sufficient to move forward with the mechanism design.

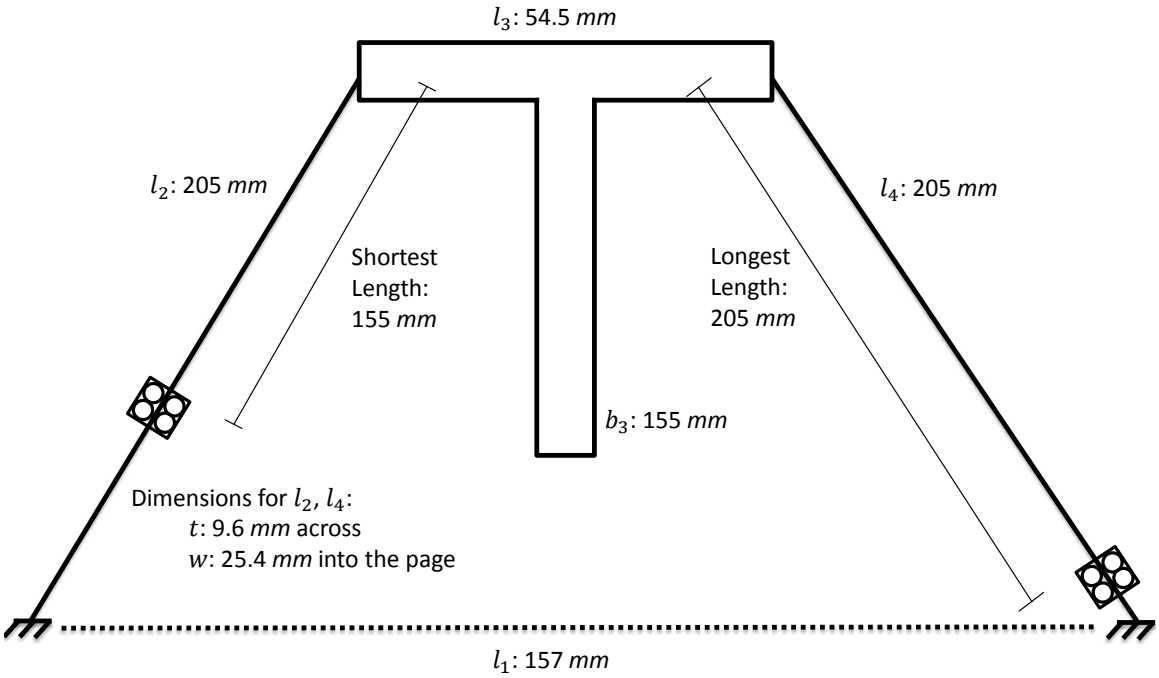


Figure 2.7: Final Compliant Mechanism Dimensions. The dimensions for the final Robert's straight-line mechanism used as the center piece for the compliant haptic interface. The black sliders show the shortest and longest effective lengths for the compliant links used. The dimensions for the compliant links are also listed

Figure 2.7 shows the final dimensions of the compliant mechanism.

2.7 Conclusion

A fully compliant Robert's straight-line mechanism was chosen to be the center piece of the compliant haptic interface. The force feedback of the device is obtained through changing the length of links 2 and 4 on the straight-line mechanism via a method described in the next chapter. The design was analyzed using the Pseudo-Rigid Body Model with the key parameters being tuned

with a Finite Element Analysis and optimized in Matlab for the specific dimensions chosen for the final mechanism.

Future work is needed to develop an algorithm to determine γ and K_Θ as links 2 and 4 change length.

Future work will also need to look more closely at the key factors used to make important decisions in the design process. In this research, decisions were made early on to simplify the control and maximize the linearity of the changing length mechanism. These choices were made to simplify the incorporation of compliant mechanisms into haptic devices. In this research it might have been better to maximize the mechanical advantage because the application required a high force output.

CHAPTER 3. MECHANICAL DESIGN, INSTRUMENTATION, AND CONTROL

3.1 Introduction

Controlling the compliant haptic interface was approached with simplicity in mind. The bulk of the research was to develop the novel idea of incorporating compliant mechanisms into haptic interfaces, and as such, the design of the controls and the base of the mechanism was made as simple as possible. In general the return-to-zero behavior inherent in compliant mechanisms presents some unique challenges in the controls. Instead of allowing this to be a problem, the return-to-zero behavior was used as the haptic force feedback. The kinematic and dimensional design to achieve a proper force feedback profile was discussed in Chapter 2, and this chapter will focus on controlling the change in length of the compliant links to achieve the force-displacement curves discussed in Chapter 2.

3.2 Mechanical and Hardware Design

3.2.1 Mechanical Motion

The key to controlling the force feedback for the compliant straight-line mechanism is controlling the motion along the compliant links. The mechanical motion of the haptic interface (see Figure 3.1) was designed around the need to change the lengths of links 2 and 4 together with a reasonable amount of precision. The range of forces that the mechanism is able to reach increases with the precision of the change in the link lengths. On the other hand, as the precision of the change increases, so does the time it takes to change the length of the link. As such, a balance between speed and precision was used to make the best choice for this application.

Lead screws driven by synchronized stepper motors are the main mechanism for changing the length of the compliant links, as shown in Figure 3.2. Lead screws allow for the linear systems for each link length (2 and 4) to be maintained independently, which simplified the manufacturing.

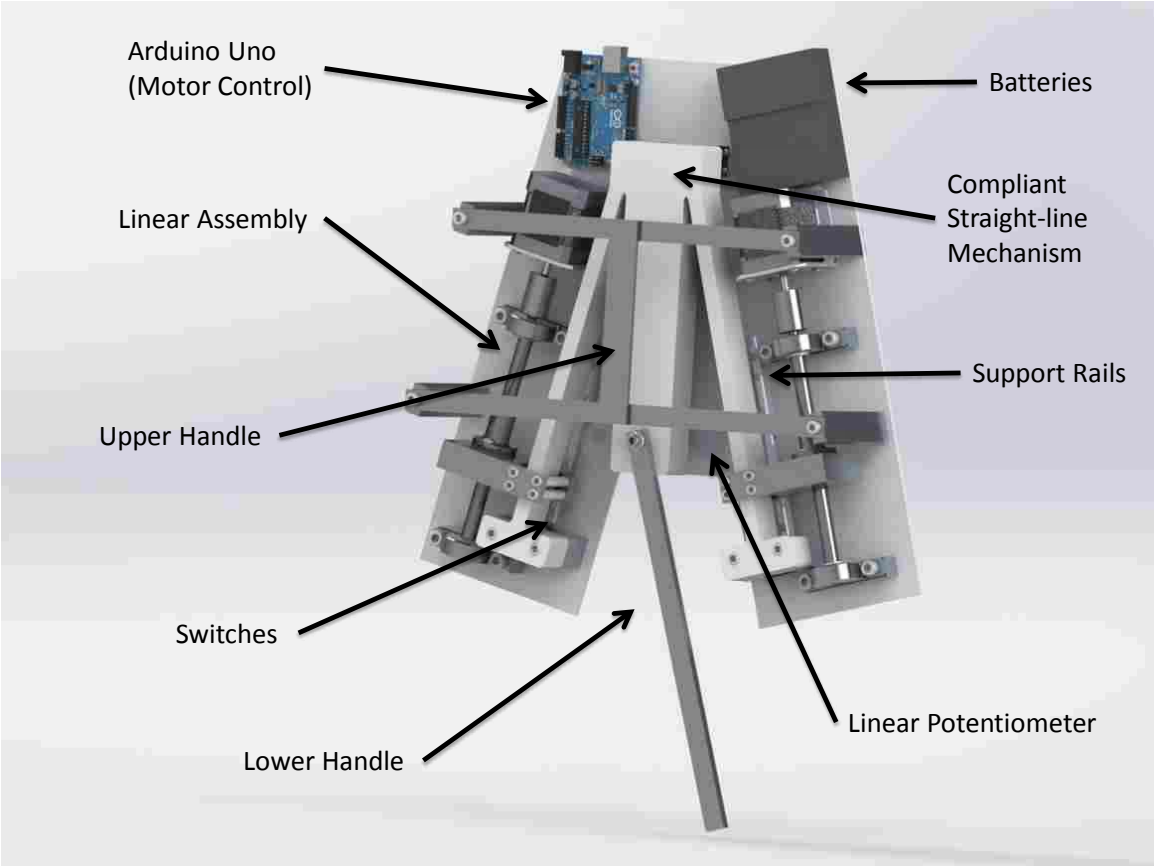


Figure 3.1: Final CAD Design. The final design of the compliant haptic interface

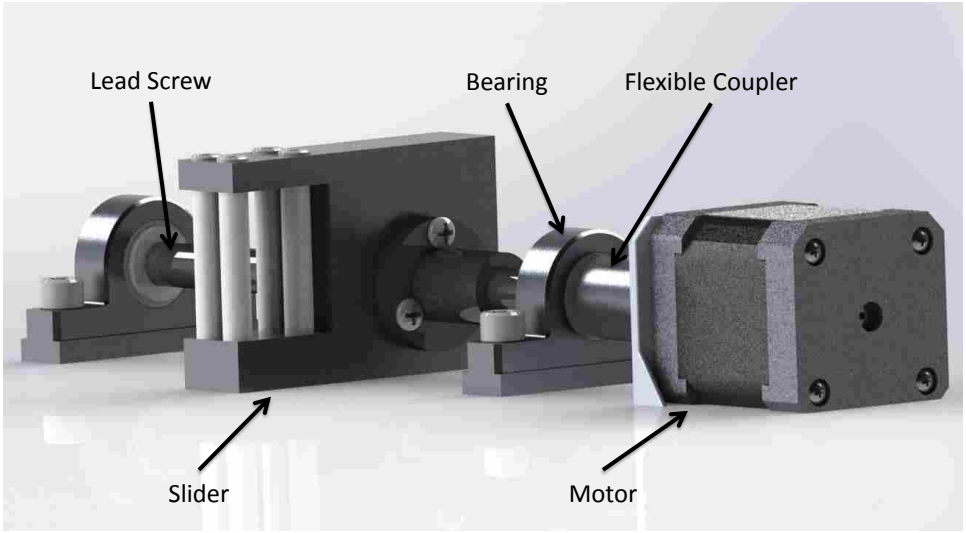


Figure 3.2: Linear Assembly. This is the linear system used to change the effective length of the compliant link. There are two identical linear systems in the haptic device, one for each compliant link. Not featured in this image is the guide rail that runs under the front end of the slider.

The lead screws are $\frac{3}{8}$" thick to be rigid and strong enough to withstand the force applied on them from the compliant mechanism. A lead of $\frac{1}{2}$ inch allows for the lead nut to move along the lead screw quickly without losing too much precision. The final speed and precision of changing the link length depends on the motors, which will be discussed later in this chapter.

As seen in Figure 3.2, several other parts are needed to complete the linear system. The steel lead screw is supported by two aluminum bearings with bronze bushings. The lead screw is connected to the motor with a flexible coupler to handle the potential axial and angular misalignment between the motor and the lead screw. As the motor turns the lead screw a plastic nut travels along the screw. The nut is attached to the slider, which moves along the compliant link to change its effective length. Finally, to aid the travel of the slider, a support rail was placed just below the compliant link (shown in Figure 3.1).

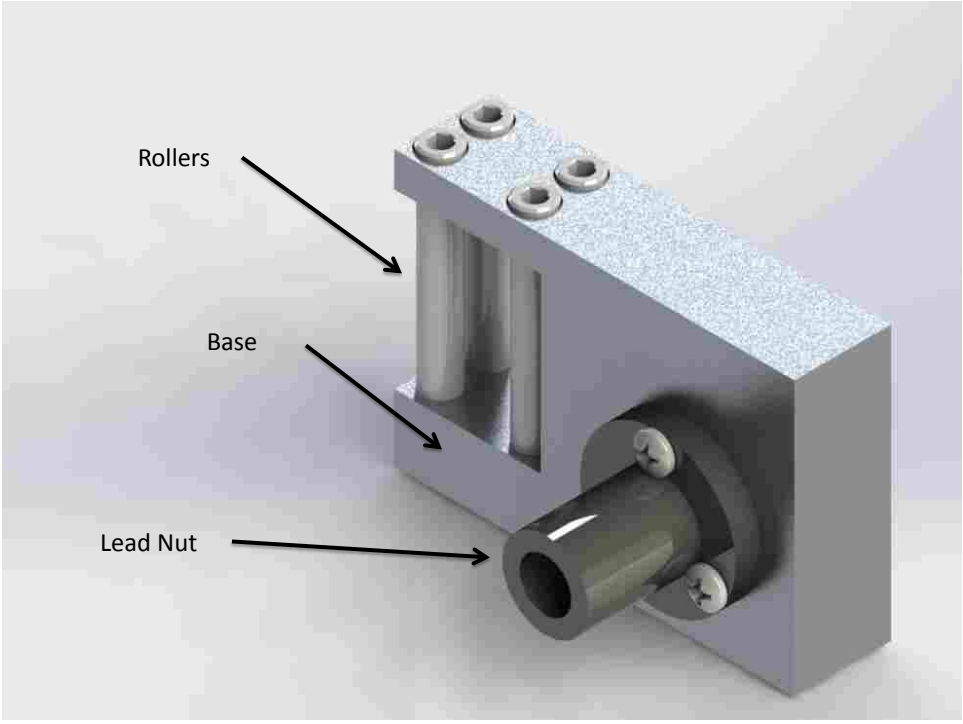


Figure 3.3: Slider. As the nut moves along the lead screw the rollers move along the compliant link. The rollers are made out of nylon to minimize the fiction between the compliant link, and as a result the rollers slide as often as they roll.

A slider was designed to connect the nut of the lead screw to the compliant mechanism (see Figure 3.3). The slider is made of an aluminum base, strong and lightweight, with plastic wheels.

The four plastic wheels roll along the compliant link providing a changeable fixed point for the links, meaning that as the slider moves along the compliant link the length of the link is effectively shortened or lengthened (see Figure 2.7).

Two stepper motors were chosen to drive the linear system because of the simplicity of using them in open-loop control, which can significantly decrease the complexity of the control system. To ensure that open-loop control was possible the stepper motors needed to have the capability to produce twice as much torque as required. The compliant mechanism was analyzed using the Newton Euler method, and it was determined that the compliant mechanism would apply a force of 42 N along the lead screw, which is equivalent to a torque of about 17.4 oz-in applied on the motor when using a $\frac{3}{8}$" lead screw with a $\frac{1}{2}$" lead. The other key to selecting motors was to ensure that they could move fast enough, and the goal was to move the full range of motion in 1 sec. The full range of change of the link length was 50 mm or about 2 in, and the lead of the screw was $\frac{1}{2}$". So the motor needed to be able to rotate 4 revolutions per second.

The stepper motor was purchased from Pololu, and had the following specifications.

- Physical size NEMA 17
- 1.8 deg/step (200 steps/rev)
- 44 oz-in (0.314 Nm)
- 1.2 A per phase (2.4 A per motor)
- Driven at 8 V
- Capable of both bipolar and unipolar (bipolar was used)
- 5 mm shaft

Two handles were included to enable the user to interface with the haptic device, as shown in Figure 3.1. The upper handle is made to be grounded, held still, by the user, while the lower handle is designed to be manipulated by the user. In application the user would hold the upper handle steady in one hand, and then move the lower handle in a linear motion with the opposition hand. Also the lower handle is able to rotate freely, so the lower handle will only produce a force feedback from the compliant mechanism when it is displaced via a linear motion.

A flat plastic base was designed to combine the mechanical parts, compliant mechanism, linear system, and control system together into a cohesive haptic interface. The full drawing package of the design can be found in appendix D. There are several other parts that make up the mechanism, and these will be discuss in the subsequent sections where the hardware and control of the mechanism are discussed.

3.2.2 Hardware Instrumentation

The nature of stepper motors led to the need for two sensors. Since the stepper motors were set up in an open-loop control system they lose track of their position when the system is power cycled. Thus, there is a need to calibrate or zero out the position of the stepper motors when the power is turned on. Two normally open limit switches are used to zero out the motors. When power is provided to the system the motors run in one direction until they contact the limit switches. Once the switches are closed the stepper motors have a ”home base” from which to count the steps they take, and thus can keep track of their respective location. The switches also play a secondary role in the control system. If something happens to cause the motors to lose count (e.g. too much torque causes the motors to slip) the switches protect the nut and slider from running into the rest of the mechanism, and trigger a restart to the program.

A five volt signal is supplied to the switches, and the output is filtered at 16 hertz because that is slow enough to filter out most of the switch bouncing, but still fast enough to stop the motor before it takes the next step. Often a switch debouncing circuit is also used to ensure that multiple signals are not obtained all at once. This problem was addressed in the software because there was a limited amount of circuit space. The circuit drawing can be seen in Figure 3.4.

Since the force-displacement curve is based on the coupler’s position, a softpot linear potentiometer (see Figure 3.5) is used to measure the coupler’s position. The measurement is coupled with a look-up table to transform the coupler’s position to a desired link length along the compliant member. The softpot linear potentiometer is used because of its small size and relatively high accuracy. The softpot outputs a different voltage depending on where contact is made with its surface. Thus, there is a different voltage output along the whole coupler path. The linear potentiometer is essentially a voltage divider where the contact point on the sensor determines the ratio of the two resistors. The output of the potentiometer is low pass filtered to attenuate any noise above 16 hertz

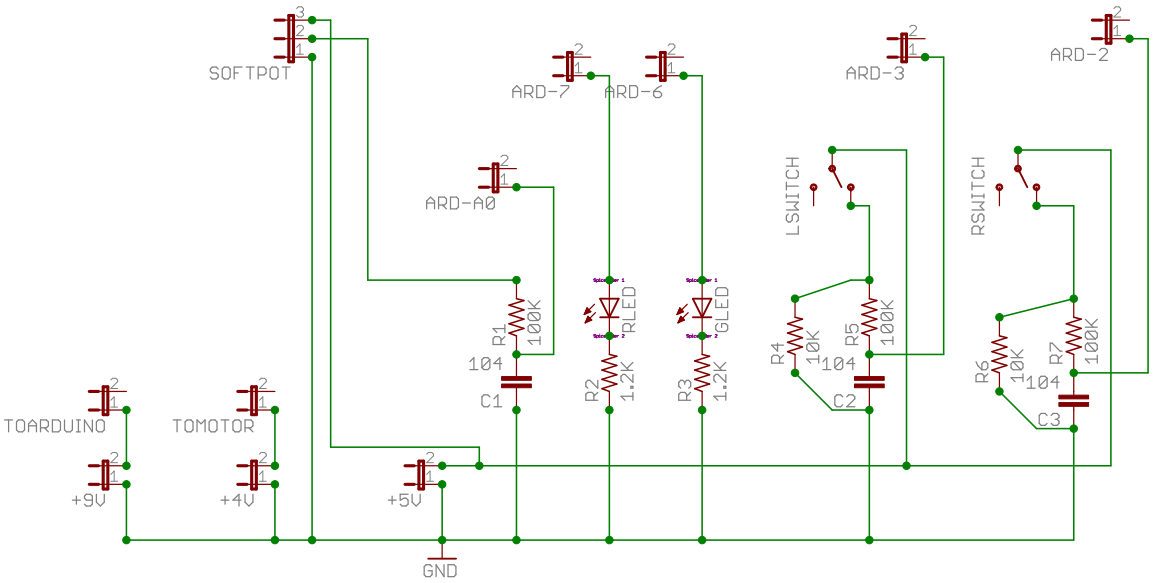


Figure 3.4: Circuit Diagram. This is the circuit demonstrated as it was laid out in Eagle (a circuit designing program). The two LED lights (RLED indicates power to the system and GLED indicates the system is ready to use). The linear potentiometer is used to measure the position of the coupler point. The two switches are used to zero out the position of the motors.

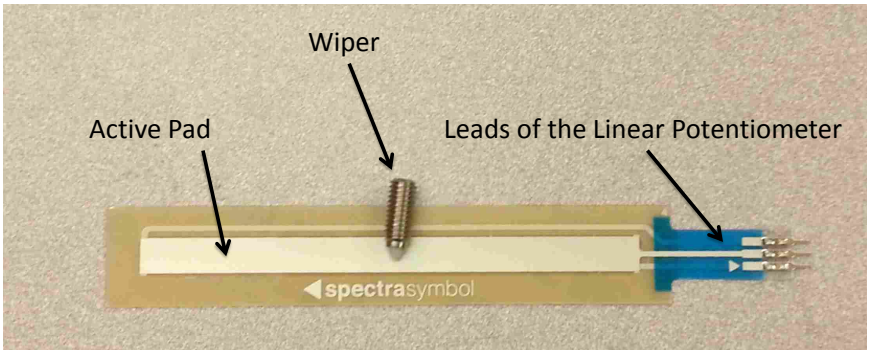


Figure 3.5: Linear Potentiometer. The linear potentiometer is a resistor whose circuit is complete when the top layer is pushed into the bottom layer. This is done by a wiper that moves along its surface, and as it does so it changes the ratio of the resistors in the voltage divider. So there is a unique voltage output along the whole length of the potentiometer (50 mm)

which is low enough to remove typical instrumentation noise but high enough to allow for most human motion. The signal is converted from an analog to a digital signal by a 10 bit A-D converter internal to the Arduino Uno [38] (see Figure 3.4).

The specifications for the linear potentiometer are listed below.

- 10 k Ohms
- $\pm$20% Resistance Tolerance
- Effective travel 50 mm
- Linearity $\pm$1%
- max power 1 watt

The last two pieces of hardware are the Arduino Uno [38] along with an Adafruit motor shield. The Arduino Uno is a programmable microcontroller that is used to control the inputs and outputs needed to run the motors and read the sensors. The motor shield provides the necessary circuitry to drive the stepper motors. Below the listed motor shield specifications show that it is capable of driving the chosen stepper motors. The motors were driven with 8 volts, and the current was allowed to peak at 4.8 amps.

- 4.5 to 13.5 Volts per motor connection
- 1.2 Amps per channel with a 3 A peak current
- Made to connect to an Arduino Uno
- Can drive 2 stepper motors, or 4 DC motors, or 2 servo motors

3.3 Software Design

3.3.1 Mathematical Algorithm

Data from knee studies was used to build the desired force-displacement curve, and was processed with a 5^{th} order polynomial curve fit. The curve fit was verified graphically to ensure it

matched the original data. Figure 3.6 shows the 3 curves that were produced. The KT-1000 knee data was chosen to be reproduced by the haptic interface because the forces were relatively low and the force profile has a recognizable shape.

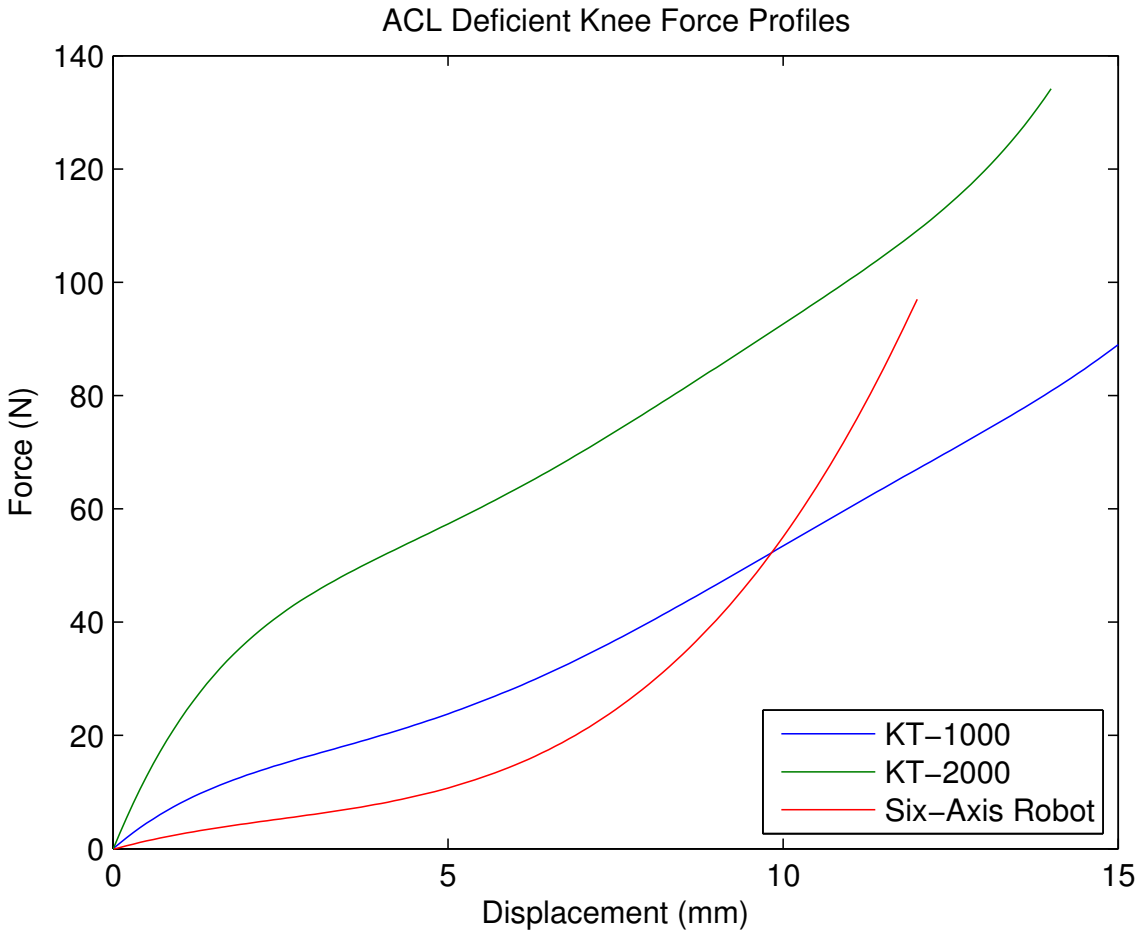


Figure 3.6: ACL Force-Deflection Profile. Force-deflection profile of an ACL injured knee as measured by three different studies KT-1000 arthrometer [2] pg.427-445, KT-2000, and Frey-Riener [4]

The force-displacement profile was used to build a look-up table to control the length of the compliant links. The look-up table connects the location of the coupler point with the requisite link lengths to produce the desired force at the current coupler point. There is a direct connection between the desired force and the location of the coupler point. There is also a direct relationship between the force produced and the length of the compliant links. An algorithm was built to make the connection between these two relationships to produce the desired one to one mapping of

the coupler points current location to the necessary length of the compliant link. The subsequent paragraphs will describe how the look-up table was formed.

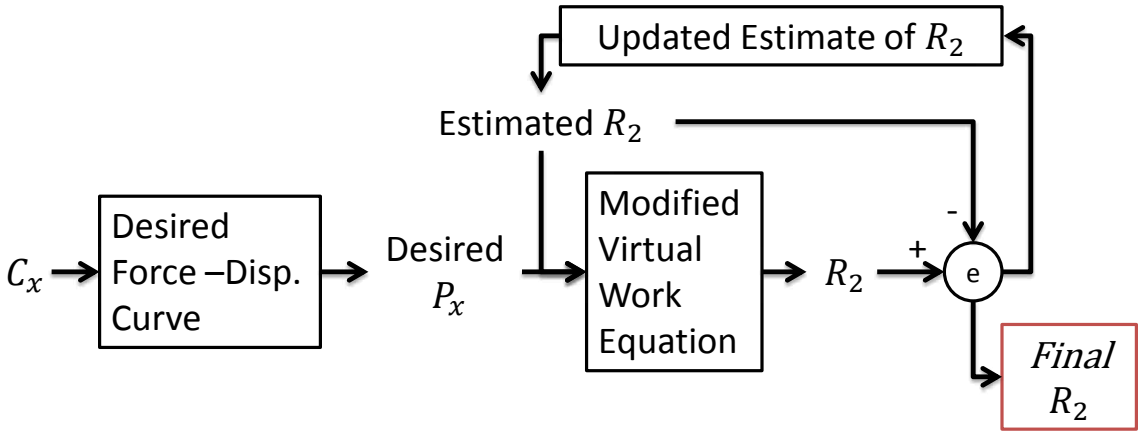


Figure 3.7: R_2 Diagram. This diagram describes the process used to find the relationship between the coupler point and the PRBM link length to produce the desired force-displacement profile. R_2 is the PRBM link length, C_x is the horizontal coupler position, P_x is the desired horizontal force at the coupler point

The algorithm (see Figure 3.7) starts with the location of the coupler point, which is transformed into a desired force. The desired force and an estimate of the PRBM link length, r_2, are then transformed into a new link length, r_2. The new r_2 value is then compared to the estimated r_2, and the process is iterated until the error between the estimated r_2 and the new r_2 is less than 0.1%.

The transformation from the coupler point to the desired force uses the equation of the line for the desired force-displacement curve. The coupler point is put into the equation, and the desired force is the output of the solved equation.

The transformation from the estimated r_2 value and the desired force takes advantage of the virtual work equations developed in Chapter 2, see Equations 2.1 - 2.5. The virtual work equation was rearranged to have r_2 as the output of the equation with the link lengths, angles and desired force as the inputs. The need for the link lengths and angles as inputs to the equation forced the process to be iterative, as shown in Figure 3.7. The final step in building the look-up table was a simple transformation from r_2 to l_2 following the equation, $l_2 = r_2/\gamma$

The iterative nature of the algorithm takes too long to solve in real time, thus a look-up table was built using the algorithm described above that could then be used in the control program. Below, the key equations for the modified virtual work equations are shown.

$$ar_2^2 + br_2 + c = 0 \tag{3.1}$$

$$a = P_x sin\theta_2 - \frac{P_x a_3 sin\theta_3 sin(\theta_4 - \theta_2)}{r_3 sin(\theta_3 - \theta_4)} - \frac{P_x b_3 cos\theta_3 sin(\theta_4 - \theta_2)}{r_3 sin(\theta_3 - \theta_4)} \tag{3.2}$$

$$b = \frac{K^*(\theta_2 - \theta_{2_0})sin(\theta_4 - \theta_2)}{r_3 sin(\theta_3 - \theta_4)} + \frac{K^*(\theta_4 - \theta_{4_0})sin(\theta_4 - \theta_2)}{r_3 sin(\theta_3 - \theta_4)} - \frac{2K^*(\theta_3 - \theta_{3_0})sin(\theta_4 - \theta_2)}{r_3 sin(\theta_3 - \theta_4)} \tag{3.3}$$

$$c = -K^*(2\theta_2 - 2\theta_{2_0} - \theta_3 + \theta_{3_0}) - \frac{2K^*(\theta_4 - \theta_{4_0})sin(\theta_3 - \theta_2)}{sin(\theta_3 - \theta_4)} + \frac{K^*(\theta_3 - \theta_{3_0})sin(\theta_3 - \theta_2)}{sin(\theta_3 - \theta_4)} \tag{3.4}$$

3.3.2 Open-Loop Control

The control program runs through simple, reliable steps to ensure that the haptic device is outputting the correct force at every position of the coupler point. In the code there are two distinct sections, the setup and the main loop (see Appendix E).

In the setup the haptic device is prepared for interaction with people. First, all of the variables such as the coupler position are initialized. This is especially important because the coupler position is measured from a linear potentiometer. The program takes the current value and sets it as the neutral or zero position for the coupler point. That way any movement from the neutral position is measured correctly regardless if the compliant mechanism or potentiometer shifted between uses. Next the motors run through their zeroing out processes using the switches. Finally, the switches are set up as interrupts to prevent a collision between the lead nut and the rest of the haptic device.

In the main loop the current coupler position is found. To ensure an accurate reading of the linear potentiometer the value is averaged over 10 readings. Then with the help of the look-up table the desired compliant link length is found. Finally, the motors are engaged until the compliant link lengths are the right length. To control the motors simultaneously AccelStepper, an Arduino library, was used to command the stepper motors by sending one step at a time alternating between the left and the right stepper motor. Without this library the first stepper motor would have moved

to the desired location before the second motor, which would have cause the four-bar to not be a straight-line mechanism during the motion of the motors. No sensors are employed to measure the length of the compliant link length since the motors are counting their steps.

3.4 Conclusion

A simple open-loop control is used to move a lead nut and slider along a lead screw to change the effective length of the compliant link. This change in the compliant link enables a non-linear force-displacement profile to be produced to match data recorded for an injured ACL knee by KT-1000. Stepper motors enable the open-loop control. Limit switches allow the stepper motors to find the correct zero position. A linear potentiometer measures the coupler's location from which the desired length of the compliant mechanism is calculated. Finally, the stepper motors steps are counted to control their positioning along the compliant links.

CHAPTER 4. TESTING

4.1 Introduction

This chapter will discuss the testing of the compliant mechanism and haptic interface discussed in the previous chapters. It will also show the ability of the haptic interface to follow prescribed force-displacement profiles.

Two tests were performed to validate the methods developed in this thesis. The first test was designed to validate the model of the compliant mechanism developed in Chapter 2. In this test force-deflection profiles were measured while holding the compliant links at a constant length. The test results for the first test are presented in Section 4.3. The second test demonstrates the ability of the haptic interface to generate controlled force-deflection profiles, in which the link lengths are actively varied. The test results for the second test are presented in Section 4.4

4.2 Experimental Setup

Both tests used an Instron load frame to measure force-deflection profiles. Figure 4.1 shows the image of the haptic interface ready to be tested in the Instron. The haptic interface and/or the compliant mechanism is connected to an L shaped bar that is clamped in to the lower arm of the Instron. The coupler bar is connected to a 22 pound load cell that measures the force output at the coupler point. The load cell is attached to the upper arm of the Instron.

Two very similar methods were followed for each of the tests mentioned above. The process followed to validate the compliant mechanism is listed below. The process used to validate the haptic interface will then be described.

Compliant mechanism validation with constant link lengths:

1. The sliders were positioned at the desired link length by the motors.

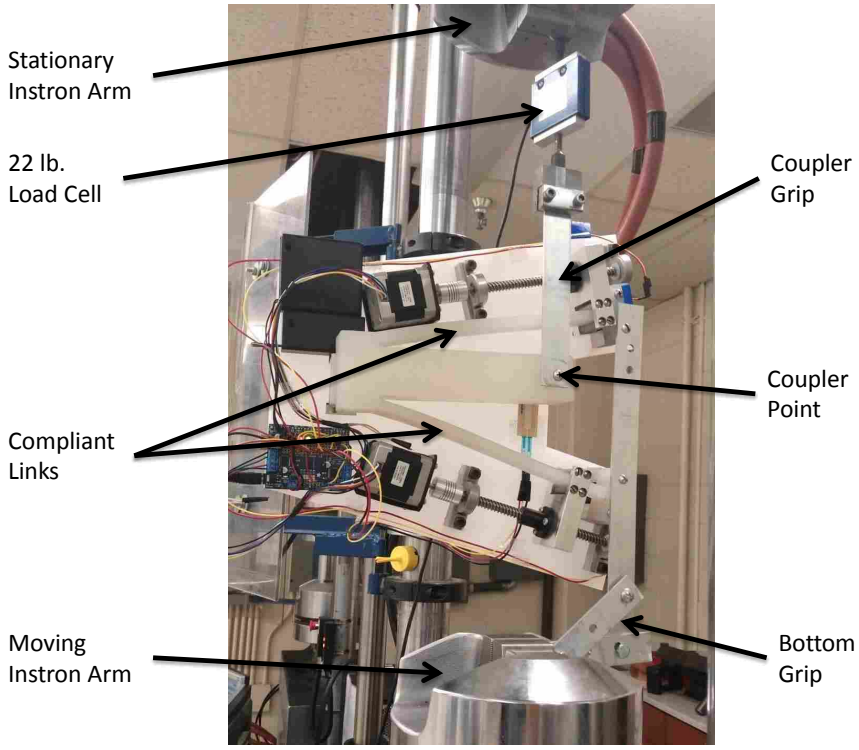


Figure 4.1: Instron Test Setup. The moving Instron arm moved down as programmed to translate the coupler point. The Instron recorded the distance traveled while the 22 lb. load cell recorded the associated force applied by the compliant links on the coupler grip.

2. The compliant link length was measured to check the accuracy of the motors. The measurements were performed frequently enough to ensure that the motors were performing properly.

3. The Instron program was run.

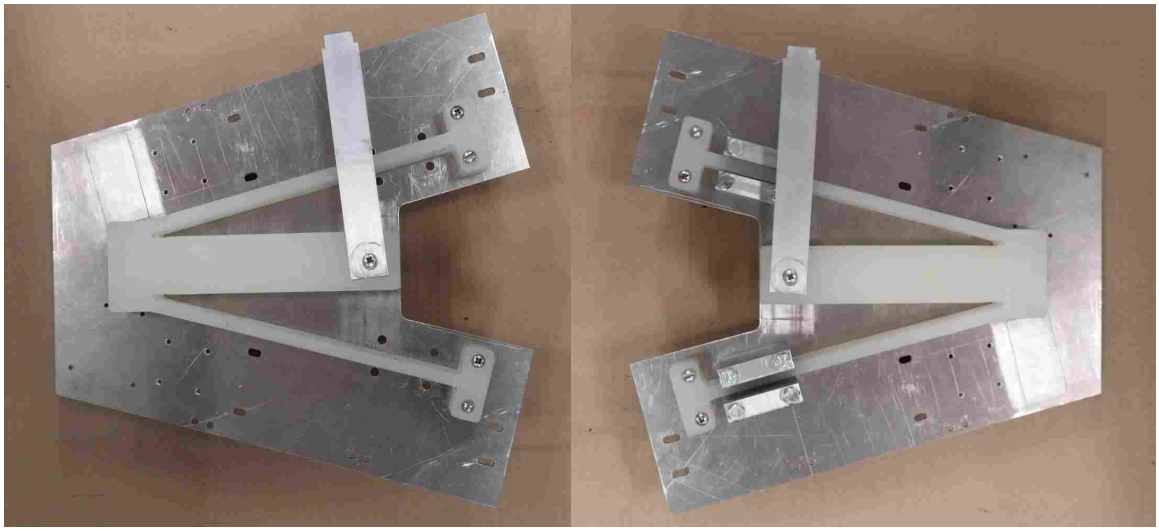
    - The lower arm of the Instron moved down, as viewed in Figure 4.1, at a rate of 0.5 mm/s.
    - The Instron moved a total of 15 mm
    - The Instron recorded the force-displacement profile

4. The process was repeated for 2 different link length settings.

In the haptic interface validation tests step 1 from above consisted of zeroing out the stepper motors, and allowing them to reach their first position before the Instron program was started. Once

the Instron program was started the motors moved the sliders according to the coupler's position to produce the desired force-displacement profile.

4.3 Compliant Mechanism Validation



a. b.

Figure 4.2: Compliant Mechanism Test. This figure shows the compliant mechanism isolated from the haptic interface prepared for testing. a. is the compliant mechanism prepared to be tested at the longest link length possible (216 mm). b. is the compliant mechanism prepared to be tested at the shortest link length used in the haptic interface (155 mm)

In the haptic interface there are many moving parts and sources of error. To verify the design and model of the compliant mechanism described in Chapter 2, an Instron test was performed on the compliant mechanism without the surrounding haptic interface. The compliant mechanism was tested by clamping the compliant links at specified lengths, shown in Figure 4.2, then put through the Instron test.

A force-displacement profile was created for two different lengths of the compliant links. As shown in Figure 4.3, the measured profiles match the predicted values within 2 N. The difference between the predicted and measured curves can be due to a number of sources of error, i.e. manufacturing inaccuracy, slight differences in the material properties, and numerical error in the PRBM of the compliant mechanism.

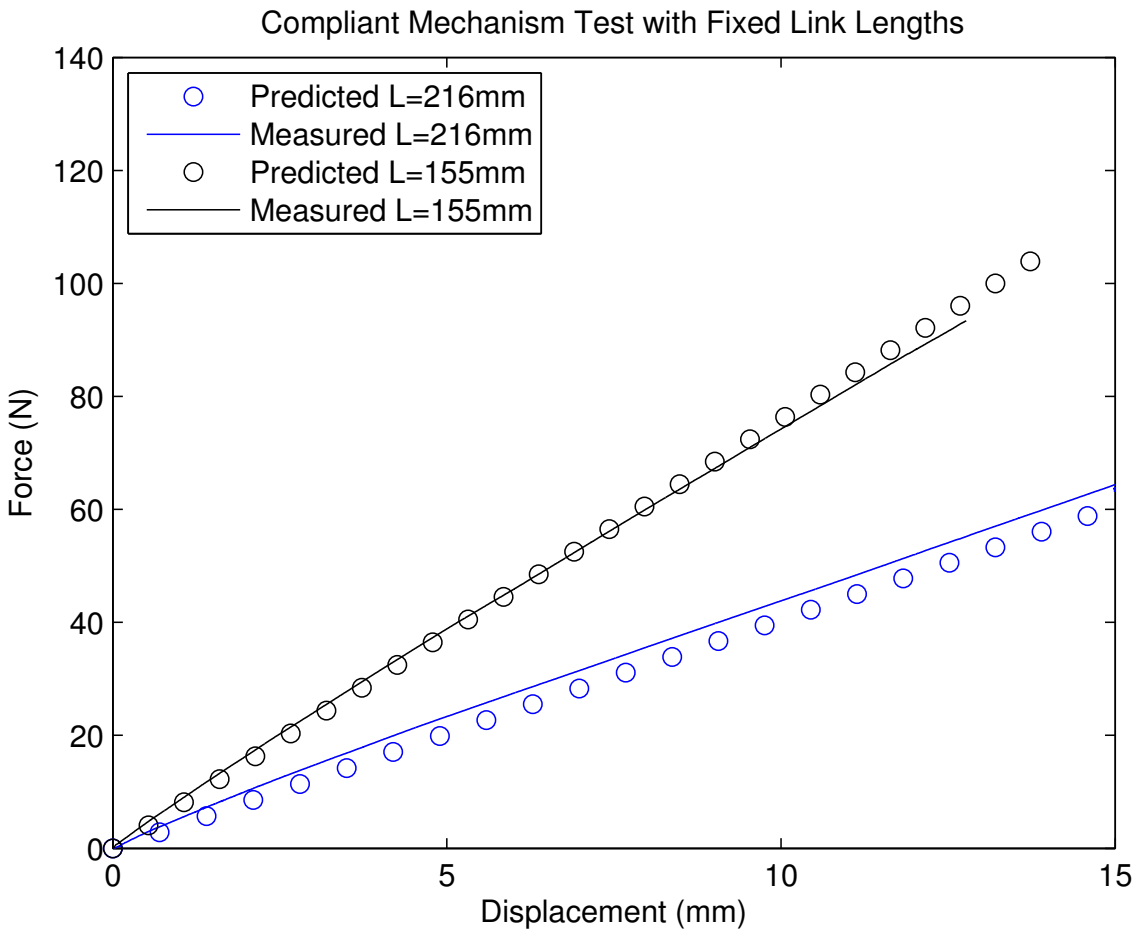


Figure 4.3: Compliant Mechanism Test. In this test the compliant mechanism was tested outside the haptic interface to verify the model developed in Chapter 2. The circles represent the predicted force profiles, while the solid lines represent the measured values on the compliant mechanism by the Instron. The black lines represent a link length of 155 mm, and the blue lines represent a link length of 216 mm

Along with the force profiles matching, the compliant mechanism demonstrated the ability to cover a range of stiffnesses. This is important because the variable stiffness of the compliant mechanism is a key factor in developing various force profiles with the haptic interface. The compliant mechanism was able to have a stiffness as low as 4200 N/m and as high as 7200 N/mm.

4.4 Haptic Interface Validation

After validating the force-deflection profile of the compliant mechanism, the haptic system as a whole was tested. The haptic interface had three steps of testing. First, the haptic interface

was tested with the compliant link lengths held constant by the motors and sliders through the Instron test. Second, the haptic interface was tested to see its capability to follow an oscillating force profile. Finally, the haptic interface was tested to see if it could accurately simulate the force profile produced by a knee with an injured ACL during Lachman's test.

4.4.1 Individual Lengths

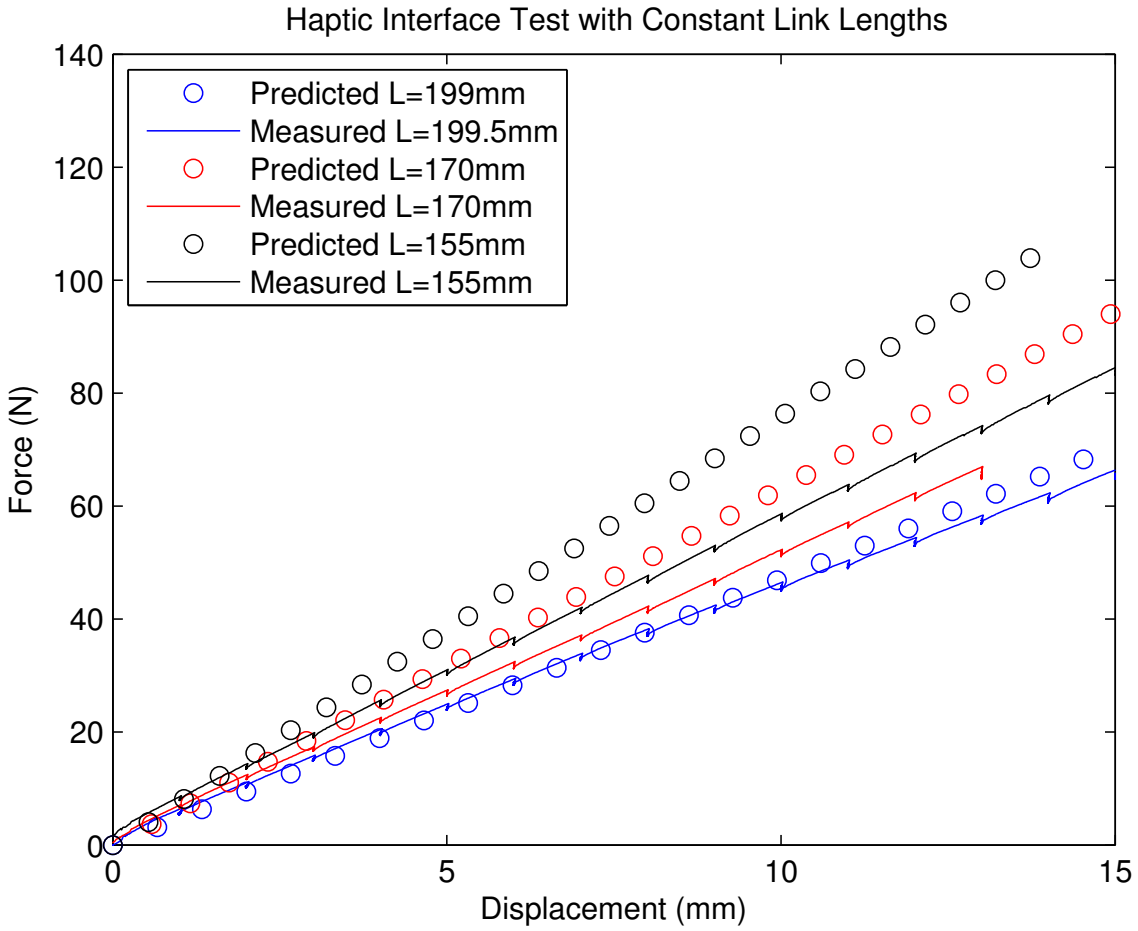


Figure 4.4: Constant Compliant Link Lengths. The circles represent the predicted force profiles at various link lengths. The solid lines represent the measured force profiles of the haptic device as measured by the Instron. The black lines represent a link length of 155 mm, the red lines represent a link length of 170 mm, and the blue lines represent a link length of 199 mm.

The purpose of this test was to compare the force-deflection profile of the motors and sliders holding the links at constant lengths to the PRBM developed in Chapter 2. During the constant

length tests, shown in Figure 4.4, the higher the forces became the greater the error was in the measured force-displacement curve. There were two main sources that caused the error. First, there was some looseness between the slider and compliant link and between the nut and lead screw. Second, the system was modeled as if the compliant links were grounded at the slider. As the slider moved, the ground point was supposed to move along the link. The slider was not tight enough to act as a ground point, and there was some bending in the compliant link below the slider. The slider acted more like something between a fixed point and a simple support, and the looseness in the slider was amplified as the forces increased.

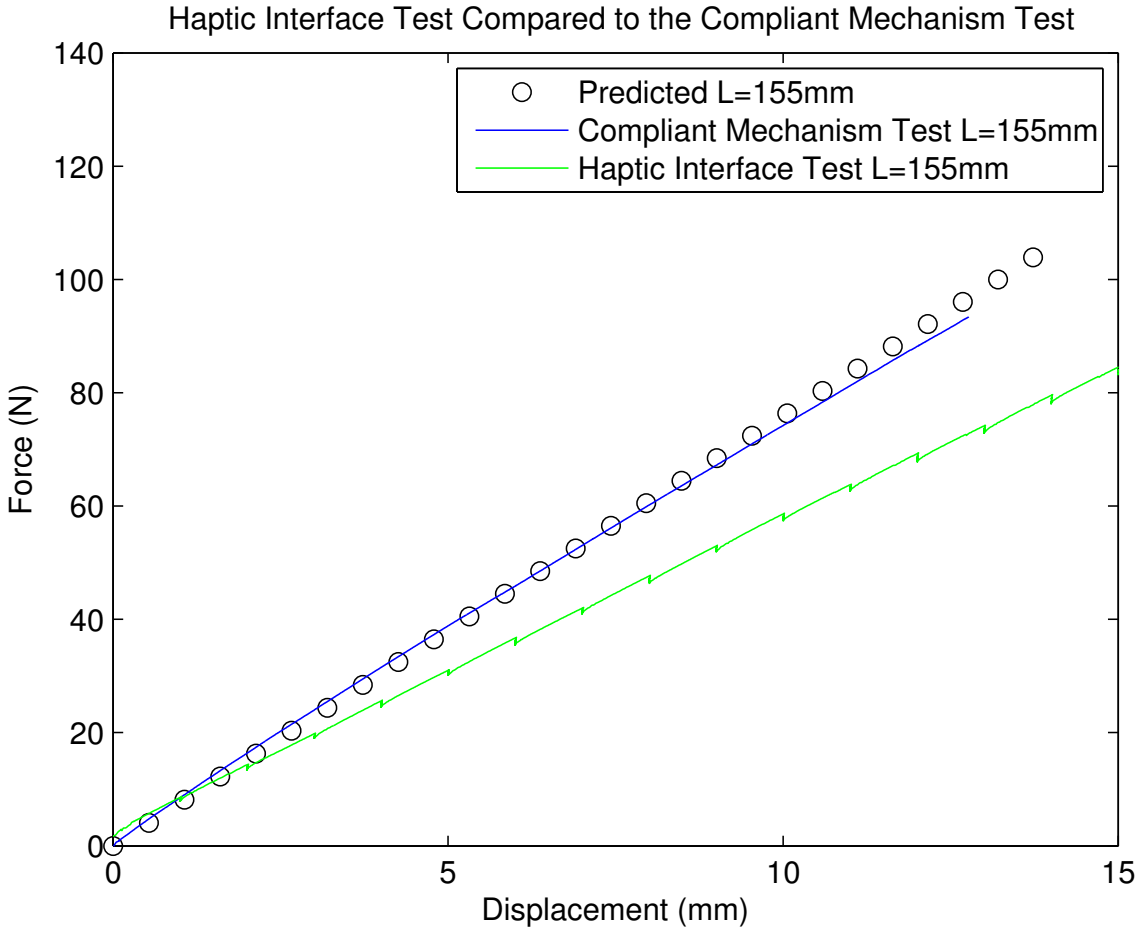


Figure 4.5: Haptic Interface Test Compared to the Compliant Mechanism Test. This graph shows the results of both the Compliant Mechanism Test and the Haptic Interface Test at a link length of 155 mm. The black circles represent the predicted force profile, the blue line represents the results from the Compliant Mechanism Test, and the green line represents the results from the Haptic Interface Test.

As can be seen in Figure 4.5, the PRBM developed in Chapter 2 more accurately predicts the force-displacement profile of the compliant mechanism with its link lengths rigidly fixed than it does the haptic interface with the link lengths held constant by the sliders. The compliant mechanism test results closely resembles the PRBM developed in Chapter 2 because the mechanical fixtures are tightly positioned keeping the lower portion of the compliant mechanism from displacing. In the compliant mechanism test there was an average percent relative error of 5% between the model and the measured values, and the two never deviated more than 4 N. On the other hand the haptic interface test results did not follow the predicted PRBM accurately because of the looseness between the slider and the compliant mechanism, leaving room for the compliant link below the slider to move. In the haptic interface test there was an average percent relative error of 27% between the model and the measured values, and the two deviated as much as 30 N.

As a result of the constant link length tests, it is known that the current design of the haptic interface is not capable of exactly matching the desired force profiles discussed in the following sections. With this in mind, the haptic interface was still tested to verify its ability to follow the same pattern as the desired force profiles even though the will be skewed to a lower force.

4.4.2 Oscillating Pattern

An oscillating pattern was created to test the ability of the system to generate a nonlinear force-displacement profile. During this test the motors were programmed to start at a link length of 195 mm. When signaled by the linear potentiometer the motors would change the link length to 155 mm. After 5 mm of displacement the motors returned the link length to 195 mm. The oscillation test did not reach the predicted values because of reasons discussed above, but it did reach the forces expected based on the constant link length tests as shown in Figure 4.6. In Figure 4.6 it can be seen that the coupler point deflects 2 mm in the time it takes the motors to travel from the 195 mm link length to the 155 mm link length. Future models can incorporate faster motors to decrease this lag in the controls.

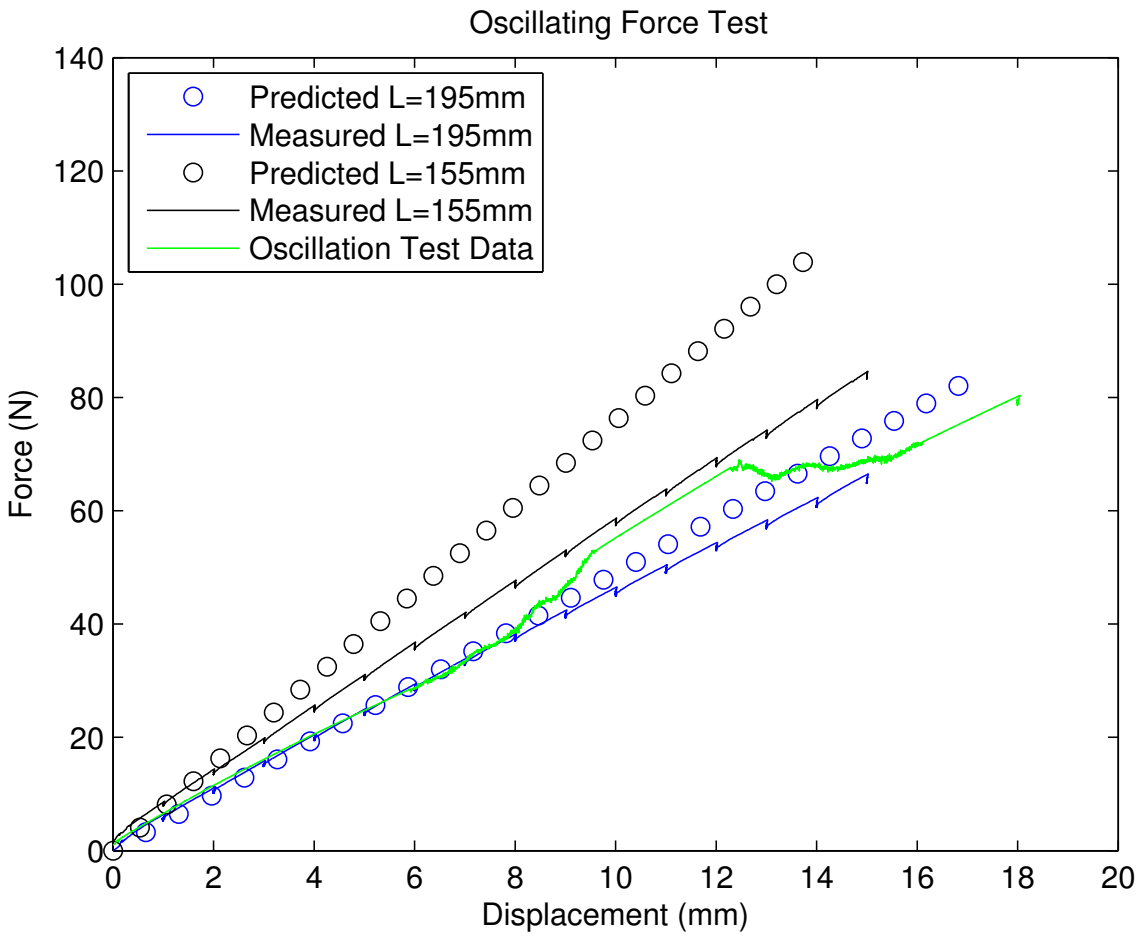


Figure 4.6: Oscillation Test. The black and blue lines show the predicted and measured limits that the oscillating force profile should move between. The green line is the force profiles as measured by the Instron.

4.4.3 Knee Profile

Finally the haptic interface was tested to see how well it could follow the force profile of an injured ACL knee during Lachman's test. As shown in Figure 4.7, the haptic interface was able to follow the basic shape of the injured ACL force profile. There was an average percent relative error of 13% between the two force profiles, but the measured force profile did not deviate more than 10 N from the desired force profile. This decrease in force from the desired force profile and the measure profile was expected as explained in Section 4.4.1. Even though there is a moderate difference in the force levels, the two force profiles are of a similar shape.

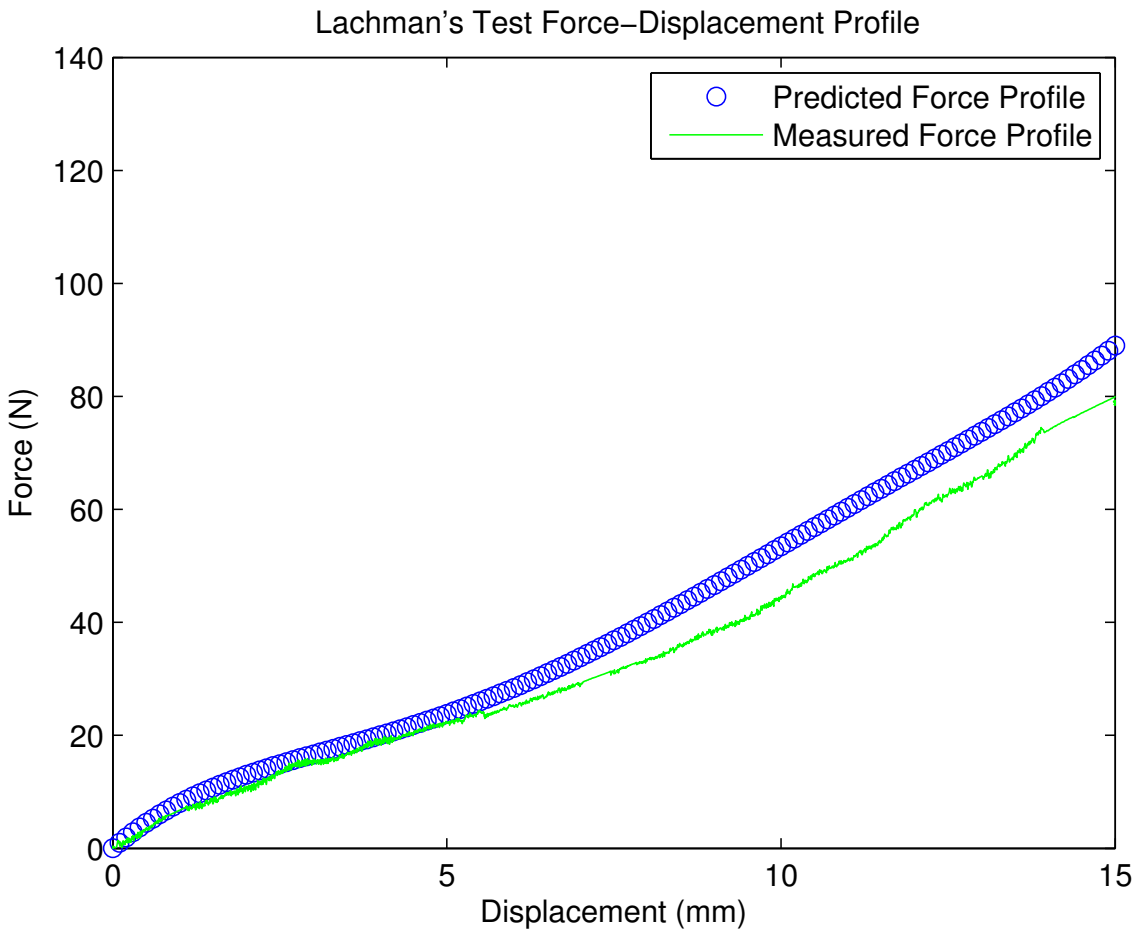


Figure 4.7: Lachman's Test Force-Displacement Profile. The blue circles are the desired force-displacement profile that the haptic interface is trying to recreate. The green line is the measured force-displacement profiles from the haptic interface as measured by the Instron.

4.5 Conclusion

The tests with the compliant mechanism outside the haptic interface in Section 4.3 show that the predicted force profiles for the compliant mechanism are accurate. But once the compliant mechanism was incorporated into the haptic device, the constant link lengths test showed the slider was not able to act as a fixed point, which led to moderate error between the PRBM and the measured force profiles. Despite the error in the force profiles, the ACL knee test and the oscillating force test show that the compliant haptic device is capable of matching various nonlinear force profiles that fit within the force range of the device.

Future work may improve upon the work presented in this research in three ways. Either the PRBM can be improved to accommodate the looseness of the slider, or the slider can be improved to eliminate the looseness found in this work. Both approaches would allow for an accurate prediction of the force profiles output by the haptic interface. Finally, a stronger motor would allow for quicker and more accurate positioning of the slider.

CHAPTER 5. CONCLUSION

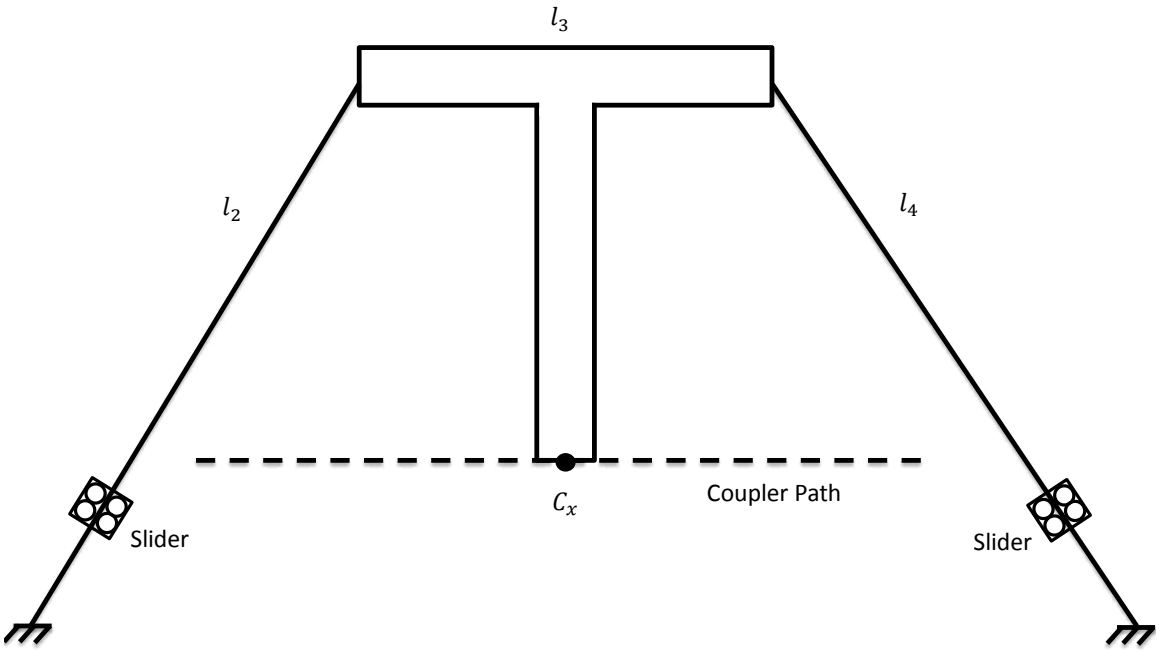


Figure 5.1: Final Compliant Mechanism Design. The final Robert's straight-line mechanism used as the center piece for the compliant haptic interface. The sliders move along the compliant links (l_2 and l_4) to change their effective length. The coupler point (C_x), the end point of the ridge link (l_3), is manipulated by the end user and follows the coupler path at all lengths of the compliant links

In this research a novel one degree of freedom haptic interface was developed, as shown in Figure 5.1. Unlike traditional haptic interfaces, in which the force is controlled using motors and rigid links, the haptic interface developed in this work displays haptic stiffness via compliant members. The objective of this work was to explore the use of compliant mechanisms in haptic

interfaces. This chapter is an overview of the accomplishments reached in this work, and a look into how future work can improve upon this research.

5.1 Accomplishments

This thesis demonstrated the possibility of using compliant mechanisms in haptic interfaces to generate prescribed force-deflection profiles. This was accomplished by developing a compliant mechanism that could change the length of its links without changing the coupler path.

The haptic interface also demonstrated that the return-to-zero behavior of a compliant mechanism can be taken advantage of to provide the force feedback of the haptic device. Often, the return-to-zero behavior of compliant mechanisms is viewed as a stumbling block, when it should be viewed as a resource.

A modified variable-stiffness Robert's straight-line mechanism was designed, and a PRBM was developed to predict its force output. The stiffness was varied by changing the effective length of the compliant links, 2 and 4, shown in Figure 5.1. The model assumes that the sliders act as an adjustable fixed point, and can be used to help build future PRBM's that incorporate variable link lengths.

Finally, a one degree of freedom prototype, shown in Figure 5.2, was developed to simulate the feel of Lachman's knee test. The prototype is simple to use, and provides a pattern to develop a more complex haptic interface to model the knee.

5.2 Limitations and Suggested Improvements

5.2.1 Design and Testing

Several lessons were learned during the design, building and testing of the compliant haptic interface prototype. The lessons learned and recommended solutions are presented below for the improvement of the haptic interface's design.

- Choose stronger and faster motors to increase the positioning accuracy of the sliders. The motors in this research lead to a lag in the response and slight deviations in positioning.

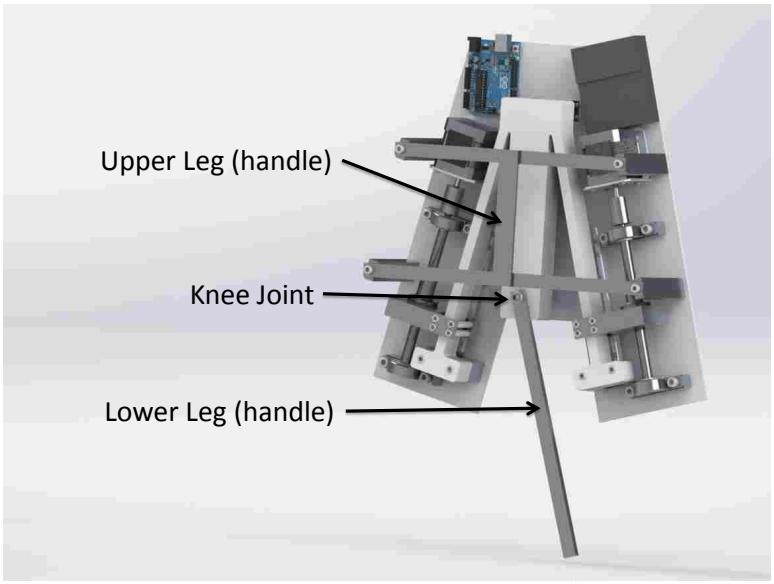


Figure 5.2: Haptic Knee Simulator

- Use closed-loop control to increased the position accuracy of the sliders. For example, two linear potentiometers could have been added to measure the sliders' position, and this information would have been used to correct for any skipping that might occur in the motors.

- Design the slider to fit more tightly onto the compliant links so that the haptic interface produces force-displacement profiles closer to the predicted model. The looseness in the sliders prevented them from being modeled as a fixed end condition for the beam. Also the two points of contact on the slider should be moved farther apart such that they are farther apart than the compliant mechanism's thickness (9.6 mm) to avoid binding issues.

- Design a smaller haptic interface. The compliant mechanism designed in this research was bulky, and further investigation into designing small compliant mechanisms with high mechanical advantage would allow for a smaller haptic interface.

- Design for the proper requirements. In this research several design decisions were made to simplify the control of the system. It might have been more helpful to base the design decisions on other parameters, such as, increasing the mechanical advantage.

- Design for the common benefits of compliant mechanisms. In this research the focus was on successfully incorporating compliant mechanisms into a haptic interface, and time was not

allotted to capitalize on the common benefits of compliant mechanisms, which include less wear, fewer parts, lower weight, and lower costs compared to similar rigid-link mechanisms.

5.2.2 Model Development

There are some changes that would need to take place if the development of the model that predicts the force output of a compliant haptic interface was preformed again. Below is a list of the improvements that could be incorporated into the method described in this research.

- Develop the key parameters in the PRBM. In this research γ and K_Θ were found for one of the compliant link lengths, and they were applied to the other link lengths. Future work can add to this model by developing an algorithm that can solve for γ and K_Θ as the link lengths change.

- Create a model to predict the force-displacement profile for a fixed-fixed beam with a moving simple support. The slider in this research was not good enough to represent a fixed support, and the simple support model would improve the ability to predict the force output of the haptic device.

- Increase the sophistication of the optimization algorithm. In this research an optimization algorithm was used to find the best Robert's straight-line mechanism for changing link length, and a separate optimization algorithm to find the best dimension for the compliant links. It turns out that these two algorithms are interdependent, and a more sophisticated algorithm is needed to solve for the best compliant Robert's straight-line mechanism.

- Use a more complete FEA model. In this research the stress concentration areas were left in the FEA model, which limited the accuracy of the force and stress predictions. A FEA model that incorporated the rounded edges from the final compliant mechanism would have improved the accuracy in solving for the PRBM parameters of γ and K_Θ.

5.3 Future Work

Along with the improvements specific to this research there are two big tasks that future work will need to answer within the scope of developing compliant haptic interfaces.

First, using only a fully compliant mechanism with changing link lengths provided a relatively narrow range of force outputs. With this limitation, haptic interfaces need to be designed with a specific application in mind. Future work will need to expand the range of forces that compliant haptic interfaces are capable of producing to expand their use beyond specific applications.

Second, a one degree of freedom compliant haptic interface was developed, but there are many situations when two to three degrees of freedom are needed to fully simulate a virtual environment. Future work can expand the number of degrees of freedom that compliant haptic interfaces are capable of providing. This will also increase the complexity of predicting the force output of the mechanism as the multiple degrees of freedom potentially interfere with one another.

As these challenges are taken on compliant mechanisms have the potential to improve the human-robot interface in both haptic interfaces and the greater field of robotics.

REFERENCES

[1] Howell, L. L., 2001. *Compliant Mechanisms*. John Wiley & Sons, Inc., New York. 2, 16, 17, 20

[2] Daniel, D. M., Akeson, W. H., and J., O. J., eds., 1990. *Knee Ligaments: Structure, Function, Injury, and Repair*. Raven Press, New York. 3, 4, 36

[3] http://www.medmetric.com. 4

[4] Frey, M., Riener, R., Michas, C., Regenfelder, F., and Burgkart, R., 2006. "Elastic properties of an intact and ACL-ruptured knee joint: measurement, mathematical modelling, and haptic rendering." *Journal of biomechanics,* **39**(8), Jan., pp. 1371–82. 4, 7, 36

[5] Pratt, G.A. and Williamson, M., 1995. "Series Elastic Actuators.pdf." In *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, pp. 399– 406. 4

[6] Williamson, M. M., 1995. "Series elastic actuators." PhD thesis, Massachusetts Institute of Technology. 4

[7] Schiavi, R., Grioli, G., Sen, S., and Bicchi, A., 2008. "VSA-II: a novel prototype of variable stiffness actuator for safe and performing robots interacting with humans." In *2008 IEEE International Conference on Robotics and Automation*, IEEE, pp. 2171–2176. 4

[8] Choi, J., Hong, S., Member, S., Lee, W., Kang, S., and Kim, M., 2011. "A Robot Joint with Variable Stiffness Using Leaf Springs." *Robotics, IEEE Transactions on,* **27**(2), pp. 229–238. 4

[9] Ishibashi, R., Ozawa, R., and Kawamura, S., 2007. "Mass estimation in microgravity with a variable stiffness mechanism." In *Advanced intelligent mechatronics, 2007 IEEE/ASME international conference on*. 4

[10] Galloway, K. C., Clark, J. E., and Koditschek, D. E., 2009. "Design of a tunable stiffness composite leg for dynamic locomotion." In *ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*. 4

[11] Galloway, K. C., 2013. "Variable Stiffness Legs for Robust, Efficient, and Stable Dynamic Running." *Journal of Mechanisms and Robotics,* **5**(1), Jan., p. 011009. 4

[12] Wolf, S., Eiberger, O., and Hirzinger, G., 2011. "The DLR FSJ: Energy based design of a variable stiffness joint." In *2011 IEEE International Conference on Robotics and Automation*, DLR - German Aerospace Center, Institute of Robotics and Mechatronics, D-82234 Wessling, Germany, Ieee, pp. 5082–5089. 4

[13] Kim, B., and Song, J., 2010. "Hybrid dual actuator unit: A design of a variable stiffness actuator based on an adjustable moment arm mechanism." *Robotics and Automation ICRA 2010 IEEE International Conference on*, pp. 1655–1660. 4

[14] Kim, B., and Song, J., 2012. "Design and Control of a Variable Stiffness Actuator Based on AdjustableMoment Arm." *Robotics, IEEE Transactions on,* **28**(5), pp. 1145–1151. 4

[15] Leishman, L. C., Ricks, D. J., and Colton, M. B., 2010. "Design and Evaluation of Statically Balanced Compliant Mechanisms for Haptic Interfaces." In *ASME 2010 Dynamic Systems and Control Conference, Volume 1*, Asme, pp. 859–866. 5

[16] Leishman, L. C., and Colton, M. B., 2011. "A Pseudo-Rigid-Body Model Approach for the Design of Compliant Mechanism Springs for Prescribed Force-Deflections." In *ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, ASME, ed. 5

[17] Merriam, E. G., Howell, L. L., Magleby, S., and Colton, M., 2013. "The Design of a Fully Compliant Statically Balanced Mechanism." In *Proceedings of ASME 2013 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, ASME. 5

[18] Basafa, E., Sheikholeslami, M., Mirbagheri, A., Farahmand, F., and Vossoughi, G. R., 2009. "Design and implementation of series elastic actuators for a haptic laparoscopic device.." In *Conference Proceedings of the International Conference of IEEE Engineering in Medicine and Biology Society*, Vol. 2009, Johns Hopkins University, Baltimore, MD 21218, USA. basafa@jhu.edu, pp. 6054–6057. 5

[19] Lu, K., 2005. "Design and Application of Compliant Mechanisms for Surgical Tools." *Journal of Biomechanical Engineering,* **127**(6), July, p. 981. 5

[20] Awtar, S., Trutna, T. T., Nielsen, J. M., Abani, R., and Geiger, J., 2010. "FlexDex: A Minimally Invasive Surgical Tool With Enhanced Dexterity and Intuitive Control." *Journal of Medical Devices,* **4**(3), p. 035003. 5

[21] Gillespie, R. B., Shin, T., Huang, F., and Trease, B., 2008. "Automated Characterization and Compensation for a Compliant Mechanism Haptic Device." *IEEE/ASME Transactions on Mechatronics,* **13**(1), Feb., pp. 136–146. 5

[22] Sung, E., Slocum, A. H., Ma, R., Bean, J. F., and Culpepper, M. L., 2011. "Design of an Ankle Rehabilitation Device Using Compliant Mechanisms." *Journal of Medical Devices,* **5**(1), p. 011001. 5

[23] Norton, R. L., 2008. *Design of Machinery: An Introduction to the Synthesis and Analysis of Mechanisms and Machines*., fourth edi ed. McGraw-Hill, New York. 5, 13

[24] Joinie-Maurin, M., LSIIT, ULP, Illkirch, F., Barbe, L., Piccin, O., Gangloff, J., Bayle, B., and Rump, R., 2010. "Design of a Linear Haptic Display Based on Approximate Straight Line Mechanisms." In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, IEEE, pp. 5048–5053. 5, 15

[25] Lei, H., Sch. of Mech. Instrum. Eng., Xi"an Univ. of Technol., Xi"an, C., Xiaomin, J., and Pengtao, H., 2012. "The Low Coupling Compliant Straight -Line Mechanisms Structure and Parameter Analysis." In *Computer Science and Information Processing (CSIP), 2012 International Conference on*, IEEE, ed., pp. 833–836. 6

[26] Hubbard, N. B., Wittwer, J. W., Kennedy, J. A., Wilcox, D. L., and Howell, L. L., 2004. "A Novel Fully Compliant Planar Linear-Motion Mechanism." In *ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 2, ASME, pp. 1–5. 6, 17

[27] Kuchta, M., Iwanejko, L., Szulim, M., and Sokolowski, Z., 2006. "Modelling and Dynamics of a Knee-Joint." In *Modern Problems of Radio Engineering, Telecommunications, and Computer Science, 2006. TCSET 2006. International Conference*, pp. 665–667. 7

[28] Manamanni, N., Moughamir, S., Gasmi, M. a., and Zaytoon, J., 2005. "A dynamic three dimensional tibio femoral modeling.." In *Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, Vol. 3, pp. 2989–92. 7

[29] Franken, H. M., Veltink, P. H., Tijsmans, R., Member, S., Boom, H. B. K., and Member, A., 1993. "Identification of Passive Knee Joint and Shank Dynamics in Paraplegics Using Quadriceps Stimulation." *Rehabilitation Engineering, IEEE Transactions on,* **I**(3), pp. 154–164. 7

[30] Abdel-Rahman, E., Hefzy, M., and Cooke, T. "Determination of the ligamentous and contact forces in the human tibio-femoral joint using a three-dimensional dynamic anatomical model." In *Proceedings of the 1996 Fifteenth Southern Biomedical Engineering Conference*, IEEE, pp. 373–376. 7

[31] Erickson, A., Beynnon, B., Werthiemer, C., Fleming, B., Pope, M., Johnson, R., Howe, J., and Nichols, C., 1988. "An in-vivo study of ACL strain in the normal knee during Lachman and Drawer tests." In *Bioengineering Conference, 1988., Proceedings of the 1988 Fourteenth Annual Northeast*, IEEE, pp. 19–21. 7

[32] Frey, M., Burgkart, R., Regenfelder, F., and Riener, R., 2004. "Optimised robot-based system for the exploration of elastic joint properties.." *Medical & biological engineering & computing,* **42**(5), Sept., pp. 674–8. 7

[33] Kawaguchi, S., Nagamune, K., Araki, D., Kubo, S., Kuroda, R., and Kurosaka, M., 2010. "A Quantitative Measurement System of Lachman Test with Force Sensor." In *World Automation Congress (WAC)*, pp. 1–6. 7

[34] Lin, H., Hsu, H., Chang, C., and Lai, W., 2009. "Analyzing Anterior Knee Laxity with an Arthrometer for Assisting Diagnosis of Anterior Cruciate Ligament Rupture I :." In *Bioengineering Conference, 2009 IEEE 35th Annual Northeast*, IEEE, pp. 1–2. 7

[35] Riener, R., Frey, M., Pröll, T., Regenfelder, F., and Burgkart, R., 2004. "Phantom-based multimodal interactions for medical education and training: the Munich Knee Joint Simulator.." *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society,* **8**(2), June, pp. 208–16. 7

[36] Morita, Y., Kawai, Y., Hayashi, Y., Hirano, T., Ukai, H., Sanaka, K., Nakamuta, H., and Takao, K., 2010. "Development of Knee Joint Robot for Students Becoming Therapist - Design of Prototype and Fundamental Experiments -." In *International Conference on Control, Automation and Systems*, pp. 151–155. 7

[37] Norton, R. L., 1996. *Machine Design An Integrated Approach*., first edi ed. Prentice Hill, Englewood Cliffs, N.J. 26

[38] www.ardnuino.cc/en/Main/ArduinoBoardUno. 35

[39] https://learn.adafruit.com/adafruit-motor-shield-v2-for arduino. 132

APPENDIX A. STRAIGHT-LINE MECHANISM ANALYSIS

A.1 Matlab Code

A.1.1 Robert Straight-line Mechanism

```
% Four Bar Kinematic Analysis ROBERTS
% Name: Jeffrey Hawks
% Date: 28 June 2013
% Description: Roberts Straight Line analysis

clear;
clc;

N = 15;

for j = 1:N

%% Position Analysis — Four Bar
    % (See 'Four—Link Mechanism Analytical Positon Analysis' handout from Dr.
    % Brian Jensen)
% Link Lengths
% Robert's

% Optimized (N = 10)
%r1 = 77.4789
%r2 = 34.3175
%r3 = 41.4145
%r4 = 34.3175
%rp = 55.4009
```

```
% Optimized (N = 15)
%r1 = 77
%r2 = 23.6
%r3 = 51.5
%r4 = 23.6
%rp = 95

r1o = 77.5; % Ground Link
r2o = 34.3;
theta2o = 60*pi/180;

r2 = r2o + j-1; % Input Link
r3 = 41.4; % Coupler Link
r4 = r2o + j-1; % Output Link
r1 = r1o + 2*(r2-r2o)*cos(theta2o);
a3 = .5*r3;%1.5*r2o*cos(70.53*pi/180); % Coupler Pt (along the Coupler Link)
b3 = -sqrt(55.4^2 - a3^2);%-1.5*r2o*sin(70.53*pi/180); % Coupler Pt ...
    (Normal from the Coupler Link)

% Input angle (the range depends on the type of fourbar)
theta2 = [41*pi/180:1*pi/180:71*pi/180]; % radians

% Open vs. Crossed Solution
mu = -1; % Open Solution (mu = -1) or Crossed Solution (mu = 1)

% Solve for vector r7
r7x = r2*cos(theta2) - r1; % x component
r7y = r2*sin(theta2); % y compnent
theta7 = atan2(r7y,r7x); % angle
r7 = (r7x.^2 + r7y.^2).^(0.5); % magnitude

% Solve for psi
psi = acos((r4^2 + r7.^2 - r3^2)./(2*r4*r7));

% Solve for theta4
```

```
theta4prime = theta7 + mu*psi; % This may be in the wrong quadrant
theta4 = atan2(sin(theta4prime), cos(theta4prime)); % radians

% Solve for theta3
r3x = r1 + r4*cos(theta4) - r2*cos(theta2); % x component
r3y = r4*sin(theta4) - r2*sin(theta2); % y component
theta3 = atan2(r3y,r3x); % radians

% Solve for the coupler pt. location
cpx = r2*cos(theta2) + a3*cos(theta3) - b3*sin(theta3); % Along the x axis
cpy = r2*sin(theta2) + a3*sin(theta3) + b3*cos(theta3); % Along the y axis

% Plot Coupler Path
figure(1)
subplot(2,2,1)
plot(cpx,cpy)
xlabel('cp_x')
ylabel('cp_y')
title('Roberts - Coupler Path')
hold on

subplot(2,2,2)
plot(theta2,cpx)
xlabel('\theta_2 (rad)')
ylabel('cp_x')
hold on

subplot(2,2,3)
plot(theta2,cpy)
xlabel('\theta_2 (rad)')
ylabel('cp_y')
hold on

%% Linearity Check

for i = 1:length(theta2)-1
```

```
    % For the straight line deviation check
    deltay(i) = cpy(i) - cpy(20);

    % For the slope
    dely(i) = cpy(i) - cpy(i+1);
    delx(i) = cpx(i) - cpx(i+1);

    slope(i) = dely(i)/delx(i);
end

deltay(length(theta2)) = cpy(length(theta2)) - cpy(20);

% Max deviation from straight line
maxdeltay = abs(max(deltay));
mindeltay = abs(min(deltay));
if maxdeltay > mindeltay
    mdeltay(j) = maxdeltay;
else
    mdeltay(j) = mindeltay;
end

% Rectilinear check
maxslope = abs(max(slope));
minslope = abs(min(slope));
if maxslope > minslope
    mslope(j) = maxslope;
else
    mslope(j) = minslope;
end

distance(j) = cpx(1) - cpx(end);
end
```

A.1.2 Watt Straight-line Mechanism

```
%% Four Bar Kinematic Analysis Watt
% Name: Jeffrey Hawks
% Date: 28 June 2013
% Description: Watt Straight Line analysis

clear;
clc;

N = 10;

for j = 1:N

%% Position Analysis - Four Bar
    % (See 'Four-Link Mechanism Analytical Positon Analysis' handout from Dr.
    % Brian Jensen)
% Link Lengths
% Robert's
Scale = 25; %Scale Factor
r1o = 4*Scale; % Ground Link
r2o = 2*Scale; % Input Link
r3 = 1*Scale; % Coupler Link
r4o = 2*Scale; % Output Link
a3 = 0.5*Scale*cos(0*pi/180); % Coupler Pt (along the Coupler Link)
b3 = 0*Scale*sin(0*pi/180); % Coupler Pt (Normal from the Coupler Link)

theta2o = 41.41*pi/180;

r2 = 2*Scale + j-1;
r4 = 2*Scale + j-1;
r1 = r1o + 2*(r2-r2o)*cos(theta2o);

% Input angle (the range depends on the type of fourbar)
theta2 = [325*pi/180:1*pi/180:365*pi/180]; % radians
```

```
% Open vs. Crossed Solution
mu = -1; % Open Solution (mu = -1) or Crossed Solution (mu = 1)

% Solve for vector r7
r7x = r2*cos(theta2) - r1; % x component
r7y = r2*sin(theta2); % y compnent
theta7 = atan2(r7y,r7x); % angle
r7 = (r7x.^2 + r7y.^2).^(0.5); % magnitude

% Solve for psi
psi = acos((r4^2 + r7.^2 - r3^2)./(2*r4*r7));

% Solve for theta4
theta4prime = theta7 + mu*psi; % This may be in the wrong quadrant
theta4 = atan2(sin(theta4prime), cos(theta4prime)); % radians

% Solve for theta3
r3x = r1 + r4*cos(theta4) - r2*cos(theta2); % x component
r3y = r4*sin(theta4) - r2*sin(theta2); % y component
theta3 = atan2(r3y,r3x); % radians

% Solve for the coupler pt. location
cpx_ = r2*cos(theta2) + a3*cos(theta3) - b3*sin(theta3); % Along the x axis
cpy_ = r2*sin(theta2) + a3*sin(theta3) + b3*cos(theta3); % Along the y axis

%Rotate so that the straightline of the mechanism is the x axis
for i=1:length(theta2)
cp_ = [cpx_(i); cpy_(i); 0];

t = [r1/2; 0; 0]; % Translation

cp = cp_ - t;

R = [cos(75*pi/180), sin(75*pi/180), 0;...
    -sin(75*pi/180), cos(75*pi/180), 0;...
```

```
    0, 0, 1]; % Rotation

cp = R*cp;
cpx(i) = cp(1);
cpy(i) = cp(2);

end

% Plot Coupler Path
figure(2)
subplot(2,2,1)
plot(cpx,cpy)
xlabel('cp_x')
ylabel('cp_y')
title(' Watt - Coupler Path')
hold on

subplot(2,2,2)
plot(theta2,cpx)
xlabel('\theta_2 (rad)')
ylabel('cp_x')
hold on

subplot(2,2,3)
plot(theta2,cpy)
xlabel('\theta_2 (rad)')
ylabel('cp_y')
hold on

%% Linearity Check

for i = 1:length(theta2)-1
    % For the straight line deviation check
    deltay(i) = cpy(i) - cpy(21);

    % For the slope
```

```
    dely(i) = cpy(i) - cpy(i+1);
    delx(i) = cpx(i) - cpx(i+1);

    slope(i) = dely(i)/delx(i);
end

deltay(length(theta2)) = cpy(length(theta2)) - cpy(21);

% Max deviation from straight line
maxdeltay = abs(max(deltay));
mindeltay = abs(min(deltay));
if maxdeltay > mindeltay
    mdeltay(j) = maxdeltay;
else
    mdeltay(j) = mindeltay;
end

% Rectilinear check
maxslope = abs(max(slope));
minslope = abs(min(slope));
if maxslope > minslope
    mslope(j) = maxslope;
else
    mslope(j) = minslope;
end

distance(j) = abs(cpx(1) - cpx(end));

end
```

A.1.3 Chebysev Straight-line Mechanism

```
%% Four Bar Kinematic Analysis Chebysev
% Name: Jeffrey Hawks
% Date: 28 June 2013
```

```
% Description: Chebysev Straight Line analysis

clear;
clc;

N = 10;

for j = 1:N

%% Position Analysis — Four Bar
    % (See 'Four—Link Mechanism Analytical Positon Analysis' handout from Dr.
    % Brian Jensen)
% Link Lengths
% Robert's
Scale = 25; %Scale Factor
r1 = 2*Scale; % Ground Link
r2 = 2.5*Scale + j—1; % Input Link
r3 = 1*Scale; % Coupler Link
r4 = 2.5*Scale + j—1; % Output Link
a3 = 0.5*Scale*cos(0*pi/180); % Coupler Pt (along the Coupler Link)
b3 = 0*Scale*sin(0*pi/180); % Coupler Pt (Normal from the Coupler Link)

% Input angle (the range depends on the type of fourbar)
theta2 = [41*pi/180:1*pi/180:80*pi/180]; % radians

% Open vs. Crossed Solution
mu = 1; % Open Solution (mu = —1) or Crossed Solution (mu = 1)

% Solve for vector r7
r7x = r2*cos(theta2) — r1; % x component
r7y = r2*sin(theta2); % y compnent
theta7 = atan2(r7y,r7x); % angle
r7 = (r7x.^2 + r7y.^2).^(0.5); % magnitude

% Solve for psi
psi = acos((r4^2 + r7.^2 — r3^2)./(2*r4*r7));
```

```
% Solve for theta4
theta4prime = theta7 + mu*psi; % This may be in the wrong quadrant
theta4 = atan2(sin(theta4prime), cos(theta4prime)); % radians

% Solve for theta3
r3x = r1 + r4*cos(theta4) - r2*cos(theta2); % x component
r3y = r4*sin(theta4) - r2*sin(theta2); % y component
theta3 = atan2(r3y,r3x); % radians

% Solve for the coupler pt. location
cpx = r2*cos(theta2) + a3*cos(theta3) - b3*sin(theta3); % Along the x axis
cpy = r2*sin(theta2) + a3*sin(theta3) + b3*cos(theta3); % Along the y axis

% Plot Coupler Path
figure(3)
subplot(2,2,1)
plot(cpx,cpy)
xlabel('cp_x')
ylabel('cp_y')
title('Chebysev - Coupler Path')
hold on

subplot(2,2,2)
plot(theta2,cpx)
xlabel('\theta_2 (rad)')
ylabel('cp_x')
hold on

subplot(2,2,3)
plot(theta2,cpy)
xlabel('\theta_2 (rad)')
ylabel('cp_y')
hold on

%% Linearity Check
```

```
for i = 1:length(theta2)-1
    % For the straight line deviation check
    deltay(i) = cpy(i) - cpy(13);

    % For the slope
    dely(i) = cpy(i) - cpy(i+1);
    delx(i) = cpx(i) - cpx(i+1);

    slope(i) = dely(i)/delx(i);
end

deltay(length(theta2)) = cpy(length(theta2)) - cpy(13);

% Max deviation from straight line
maxdeltay = abs(max(deltay));
mindeltay = abs(min(deltay));
if maxdeltay > mindeltay
    mdeltay(j) = maxdeltay;
else
    mdeltay(j) = mindeltay;
end

% Rectilinear check
maxslope = abs(max(slope));
minslope = abs(min(slope));
if maxslope > minslope
    mslope(j) = maxslope;
else
    mslope(j) = minslope;
end

distance(j) = cpx(1) - cpx(end);
end
```

A.1.4 Evan's 3 Straight-line Mechanism

```
%% Four Bar Kinematic Analysis Evans3
% Name: Jeffrey Hawks
% Date: 28 June 2013
% Description: Evans 3 Straight Line analysis

clear;
clc;

N = 10;

for j = 1:N

%% Position Analysis - Four Bar
    % (See 'Four-Link Mechanism Analytical Positon Analysis' handout from Dr.
    % Brian Jensen)
% Link Lengths
% Robert's
Scale = 25; %Scale Factor
r1 = 2*Scale; % Ground Link
r2 = 1*Scale + j-1; % Input Link
r3 = 1*Scale; % Coupler Link
r4 = 1*Scale + j-1; % Output Link
a3 = 2*Scale*cos(0*pi/180); % Coupler Pt (along the Coupler Link)
b3 = 0*Scale*sin(0*pi/180); % Coupler Pt (Normal from the Coupler Link)

% Input angle (the range depends on the type of fourbar)
theta2 = [55*pi/180:1*pi/180:75*pi/180]; % radians

% Open vs. Crossed Solution
mu = -1; % Open Solution (mu = -1) or Crossed Solution (mu = 1)

% Solve for vector r7
r7x = r2*cos(theta2) - r1; % x component
```

```
r7y = r2*sin(theta2); % y compnent
theta7 = atan2(r7y,r7x); % angle
r7 = (r7x.^2 + r7y.^2).^(0.5); % magnitude

% Solve for psi
psi = acos((r4^2 + r7.^2 - r3^2)./(2*r4*r7));

% Solve for theta4
theta4prime = theta7 + mu*psi; % This may be in the wrong quadrant
theta4 = atan2(sin(theta4prime), cos(theta4prime)); % radians

% Solve for theta3
r3x = r1 + r4*cos(theta4) - r2*cos(theta2); % x component
r3y = r4*sin(theta4) - r2*sin(theta2); % y component
theta3 = atan2(r3y,r3x); % radians

% Solve for the coupler pt. location
cpx_ = r2*cos(theta2) + a3*cos(theta3) - b3*sin(theta3); % Along the x axis
cpy_ = r2*sin(theta2) + a3*sin(theta3) + b3*cos(theta3); % Along the y axis

%Rotate so that the straightline of the mechanism is the x axis
for i=1:length(theta2)
cp_ = [cpx_(i); cpy_(i); 0];

t = [r1; 0; 0]; % Translation

cp = cp_ - t;

R = [cos(60*pi/180), sin(60*pi/180), 0;...
    -sin(60*pi/180), cos(60*pi/180), 0;...
    0, 0, 1]; % Rotation

cp = R*cp;
cpx(i) = cp(1);
cpy(i) = cp(2);
```

```
end

% Plot Coupler Path
figure(4)
subplot(2,2,1)
plot(cpx,cpy)
xlabel('cp_x')
ylabel('cp_y')
title('Evans – Coupler Path')
hold on

subplot(2,2,2)
plot(theta2,cpx)
xlabel('\theta_2 (rad)')
ylabel('cp_x')
hold on

subplot(2,2,3)
plot(theta2,cpy)
xlabel('\theta_2 (rad)')
ylabel('cp_y')
hold on

%% Linearity Check

for i = 1:length(theta2)-1
    % For the straight line deviation check
    deltay(i) = cpy(i) - cpy(21);

    % For the slope
    dely(i) = cpy(i) - cpy(i+1);
    delx(i) = cpx(i) - cpx(i+1);

    slope(i) = dely(i)/delx(i);
end
```

```
deltay(length(theta2)) = cpy(length(theta2)) - cpy(21);

% Max deviation from straight line
maxdeltay = abs(max(deltay));
mindeltay = abs(min(deltay));
if maxdeltay > mindeltay
    mdeltay(j) = maxdeltay;
else
    mdeltay(j) = mindeltay;
end

% Rectilinear check
maxslope = abs(max(slope));
minslope = abs(min(slope));
if maxslope > minslope
    mslope(j) = maxslope;
else
    mslope(j) = minslope;
end

distance(j) = cpx(1) - cpx(end);
end
```

A.2 Matlab Plots

This section provides the plots used to visualize the four straight-line mechanisms. Each plots shows the relationship between the x and y components of the coupler position, the y component of the coupler position and the angle of link 2, and the x component of the coupler position and the angle of link 2.

It is important to pay attention to the scale of the plots presented in this section. On some of the plots the lines do not appear to be very flat, but when considering the scale they are relatively flat.

A.2.1 Robert Straight-line Mechanism

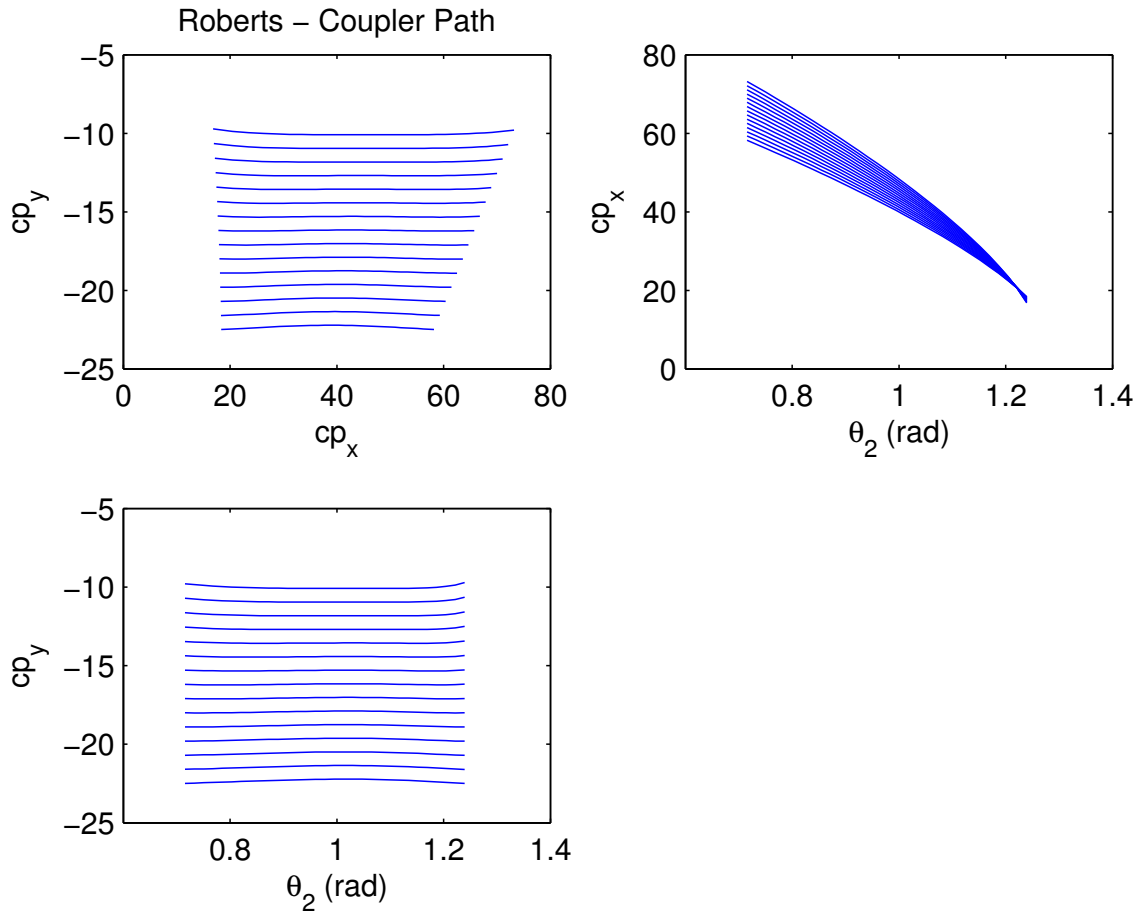


Figure A.1: Robert's Initial Analysis. cp_y is the position of the coupler in the y direction. cp_x is the position of the coupler in the x direction. θ_2 is the angular position of link 2 measured from the horizontal

A.2.2 Watt Straight-line Mechanism

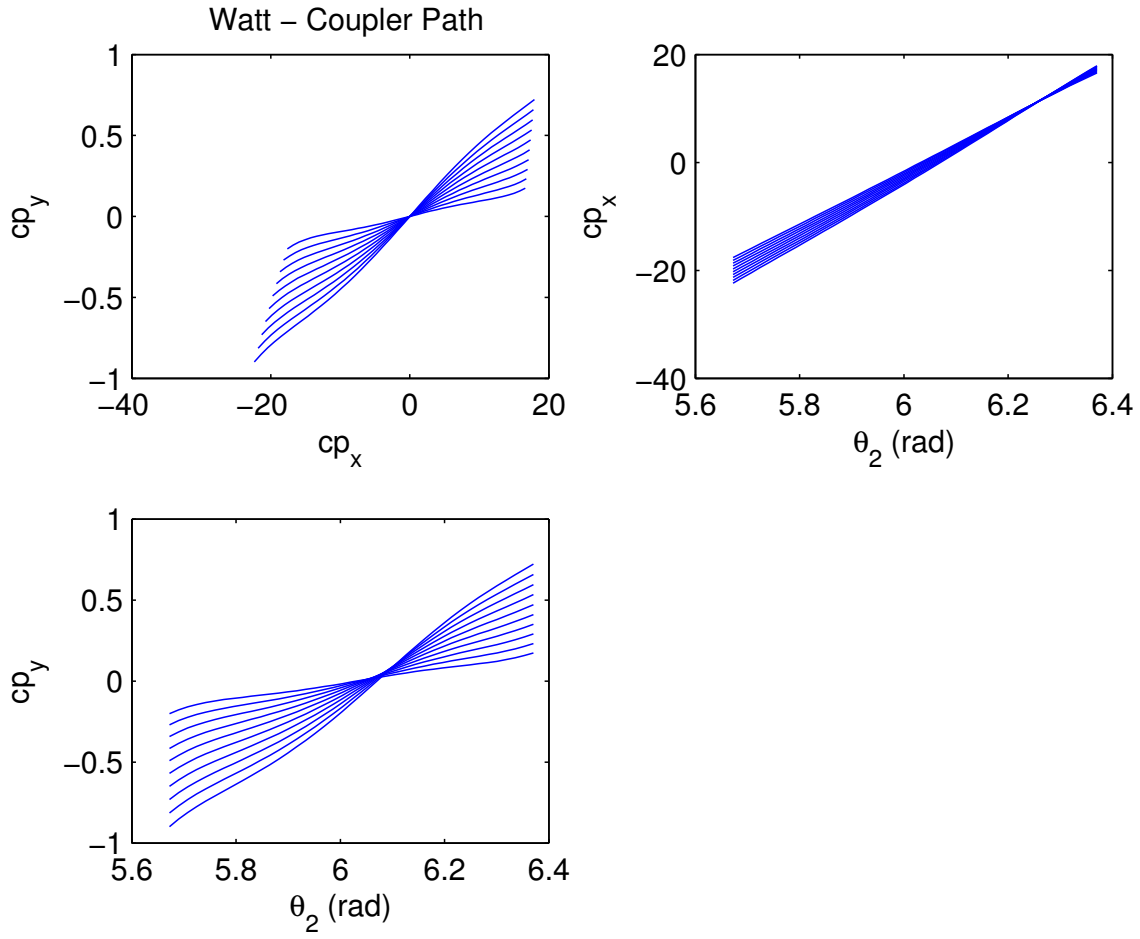


Figure A.2: Watt's Initial Analysis. cp_y is the position of the coupler in the y direction. cp_x is the position of the coupler in the x direction. θ_2 is the angular position of link 2 measured from the horizontal

A.2.3 Chebysev Straight-line Mechanism

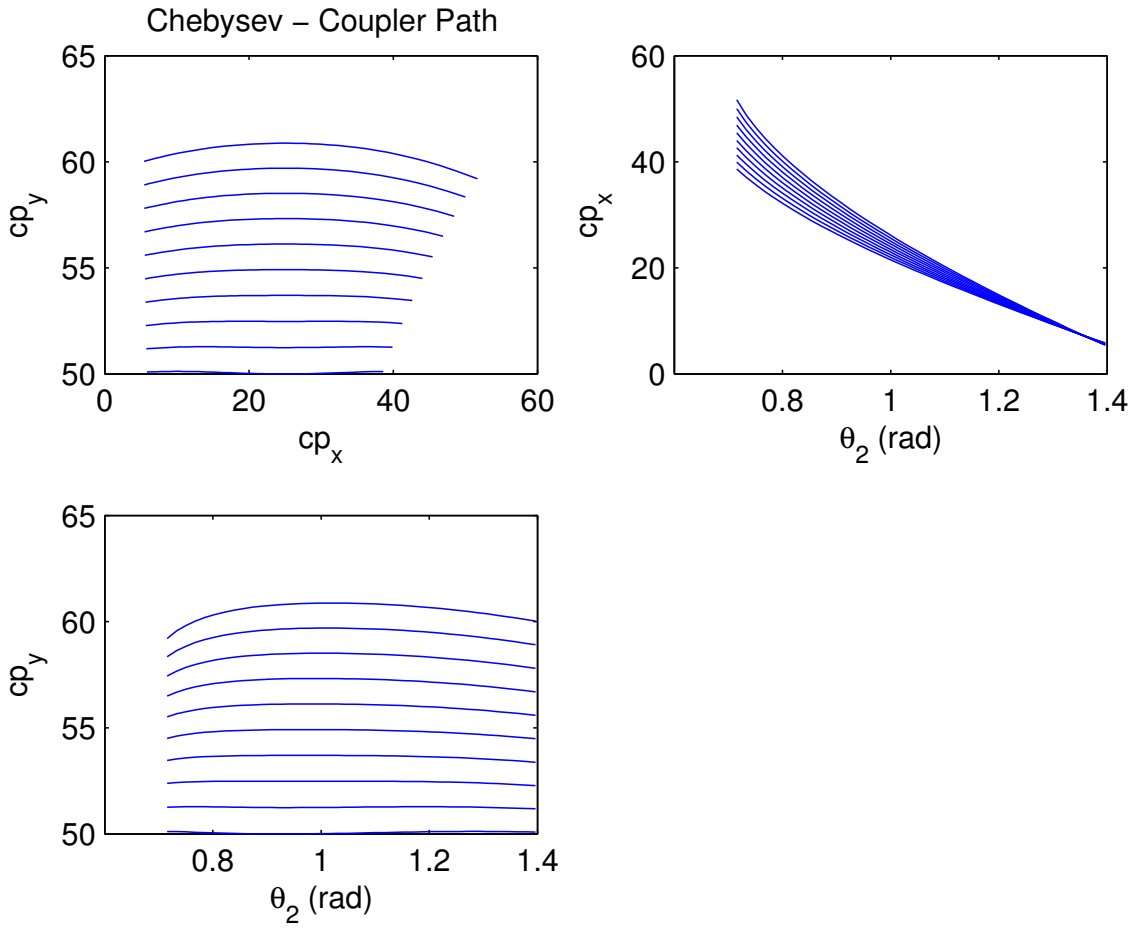


Figure A.3: hebysev Initial Analysis. cp_y is the position of the coupler in the y direction. cp_x is the position of the coupler in the x direction. θ_2 is the angular position of link 2 measured from the horizontal

A.2.4 Evan's 3 Straight-line Mechanism

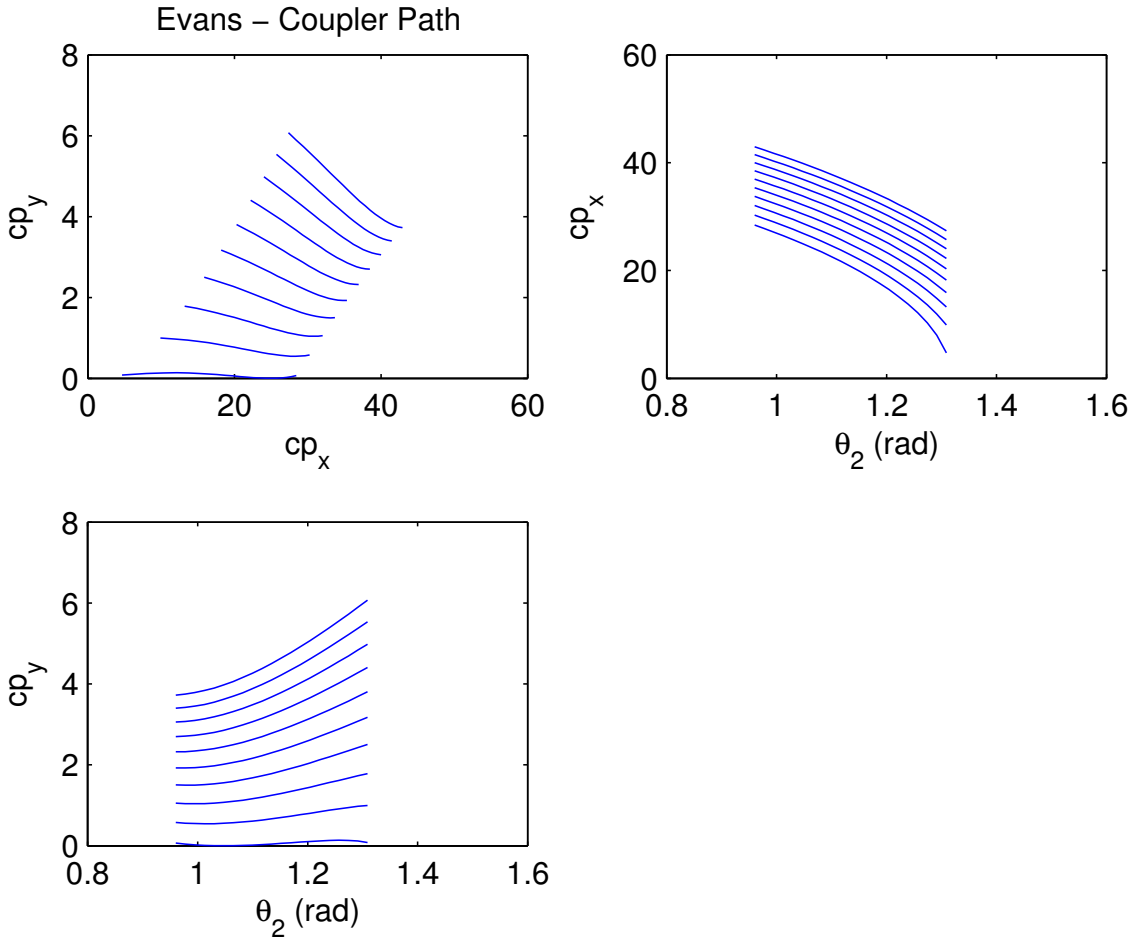


Figure A.4: Evan's Initial Analysis. cp_y is the position of the coupler in the y direction. cp_x is the position of the coupler in the x direction. θ_2 is the angular position of link 2 measured from the horizontal

APPENDIX B. VARIABLE STIFFNESS MECHANISM ANALYSIS

B.1 Matlab Code

B.1.1 Main Program

```
function varargout = AnalysisGUI(varargin)
% ANALYSISGUI MATLAB code for AnalysisGUI.fig
%      ANALYSISGUI, by itself, creates a new ANALYSISGUI or raises the ...
   existing
%      singleton*.
%
%      H = ANALYSISGUI returns the handle to a new ANALYSISGUI or the ...
   handle to
%      the existing singleton*.
%
%      ANALYSISGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ANALYSISGUI.M with the given input ...
   arguments.
%
%      ANALYSISGUI('Property','Value',...) creates a new ANALYSISGUI or ...
   raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before AnalysisGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to AnalysisGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
```

```
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help AnalysisGUI

% Last Modified by GUIDE v2.5 23-May-2014 14:37:46

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @AnalysisGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @AnalysisGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before AnalysisGUI is made visible.
function AnalysisGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to AnalysisGUI (see VARARGIN)

% Choose default command line output for AnalysisGUI
handles.output = hObject;
```

```
handles.mat = 3;
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes AnalysisGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = AnalysisGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function w_edit_Callback(hObject, eventdata, handles)
% hObject    handle to w_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.w = str2double(get(hObject, 'String'));
% Hints: get(hObject,'String') returns contents of w_edit as text
%        str2double(get(hObject,'String')) returns contents of w_edit as a ...
    double
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function w_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to w_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function h_edit_Callback(hObject, eventdata, handles)
% hObject    handle to h_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.h = str2double(get(hObject, 'String'));
% Hints: get(hObject,'String') returns contents of h_edit as text
%        str2double(get(hObject,'String')) returns contents of h_edit as a ...
    double
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function h_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to h_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function L1_edit_Callback(hObject, eventdata, handles)
% hObject    handle to L1_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.r1 = str2double(get(hObject, 'String'));
```

```
% Hints: get(hObject,'String') returns contents of L1_edit as text
%        str2double(get(hObject,'String')) returns contents of L1_edit as ...
    a double
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function L1_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to L1_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
     set(hObject,'BackgroundColor','white');
end



function L2_edit_Callback(hObject, eventdata, handles)
% hObject    handle to L2_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.r2 = str2double(get(hObject, 'String'));
% Hints: get(hObject,'String') returns contents of L2_edit as text
%        str2double(get(hObject,'String')) returns contents of L2_edit as ...
    a double
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function L2_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to L2_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function L3_edit_Callback(hObject, eventdata, handles)
% hObject    handle to L3_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.r3 = str2double(get(hObject, 'String'));
% Hints: get(hObject,'String') returns contents of L3_edit as text
%        str2double(get(hObject,'String')) returns contents of L3_edit as ...
    a double
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function L3_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to L3_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function L4_edit_Callback(hObject, eventdata, handles)
% hObject    handle to L4_edit (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.r4 = str2double(get(hObject, 'String'));
% Hints: get(hObject,'String') returns contents of L4_edit as text
%        str2double(get(hObject,'String')) returns contents of L4_edit as ...
    a double
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function L4_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to L4_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
     set(hObject,'BackgroundColor','white');
end



function B3_edit_Callback(hObject, eventdata, handles)
% hObject    handle to B3_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.rp = str2double(get(hObject, 'String'));
% Hints: get(hObject,'String') returns contents of B3_edit as text
%        str2double(get(hObject,'String')) returns contents of B3_edit as ...
    a double
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function B3_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to B3_edit (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in run_button.
function run_button_Callback(hObject, eventdata, handles)
% hObject    handle to run_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


%% Clear old plots or not
if(get(handles.clear_checkbox, 'Value'))
    cla(handles.axes1);
    cla(handles.axes2);
end
%% Get constants from edit fields ...
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
gamma = .8872; % Xbob .85; opt long .8656; opt short .9015
Ktheta = 2.0977; % XBob 2.65; opt long 2.1959; opt short 2.2479; opt long ...
    for gamma and combo for Ktheta 2.182

h = str2double(get(handles.h_edit, 'String'))/1000;
w = str2double(get(handles.w_edit, 'String'))/1000;
L1 = str2double(get(handles.L1_edit, 'String'))/1000;
L2 = str2double(get(handles.L2_edit, 'String'))/1000;
L3 = str2double(get(handles.L3_edit, 'String'))/1000;
L4 = str2double(get(handles.L4_edit, 'String'))/1000;
B3 = str2double(get(handles.B3_edit, 'String'))/1000;
```

```
N = str2double(get(handles.n_edit, 'String'));

%% Choose material
handles.mat = get(handles.material_menu, 'Value');
mat = handles.mat;
switch mat
    case 1 % Steel
        E = 207000000000; % Young's Modulus (Pa)
        Sy = 179000000; % Yield strength (Pa)
        Sut = 320000000; % Ultimate Strength (Pa)
        Sf = 0.5*Sut; % Fatigue strength (Pa)
        material = 'Steel';
    case 2 % Aluminim
        E = 71700000000; % Young's Modulus (Pa)
        Sy = 55000000; % Yield strength (Pa)
        Sut = 124000000; % Ultimate Strength (Pa)
        Sf = 0.4*Sut; % Fatigue strength (Pa)
        material = 'Aluminum';
    case 3 % Polypropylene
        E = 1.4*10^9; % Young's Modulus (Pa) for xbob 1.138*10^9
        Sy = 34*10^6; % Yield strength (Pa)
        Sut = 55*10^6; % Ultimate Strength (Pa)
        Sf = 0.3*Sut; % Fatigue strength (Pa)
        material = 'Polypropylene';
    case 4 % ABS
        E = 1627000000; % Young's Modulus (Pa)
        Sy = 22000000; % Yield strength (Pa)
        Sut = 40000000; % Ultimate Strength (Pa)
        Sf = 0.3*Sut; % Fatigue strength (Pa)
        material = 'ABS';
    case 5 % Titanium (Ti6A14V)
        E = 113800000000; % Young's Modulus (Pa)
        Sy = 880000000; % Yield strength (Pa)
        Sut = 950000000; % Ultimate Strength (Pa)
        Sf = 240000000; % Fatigue strength (Pa) (maybe up to 510 MPa)
        material = 'Titanium';
```

```
    case 6 % 301 stainless steel
        E = 190000000000; % Young's Modulus (Pa)
        Sy = 979000000; % Yield strength (Pa)
        Sut = 1379000000; % Ultimate Strength (Pa)
        Sf = 390000000; % Fatigue strength (Pa) Need to modify
        material = 'Stainless Steel';
end

%% Run a position analysis on the mechanism. - VWKinAnalysis3.m
axes(handles.axes1)
[r1 r2 r3 r4 a3 b3 theta2o] = RSMkin3(L1, L2, L3, L4, B3, N, gamma);

delta_theta2_deg = str2double(get(handles.deltaTheta2_edit, 'String'));
delta_theta2 = delta_theta2_deg*pi/180;

[theta2, theta3, theta4, deltax, deltay] = RobertTheta2(theta2o, ...
   delta_theta2, r1, r2, r3, r4, a3, b3, N);

%% Solve for kstar (material properties and w and h as inputs)
Kstar = (2*gamma^2*Ktheta*E*(w*h^3)/12);

%% Find the forces in the mechanism.
[F12x,F12y,F32x,F32y,F43x,F43y,F14x,F14y,Fp] = ...
   StaticForce(Kstar,r1,r2,r3,r4,a3,b3,theta2,theta3,theta4,theta2o);
[sigmamax,sigmaMpa] = Stress(w, h, Ktheta, gamma, r1, r2, r3, r4, F32x, ...
   F32y, F43x, F43y, theta2, theta2o);

% Fcheck = (4*gamma^2*Ktheta*E*w*h^3/12*(deltaTheta2_edit - ...
   theta2s)*pi/180)/(r2(1)^2*cosd(deltaTheta2_edit - theta2s));

% Find the forces perpendicular to and along the compliant beam
[F12xp, F12yp] = Rotation(F12x,F12y,r1,r2,r3);
set(handles.txt_F12xp, 'String', F12xp(1,1));
set(handles.txt_F12yp, 'String', F12yp(1,1));

% Solve for the factor of safety
```

```
sigmaalt = 0.5*sigmamax; % Alternating stress
sigmamean = 0.5*sigmamax; % Mean stress
SFyield = Sy/sigmamax; % Safety Factor for yield
SFfatigue =1/((sigmaalt/Sf)+(sigmamean/Sut));
%Output SF on the GUI
set(handles.sfy_text, 'String', SFyield);
set(handles.sff_text, 'String', SFfatigue);

axes(handles.axes2)
% Old stuff
%Xplot = Px*1000; % Set the x axis to mm
%Xplot = Xplot - min(min(Xplot));

Xplot = deltax*1000; % change mm to m for the location of the coupler pt.
hold on
grid on
%grid minor
title('Force vs Displacement (N mm change)')
xlabel('Coupler Position (mm)')
ylabel('Force (N)')

% PLOT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M{1} = 'KT-1000';
M{2} = 'KT-2000';
M{3} = 'Frey-Riener';

for i = 1:N
    plot(Xplot(i,:),Fp(i,:),'c')
end

% Get data for hurt ACL's
[xnew_kt1,ynew_kt1,xnew_kt2,ynew_kt2,xnew_fr,ynew_fr] = Data_hurtACL();

plot(xnew_kt1,ynew_kt1,'k','LineWidth',2)
plot(xnew_kt2,ynew_kt2,'g','LineWidth',2)
plot(xnew_fr,ynew_fr,'r','LineWidth',2)
```

```
title('ACL Defecient Knee Force Profiles')
for i = 1:N
    M{i} = ['r2 = ',num2str(r2(i)*1000), ' mm'];
    %legend (M(i,:))
end
M{end+1} = 'KT-1000';
M{end+1} = 'KT-2000';
M{end+1} = 'Frey-Riener';

display ('done');

% legend(M(end-2:end),...
%     'Location','NorthEast')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Code for the clearing checkbox
% --- Executes on button press in clear_checkbox.
function clear_checkbox_Callback(hObject, eventdata, handles)
% hObject    handle to clear_checkbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of clear_checkbox


% --- Executes on selection change in material_menu.
function material_menu_Callback(hObject, eventdata, handles)
% hObject    handle to material_menu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.mat = get(hObject,'Value');
% Hints: contents = cellstr(get(hObject,'String')) returns material_menu ...
    contents as cell array
%        contents{get(hObject,'Value')} returns selected item from ...
    material_menu
```

```
% --- Executes during object creation, after setting all properties.
function material_menu_CreateFcn(hObject, eventdata, handles)
% hObject    handle to material_menu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function n_edit_Callback(hObject, eventdata, handles)
% hObject    handle to n_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of n_edit as text
%        str2double(get(hObject,'String')) returns contents of n_edit as a ...
    double


% --- Executes during object creation, after setting all properties.
function n_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to n_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end



function theta2s_Callback(hObject, eventdata, handles)
% hObject    handle to theta2s (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of theta2s as text
%        str2double(get(hObject,'String')) returns contents of theta2s as ...
    a double


% --- Executes during object creation, after setting all properties.
function theta2s_CreateFcn(hObject, eventdata, handles)
% hObject    handle to theta2s (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function deltaTheta2_edit_Callback(hObject, eventdata, handles)
% hObject    handle to deltaTheta2_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of deltaTheta2_edit as text
```

```
%        str2double(get(hObject,'String')) returns contents of ...
    deltaTheta2_edit as a double


% --- Executes during object creation, after setting all properties.
function deltaTheta2_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to deltaTheta2_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function txt_F12xp_Callback(hObject, eventdata, handles)
% hObject    handle to txt_F12xp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of txt_F12xp as text
%        str2double(get(hObject,'String')) returns contents of txt_F12xp ...
    as a double


% --- Executes during object creation, after setting all properties.
function txt_F12xp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txt_F12xp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function txt_F12yp_Callback(hObject, eventdata, handles)
% hObject    handle to txt_F12yp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of txt_F12yp as text
%        str2double(get(hObject,'String')) returns contents of txt_F12yp ...
    as a double


% --- Executes during object creation, after setting all properties.
function txt_F12yp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txt_F12yp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function theta2o_edit_Callback(hObject, eventdata, handles)
% hObject    handle to theta2o_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of theta2o_edit as text
%        str2double(get(hObject,'String')) returns contents of ...
   theta2o_edit as a double


% --- Executes during object creation, after setting all properties.
function theta2o_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to theta2o_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --------------------------------------------------------------------
function Untitled_1_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

B.1.2 PRBM Conversion

```
%% Find link lengths
% Name: Jeffrey Hawks


%% The change in this function is how r1 and/or theta2o are solved for.
```

```
function [r1, r2, r3, r4, a3, b3, theta2o] = RSMkin3(L1o, L2o, L3, L4o, ...
    B3, N, gamma)


%solving for the nuetral theta2
y = (L1o -  L3)/2;
theta2o = acos(y/L2o);

% Solve for A3
A3 = L3/2;

% Get all of the changes in L2
for j = 1:N
    L2(j,1) = L2o - ((j-1)*0.001);
    L4(j,1) = L2(j);
    L1(j,1) = L1o - 2*(L2o-L2(j))*cos(theta2o);
end

% Convert to PRBM
for j = 1:N
    r2(j,1) = gamma*L2(j);
    r4(j,1) = gamma*L4(j);

    r1(j,1) = L1(j) - (1-gamma)*L2(j)*cos(theta2o);
    r3(j,1) = L3 + (1-gamma)*L2(j)*cos(theta2o);

    a3(j,1) = r3(j)/2;
    b3(j,1) = B3 + 0.5*(1-gamma)*L2(j)*sin(theta2o); % B3 is negative, so ...
       + the small change will make b3 smaller than B3
    rp(j,1) = sqrt(a3(j)^2 + b3(j)^2);
end

end
```

B.1.3 Kinematics

```
% RobertTheta2
% Name: Jeffrey Hawks
% Date: 20 Aug 2013 update: 27 Feb 2013
% Description: Roberts Straight Line analysis with Theta2 as the input

function [theta2, theta3, theta4, deltax, deltay] = RobertTheta2(theta2o, ...
    delta_theta2, r1, r2, r3, r4, a3, b3, N)

for i = 1:N

    %% Position Analysis - Four Bar
    % (See 'Four-Link Mechanism Analytical Positon Analysis' handout from Dr.
    % Brian Jensen)
    % Link Lengths
    % Robert's

    theta2(i,:) = theta2o:-0.1*pi/180:(theta2o-delta_theta2); % radians
    M = size(theta2,2); % The size of Theta2 for each analysis

    for j = 1:M

        % Open vs. Crossed Solution
        mu = -1; % Open Solution (mu = -1) or Crossed Solution (mu = 1)

        % Solve for vector r7
        r7x = r2(i)*cos(theta2(i,j)) - r1(i); % x component
        r7y = r2(i)*sin(theta2(i,j)); % y compnent
        theta7 = atan2(r7y,r7x); % angle
        r7 = sqrt(r7x^2 + r7y^2); % magnitude

        % Solve for psi
        psi = acos((r4(i)^2 + r7^2 - r3(i)^2)/(2*r4(i)*r7));
```

```
        % Solve for theta4
        theta4prime = theta7 + mu*psi; % This may be in the wrong quadrant
%         disp('Angle is: ')
%         disp(theta4prime)
        theta4(i,j) = atan2(sin(theta4prime), cos(theta4prime)); % radians

        % Solve for theta3
        r3x = r1(i) + r4(i)*cos(theta4(i,j)) - r2(i)*cos(theta2(i,j)); % x ...
            component
        r3y = r4(i)*sin(theta4(i,j)) - r2(i)*sin(theta2(i,j)); % y component
        theta3(i,j) = atan2(r3y,r3x); % radians


        % Solve for the coupler pt. location (accounting for the moving orgin)
        Px(i,j) = r2(i)*cos(theta2(i,j)) + a3(i)*cos(theta3(i,j)) - ...
            b3(i)*sin(theta3(i,j)); % Along the x axis
        Py(i,j) = r2(i)*sin(theta2(i,j)) + a3(i)*sin(theta3(i,j)) + ...
            b3(i)*cos(theta3(i,j)); % Along the y axis

        x0(i) = r2(i)*cos(theta2o) + a3(i); % Coupler position at the ...
            nuetral angle
        y0(i) = r2(i)*sin(theta2o) + b3(i); % Coupler position at the ...
            nuetral angle

        deltax(i,j) = Px(i,j) - x0(i);
        deltay(i,j) = Py(i,j) - y0(i);

    end

    % Plot Coupler Path
    plot(deltax(i,:)*1000,deltay(i,:)*1000)
    xlabel('P_x (mm)')
    ylabel('P_y (mm)')
    title('Roberts - Coupler Path')
    axis equal
    hold on
```

```
    grid on

%     figure(155)
%     P = polyfit(deltax(i,:),deltay(i,:),2);
%     plot(deltax(i,:)*1000,deltay(i,:)*1000)
%     axis normal
%     xlabel('P_x (mm)')
%     ylabel('P_y (mm)')
%     title('Roberts - Coupler Path')
%     hold on
%     grid on
% %     legend(num2str(P(1)))

    distance(i) = Px(i,1) - Px(i,end);

end

end
```

B.1.4 Force Analysis

```
%% Static Force Analysis
% Name: Jeffrey Hawks
% Date: 17 Sep 2013
% Description: This program uses Newton-Euler equation to analysis the
% force at each joint of the Robert Straightline Mechanism.  The analysis
% is performed statically (aka no inertial forces or torques are
% considered)

function [F12x,F12y,F32x,F32y,F43x,F43y,F14x,F14y,Fp] = ...
    StaticForce(Kstar,r1,r2,r3,r4,a3,b3,theta2c,theta3c,theta4c,theta2o)

%% Solve for the torques in  the psuedo springs and Fp

% Set up matrices that will be used
```

```
K = zeros(1, length(r1));
% Fp1 = zeros(length(r1), length(theta2c(1,:)));
T1 = zeros(length(r1), length(theta2c(1,:)));
T2 = zeros(length(r1), length(theta2c(1,:)));
T3 = zeros(length(r1), length(theta2c(1,:)));
T4 = zeros(length(r1), length(theta2c(1,:)));
Fp1 = zeros(length(r1), length(theta2c(1,:)));

% Set up initial theta values
theta3o = 0;
theta4o = pi - theta2o;

for i=1:length(r1) % This steps through the changes in the length of r2

    % The spring constant for the compliant beams (Nm/rad)
    K(i) = Kstar/r2(i);

    for j=1:length(theta2c(1,:)) % This steps through the motion of the ...
       coupler point for a given length of r2

        % Torques (Nm)
        T1(i,j) = -K(i)*(theta2c(i,j) - theta2o);
        T2(i,j) = -K(i)*((theta2c(i,j) - theta2o) - (theta3c(i,j) - theta3o));
        T3(i,j) = -K(i)*((theta4c(i,j) - theta4o) - (theta3c(i,j) - theta3o));
        T4(i,j) = -K(i)*(theta4c(i,j) - theta4o);

        % This section was a check on the force solved for at the coupler
        % pt. (Virtual Work method vs. Netwon-Euler Free Body Diagrams)
        % Kinematic Coeffecients
        h32 = (r2(i)*sin(theta4c(i,j) - ...
           theta2c(i,j)))/(r3(i)*sin(theta3c(i,j) - theta4c(i,j)));
        h42 = (r2(i)*sin(theta3c(i,j) - ...
           theta2c(i,j)))/(r4(i)*sin(theta3c(i,j) - theta4c(i,j)));

        % Force equation (N)
```

```
        Fp1(i,j) = (T1(i,j) + T2(i,j)*(1-h32) + T3(i,j)*(h42-h32) + ...
            T4(i,j)*h42)...
             /(r2(i)*sin(theta2c(i,j)) + a3(i)*sin(theta3c(i,j))*h32 + ...
                b3(i)*cos(theta3c(i,j))*h32);
    end
end

%% Newton-Euler (F = inv(A)*b)

% Set up Gravity
m2 = 0; % (kg) This may change, but for now the mass is considered negligent
m3 = 0; % (kg)
m4 = 0; % (kg)

g = 9.81; % Gravity (m/s^2)

% centroid for link 3 (Only needed if we decide to include the mass of the ...
    link)
c = 0;
psi = 0;

% Set up force matrices
F12x = zeros(length(r1),length(theta2c(1,:)));
F12y = zeros(length(r1),length(theta2c(1,:)));
F32x = zeros(length(r1),length(theta2c(1,:)));
F32y = zeros(length(r1),length(theta2c(1,:)));
F43x = zeros(length(r1),length(theta2c(1,:)));
F43y = zeros(length(r1),length(theta2c(1,:)));
F14x = zeros(length(r1),length(theta2c(1,:)));
F14y = zeros(length(r1),length(theta2c(1,:)));
Fp = zeros(length(r1),length(theta2c(1,:)));

for i=1:length(r1) % This steps through the changes in the length of r2
    for j=1:length(theta2c(1,:)) % This steps through the motion of the ...
        coupler point for a given length of r2
         % A*F = b
```

```
% Matrix A: The constants that are multiplied by the force matrix
A = [1, 0, 1, 0, 0, 0, 0, 0, 0;...
    0, 1, 0, 1, 0, 0, 0, 0, 0;...
    0, 0, -r2(i)*sin(theta2c(i,j)), r2(i)*cos(theta2c(i,j)), 0, 0, ...
       0, 0, 0;...
    0, 0, -1, 0, 1, 0, 0, 0, 1;...
    0, 0, 0, -1, 0, 1, 0, 0, 0;...
    0, 0, 0, 0, -r3(i)*sin(theta3c(i,j)), r3(i)*cos(theta3c(i,j)), ...
       0, 0, ...
       (a3(i)*sin(theta3c(i,j))+abs(b3(i))*cos(theta3c(i,j)));...
    0, 0, 0, 0, -1, 0, 1, 0, 0;...
    0, 0, 0, 0, 0, -1, 0, 1, 0;...
    0, 0, 0, 0, r4(i)*sin(theta4c(i,j)), -r4(i)*cos(theta4c(i,j)), ...
       0, 0, 0];
% Vector b: the known Torques and masses
b = [m2*g;...
    0;...
    -T1(i,j) - T2(i,j) - m2*g*r2(i)/2*sin(theta2c(i,j));...
    m3*g;...
    0;...
    T2(i,j) + T3(i,j) + m3*g*c*sin(theta3c(i,j)+psi);...
    m4*g;...
    0;...
    -T3(i,j) - T4(i,j) - m4*g*r4(i)/2*sin(theta4c(i,j))];

F = A\b;

% Split up the Force verter into its components and store.
F12x(i,j) = F(1);
F12y(i,j) = F(2);
F32x(i,j) = F(3);
F32y(i,j) = F(4);
F43x(i,j) = F(5);
F43y(i,j) = F(6);
F14x(i,j) = F(7);
F14y(i,j) = F(8);
```

```
        Fp(i,j) = F(9);
    end

end

end
```

B.1.5 Stress Analysis

```
%% Rotate the Forces
% Name: Jeffrey Hawks
% Date: 20 Nov 2013
% Description: Rotate the Forces on the base to be along the beam and
% perpendicular to the beam.

function [F12xp, F12yp] = Rotation(F12x,F12y,r1,r2,r3)

% Loop through the rows and columns of the F12 force to find the components
% at each spot

% Get the Diminsions for the Force Matrices
r = size(F12x,1);
c = size(F12x,2);
F12xp = zeros(r,c);
F12yp = zeros(r,c);

for i=1:r
    % Set up initial theta values
    y = (r1(i) -  r3(i))/2;
    theta2o = acos(y/r2(i));

    for j=1:c

        % Rotation matrix to line up the forces with the compliant beam
        R = [cos(theta2o), sin(theta2o), 0;...
```

```
            -sin(theta2o), cos(theta2o), 0;...
            0, 0, 1];
        % Mltipluy forces by Rotation Matrix
        F12 = [F12x(i,j); F12y(i,j);0];
        F12p = R*F12;
        F12xp(i,j) = F12p(1);
        F12yp(i,j) = F12p(2);
    end
end

end


%% Stress Analysis
% Name: Jeffrey Hawks
% Date: 7 Oct 2013
% Description: This function analysis the stress on link 2 based off of the
% given forces, dimensions, and angle of the fourbar mechanism.  The output
% is the stress, which will be minimized by optDim.m

function [sigmamax,sigmaMpa] = Stress(w, h, Ktheta, gamma, r1, r2, r3, r4, ...
    F32x, F32y, F43x, F43y, theta2c, theta2o)

%% Solve for the Stress
% Rotation matrix to line up the forces with the compliant beam
R = [cos(theta2o), sin(theta2o), 0;...
    -sin(theta2o), cos(theta2o), 0;...
    0, 0, 1];
for i=1:length(r2)
    for j=1:size(F32x,2)
        % Rotate the forces to be lined up with the compliant beam (links ...
           2 and 4)
        F32 = [F32x(i,j); F32y(i,j);0];
        F32p = R*F32;
        F32xp = F32p(1);
        F32yp = F32p(2);
```

```
        % Variables to solve for stress
        % Variables needed for the stress equation
        P = abs(F32yp); % Force that is perpendicular to the beams ...
            original orientation ((4*Ktheta*E*I*(theta2o - ...
            theta2c(1,1)))/(L^2*cos(theta2o - theta2c(1,1)));)
        n = -F32xp/P; % Force component that is along the beams original ...
            orientation
        L = r2(i)/gamma; % length of the compliant beam
        a = L*(1 - gamma*(1 - cos(theta2o - theta2c(i,j))));
        b = gamma*L*sin(theta2o - theta2c(i,j));
        c = h/2; % half of the thickness of the beam
        I = (w*h^3)/12; %area moment of inertia
        A = w*h; % Cross Sectional area of the beam

        % Stress EQ
        sigmaneg = -1*(P*(a + n*b)*c)/(2*I) - (n*P)/A; % Equation with I
        sigmapos = (P*(a + n*b)*c)/(2*I) - (n*P)/A;


        if (abs(sigmaneg) > abs(sigmapos))
            sigma(i,j) = abs(sigmaneg);
        else
            sigma(i,j) = abs(sigmapos);
        end
    end
end
Kt = 1.285; %Stress Concentration factor difference from my analytical to ...
    ansys Kt is between 1.24-1.33  (quick chart check Kt ~ 1.4)

sigmaMpa = Kt*sigma/(1e6);

%% Max Stress
sigmamax = Kt*max(max(sigma));

end
```

B.2 GUI Interface

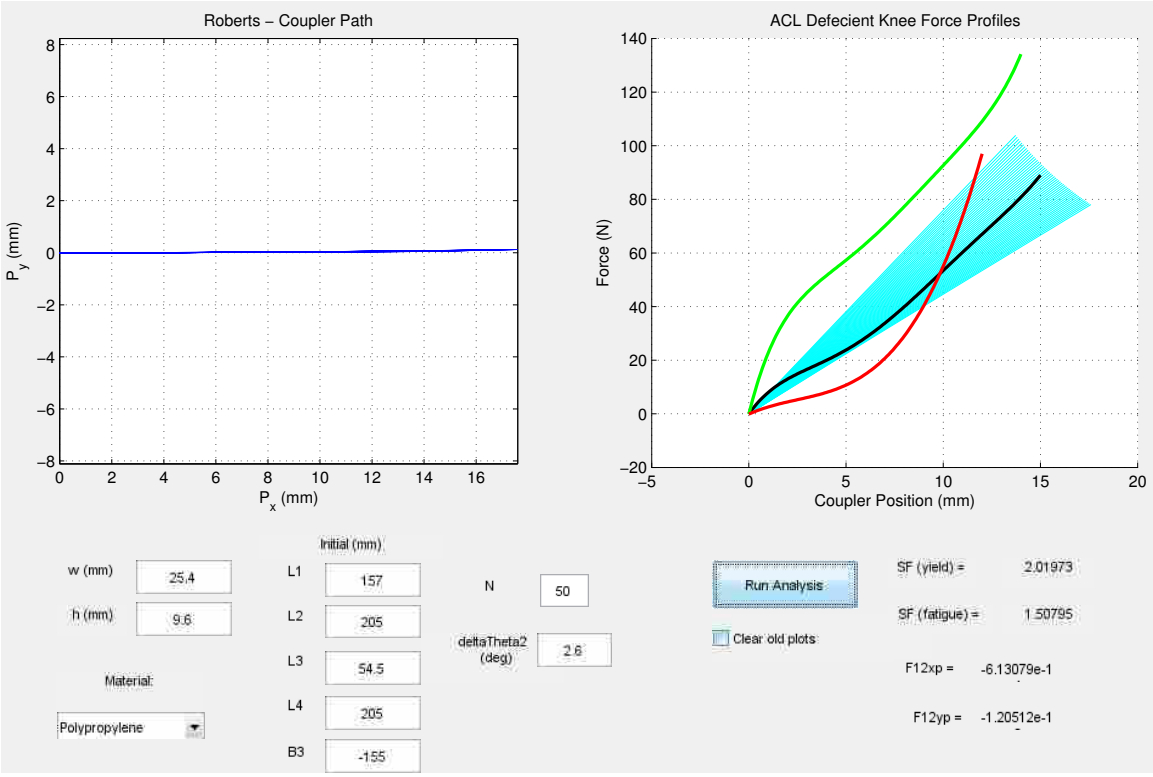


Figure B.1: Analysis GUI. This is a GUI interface that was used to change the dimensions of the compliant mechanism (length, width, etc) and be able to see the effect on the mechanism. The right graph shows the path of the coupler point. The right graph shows the force output of the compliant mechanism The red, black and green lines represent potential force profiles. The blue lines represent the force displacement combinations that the compliant mechanism can achieve. In this figure the blue lines surrounding the black line demonstrate that the given dimensions for a compliant mechanism can achieve the force profile given by the black line.

APPENDIX C. FINITE ELEMENT ANALYSIS

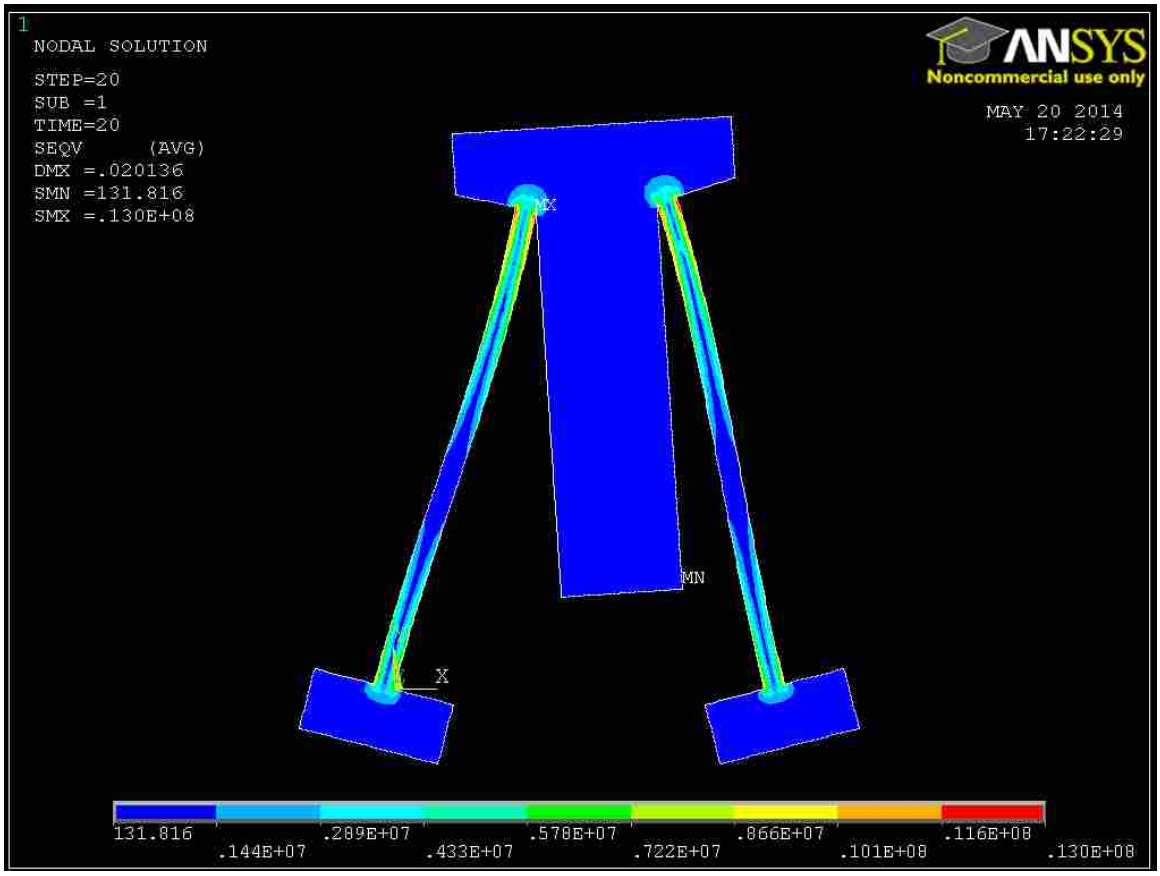


Figure C.1: FEA Image of Robert's Straight-line Mechansim

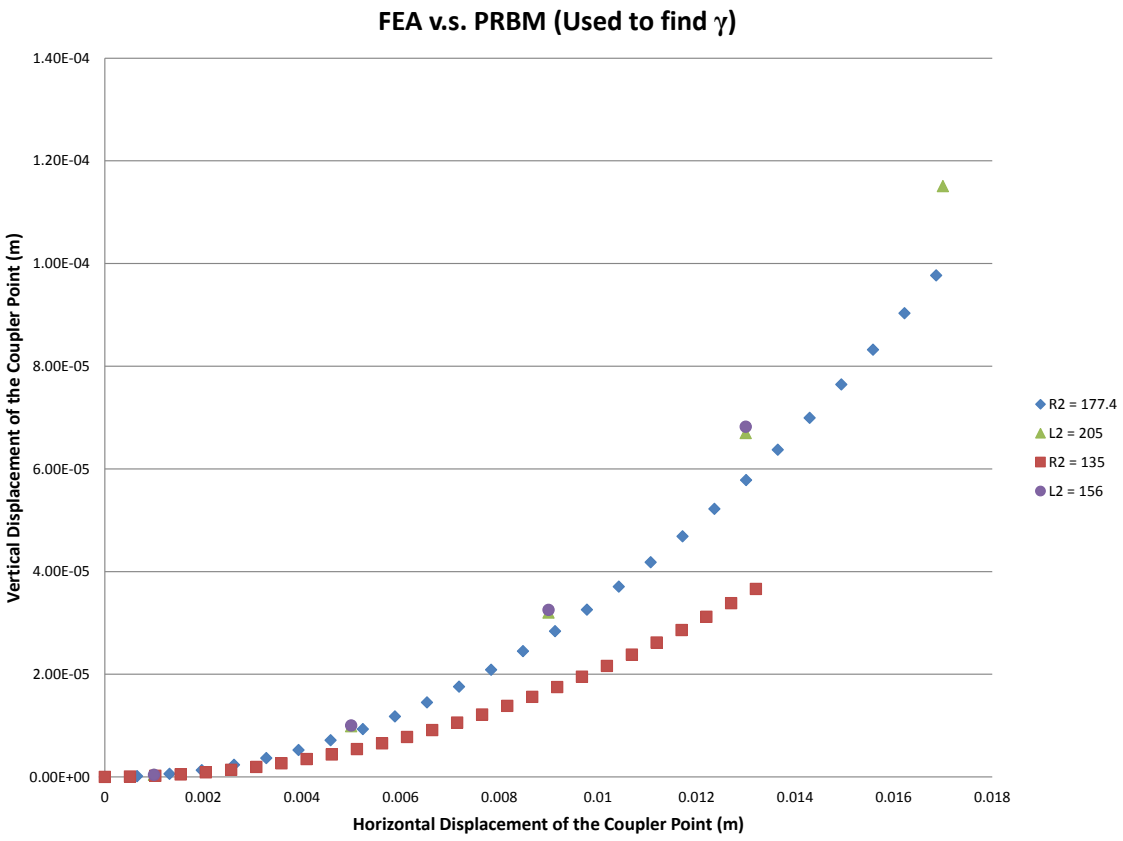


Figure C.2: Finding γ Graphic

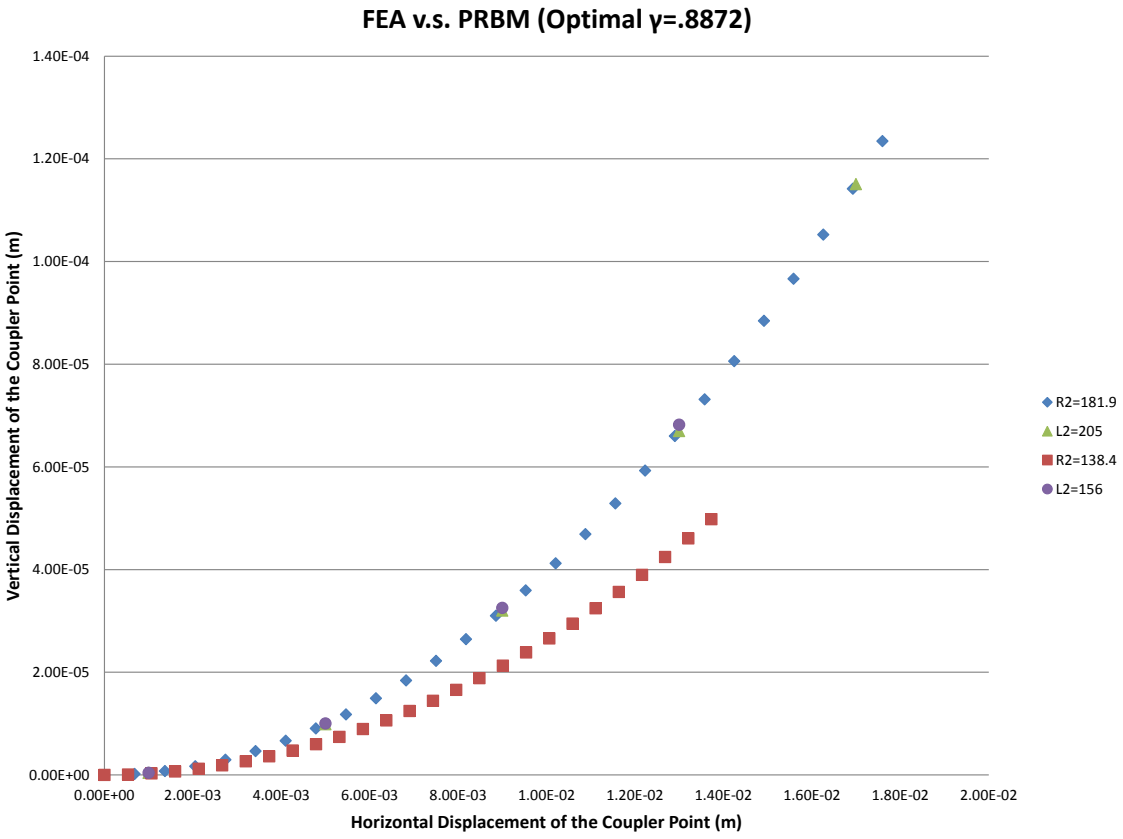


Figure C.3: Optimal γ Graphic

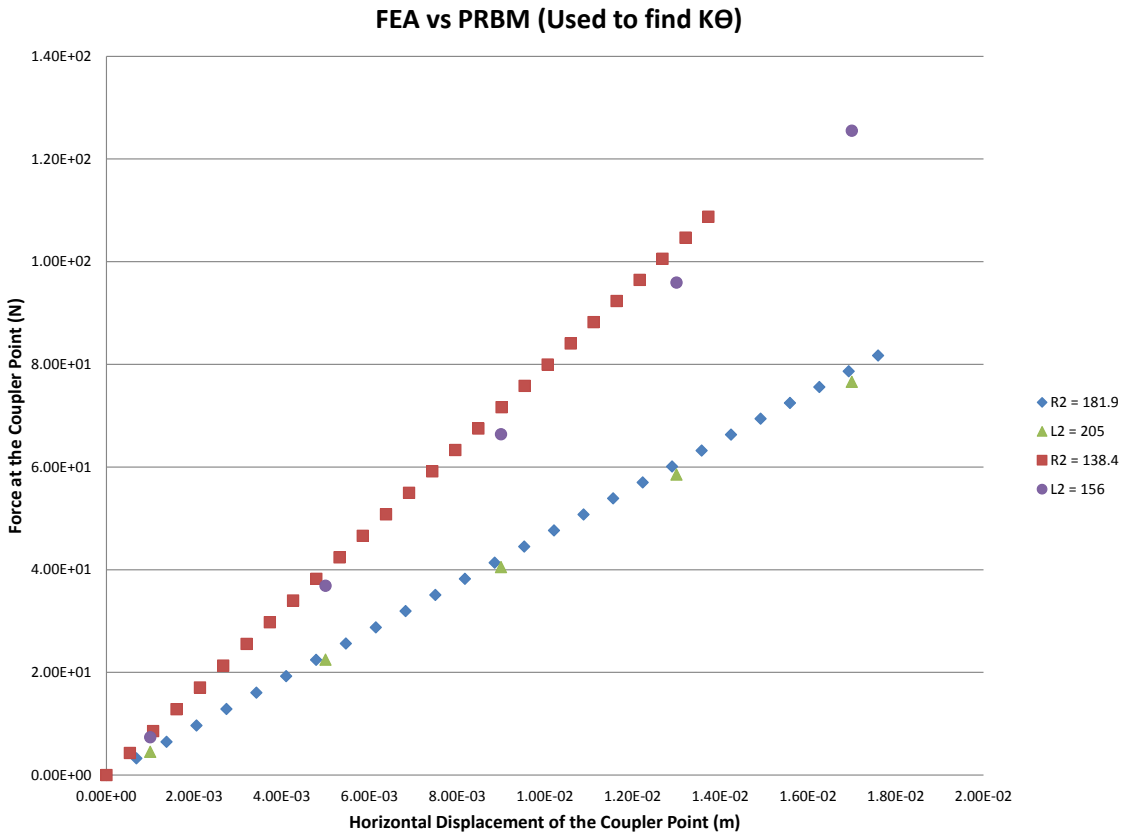


Figure C.4: Finding K_Θ Graphic

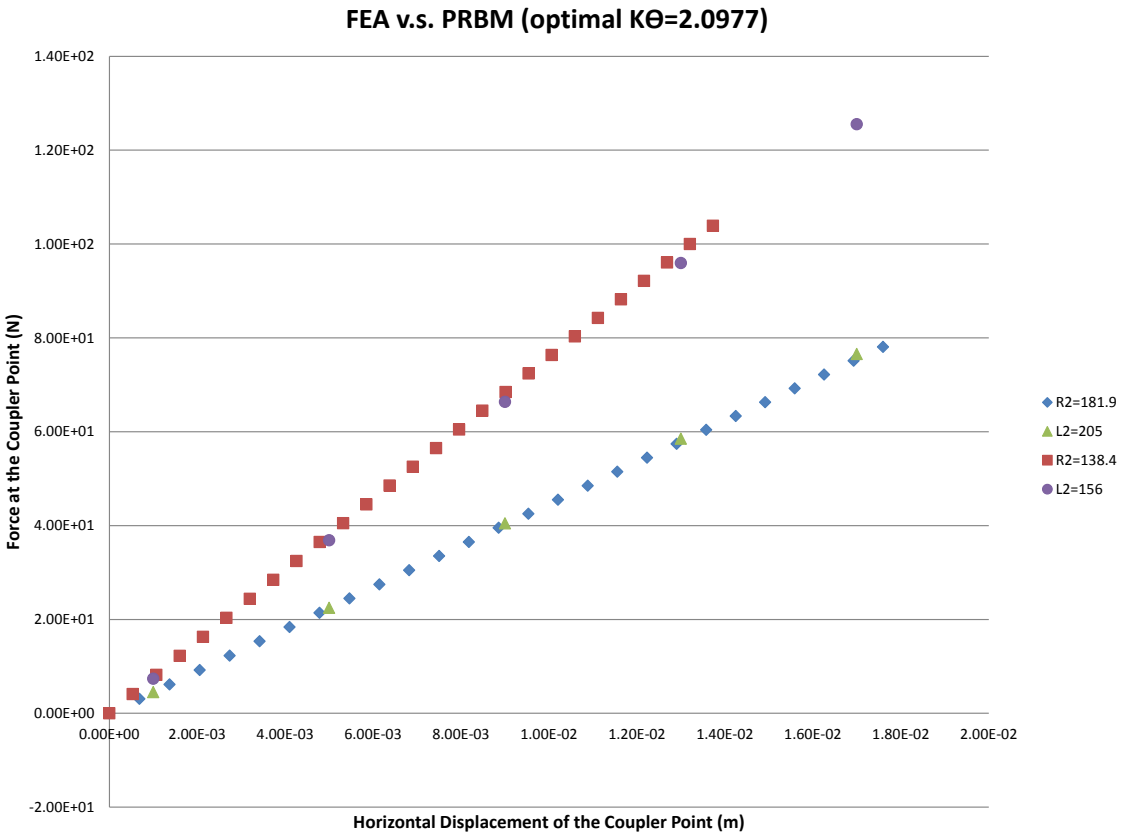
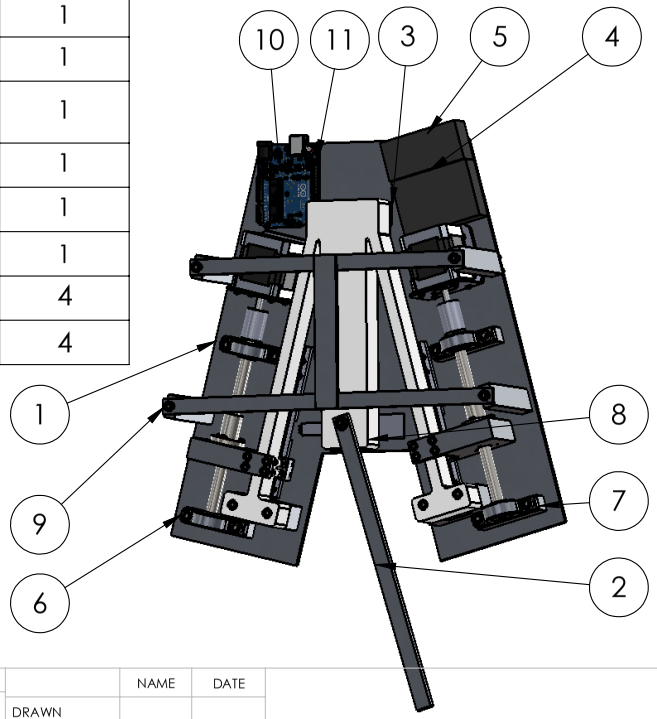


Figure C.5: Optimal K_Θ Graphic

APPENDIX D. DRAWING PACKAGE

ITEM NO.	Name	DESCRIPTION	QTY.
1	backplate		1
2	Coupler_assembly		1
3	SLM_assembly		1
4	Electronics_3AAcase	BYU EE Shop	1
5	Electronics_9vcase	Adafruit (#67)	1
6	Linear_assembly2		1
7	Linear_assemblyleft002		1
8	Coupler_linpot	SparkFun (#SEN-08680)	1
9	Handle_assembly		1
10	arduino uno	Adafruit (#50)	1
11	Pan 2-56 length 0.5	McMaster (#91735A021)	4
12	Nut 2-56	McMaster (#90480A003)	4



UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±

INTERPRET GEOMETRIC TOLERANCING PER:

MATERIAL

FINISH

DO NOT SCALE DRAWING

	NAME	DATE
DRAWN		
CHECKED		
ENG APPR.		
MFG APPR.		
Q.A.		

COMMENTS:

NEXT ASSY | USED ON

APPLICATION

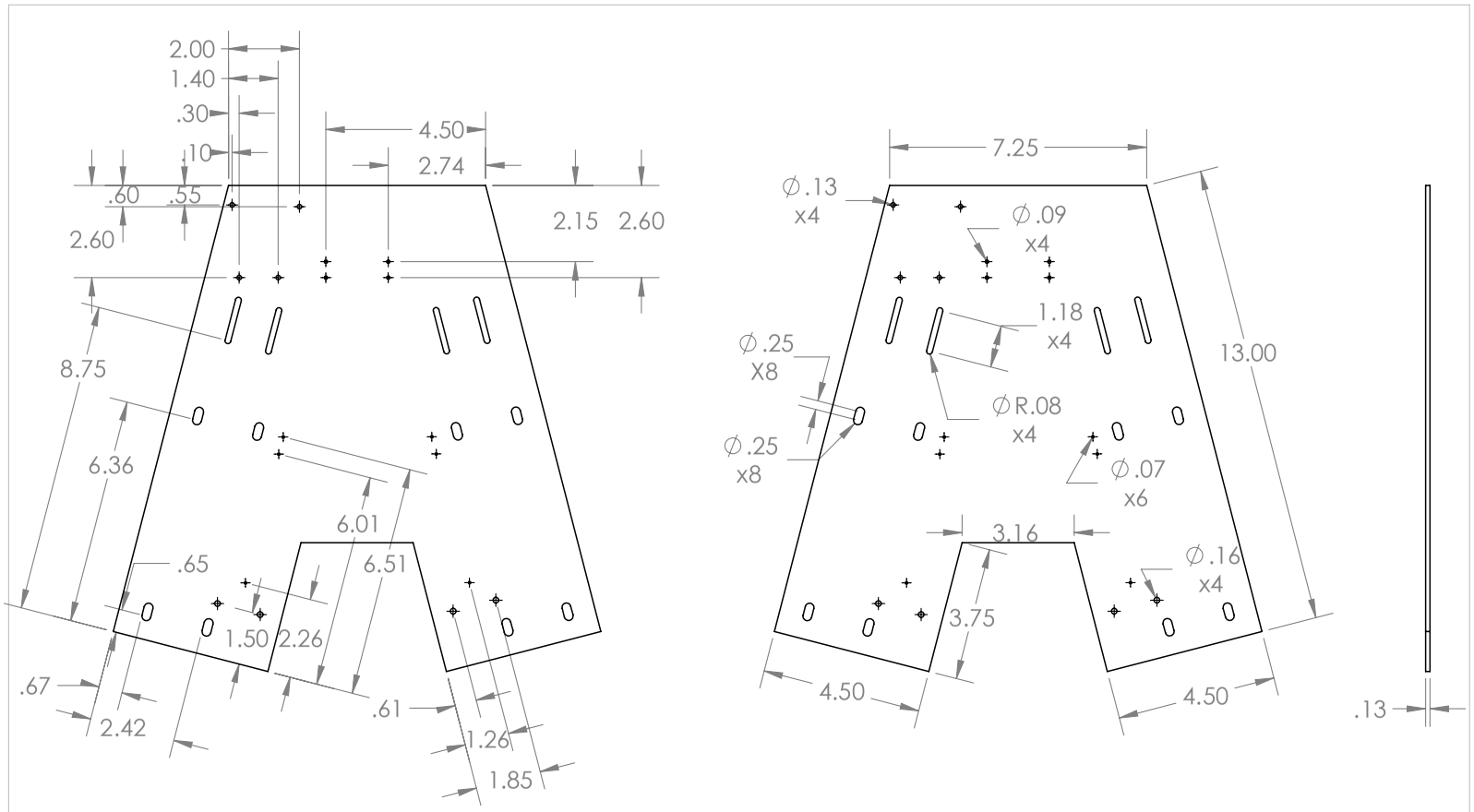
TITLE:

SIZE | DWG. NO. | REV

A _MainAssembly002

SCALE: 1:10 | WEIGHT: | SHEET 1 OF 1

5 4 3 2 1



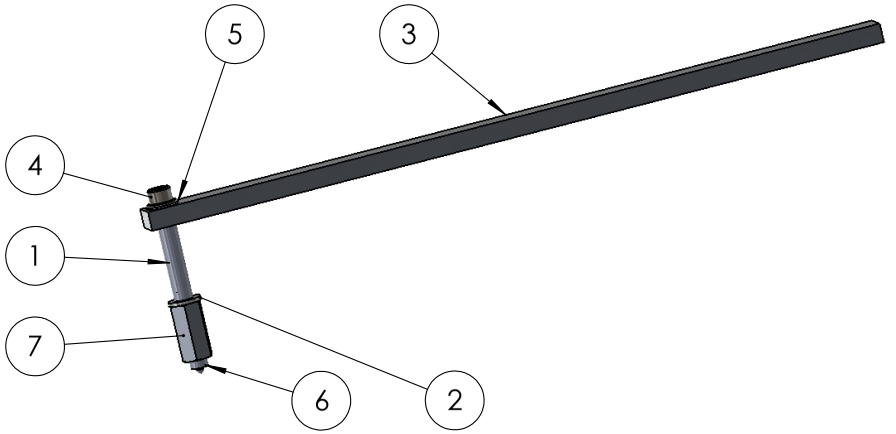
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

SIZE	DWG. NO.	REV
A	backplate	
SCALE: 1:4	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1

ITEM NO.	Name	DESCRIPTION	QTY.
1	Stand-Off	McMaster (#96110A044)	1
2	Coupler_washerb	McMaster (#93286A043)	1
3	Coupler_lowerleg		1
4	Screw 10_32	McMaster(#91251A344)	1
5	Coupler_washeru	McMaster(#90062A011)	1
6	Coupler_wiper	SparkFun(#SEN-09075)	1
7	Adapter_1032_6m	McMaster(#92499A379)	1

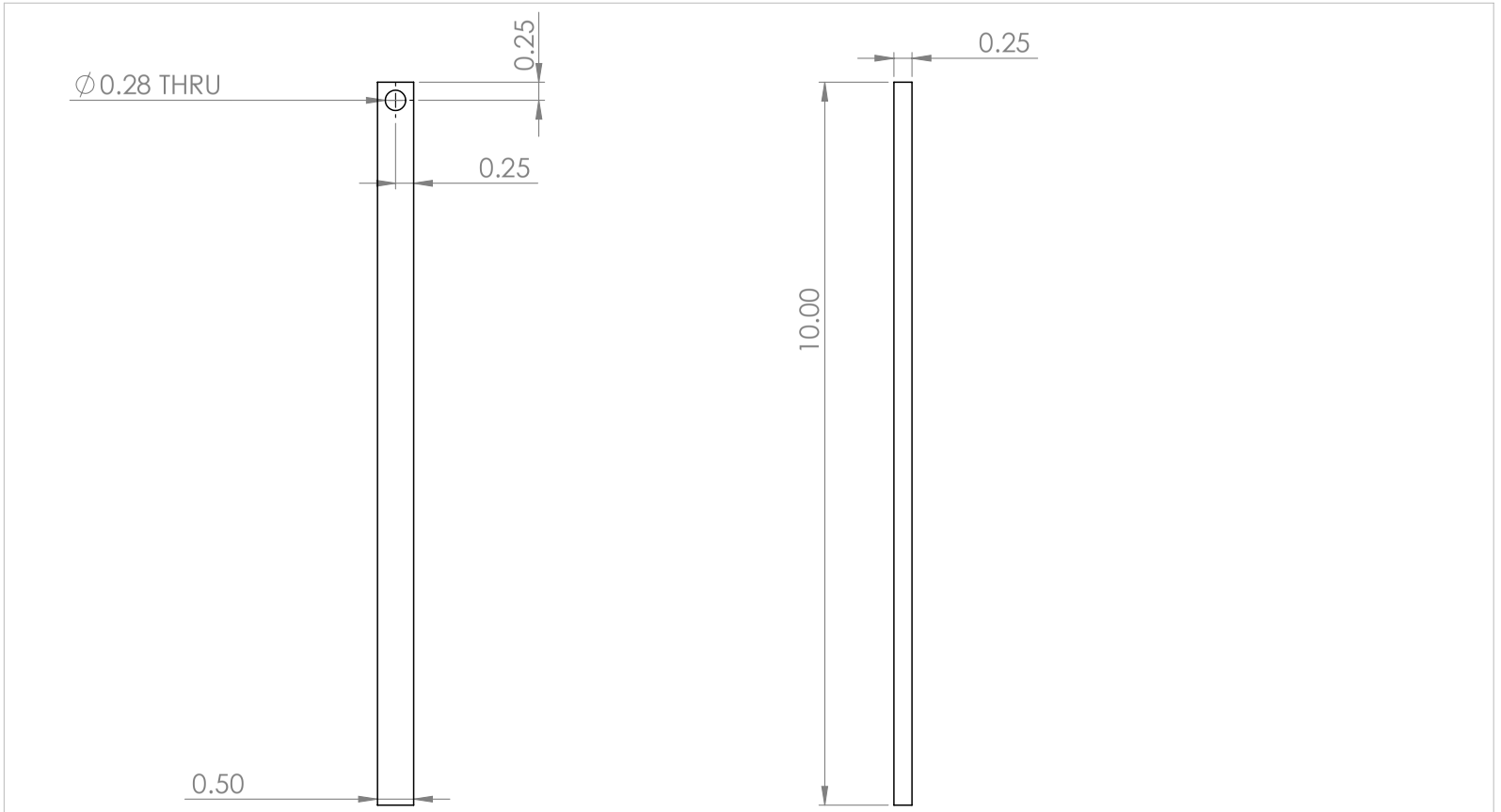


		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

SIZE	DWG. NO.	REV
A	Coupler_assembly	
SCALE: 1:5	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1



		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

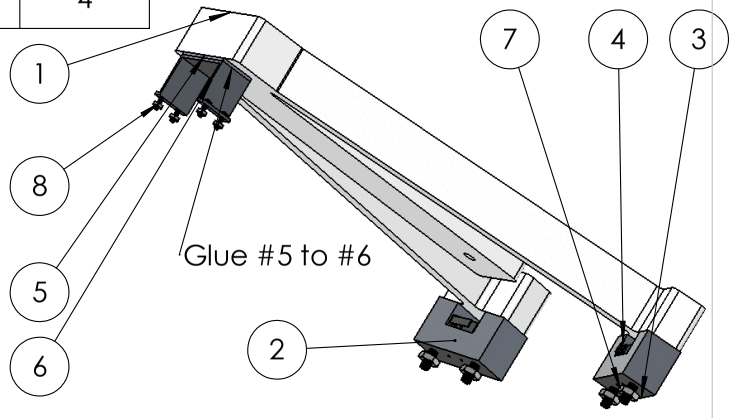
TITLE:

SIZE	DWG. NO.	REV
A	Coupler_lowerleg	
SCALE: 1:2	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1

ITEM NO.	Name	DESCRIPTION	QTY.
1	SLM_compliantpart	The key force feedback piece	1
2	SLM_baseright		1
3	SLM_base		1
4	SLM_pololuswitch	Pololu (#1402)	2
5	SLM_topbearing		1
6	SLM_topsupport		1
7	10_32 Socket Screw, Length 2.25	McMaster (#91251A354)	4
8	2_56 Pan Screw, length 0.3125	McMaster (#91772A078)	8
9	Nut 2-56	McMaster (#90480A003)	4
10	Nut 10-32	McMaster (#90545A111)	4

#8 is also the screw that connects the switch (#4) to the base (#2 and #3)



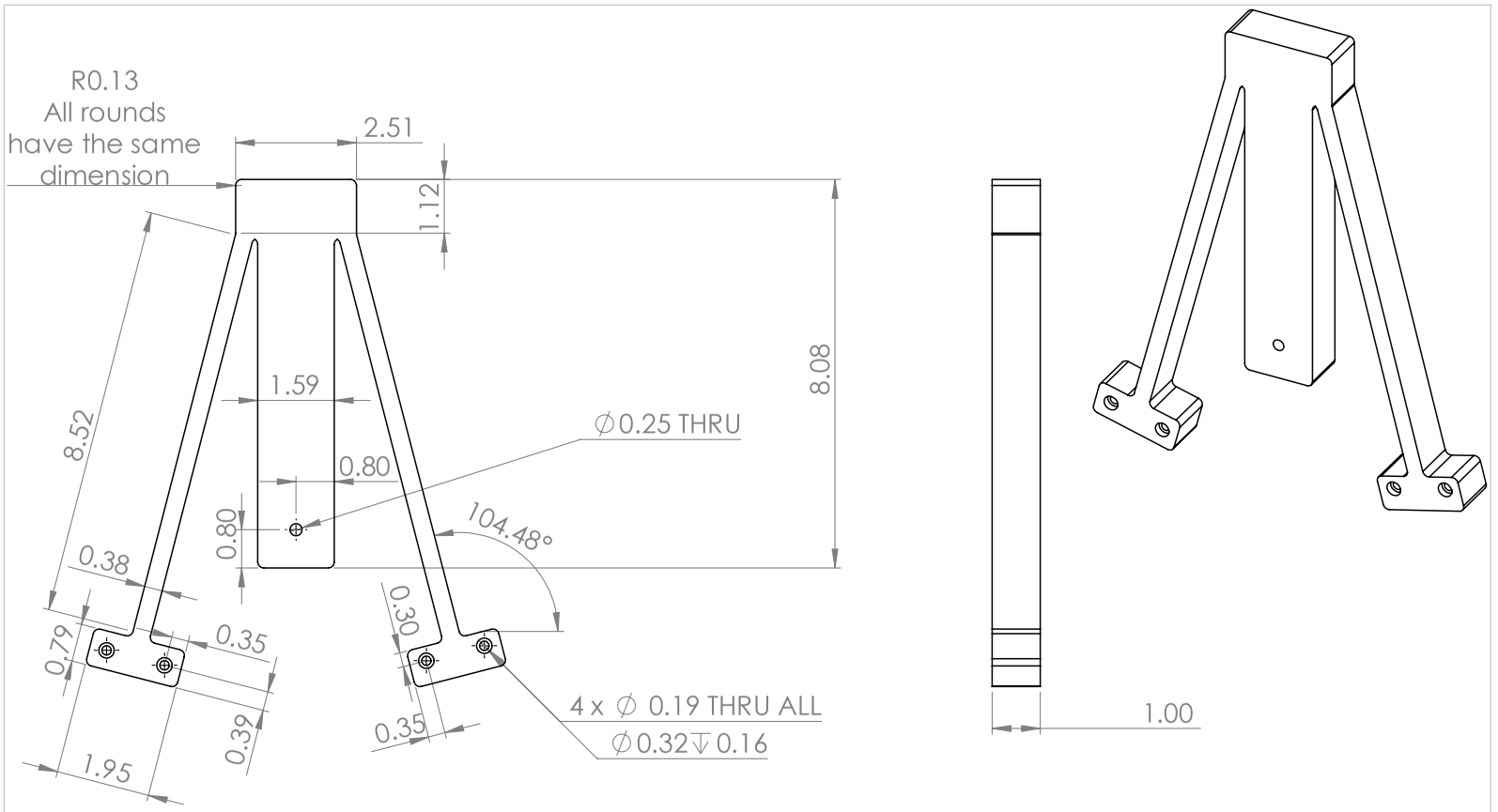
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

SIZE A | DWG. NO. SLM_assembly | REV

SCALE: 1:3 | WEIGHT: | SHEET 1 OF 1

5 4 3 2 1

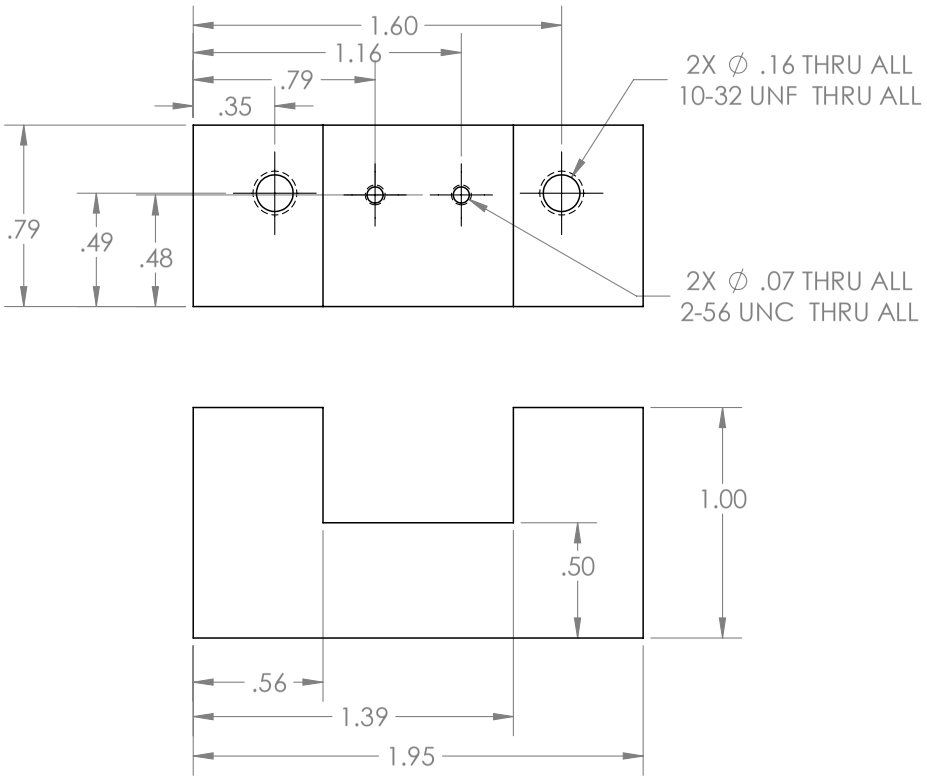


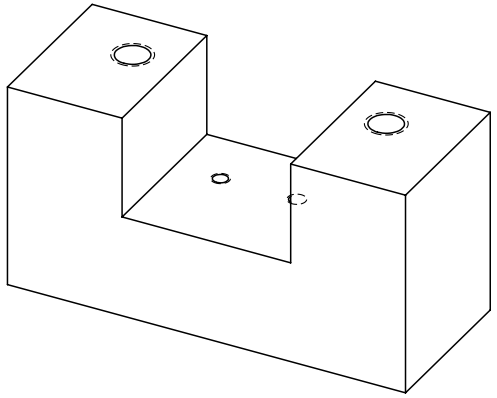
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

SIZE	DWG. NO.	REV
A	SLAM_compliantpart	
SCALE: 1:3	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1



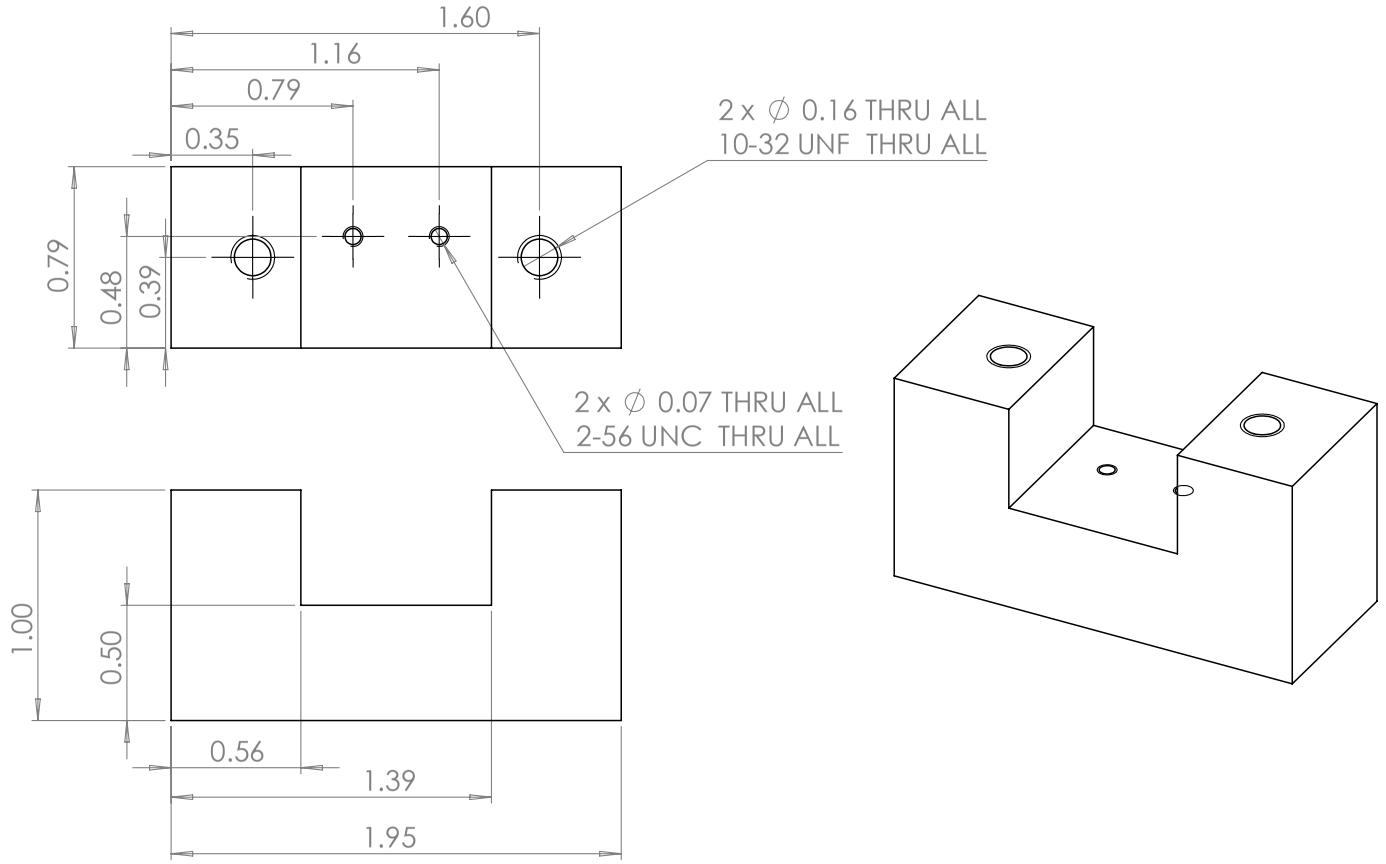


		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

SIZE	DWG. NO.	REV
A	ALM_baseright	
SCALE: 1.5:1	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1

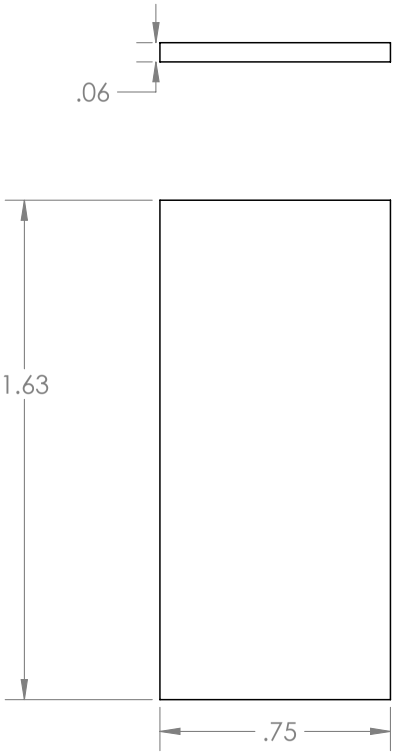


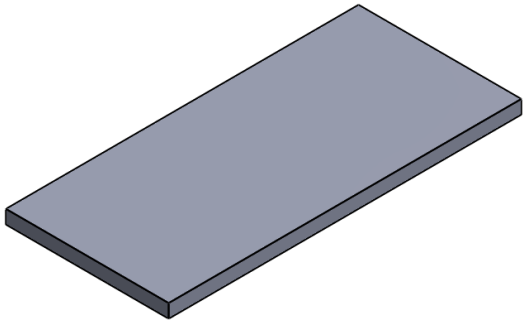
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

SIZE	DWG. NO.	REV
A	SLM_base	
SCALE: 1.5:1	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1



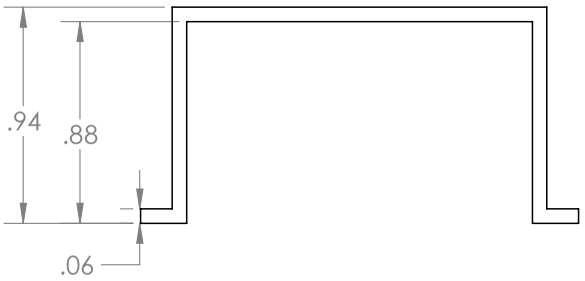


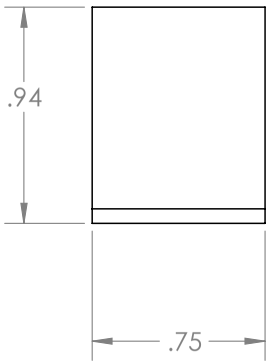
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

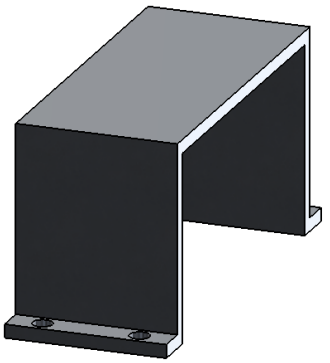
TITLE:

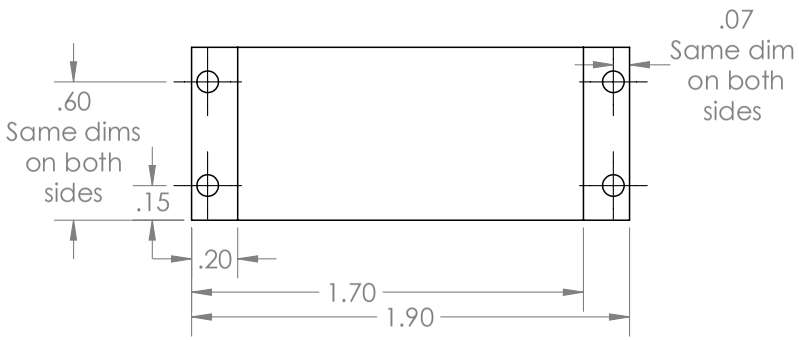
SIZE	DWG. NO.	REV
A	SAM_topbearing	
SCALE: 2:1	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1









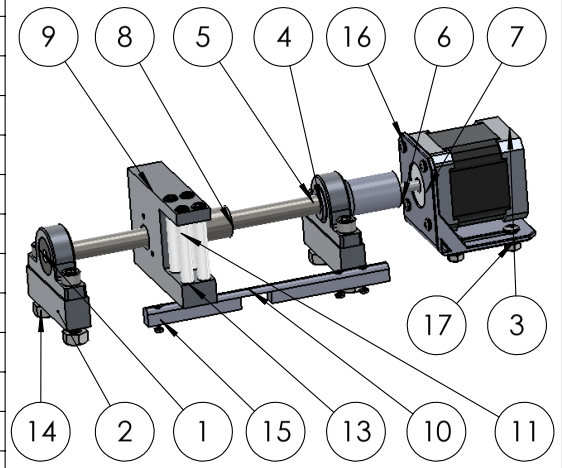
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

SIZE	DWG. NO.	REV
A	AM_topsupport	
SCALE: 1.5:1	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1

ITEM NO.	Name	DESCRIPTION	QTY.
1	Aluminum Bearing	McMaster (#5912K3)	2
2	Linear_bearingstand		2
3	NEMA 17 Stepper Motor Assembly (1)	Pololu (#1200)	1
4	Linear_leadscrew002	Roton (#61016)	1
5	Linear_thrustwasher	McMaster (#2797T1)	2
6	Linear_coupler	AdaFruit (#1175)	1
7	Linear_stepperbracket	AdaFruit (#1297)	1
8	Linear_nut	Roton (#91956)	1
9	Linear_slider002		1
10	Linear_nylonrail		1
11	Nylon Stand-Off	McMaster (#96110A039)	4
12	Pan 6_32 leng .75	McMaster (#91735A146)	3
13	Hex 8_32 leng 1.875	McMaster (#91251A188)	4
14	Hex 1/4-20 leng 1.25	McMaster (#91251A544)	4
15	Pan 0-80 leng 0.5	McMaster(#91772A059)	3
16	Pan M3 leng 8mm	McMaster (#90116A153)	4
17	Pan M4 leng 10mm	McMaster(#90116A207)	4
18	Nut 0-80	McMaster(#90480A001)	3
19	Nut 1/4 - 20	McMaster(#93827A211)	4
20	Nut M4	McMaster (#90592A011)	4
21	Nut 8-32	McMaster (#91125A359)	4



NEXT ASSY | USED ON

APPLICATION | DO NOT SCALE DRAWING

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±

INTERPRET GEOMETRIC TOLERANCING PER:

MATERIAL

FINISH

	NAME	DATE
DRAWN		
CHECKED		
ENG APPR.		
MFG APPR.		
Q.A.		

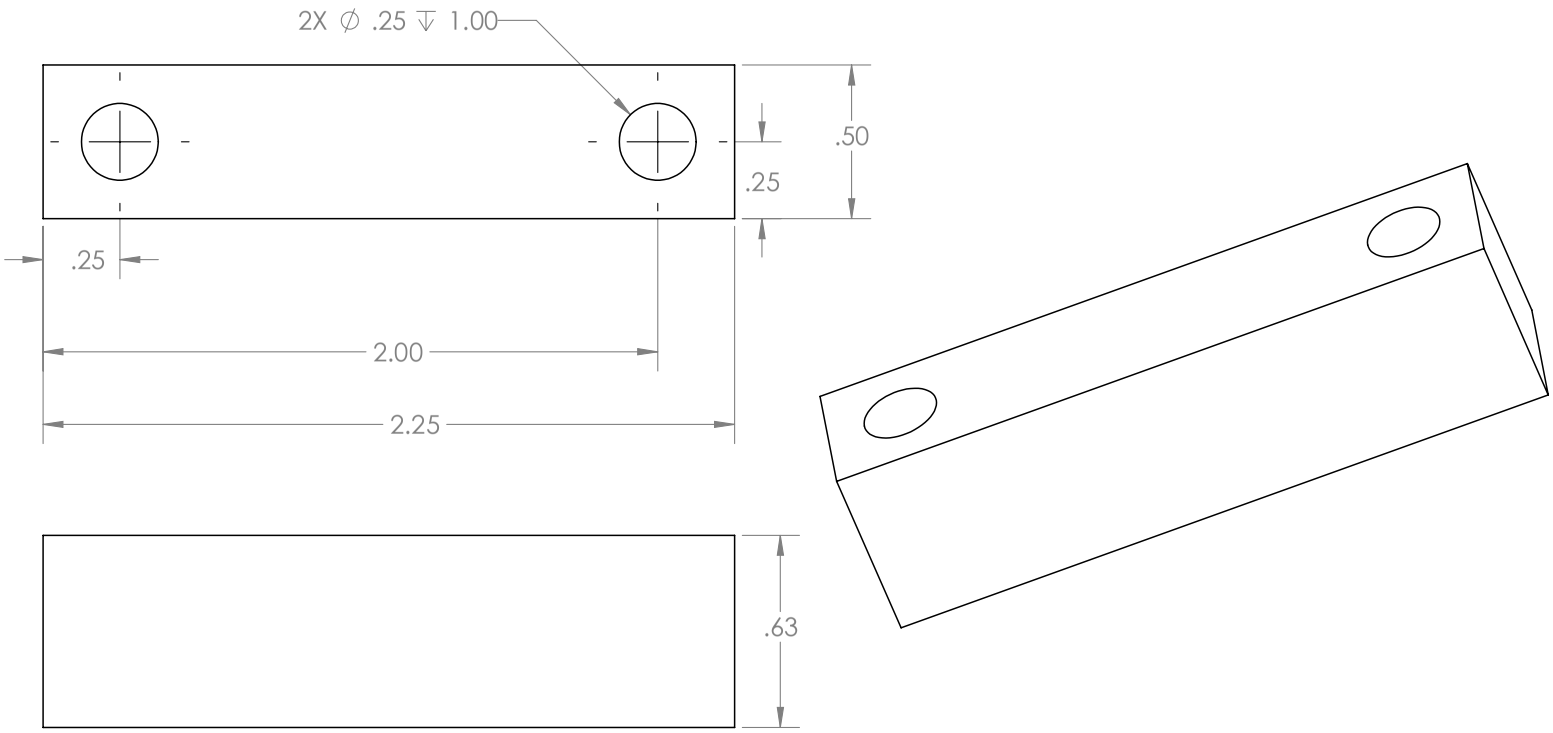
COMMENTS:

TITLE:

SIZE A | DWG. NO. Linear_assembly2 | REV

SCALE: 1:3 | WEIGHT: | SHEET 1 OF 1

5 4 3 2 1



		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

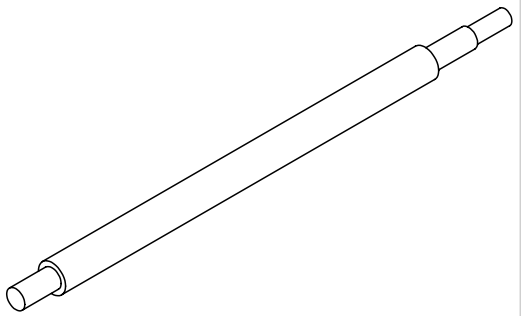
PROPRIETARY AND CONFIDENTIAL

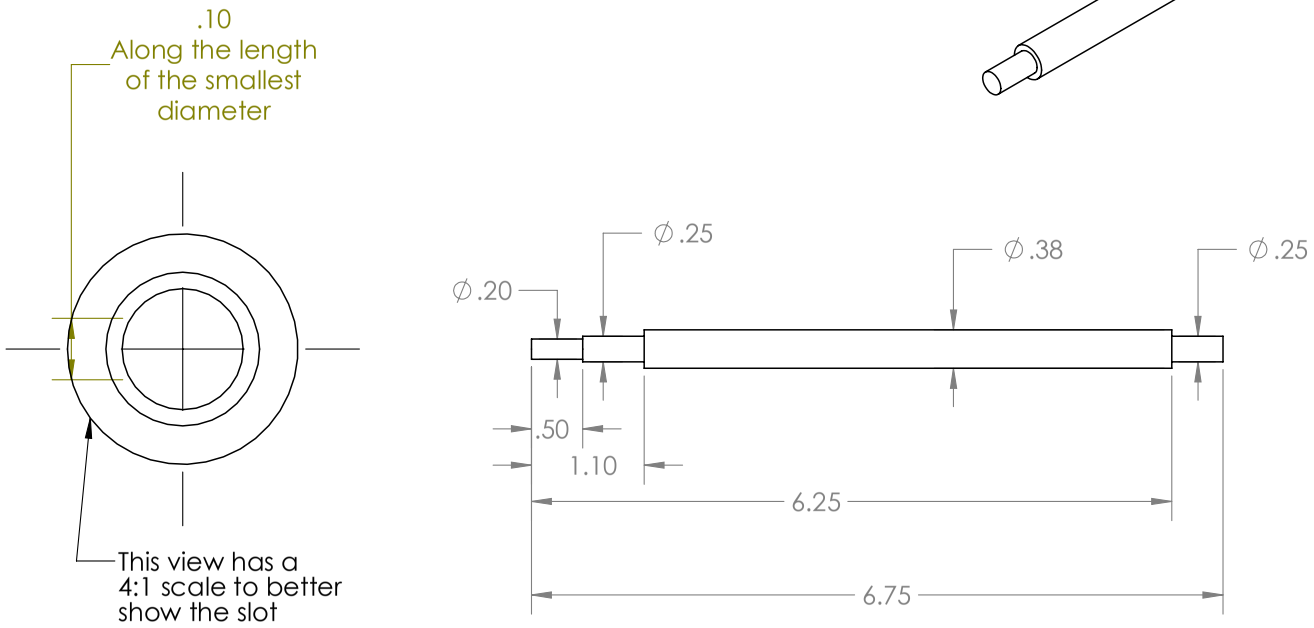
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.

TITLE:

SIZE	DWG. NO.	REV
A	Linear_bearingstand	
SCALE: 2:1	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1



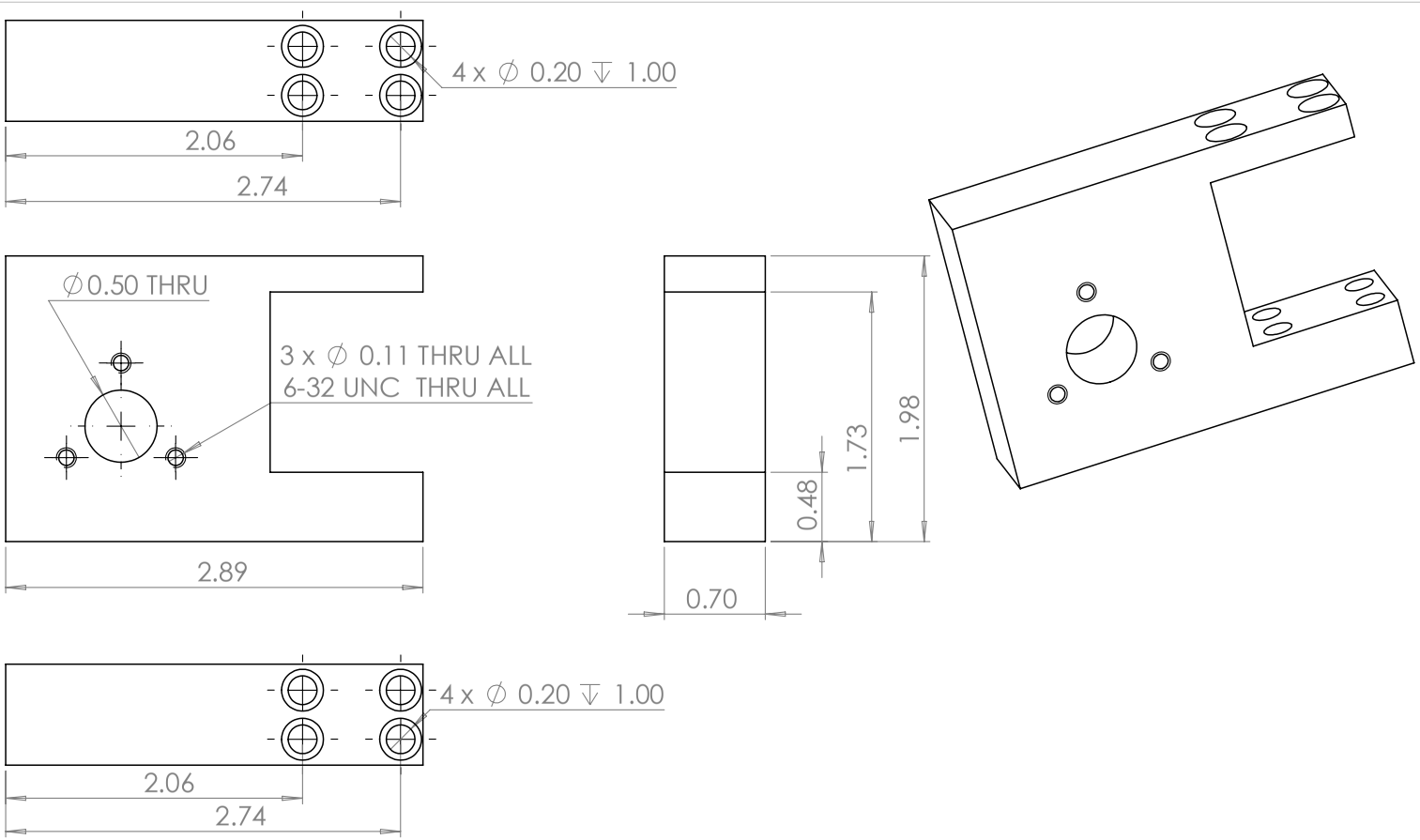


		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

SIZE	DWG. NO.	REV
A	Linear_leadscrew002	
SCALE: 1:1.5	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1

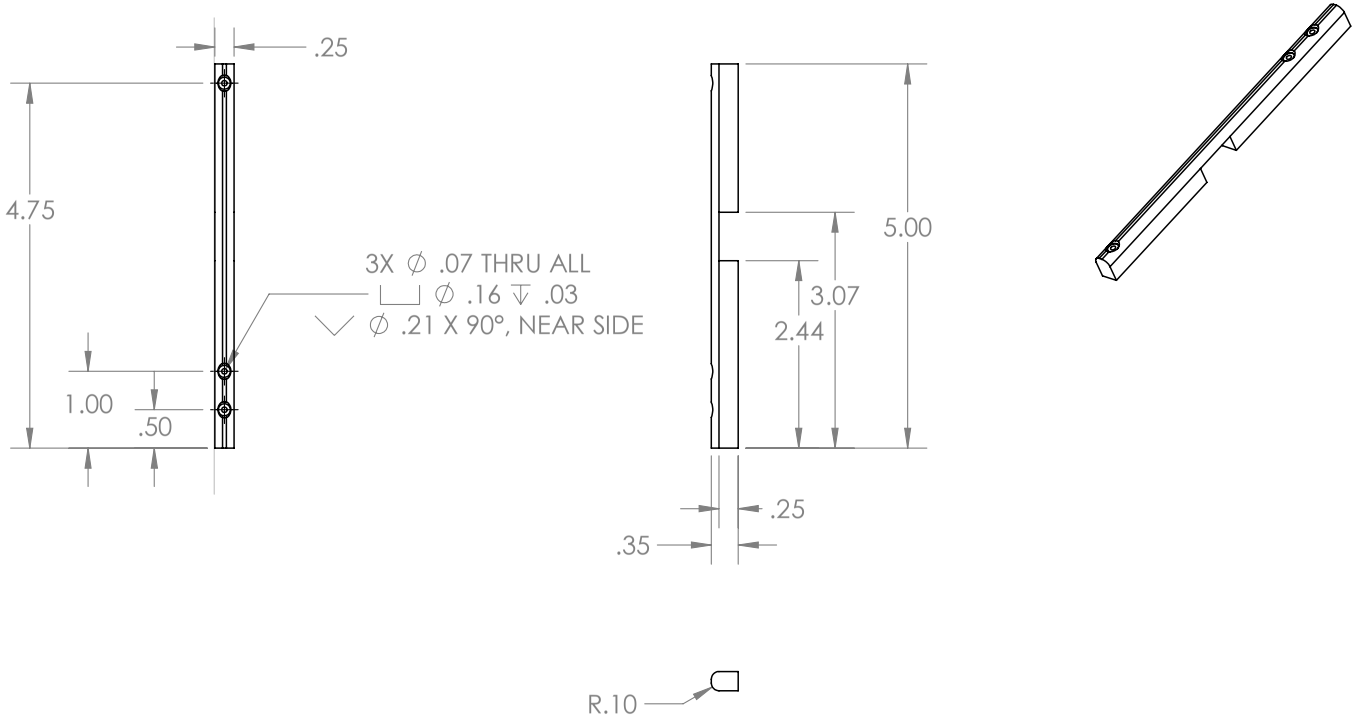


		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

SIZE	DWG. NO.	REV
A	Linear_slider002	
SCALE: 1:1	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1



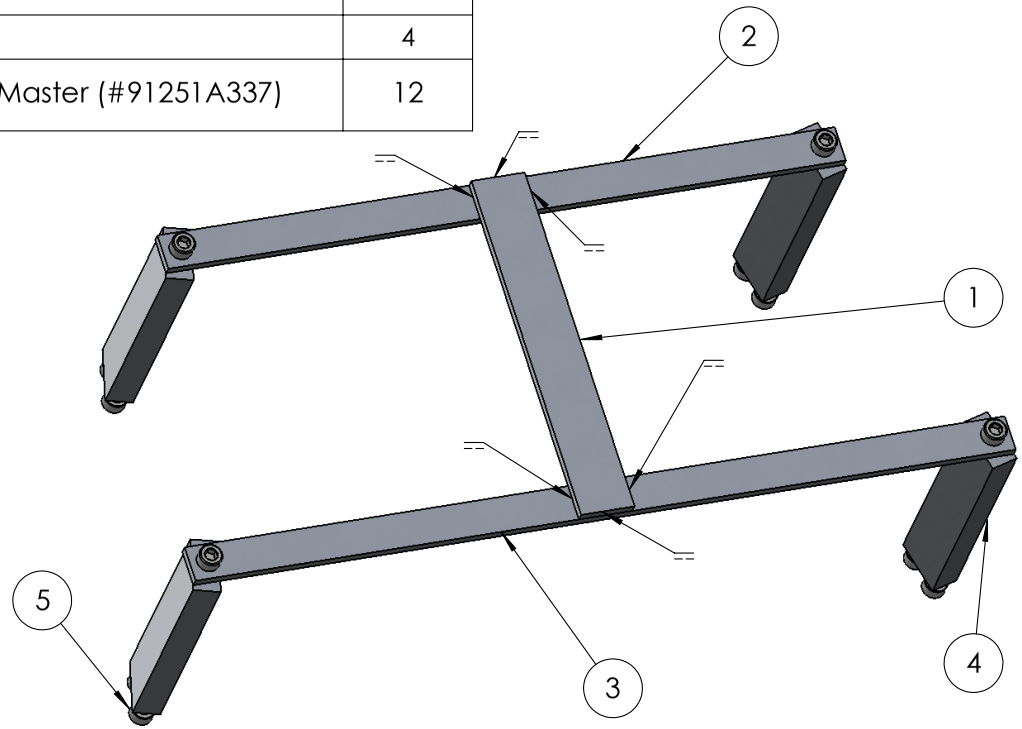
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

SIZE	DWG. NO.	REV
A	linear_nylonrail	
SCALE: 1:2	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1

ITEM NO.	Name	DESCRIPTION	QTY.
1	Handle_grip		1
2	Handle_lowersupport		1
3	Handle_uppersupport		1
4	Handle_stand		4
5	10_32 Socket Screw, Length 0.3125	McMaster (#91251A337)	12

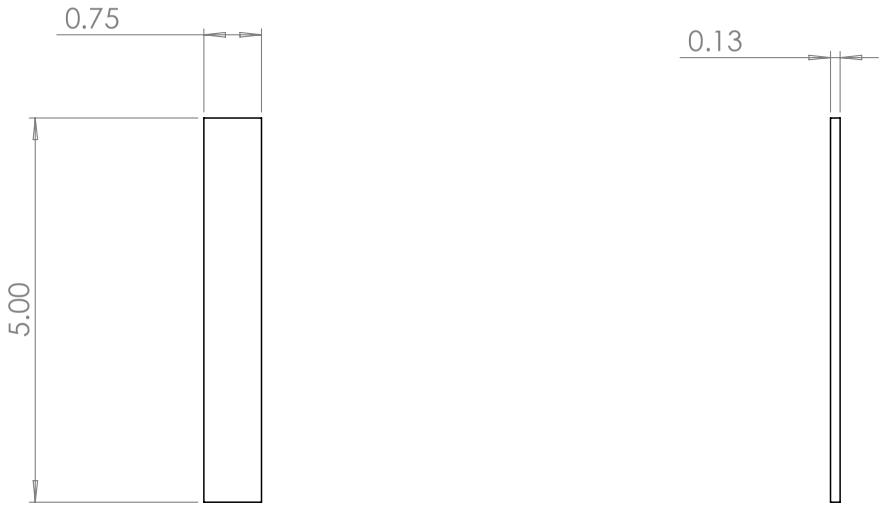


		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

SIZE	DWG. NO.	REV
A	Handle_assembly	
SCALE: 1:5	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1



		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

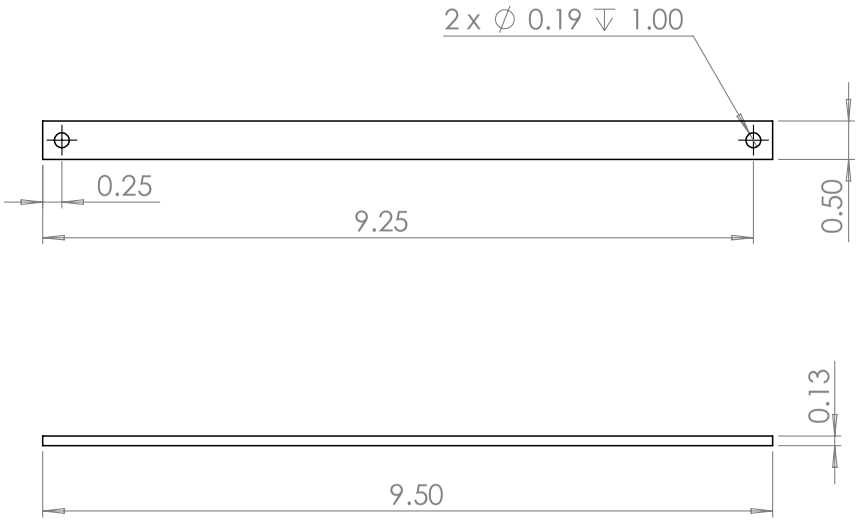
PROPRIETARY AND CONFIDENTIAL

THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.

TITLE:

SIZE	DWG. NO.	REV
A	Handle_grip	
SCALE: 1:2	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1



		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

PROPRIETARY AND CONFIDENTIAL

THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.

TITLE:

SIZE	DWG. NO.	REV
A	Handle_lowersupport	
SCALE: 1:2	WEIGHT:	SHEET 1 OF 1

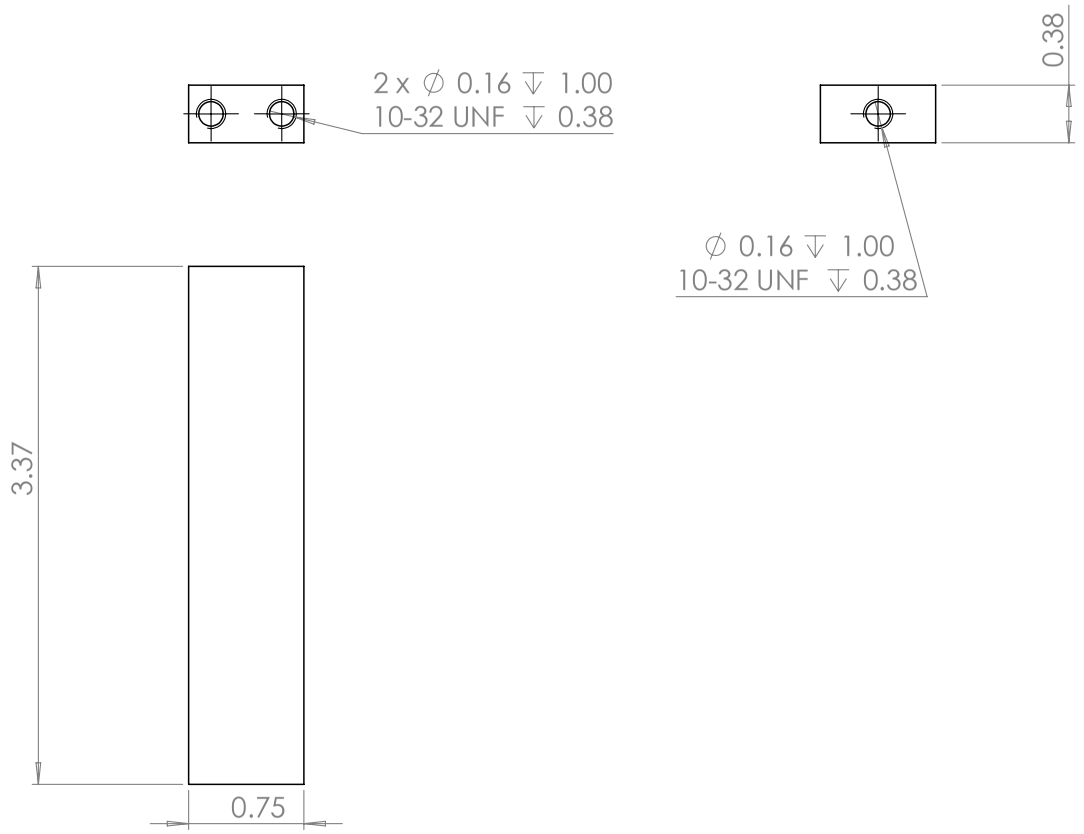
5 4 3 2 1

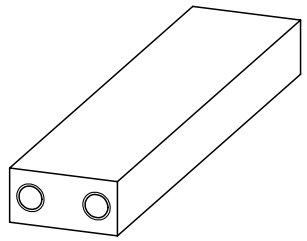
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

SIZE	DWG. NO.	REV
A	Handle_uppersupport	
SCALE: 1:2	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1





		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

SIZE	DWG. NO.	REV
A	Handle_stand	
SCALE: 1:1	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1

APPENDIX E. CONTROL PROGRAM

The code used to control the Haptic knee simulator was written for an Arduino micro controller, which uses a modified version of the C programming language. The code has been attached below to make it as easy as possible to recreate this work. Besides using the base Arduino library three other free libraries were used to create this code, and they are referenced on the Adafruit motor shield tutorial [39].

- Adafruit Motorshield: This library was needed to easily interface with the AdaFruit Motor shield that was used to driver and control the stepper motors
- AccelStepper: This library made it possible to drive the stepper motors simultaneously, so that links 2 and 4 ould change effective length at the same time
- Adafruit PWMServoDriver: This library allows the Adafruit motor shield to drive stepper motors

```
/////////////////////////// ////////////////////// ////////////////////////// ...
   ///////////////////
// HEADER

// Title: Haptic Knee Simulator (Injured ACL during Lachmans Test)
// Name: Jeffrey Hawks
// Date: 26 June 2014
//  Updated: 12 Aug 2014
// Description: This program runs the Haptic Knee simulator.  It is set up ...
   to simulate the
//  injuried knee during Lachman's Test.  Basically the program will ...
   incoorporate 2 stepper
```

```
//  motors, 2 switches, 1 linear potentiometer.  The switches will be used ...
   to zero the
//  position of the steppers. The linear pot will be used to measure the ...
   position of the
//  coupler pt. of the compliant mechanism. The servo motors will be ...
   driven based off of
//  the position of the coupler pt. to produce a predefined force ...
   deflection curve

////////////////////////// ////////////////////// ////////////////////////// ...
   //////////////////
// LIBRARIES to include

#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_PWMServoDriver.h"
#include <AccelStepper.h>

////////////////////////// ////////////////////// ////////////////////////// ...
   //////////////////
// MOTOR SETUP (these need to be global variables)

// Create the motor shield object
Adafruit_MotorShield AFMS = Adafruit_MotorShield();

// Create object pointers to the two stepper motors
Adafruit_StepperMotor *motor_L = AFMS.getStepper(200, 1);
Adafruit_StepperMotor *motor_R = AFMS.getStepper(200, 2);

// Set up AccelStepper object (This allows the motors to run simultaneously)
// motor L
void forwardstep1() {
  motor_L->onestep(FORWARD, DOUBLE); //check single for max speed
}
void backwardstep1() {
  motor_L->onestep(BACKWARD, DOUBLE);
```

```
}
// motor R
void forwardstep2() {
  motor_R->onestep(FORWARD, DOUBLE);
}
void backwardstep2() {
  motor_R->onestep(BACKWARD, DOUBLE);
}

// Put motors in AccelStepper objects
AccelStepper stepper_L(forwardstep1, backwardstep1);
AccelStepper stepper_R(forwardstep2, backwardstep2);

////////////////////////// ////////////////////// ////////////////////////// ...
   //////////////////
// VARIABLES

// LED variables
int LED_READY = 6; //Output LED pin
int LED_ON = 7; //Output LED pin to show when the system is powered up

// Switch variables
int switch_R = 2; // Input right switch pin
int switch_L = 3; // Input left switch pin
int val_switch_L = 0; // Read value of left switch (high = pressed)
int val_switch_R = 0; // Read value of right switch (high = pressed)

// Linear Potentiometer variables
int LinPot = 0; // Input pin for the linear potentiometer
volatile int val_LinPot; // Value being read in from the linear potentiometer

// Variables to calculate motion of the steppers
// Important note:
    // Measurements for the couplers position (x) and length of the ...
       compliant link (L) will
```

```
    // be whole numbers representing 10^-4 meters.  This allows the system ...
       to be accurate
    // within 0.1 mm, and still use integers to help speed up the ...
       computing time.
volatile int x = 0; // The position of the Coupler in 10^-4 meters
volatile int L = 0; // The length of L2 (the compliant arm) in 10^-4 meters
volatile int pos = 0; // The position of the slider based off of the ...
   longest length of the compliant arm being the zero position
const int lb = 1550; // This is the lower bound, 150 mm
const int ub = 2050; // This is the upper bound, 205 mm
int mid;// The mid point value will be established in the IO_Setup function

// Table of variables used to determine the desired force profile
const int dim = 200;
// Table of Coupler Positions, matches with L_t to give a desired force ...
   profile
const int x_t[dim] = { 1,2,3,4,5, ...
   6,7,8,9,10,11,12,13,14,15,16,17,18,19,20, ...
   21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40, ...
   41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60, ...
   61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80, ...
   81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100, ...
   101,102,103,104,105,106,107,108,109,110,111,112,113,114,115, ...
   116,117,118,119,120,121,122,123,124,125,126,127,128,129,130, ...
   131,132,133,134,135,136,137,138,139,140,141,142,143,144,145, ...
   146,147,148,149,150,151,152,153,154,155,156,157,158,159,160, ...
   161,162,163,164,165,166,167,168,169,170,171,172,173,174,175, ...
   176,177,178,179,180,181,182,183,184,185,186,187,188,189,190, ...
   191,192,193,194,195,196,197,198,199,200};
// Table of Link Lengths, matches with x_t to give a desired force profile
const int L_t[dim] = {1398, 1404, 1417, 1432, 1448, 1464, 1480, 1497, ...
   1513, 1530, 1547, 1559, 1576, 1593, 1610, 1627,   1644,   1665,   ...
   1680,   1692,   1712,   1723,   1737,   1756,   1771,   1786,   ...
   1800,   1814,   1827,   1840,   1853,   1865,   1877,   1882,   ...
   1893,   1903,   1912,   1921,   1929,   1937,   1944,   1951,   ...
   1957,   1963,   1968,   1973,   1977,   1981,   1984,   1987,   ...
```

```
    1990,   1992,   1993,   1994,   1995,   1995,   1995,   1995,   ...
    1995,   1994,   1993,   1991,   1989,   1987,   1985,   1983,   ...
    1980,   1978,   1975,   1972,   1969,   1966,   1962,   1959,   ...
    1956,   1952,   1948,   1945,   1941,   1938,   1934,   1930,   ...
    1927,   1923,   1919,   1916,   1912,   1909,   1905,   1902,   1898,
1895,   1892,   1889,   1886,   1882,   1879,   1882,   1879,   1876,   ...
    1873,   1870,   1868,   1865,   1863,   1860,   1858,   1856,   ...
    1853,   1851,   1849,   1847,   1845,   1843,   1842,   1840,   ...
    1838,   1836,   1835,   1833,   1832,   1830,   1829,   1827,   ...
    1826,   1824,   1823,   1821,   1820,   1819,   1817,   1816,   ...
    1814,   1812,   1811,   1809,   1807,   1806,   1804,   1802,   ...
    1800,   1798,   1795,   1793,   1791,   1788,   1785,   1782,   ...
    1779,   1776,   1773,   1769,   1765,   1761,   1757,   1753,   ...
    1748,   1739,   1734,   1729,   1725,   1719,   1714,   1712,   ...
    1704,   1695,   1686,   1684,   1677,   1670,   1663,   1650,   ...
    1641,   1633,   1623,   1614,   1605,   1595,   1585,   1574,   1564,
1558,   1548,   1537,   1526,   1515,   1504,   1492,   1481,   1469,   ...
    1458,   1446,   1434,   1422,   1410,   1398,   1385,   1373,   ...
    1361,   1348};

int last = dim - 1; // Used to refer to the last item in the array

////////////////////////// /////////////////////// ////////////////////////// ...
    ///////////////////
// FUNCTIONS

// Establish the input/ouput pins that will be used
void IO_Setup()
{
  // Set up the input and output pins (digital)
  pinMode(LED_READY,OUTPUT);
  pinMode(LED_ON, OUTPUT);
  pinMode(switch_L, INPUT);
  pinMode(switch_R, INPUT);

  // Set up the input and output pins (analog)
```

```
  pinMode(LinPot, INPUT);

  // LED Control
  digitalWrite(LED_READY, LOW); // Turn LED off to indicate that the ...
     system is not ready to use
  digitalWrite(LED_ON, HIGH); // Turn LED on to indicate that the system ...
     is powered

  //Take the current reading from the linear potentiomenter and set as the ...
     mid point
  int sample = 0;
  for(int i = 0;i<10; i++)
  {
    sample = sample + analogRead(LinPot); // The nuetral position of the ...
       coupler in counts of the linear potentiometer
  }
  mid = sample/10;
}

// Establish the default speeds and accelerations for the motors
void Motor_Setup()
{
  AFMS.begin();  // create with the default frequency 1.6KHz (Connects the ...
     motorsheild)
  // Default values for the motors
  stepper_L.setMaxSpeed(100.0); // This is in steps  per second, and is at ...
     the recommended max reliable speed
  stepper_L.setAcceleration(10000.0); //

  stepper_R.setMaxSpeed(100.0);
  stepper_R.setAcceleration(10000.0); //
}

// Set the motors zero position to match the longest link length L2
void Motor_Zero()
{
```

```
// Move the motors one at a time until the switches are pressed ...
    (individually)
// Move the motors one at a time until the switches are pressed ...
    (individually)
stepper_L.setMaxSpeed(20); //Nice and slow
stepper_R.setMaxSpeed(20); //Nice and slow

//Left
val_switch_L = digitalRead(switch_L); // Check the state of the switch
if (val_switch_L ==1)
{
  while(val_switch_L==1)
  {
    //Move the motor back a little to get off of the switch
    Linear_Motion(-1,1); // Need to be the opposite sign of the original ...
        motion
    val_switch_L = digitalRead(switch_L); // Recheck the state of the switch
  }
  stepper_L.setCurrentPosition(0);
}
else
{
  while (val_switch_L !=1)
  {
    // Check sign on here to make sure we are moving in the correct ...
        direction
    Linear_Motion(1,1); //This is equivalent to moving the left motor ...
        0.1 mm at a time
    val_switch_L = digitalRead(switch_L); // Recheck the state of the switch
  }
  delay(500);
  while(val_switch_L==1)
  {
    //Move the motor back a little to get off of the switch
    Linear_Motion(-1,1); // Need to be the opposite sign of the original ...
        motion
```

```
    val_switch_L = digitalRead(switch_L); // Recheck the state of the switch
  }

  stepper_L.setCurrentPosition(0); //Set the zero position
}

delay(1000);

 // Right
val_switch_R = digitalRead(switch_R); // Check the state of the switch
if (val_switch_R ==1)
{
  while(val_switch_R==1)
  {
    //Move the motor back a little to get off of the switch
    Linear_Motion(-1,2); // Need to be the opposite sign of the original ...
        motion
    val_switch_R = digitalRead(switch_R); // Recheck the state of the switch
  }
  stepper_R.setCurrentPosition(0);
}
else
{
  while (val_switch_R !=1)
  {
    // Check sign on here to make sure we are moving in the correct ...
        direction
    Linear_Motion(1,2); // This will move the right motor 0.1 mm
    val_switch_R = digitalRead(switch_R); // Recheck the state of the switch
  }
  delay(500);
  while(val_switch_R==1)
  {
    //Move the motor back a little to get off of the switch
    Linear_Motion(-1,2); // Need to be the opposite sign of the original ...
        motion
```

```
      val_switch_R = digitalRead(switch_R); // Recheck the state of the switch
    }
    stepper_R.setCurrentPosition(0); //Set the zero position
  }

  delay(1000);

  stepper_L.setMaxSpeed(100); //Speed up for user interaction
  stepper_R.setMaxSpeed(100); //Speed up for user interaction

 // move motors closer to the center of their common location...
Linear_Motion(-250,3);
}

// Set up switch as a safety stop, so the slider can not go too far
void Interrupt_Setup()
{
  EIFR = 1;  // clear flag for interrupt 0 (switch_R)
  EIFR = 2;  // clear flag for interrupt 1 (switch_L)
  attachInterrupt(0,Right,RISING); // matches with pin 2 double check high ...
     vs low
  attachInterrupt(1,Left,RISING); // matches with pin 3, dido from above
  noInterrupts();
}

// Measure and convert the coupler position into 10^-4 m as whole numbers
void Coupler_Position()
{
  // Read lin pot and average ten readings
  int avg=0; //average in counts from the lin pot
  val_LinPot = 0;
  for (int i=0; i < 10; i++)
  {
    val_LinPot = val_LinPot + analogRead(LinPot); // between 0 and 1024 ...
       (convert from counts to volts, then from volts to distance)
  }
```

```
  avg = val_LinPot/10;

  // Find the deviation from the mid point
    if (avg >= mid)
  {
    x = (avg - mid);

  }
  else if (avg < mid)
  {
    x = (mid - avg);
  }
////////////////////////// /////////////////////// ////////////////////////// ...
    ///////////////////
// Convert from counts to 10^-4 m (when at 5v it should be about 500/1024 ...
    counts->10^-4m  or 0.48828125)
x = 0.488*x; // This came from a simple calibration by hand
   //Serial.println("Avg");
   //Serial.println(avg);
   //Serial.println("Mid Pt");
   //Serial.println(mid);
   //Serial.println("Coupler Pt");
   //Serial.println(x);
   //delay(5000);
}

// Determines the requisite link length based off of the coupler's curent ...
    location
int Link_Length()
{
  // Use look up table to find the new length (L)
  if (x <= x_t[0]) // Coupler Position is out of range (too small)
  {
    L = L_t[0];
  }
  else if(x >= x_t[last]) // Coupler Position is out of range (too big)
```

```
  {
    L = L_t[last];
  }
  else // Normal Coupler Range
  {
    // Find the L that matched with the recorded x value
    L = L_t[x-1]; // This works because of how x_t is set up (x_t = 1, 2, ...
       3,...)
  }

  // Limit the range of the length to keep the device from breaking
  if (L < lb)
  {
    L = lb;
  }
  else if (L > ub)
  {
    L = ub;
  }
}

// Determines the necessary motor position and moves there
void Move_Motors()
{
  // Convert from link length to relative distance to travel
  //  then convert from relative distance to travel to motor position (pos)
  int relative = ub - L; //This is set up so that when the length needs to ...
     be at its longest the motors will be at their zero position
  //Serial.println(relative);
  pos = relative*1.575; //(1/254)=10^-4m->inches  & ...
     (1/0.5)=inches->revolutions & (200.0/1.0)=rev->steps
  stepper_L.moveTo(-pos); //Sends it away from the switch
  stepper_R.moveTo(-pos); //Sends it away from the switch
//    stepper_L.moveTo(0); //Sends it away from the switch
//  stepper_R.moveTo(0); //Sends it away from the switch
  stepper_L.run();
```

```
  stepper_R.run();
//Serial.println(x);
//  Serial.println(L);
//  Serial.println(pos);
//delay(7000);
}

// This function accepts a linear distance in 10^-4m that you want to ...
   drive the nut.
  // This is relative linear motion to the current position. It ...
     alsoaccepts which motor
  // will be driven
  // 1 = stepper_L
  // 2 = stepper_R
  // 3 = both
void Linear_Motion (int distance, int moto) //Units of distance are 10^-4 ...
   meters
{
  // Convert from length to motor position (pos)
  int turn;
  turn = distance*(400.0/254.0); //(1/254)=10^-4m->inches  & ...
     (1/.5)=inches->revolutions & (200.0/1.0)=rev->steps

  switch (moto)
  {
    case 1: //Left motor
    stepper_L.move(turn);
    stepper_L.runToPosition();
    break;
    case 2: //Right motor
    stepper_R.move(turn);
    stepper_R.runToPosition();
    break;
    case 3: //Both
    stepper_L.move(turn);
    stepper_R.move(turn);
```

```
    stepper_L.runToPosition();
    stepper_R.runToPosition();
    break;
  }
}

void Reset()
{
  // Flash LED as warning and delay for 1 seconds
  for (int i = 0; i < 31; i++)
  {
    digitalWrite(LED_READY, HIGH);
    delayMicroseconds(16000);
    digitalWrite(LED_READY, LOW);
    delayMicroseconds(16000);
  }

  // Reset the motors zero position to match the longest link length L2
  interrupts();
  Motor_Zero();
  noInterrupts();

  // Reset the interrupts
  Interrupt_Setup();

  digitalWrite(LED_READY, HIGH); // Turn on LED to indicate that the ...
     zeroing of the motors is finished
}
////////////////////////// ////////////////////// ////////////////////////// ...
   //////////////////
// INTERRUPTS

void Left ()
{
  // Disconnect the interrupt until the motors are re-positioned
  EIFR = 1;  // clear flag for interrupt 0 (switch_R)
```

```
  detachInterrupt(0);
  EIFR = 2;  // clear flag for interrupt 1 (switch_L)
  detachInterrupt(1);

  interrupts();
  // Stop the motors
  stepper_L.runToNewPosition(stepper_L.currentPosition());
  stepper_R.runToNewPosition(stepper_R.currentPosition());
  noInterrupts();

  // Reset arduino (aka re-zero the motors)
  Reset();
}

void Right ()
{
  // Disconnect the interrupt until the motors are re-positioned
  EIFR = 1;  // clear flag for interrupt 0 (switch_R)
  detachInterrupt(0);
  EIFR = 2;  // clear flag for interrupt 1 (switch_L)
  detachInterrupt(1);

  interrupts();
  // Stop the motors
  stepper_L.runToNewPosition(stepper_L.currentPosition());
  stepper_R.runToNewPosition(stepper_R.currentPosition());
  noInterrupts();

  // Reset arduino (aka re-zero the motors)
  Reset();
}

//////////////////////// ////////////////////// ////////////////////////// ...
   /////////////////
// INITIALIZATION LOOP
```

```
void setup()
{
  Serial.begin(9600); // Testing purposes
  Serial.println("Run1");
  IO_Setup(); // Establish the input/ouput pins that will be used
  Motor_Setup(); // Establish the default speeds and accelerations for the ...
     motors
  Motor_Zero(); // Set the motors zero position to match the longest link ...
     length L2
  Interrupt_Setup(); // Set up switch as a safety stop, so the slider can ...
     not go too far
  digitalWrite(LED_READY, HIGH); // Turn on LED to indicate that the ...
     zeroing of the motors is finished
}

////////////////////////// ////////////////////// ////////////////////////// ...
   //////////////////
// MAIN LOOP

void loop()
{
  interrupts(); // Enable the switches to work as safety limits

//  // Important note:
//    // Measurements for the couplers position (x) and length of the ...
   compliant link (L) will
//    // be whole numbers representing 10^-4 meters.  This allows the ...
   system to be accurate
//    // within 0.1 mm, and still use integers to help speed up the ...
   computing time.
  Coupler_Position(); // Measure and convert the coupler position into ...
     10^-4 m as whole numbers
  Link_Length(); // Determines the requisit link length based off of the ...
     coupler's curent location
  Move_Motors();  // Determines the necessary motor position and moves there
}
```