Reconfigurable End Effector Allowing For In-Hand Manipulation Without Finger Gaiting Or Regrasping

Jacob Ames Ziesmer
Marquette University

RECONFIGURABLE END EFFECTOR ALLOWING FOR IN-HAND MANIPULATION WITHOUT FINGER GAITING OR REGRASPING

by

Jacob A. Ziesmer, B.S.

A Thesis Submitted to the Faculty of the

Graduate School, Marquette University,

in Partial Fulfillment of the Requirements for the
Degree of Master of Science

Milwaukee, WI 53201–1881

December 2009

ABSTRACT

The goal of this thesis is to move a step towards the solution of the bin picking problem. A novel metamorphic end effector is proposed, tested for proof of concept and analyzed using standard techniques of degrees of freedom and graph theory as well as a classical dynamic analysis. Once proof of concept was achieved, the results from the analysis were formed into an optimization program with the hope of finding a more stable, predictable mechanism.

ACKNOWLEDGEMENTS

Jacob A. Ziesmer, B.S.

I could not have done this on my own. I would first like to thank Dr. Voglewede for this opportunity, as well as, for his patience and guidance with this stubborn student. Thanks also goes to Dr. Schimmels for allowing me to use his robot and to tie it up for a lot longer than planned. I also want to thank the guys in the lab for all of their support and for challenging my ideas and helping me work past the roadblocks. Last, but certainly not least, I want to thank my friends and family who have given me the encouragement that I needed to keep going and ultimately persevere.

TABLE OF CONTENTS

TABLE OF CONTENTS — *Continued*

LIST OF FIGURES

LIST OF FIGURES — *Continued*

CHAPTER 1

Introduction

1.1 Motivation

Several consumer driven factors, such as the demand for personalized products and more rapid release of new models, continue to push for improved manufacturing processes. A competitive manufacturing advantage can be gained by decreasing time required for line changes. One way to accomplish this is to make industrial automation more flexible over both the short and long term. This flexibility will allow the automation to handle more than one specific component. However, flexibility alone is not enough; the automation must also withstand the constant punishment sustained within an industrial setting.

Current trends have shown the capability of robotic manipulators to increase flexibility in some specialized tasks. The automotive industry is one such example; spot welding and painting have proven to be vastly improved through the use of robotic manipulators. There has also been growth in the use of robotic manipulators in the nuclear clean up and food packaging industries. Making use of parallel manipulators for parts having high degrees of symmetry, some structured tasks in the food industry have increased throughput. Unfortunately, most assembly tasks have proven elusive due to the requirement of specific part position and orientation when presented to the manipulator. While dedicated part orientation devices (e.g., vibratory bowl feeders) aid in this task, these orientation devices limit the overall flexibility of the system and increase capital and maintenance costs. If the time between product introductions is large enough many assembly tasks are left to dedicated tooling. If flexibility is the governing constraint then manual assembly is usually selected.

Eliminating the orientation devices leads to the bin picking problem: picking parts directly from unorganized bins and placing them into an assembly. The bin picking problem can be broken into three sections: locating, retrieving and orientating the part. Robotic vision is continuing to make strides in effectively locating and identifying the part. Coupling an effective vision system with a six degree of freedom (DOF) robotic manipulator solves most of the retrieving issues. However, to manipulate the part without regrasping (dropping the part and picking it up again), a proper end effector must also be used. With this combination of vision, six DOF robotic manipulator and end effector, there is the potential for many more industrial tasks to be automated.

The bin picking problem is not new, but previous research has focused primarily on the vision aspect of the problem; vision systems identify the part position and orientation. The robot then uses its end effector to grasp and correctly orient the part before assembly. When trying to couple the manipulator with the vision system, the dilemma facing automation designers becomes the robotic end effector itself. Primarily there exists two groups of end effectors, those with high dexterity and those with low to no dexterity. The highly dexterous end effectors (e.g., the Utah/MIT hand [1]), which allow the identified parts to be picked up and manipulated in-hand (in-hand manipulation) by using finger gaiting ("walking" the fingers in a controlled manner to move the part), currently are not feasible to be used in industrial context due to the high number of controlled DOF; this causes the hand to be too slow and not robust enough for industrial settings. These types of systems are typically only used when time is not a factor and flexibility is paramount. Conversely, there exist very fast, simple and robust end effectors like suction cups and parallel jaw end effectors. These end effectors do not allow in-hand manipulation of the parts and, therefore, most require regrasping for part manipulation when used in a bin picking situation. One such example is the commercially available Pallet Picker 3D [2]. The Pallet Picker 3D uses a camera to identify discrete parts in a bin where a robotic manipulator equipped with a suction cup end effector picks them up. To orient the parts before assembly, the robot must drop the part and pick it up again. Eliminating this extra task has the ability to drastically increase speed and throughput while simultaneously decreasing complexity. This is why most of these simple end effectors are paired with an orientation device so that no in-hand manipulation is required. With a solution to the bin picking problem, part feeders can effectively be removed, decreasing machinery cost by 30% and eliminating 50% of downtime [3].

1.2 Design Paradigm

Many industrial parts for assembly have two axes of symmetry, as defined by Boothroyd [4]; the α axis runs parallel to the plane of assembly while the β axis is perpendicular as can be seen in Figure 1.1. Assuming these lines of symmetry exist and an vision system is in place to determine orientation, a 6 DOF robot is capable of the β rotation, requiring the end effector to provide the α rotation. It is hypothesized that if the end effector can be designed that allows rotation of parts only around the α axis, it will solve many, but not all, of the bin picking end effector related problems encountered today. If the end effector must come from directly above the part, Figure 1.2 shows that rotation about the α axis is the only way to reorient the part without regrasping or finger gaiting.

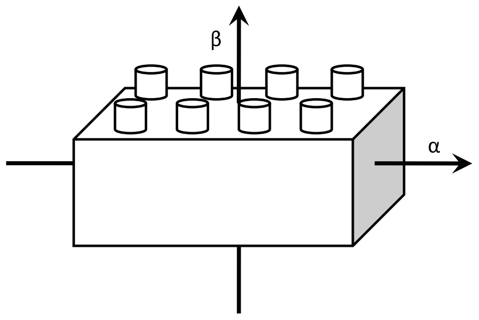


Figure 1.1: α and β Axes As Defined By Boothroyd [4]

To make the end effector as robust as possible, one train of thought is to minimizing the DOF. This reduces the number of actuators needed and the number

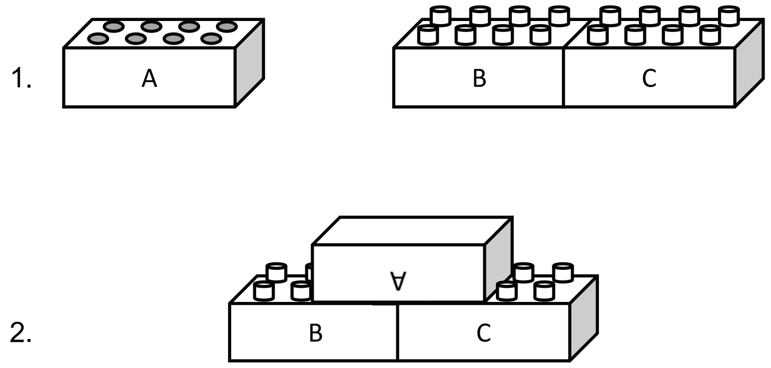


Figure 1.2: Blocks for Assembly Using α Rotation

of parts that may break. To force the block to rotate as desired a “fixed finger” will be used. This fixed finger, shown in Figure 1.3, will take the form of a stanchion in the work space of the six DOF robot allowing the part to be touched down,constraining the part’s DOF while the manipulator rotates.

If the part can be thought of as an integral link, the end effector becomes a closed loop chain with the need to manipulate a link but also be able to completely constrain the same link. Because of the dual configurations required, reconfigurable mechanisms provided promising solutions.

Mechanism selection will be looked at in greater detail in Chapters 2 and 3. Design and Testing will be discussed in Chapter 4, analysis and optimization of the mechanism will be shown in Chapters 5 and 6 followed by the conclusions in Chapter 7.

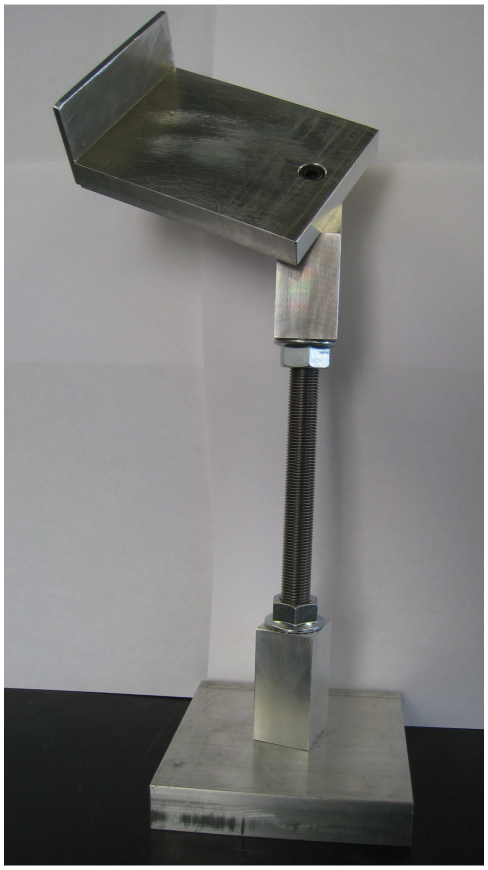


Figure 1.3: Fixed Finger

CHAPTER 2

Literature Review

Looking at the end effector and part as a linked closed loop mechanism, created the need for a mechanism with multiple poses in which the constraints on a link changes. Substituting the part for this link is believed to give the necessary changes in mobility to fully constrain the part in one pose, yet be able to manipulate the part in the next. This lead to using a mechanism that would fall into the reconfigurable mechanisms classification. The term reconfigurable mechanisms is often used interchangeably with mechanisms with variable topology (MVT). Not all reconfigurable mechanisms have a change in their topology but instead have a change in their range of motion. Calling these mechanisms MVT is *wrong*. It is thought that by making use of these changes, the desired link (the part) can go from fully constrained to having the ability to rotate about the desired axis. To rotate the part without use of an extra actuator, a fixed finger will be placed in the work space of the robot.

2.1 Metamorphic Mechanisms

Metamorphic mechanisms are one type of reconfigurable mechanism and have have been loosely defined as, "mechanisms whose number, the total of all effective links, change as they move from one configuration to another or a singularity condition in geometry occurs that makes it behave differently" [5].

If the change in mobility is caused by the actuation scheme or an operator interacting with the mechanism, the mechanism is *not* a metamorphic mechanism. Using the approach of locking actuators or manual manipulation the mechanism, all mechanisms can become metamorphic mechanisms and the classification becomes trivial. Furthermore, the governing equations for this approach do not change. It is proposed that the classification also include mechanisms where the joint changes form due to the pose or motion of the mechanism. Therefore, the definition of metamorphic mechanisms should be amended to state:

> ***Definition:*** *A metamorphic mechanism is such that, during operation, a minimum of one of the following occurs due to the kinematics of the mechanism:*
> *(a) The number of links changes*
> *(b) The number of joints changes*
> *(c) The type of a joint changes*

Example : An example of a unique application of a metamorphic mechanism, similar to origami, is presented by Dai in [5, 6]. By folding a flat piece of paper, a mechanism which changes configuration depending on its pose is created. In this mechanism, a hexahedron starts as a flat piece of paper with predetermined creases (represented by the dotted lines) and flaps as shown in Figure 2.1. In this form, it is a serial mechanism where each crease is modeled as a revolute joint.

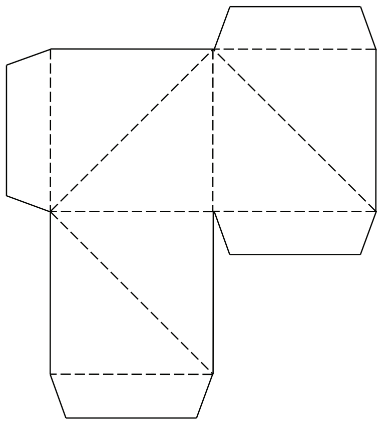


Figure 2.1: Flat Creased Card (Adapted From [5])

This is an abstract example of a metamorphic mechanism because it requires someone to physically manipulate it due to no actuators being present. If the creases, shown as dashed lines, represent revolute joints and the flaps represent mechanical clasps. As the paper is folded along the creases, the flaps come into contact with the opposing piece. This mimics an actuated revolute joint and two sides of a clasp closing. This *creates* a new joint, which now adds constraint to the system. The number of links remains the same, but the relations between the links changes. The mechanism is no longer a member of the class of serial mechanisms. Once all the folds and clasps are completed, it becomes a structure as shown in Figure 2.2. The "actuated" joints may lose the ability to move due to the geometry but at no point are any of the them ever locked.

This mechanism clearly fits into several classifications as it changes from a serial mechanism to a structure. While not all metamorphic mechanisms will change classes during operation, it is not uncommon due to the nature of the class.

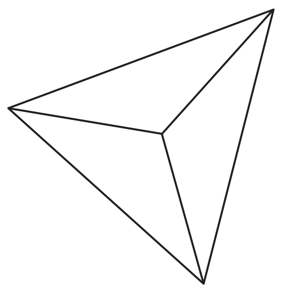


Figure 2.2: Hexahedron Formed From The Flat Card In Figure 2.1. (Adapted From [5])

2.2 Previous Work

To design a functional end effector the fundamentals of gripping must first be explored. For a good review see [7]. Expanding on the fundamentals and using grasping to reorient parts is not a new concept. Canny and Goldberg presented several options for using lower DOF manipulators and simple sensors to determine current orientation and for high precision insertion. Carlisle et al. presented an alternative to vibratory bowls in [3] for orientation. This part feeder consisted of a high speed robot arm combined with a vision system. The end effector attached to the arm was intended to allow the part to rotate about the previously defined α axis on hard finger contacts but never fully constrained the part. Zhang et al. [8] used compensatory grasping to reorient parts with an undesirable resting position. This process uses pins at key locations on parallel jaw end effectors to guide the part into the desired grasp position as the end effector closes in a form of controlled part tumbling. This method relies heavily on the geometry of the part. If there is any variation in the part geometry it may not tumble as desired. A six DOF reconfigurable end effector for fixtureless assembly of automotive body panels was developed by Yeung [9]. This end effector has three fingers each with two actuated joints, one revolute and one prismatic and was classified as a reconfigurable mechanism. By actuating the joints in the hand, “reconfiguring” the end effector, this end effector can pick up different shaped objects. The end effector is not a reconfigurable mechanism, but instead is closer to that of a simplified version of an anthropomorphic hand.

Grasping is not generally associated with reconfigurable mechanisms but for the purposes of this research the two are directly linked. The first reconfigurable mechanisms were introduced in 1996 by Wohlhart and termed kinematotropic mechanisms [10]. Wohlhart’s work presented three mechanisms that have similar properties: at least two distinct ranges of motion separated by a bifurcation pose, while in the bifurcation pose each mechanism *gains* DOF, and the mechanisms have different DOF in each of the ranges of motion. This work was followed by Galletti’s attempt to develop some single loop [11] and multi-loop [12] kinematotropic chains. However, these mechanisms only partially fulfill the definition proposed by Wohlhart in [10] as they do not pass through a bifurcation point but instead remain in a singularity.

Arguably, the first attempt to use a reconfigurable mechanism as an end effector was proposed by Voglewede [13]. When modeled as a closed kinematic chain, the mechanism would reach a singularity that allowed large unconstrained motion (aka self motion). However, *it does not function as claimed.* As *modeled* and analyzed the mechanism successfully completes the task, but unfortunately, the model lacks a *critical* nonholonomic rolling constraint. Because of this, the part to be manipulated is never fully constrained. The part can move in and out of the plane in one pose, an unintended motion, and can rotate about its center axis in the other, the desired motion. This is not due to an improper analysis, but due to an oversight in the proposed model.

In 1999 Dai introduced a group of mechanisms called metamorphic mechanisms [5]. Dai went on to propose a way to represent each phase of a metamorphic mechanism by using adjacency matrices as well as a topological graph [6, 14]. Using elementary matrix operations, a matrix representation of the topological changes of the matrix was determined. Following in Dai's work, Lan [15] proposed a way to use augmented adjacency matrices to represent metamorphic mechanisms; this would allow for the matrix to remain the same size as the mechanism changed. However, the metamorphic mechanisms proposed are not metamorphic due to joints being locked to create the change in the mechanism. Also, the use of these augmented adjacency matrices does not give meaning to the change in pose for general metamorphic mechanisms and must be constructed by inspection for each pose. One feasible way to form a metamorphic mechanism is to properly implement one or more of the variable joints, as introduced by Yan and Kuo [16], based on topological representations and able to change the kinematic pair during operation.

In 2007 Dai and Wang introduced a metamorphic robotic hand [17]. The claim of metamorphic capabilities relies on the use of locking actuators in the spherical mechanism of the palm. The number of links and joints remains constant as well as the type of joints, therefore, this mechanism is not truly metamorphic and is instead a spherical mechanism.

In 2000 ortho-planar mechanisms were introduced by Parise [18]. The primary use for these mechanisms is in micro-electro-mechanical systems (MEMS).

Many of these mechanisms are change point mechanisms which are more closely related to kinematotropic mechanisms than metamorphic. Ortho-planar mechanisms were then expanded upon by Lusk et al. with a derivation of the design space of these mechanisms [19], and then again by Caroll et al. with the development of some compliant ortho-planar metamorphic mechanisms (COPMM) [20]. These mechanisms are able to be manufactured in a planar configuration often assuming a change point pose. After undergoing a "metamorphic process," the mechanisms no longer remain planar or have a change point pose. By allowing for manufacturing of the mechanisms to be planar versus spatial, costs can be reduced. There can also be a savings seen in packaging and shipping due to the ability to ship flat and be constructed upon arrival. Because the mechanism is not useful for any task prior to the "metamorphic process" proposed, this metamorphosis is simply assembly of the mechanism. Taking the links from the manufacturing process and putting them together does not meet the criteria of a reconfigurable mechanism.

Currently there is no good way to synthesize metamorphic mechanisms leaving it solely to designers' imaginations. In an effort to change that, Zhang et al. proposed an evolutionary methodology for synthesizing and and designing metamorphic mechanisms using biological modeling and evolution [21]. This is a very complex method and not very well refined making it hard, if not impossible, to guarantee that a mechanism with the desired characteristics will be revealed.

Metamorphic mechanisms are not to be confused with metamorphic robotic systems. These systems consist of a collection of independently controlled mobile robots that are identical in structure, computational abilities, and motion constraints [22]. These robots move over and around each other to reposition themselves into new configurations.

Calculating the mobility of reconfigurable mechanisms can be difficult due to the ability to change DOF multiple times during a process. There is also a use of redundant DOF which are generally ignored in most DOF calculations. Gogu [23] did a critical review of the calculation of mobility for non-reconfigurable mechanisms. Several methods using screw systems [24, 25, 26] have been introduced to calculate the mobility of reconfigurable mechanisms in their different configurations that are considered more accurate than the Grübler or Kutzbach equations. As with any mechanism it is critical to know and understand how many DOF are present and where actuators can be placed to properly constrain the mechanism for the desired motion profile.

It appears that the only instance of a reconfigurable mechanism being used as a end effector was done by Voglewede [13]. Building upon the knowledge gained in [13] the next iteration can be designed.

CHAPTER 3

Novel End Effector

The structured environment of the industrial assembly process creates a situation where a fully anthropomorphic hand is generally not needed to allow for manipulation of the parts to be assembled, partially due to the fact that the geometry of the parts is already known. It has also been noted that there generally exists two axes of symmetry for most industrial assembly parts. Assuming that symmetry along the α and β axes does exist for a part, this vastly simplifies the problem. By making use of a six DOF robot, which can provide the necessary β rotation and to some extent α rotation as well, the end effector only needs to be able to accommodate the remaining necessary rotation of the part about the α axis. It is also assumed that a two finger grasp can be made that will fully constrain the part, and that the necessary vision system is already in place allowing for the orientation of the part to be determined.

The mechanism introduced here is the next iteration of the design introduced in [13]. As modeled, the mechanism proposed in [13] would be effective, however, due to an unforseen rolling constraint, the part is never fully constrained. Minimizing or eliminating the effects of the rolling constraint will decrease the complexity of the model required to produce useable results. Also the use of soft finger contacts necessitated the use of a nonholonomic rolling constraint. This constraint was also considered negligible in the analysis but has since been shown to be critical.

3.1 Mechanism Selection

The objective of the mechanism introduced is to allow for in-hand manipulation while being robust enough to thrive in an industrial setting. It is believed that making use of a metamorphic mechanism and a new form of in-hand manipulation that does not require the end effector to break contact with the part, the desired kinematic solution can be found. Kinematotropic mechanisms hold promising solutions as well, but also have distinct disadvantages not necessarily experienced with metamorphic mechanisms; primarily that they require singularities in the workspace while metamorphic mechanisms do not. A kinematotropic mechanism must *pass through* a singularity when the change in mobility occurs and seems to require all manipulations to be undone before they are able to pass back through the singularity. Also, actuation near a singularity poses many problems; requiring the mechanisms to predictably pass in and out of a singularity on a regular basis complicates things further. Making use of a metamorphic mechanism will reduce the likelihood of operating in or around singular positions.

The concept of this new design is based on changing the interface between the part and the end effector and thus changing the type of contact points between the part and the end effector. Assuming that the part and end effector never lose contact and that the normal force at the interface between the end effector and the part is always sufficient to constrain the part from undesired movement, as is the goal of in-hand manipulation without finger gaiting, the interface of the part and end effector can be modeled as joints. Because the end effector-part interface changes, this can be modeled as the joints changing type during operation classifying the mechanism as a class (c) metamorphic mechanism as defined in Section 2.1.

3.2 Design Process

A prototype of the proposed end effector is shown in Figures 3.1 though 3.6. The key to the mechanism is the part-finger interfaces. Modeling the interfaces as joints, the end effector and part combine to create a closed loop mechanism, and it can be seen that as the pose changes so do the part-finger interfaces; they change between hard finger contacts [27] (spherical joints), as seen in Figure 3.5, and planar contacts. It is assumed that the frictional forces are great enough to constrain motion generally associated with planar contacts constraining the part and creating rigid joints, shown in Figure 3.3.

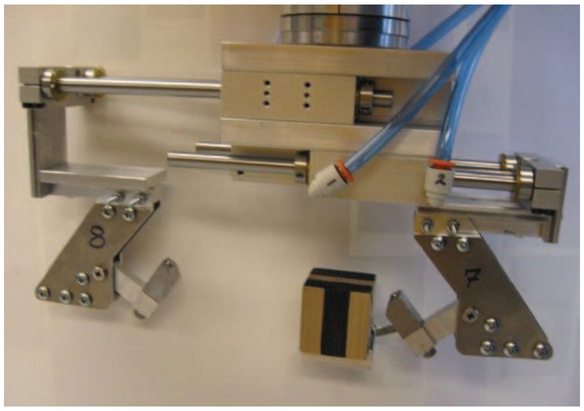


Figure 3.1: Prototype End Effector In Phase 1

This design was chosen due to the ability to move between poses and maintain the desired manipulation, as well as, minimizing the effects of the nonholonomic rolling constraint by using small hard contacts. By minimizing the nonholonomic rolling constraint, the model of the mechanism will be greatly simplified. Also, the motion provided is much more intuitive as the distance traveled while rolling on a soft contact is dependent upon many parameters other than the arc length traveled. In an ideal design, the fingers would go directly from a planar contact to a sharp point contact to eliminate any rolling; this is not possible due to material properties. The sharp point would either dig into the part damaging it or would have uncontrolled deformation damaging the end effector. Neither of these options are desirable. To avoid these, the tip of the end effector is a small ball bearing. Due to the minimal diameter, the rolling is negligible allowing the contact to be modeled as moving directly from planar joints to spherical joints.

Several other design possibilities were explored but were not feasible

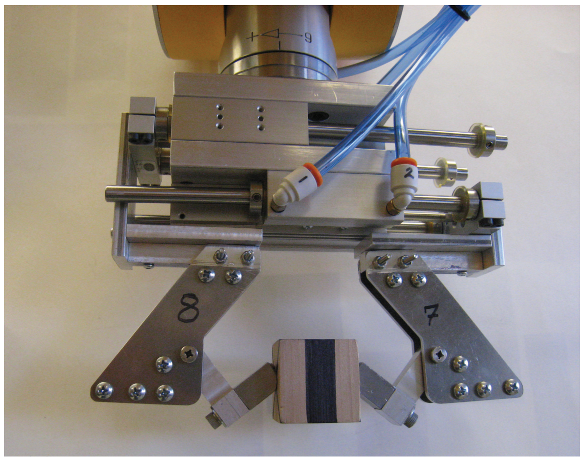


Figure 3.2: Prototype End Effector In Phase 2

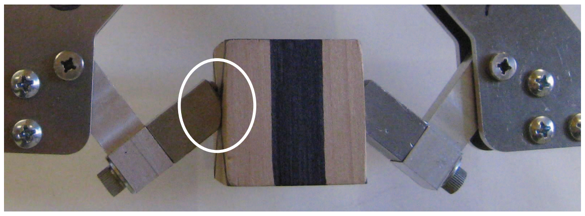


Figure 3.3: Closeup of Planar Contacts

possibilities. Another mechanism considered is shown in Figure 3.7. Concerns about tolerance stacking, the potential affects on the singular position, and the need for two actuators, one prismatic and one revolute, quickly eliminated this option.

3.3 Actuation Scheme

The selected end effector can be broken up into three distinct phases. These phases are: Phase 1: completely open - no part; Phase 2: half open - part constrained; and Phase 3: completely closed - part rotation. In order to get these three phases, a three position actuator is required. Since a three position actuator that meets the needs of this ongoing research is not commercially available, two two-position actuators were utilized instead.

In Phase 1, shown in Figure 3.1, the part is assumed to be resting in some configuration ready for assembly. The part has not yet come into contact with the

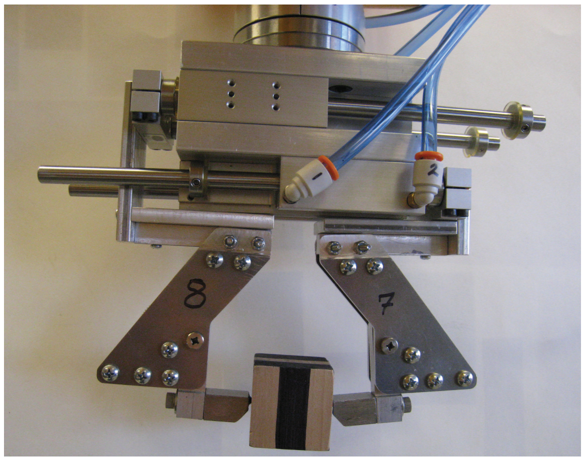


Figure 3.4: Prototype End Effector In Phase 3

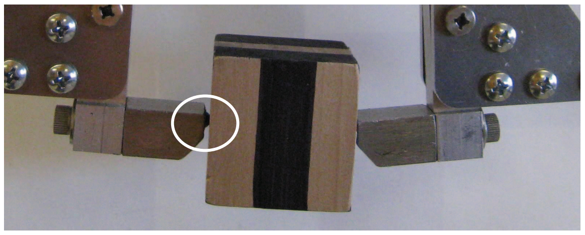


Figure 3.5: Closeup of Hard Finger Contacts

end effector. The end effector then consists of two open kinematic chains of a prismatic followed by a revolute (PR) joint. The revolute joint contains a stretched spring to keep the finger pulled back against a stop, completely constraining the finger.

In Phase 2, shown in Figure 3.2, the end effector has partially closed onto the part. Due to the geometry of the finger contacts, the end effector forms two planar contacts on the part, as shown in Figure 3.3. In this pose the end effector can be modeled as a simple parallel jaw end effector. Assuming that the coefficient of friction is high and the normal force is sufficient, the part has complete force closure [27, 28].

In Phase 3, the jaw closes to the pose shown in Figure 3.4. The fingers on the end effector rotate against the springs imbedded in the secondary joints due to the force provided by the linear actuator and place the part onto hard point finger

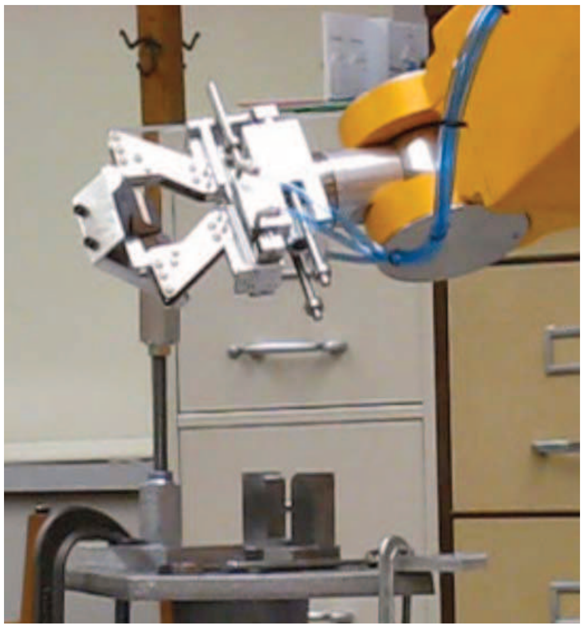


Figure 3.6: Prototype End Effector Mid-Rotation

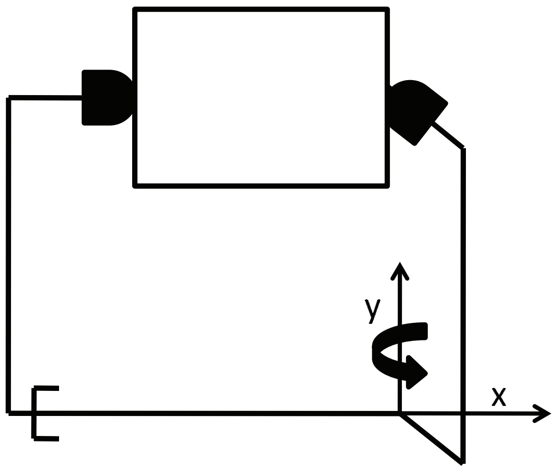


Figure 3.7: Alternate End Effector Design

contacts which can be seen in Figure 3.5. Due to the design of the points the actual rolling that takes places has been minimized such that the motion is negligible. In this position the part is held between two point contacts and it is again assumed that friction is enough to prevent the part from sliding. When the part is held between the hard finger contacts, the entire assembly can be modeled as a closed kinematic chain with the part picked up connecting the two serial links from Phase

1. Due to the nature of the point contacts, the part can now be analyzed as a link with two spherical joints. Such a configuration is a Spherical Spherical (SS) pair and is referred to as a passive DOF. Due to this passive DOF, force closure cannot be obtained [27], and the part is able to rotate around the axis between the two point contacts.

By making use of the fixed finger the part can be rotated when in Phase 3 by contacting the part with the fixed finger. The full 6 DOF of the robot can be utilized with an applicable motion planning scheme to rotate the part to a desired orientation. When reaching this orientation, the end effector will transition back to Phase 2 to fully constrain the part. When fully constrained, the part can be assembled at which time the end effector transitions back to Phase 1 to release the part and the process can be repeated.

CHAPTER 4

Prototype Manufacturing and Testing

4.1 Prototype Manufacturing

In order to validate the design, a prototype was created. The end effector body consisted of aluminum plates, separated by plastic spacers. The aluminum plates provided attachment points for the fingers as well as the springs. The spacers between the plates provide the required mechanical stops for the fingers. The revolute joint was created using a simple shoulder bolt passing through the end effector body and the machined aluminum finger. A commercially available helical spring was placed inside the end effector body to constrain the moveable fingers. At the end of the fingers, a .1875 inch diameter ball bearing was press fit into a steel finger tip. Several different diameter balls were tried to determine the proper trade-off between indentation into the cube and rotation about the axis as the highest forces are seen when the cube is held between the point contacts. No special surface was created on the contacts to increase friction. If necessary a cross hatch pattern could be machined into the planar contact or a thin layer of material with a higher coefficient of friction could be used to augment the design. A CAD model of the end effector body is shown in Figure 4.1. This assembly is bolted to the actuator assembly. Drawings of all of the components used can be found in Appendix A.

The desired actuation scheme requires a three position linear actuator. A feasible three position linear actuator was not commercially available. One was created by mating two Robohand linear actuators (Automation Technology Inc., model number DLT-08-L-C-3.5). These double acting pneumatic slides were stacked on each other in opposing directions to provide the necessary three position linear

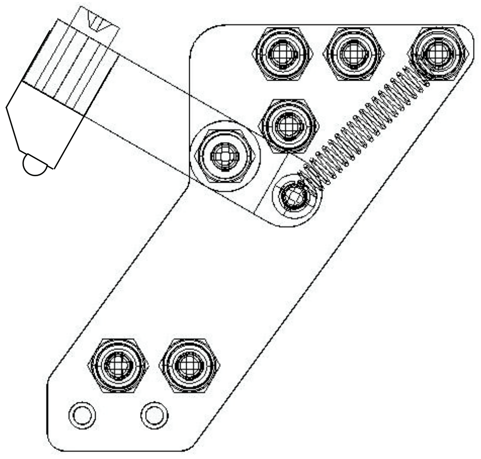


Figure 4.1: End Effector Body and Finger Assembly

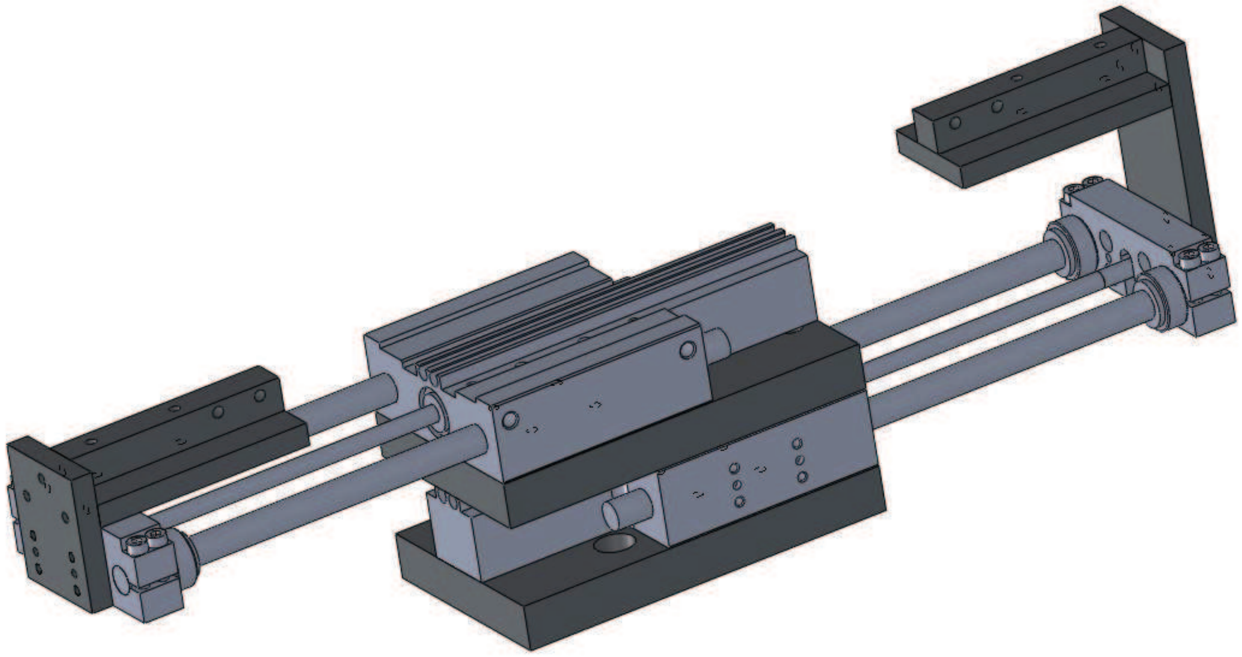


Figure 4.2: CAD Model of Linear Actuator Assembly

actuator. Adaptor plates were created to interface between the two actuators and the robot. Bracketry was also created to interface between the actuators and the end effector body. Drawings of all of the components used for the actuator assembly can be found in Appendix B.

To test the prototype the assembly was mounted to a 6 DOF Stäubli robot

(Model number RX 130 CS7). The pneumatic actuators were powered using the two auxiliary air lines contained within the robot. Control for the end effector was completely within the capability of the standard robot controller (VAL II). Thus, there was no need for additional controls, wires, conduits, or actuators other than four plastic airlines. The entire assembly is portable to most other robotic manipulators with standard two air line outputs.

Presentation of the block uses a fixture so that the block was always at the same location with the same orientation as can be seen in Figure 4.3. This allowed testing to focus on the mechanical design and eliminated the need for the robot to be connected to a vision system. The fixed finger used to rotate the block was a custom stanchion and can be seen in Figure 4.4. This shape was designed so the block could not only be rotated once in the appropriate phase but could also be repositioned so that the axis of rotation was exactly as desired. This helped to offset some of the relative motion between the block and the end effector due to the undesired dynamic effects encountered when moving between poses. The height and angle at which the platform was set was chosen to accommodate the workspace of the robot.

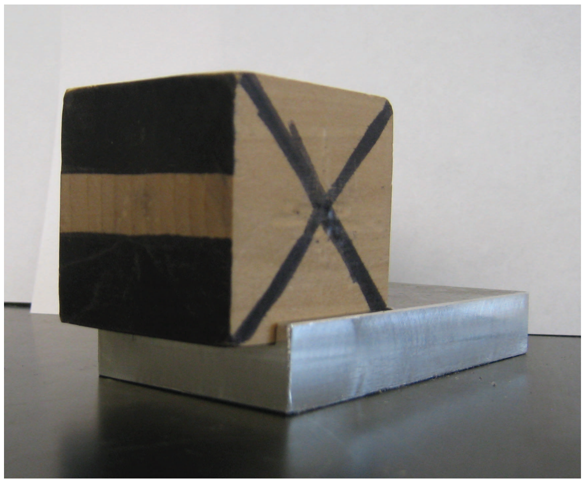


Figure 4.3: Block Starting Point Fixture

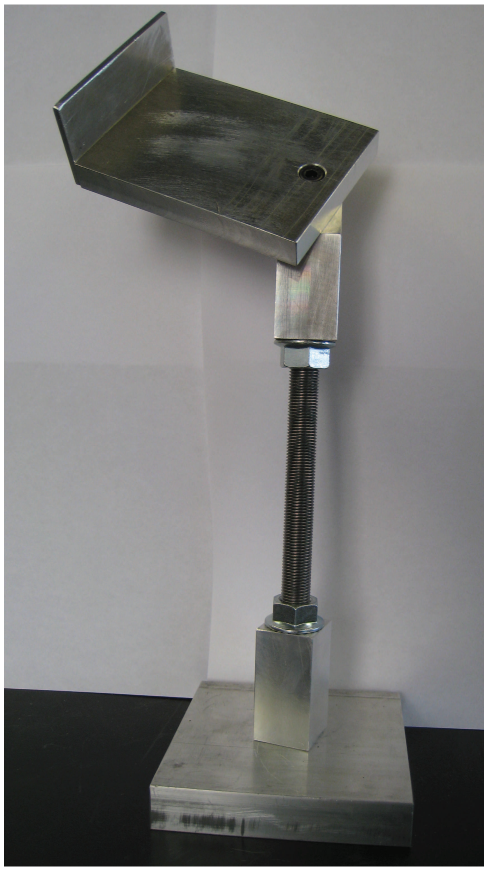


Figure 4.4: Fixed Finger

4.2 Testing

The necessary control protocol was written for the Stäubli robot allowing the configuration to be run for proof of concept. During the testing, the mechanism has been shown to effectively rotate the block as desired, but suffers from a lack of robustness. It preforms as desired kinematically, but is in an unstable equilibrium that, at times, can cause the block to rotate in the grasp causing the block to not always end up in the desired position. The change between poses occurs too quickly to see the details of what is happening with the naked eye. Using a Photron APX RS high speed video camera, the end effector was filmed at 500 frames per second. In the video, it was seen that one finger actuates first and actually loses contact with the block as shown in Figure 4.5. This occurs because the mechanism is attempting to go to the crossed configuration, a more stable equilibrium position, as verified with the model that will be discussed in Section 5.3. Upon hitting the stop, preventing the first finger from making it to the crossed configuration, the next finger actuates bringing the block with it. Particle image velocimetry (PIV) was applied to the high speed footage to obtain the velocity for the linear actuator, $\dot{d}$. Because the end effector is bulk and not particles, the PIV software was unable to give the true value of the velocity. It did however give insight to the shape of the velocity curve, Figure 4.6. As expected, it is far from constant or smooth curve.

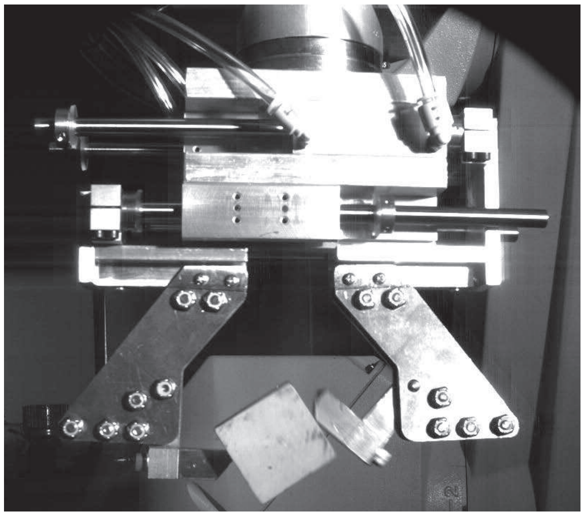


Figure 4.5: End Effector Loses Contact

One of the unforseen advantages of this design is the built in compliance during assembly, while the block is placed the grip tightens. As the end effector is withdrawn the natural motion allows the grip to loosen if the actuation timing is not perfect. Before these advantages can be taken advantage of and improvements made, it must first be fully understood why the fingers do not actuate simultaneously, causing one finger to lose contact with the block, requiring a full analysis of the mechanism.

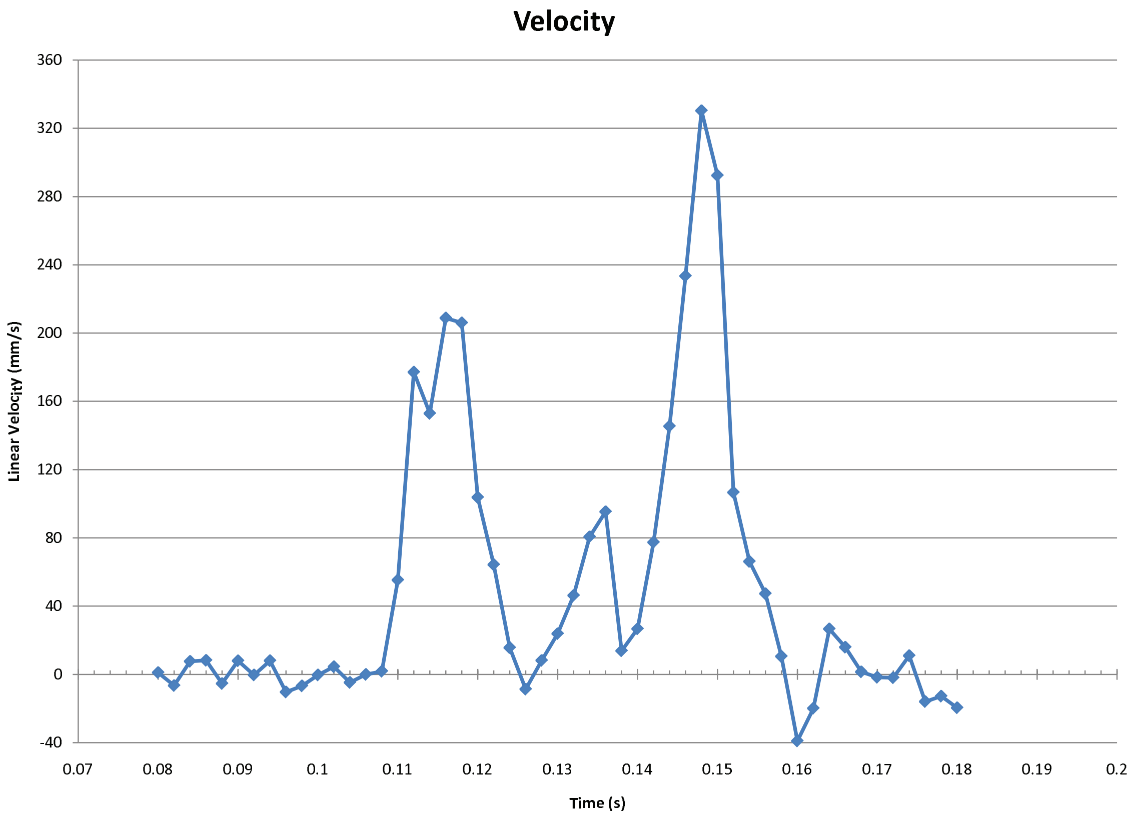


Figure 4.6: Linear Velocity ($\dot{d}$) vs. Time

CHAPTER 5

Analysis

In order to verify the design intent, elucidate opportunities for improvement and determine new design concepts, the mechanism must be fully understood. First, the mobility of the mechanism will be determined for each phase. Further analysis will be conducted using standard approaches from [29] and [30]. Due to the type of mechanism these methods provide some insight while also elucidating some confusion in the literature, but falls far short of providing the needed information. A full dynamic analysis is completed resulting in obtaining the remaining information.

5.1 Mobility

Grübler's equation [23] was used to verify the hypothesized DOF of the manipulator. To apply Grübler's equation, the mechanism must be broken up into the distinct phases discussed previously. Specifically using Tsai's formulation of Grübler's equation [30], the total degrees of freedom can be estimated by:

$$M = b(m - p - 1) + \sum_{i=1}^{p} f_i - f_p \tag{5.1}$$

where M is the total DOF of the mechanism, b is the motion parameter (3 for planar motion or 6 for spacial motion), m is the total number of links, p is the total number of joints, f_i is the DOF of joint i, and f_p is the number of passive DOF, a degree of freedom that when actuated does not affect the kinematics of the mechanism. Because the passive DOF are important for this mechanism, as this provides motion in Phase 3, a modified version is also used here:

$$M = b(m - p - 1) + \sum_{i=1}^{p} f_i \tag{5.2}$$

Phase 1 consists of two coupled, open, PR chains as seen in Figure 5.1. This is a trivial case in that the mechanism will have three degrees of freedom. In this phase, the prismatic joint is coupled and the revolute joints each have springs that pull the fingers against hard stops to fully constrain them. Thus, the mechanism has enough actuators (either active or passive) to fully constrain motion.

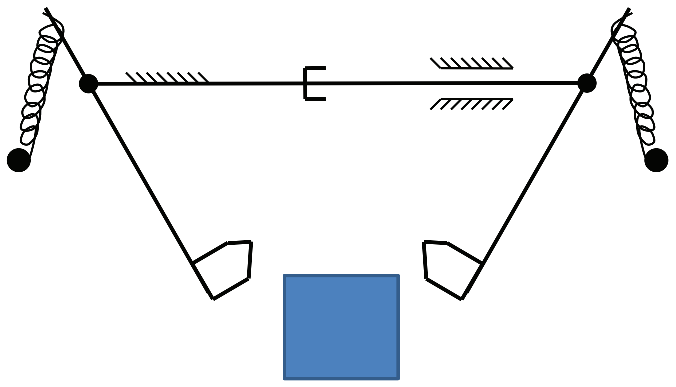


Figure 5.1: Schematic of Phase 1

In Phase 2 (Figure 5.2), the mechanism is a 5 bar (2R2E1P) mechanism and can be modeled as five links, two revolute joints, one prismatic joint, two planar contacts, and no passive DOF. Because of the friction in the planar contacts there are no available DOF resulting in:

$$b = 6, m = 5, p = 5, \sum_{i=1}^{p} f_i = 3, f_p = 0$$

Because there are no passive degrees of freedom in this pose the choice of equation is less important. Substituting into into Equation 5.1 or 5.2, the number of independent DOF is found to be negative three denoting that the mechanism is a structure.

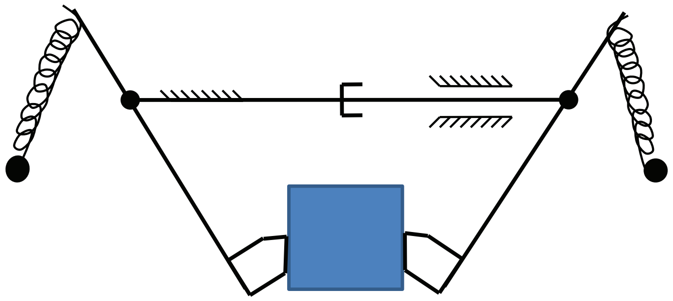


Figure 5.2: Schematic of Phase 2

In Phase 3 (Figure 5.3), the mechanism is a 5 bar (2R2S1P) mechanisms and can be modeled with five links, two revolute joints, one prismatic joint, two spherical joints, and one passive DOF. In this case:

$$b = 6, m = 5, p = 5, \sum_{i=1}^{p} f_i = 9, f_p = 1$$

Due to the passive degree of freedom present in this pose the equation choice becomes critical. By substitution into Equation (5.1), the number of independent DOF is found to be two. This indicates that constraining the prismatic joint and one of the revolute joints completely constrains the system. However, when utilizing Equation (5.2), the passive degree of freedom is not subtracted, and the total DOF is three. The spring and prismatic joint constrains two of the DOF leaving the critical passive DOF available.

5.2 Graph Theory

One prominent way to analyze reconfigurable mechanisms is using graph theory [6, 15, 26]. In graph theory the vertices are numbered 1 through n and represent the links of the mechanisms while the lines between represent the joints and are labeled accordingly. Figure 5.4 and 5.5 show the graphical representation of Phases 2 and 3 respectively. In the graphical representations, the difference between

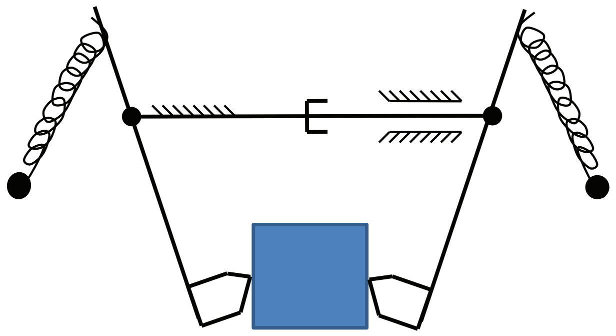


Figure 5.3: Schematic of Phase 3

the two phases can be seen as the planar contacts are replaced with spherical joints in Phase 2.

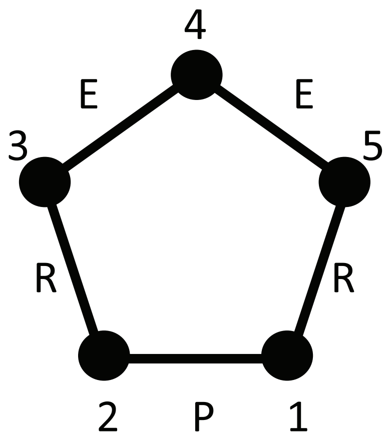


Figure 5.4: Graph of Phase Two

5.2.1 Adjacency Matrices

The adjacency matrix is a way of putting graph theory into a format that can easily be handled by computers and mathematical software. As shown in [29], the adjacency matrix can elucidate the topological changes undergone by metamorphic mechanisms by showing the relationships between joints and links in a mechanism. The columns and rows represent a corresponding joint and link number of the mechanism. When links are connected by a joint, the entry in the matrix is 1. When it is not, it is 0.

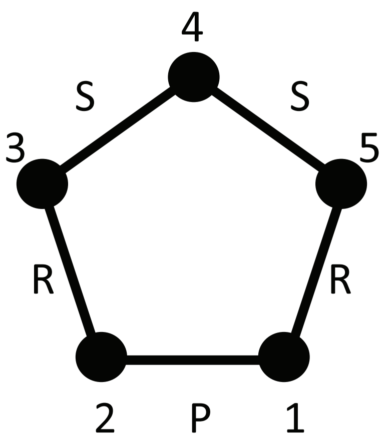


Figure 5.5: Graph of Phase Three

The links are numbered 1 to 5 as shown in Figure 5.4. This analysis will focus only on Phases 2 and 3; Phase 1 is again trivial. When the mechanism is in Phase 2 (Figure 3.2), the adjacency matrix takes the form:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.3}$$

As an example the (2,1) element is 1 as the base link is connected to the second link through a prismatic joint.

When in Phase 3, the adjacency matrix remains the same. Because the adjacency matrix only detects metamorphic mechanisms that change the number of links or joints, it does not appear in this analysis. This observation is in direct contrast to the conclusion of [29]. Adjacency matrices are only relevent for metamorphic mechanisms of type (a) and (b).

5.2.2 Incidence Matrices

An incidence matrix is similar to an adjacency matrix in how they are formed as both come from the mechanism's graph. Each row of the incidence matrix corresponds to a joint and each column a link. If joint v_i is joined to link l_i, a 1 would be placed at (v_i, l_i) in the matrix. The numbering of the links and joints can be seen in Figure 5.6. As with the adjacency matrix, Phase 1 is also trivial for

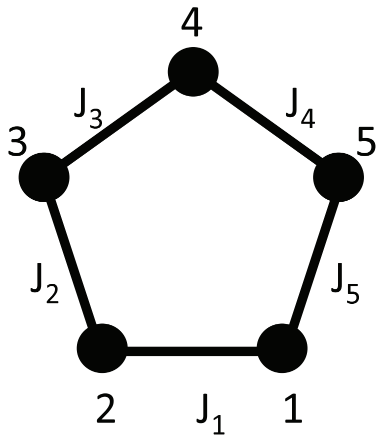


Figure 5.6: Graph Showing Link and Joint Numbering

the incidence matrix. In Phase 2, shown in Figure 3.2, the incidence matrix is:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

Like adjacency matrices, for this mechanism, the incidence matrix does not change between Phase 2 and 3 because the same number of links and joints are retained as well as the same relationships between what links connect to what joints. Only the type of joint changes, therefore, this form of analysis is again only good for metamorphic mechanisms of type (a) and (b).

Graph theory is a standard form of analysis for reconfigurable mechanisms, but only gives information on how and where things are connected. This does little to nothing to help with understanding why one finger is actuated before the other causing it to lose contact with the block. To determine that a standard dynamic analysis must be done.

5.3 Dynamic Analysis

Before a dynamic analysis can be done some assumptions must be made. During testing, high speed video revealed that the end effector primarily operates in the plane with negligible out of the plane motion. It also revealed that there was a negligible amount of deflection allowing for the bodies to be assumed rigid. With this in mind, it was decided to model the mechanism while moving between Phases 2 and 3 focusing on the value of θ_3. θ_3 is defined as the angle between the top of the block and the x-axis; θ_3 in Figure 5.7 is zero. In this phase it is a 5 bar (2R2S1P) mechanism. Since it had already been shown that the passive degree of freedom functioned as desired the model was simplified to a 5 bar (4R1P). Imposing this over the end effector as shown in Figure 5.7 it appears very similar to a fourbar. Modeling link d as a prismatic joint, such that $\dot{d}$ is the linear velocity and $\ddot{d}$ is the linear acceleration of d, effectively creates a "two DOF fourbar." In this configuration it is assumed that d stays parallel with the ground and θ_1 is therefore always zero.

Using the dynamic analysis of a fourbar done in [31] as a reference, a dynamic analysis of the end effector was preformed. The newly derived equations were checked against those presented in [31] by setting $\dot{d}$ and $\ddot{d}$ equal to zero.

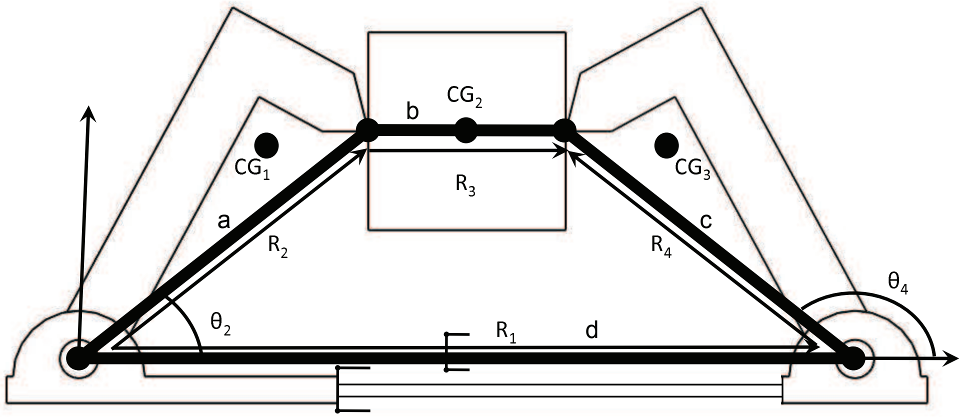


Figure 5.7: Fourbar Imposed Over the End Effector

5.3.1 Position Analysis

The first step was to complete the position analysis. The position analysis is not a continuous analysis; instead it is instantaneous and requires discrete points in time. Because of this, the fact that d is changing has no affect on the analysis as the instantaneous value of d is still known at all times. Therefore, this analysis follows *directly* from [31]. The same notation will also be used throughout for easier reference. Using the vectors displayed in Figure 5.7 the vector loop equation must be formed as shown in Equation 5.5.

$$\overrightarrow{R}_2 + \overrightarrow{R}_3 - \overrightarrow{R}_4 - \overrightarrow{R}_1 = 0 \tag{5.5}$$

The scalar length of the vectors will be represented by a, b, c and d in reference to R_2, R_3, R_4 and R_1 respectively. Next the vector equation is converted into complex form becoming:

$$ae^{j\theta_2} + be^{j\theta_3} - ce^{j\theta_4} - de^{j\theta_1} = 0 \tag{5.6}$$

Using complex form reduces the number of equations and helps keep the derivation cleaner. To allow this equation to be separated into real and imaginary components the Euler equivalents must be used. With this substitution Equation 5.6 becomes

Equation 5.7.

$$a(\cos\theta_2+j\sin\theta_2)+b(\cos\theta_3+j\sin\theta_3)-c(\cos\theta_4+j\sin\theta_4)-d(\cos\theta_1+j\sin\theta_1)=0 \quad (5.7)$$

Separating into the real (x) and imaginary (y) components gives Equation 5.8 and 5.9.

$$a\cos\theta_2+b\cos\theta_3-c\cos\theta_4-d\cos\theta 1=0 \quad (5.8)$$

$$ja\sin\theta_2+jb\sin\theta_3-jc\sin\theta_4-jd\sin\theta_1=0 \quad (5.9)$$

After simplifications and solving for θ_3 Equation 5.8 and 5.9 become Equation 5.10 and 5.11.

$$b\cos\theta_3=-a\cos\theta_2+c\cos\theta_4+d \quad (5.10)$$

$$b\sin\theta_3=-a\sin\theta_2+c\sin\theta_4 \quad (5.11)$$

Squaring Equation 5.10 and 5.11 and adding them results in 5.12:

$$b^2(\sin^2\theta_3+\cos^2\theta_3)=(-a\sin\theta_2+c\sin\theta_4)^2+(-a\cos\theta_2+c\cos\theta_4+d)^2 \quad (5.12)$$

Simplifying and solving for θ_3 and θ_4 leads to:

$$\theta_3=2\tan^{-1}\left(\frac{-E\pm\sqrt{E^2-4DF}}{2D}\right) \quad (5.13)$$

$$\theta_4=2\tan^{-1}\left(\frac{-B\pm\sqrt{B^2-4AC}}{2A}\right) \quad (5.14)$$

Such that:

$$\begin{aligned} A &= \cos\theta_2 - \frac{d}{a} - \frac{d}{c}\cos\theta_2 + \frac{a^2 - b^2 + c^2 + d^2}{2ac} \\ B &= -2\sin\theta_2 \\ C &= \frac{d}{a} - \left(\frac{d}{c} + 1\right)\cos\theta_2 + \frac{a^2 - b^2 + c^2 + d^2}{2ac} \\ D &= \cos\theta_2 - \frac{d}{a} + \frac{d}{b}\cos\theta_2 + \frac{c^2 - d^2 - a^2 - b^2}{2ab} \\ E &= -2\sin\theta_2 \\ F &= \frac{d}{a} + \left(\frac{d}{b} - 1\right)\cos\theta_2 + \frac{c^2 - d^2 - a^2 - b^2}{2ab} \end{aligned} \tag{5.15}$$

Note that there are two solutions for both θ_3 and θ_4. This corresponds to the open and crossed kinematic configurations.

5.3.2 Velocity Analysis

The velocity analysis, like the position analysis, follows Norton [31], but the velocity analysis done in Norton is only for a one DOF fourbar so this derivation does not follow directly. As in the position analysis, the velocity analysis starts off with the vector loop equation shown in Equation 5.5. Equation 5.5 is again converted into complex form as in Equation 5.6. Taking the derivative of Equation 5.6 with respect to time leads to:

$$jae^{j\theta_2}\frac{d\theta_2}{dt} + jbe^{j\theta_3}\frac{d\theta_3}{dt} - jce^{j\theta_4}\frac{d\theta_4}{dt} - \dot{d}e^{j\theta_1} = 0 \tag{5.16}$$

Substituting the Euler equivalents, ω_i for $\frac{d\theta_i}{dt}$ and multiplying the j's through:

$$a\omega_2(j\cos\theta_2 - \sin\theta_2) + b\omega_3(j\cos\theta_3 - \sin\theta_3) - c\omega_4(j\cos\theta_4 - \sin\theta_4) - \dot{d}(\cos\theta_1 + j\sin\theta_1) = 0 \tag{5.17}$$

Equation 5.17 is then separated into real (x) and imaginary (y) components respectively.

$$-a\omega_2 \sin\theta_2 - b\omega_3 \sin\theta_3 + c\omega_4 \sin\theta_4 - \dot{d}\cos\theta_1 = 0 \tag{5.18}$$

$$ja\omega_2 \cos\theta_2 + jb\omega_3 \sin\theta_3 - jc\omega_4 \cos\theta_4 - j\dot{d}\sin\theta_1 = 0 \tag{5.19}$$

Solving Equation 5.18 for ω_3 and Equation 5.19 for ω_4:

$$\omega_3 = \frac{-a\omega_2 \sin\theta_2 + c\omega_4 \sin\theta_4 - \dot{d}\cos\theta_1}{b\sin\theta_3} \tag{5.20}$$

$$\omega_4 = \frac{a\omega_2 \cos\theta_2 + b\omega_3 \sin\theta_3 - dotd\sin\theta_1}{c\cos\theta_4} \tag{5.21}$$

Substituting Equation 5.20 back into Equation 5.19, Equation 5.21 into Equation 5.18 and simplifying yields:

$$\omega_3 = \frac{a\omega_2 \sin(\theta_4 - \theta_2) - \dot{d}\cos(\theta_1 - \theta_4)}{b\sin(\theta_3 - \theta_4)} \tag{5.22}$$

$$\omega_4 = \frac{a\omega_2 \sin(\theta_2 - \theta_3) + \dot{d}\cos(\theta_1 - \theta_3)}{c\sin(\theta_4 - \theta_3)} \tag{5.23}$$

As previously stated the newly derived equations were checked against those from [31]. It can be seen that Equation 5.22 and 5.23 differ from the equations given by Norton [31] by only one term, the $\dot{d}$ term. Allowing $\dot{d} = 0$ the equations become exactly the same.

5.3.3 Acceleration Analysis

The acceleration analysis starts at Equation 5.16. Taking the derivative of Equation 5.16 with respect to time again, the second derivative of the original vector loop Equation 5.6 yields:

$$\left(ja\alpha_2 e^{j\theta_2} + j^2 a\omega_2^2 e^{j\theta_2}\right) + \left(jb\alpha_3 e^{j\theta_3} + j^2 b\omega_3^2 e^{j\theta_3}\right) - \left(jc\alpha_4 e^{j\theta_4} + j^2 c\omega_4^2 e^{j\theta_4}\right) - \left(\ddot{d}e^{j\theta_1}\right) = 0 \quad (5.24)$$

Substituting the Euler equivalents and simplifying:

$$\begin{aligned} &a\alpha_2(-\sin\theta_2 + j\cos\theta_2) - a\omega_2^2(\cos\theta_2 + j\sin\theta_2) + b\alpha_3(-\sin\theta_3 + j\cos\theta_3) \\ &\quad - b\omega_3^2(\cos\theta_3 + j\sin\theta_3) - c\alpha_4(-\sin\theta_4 + j\cos\theta_4) - a\omega_4^2(\cos\theta_4 + j\sin\theta_4) \\ &\quad - \ddot{d}(\cos\theta_1 + j\sin\theta_1) = 0 \end{aligned} \quad (5.25)$$

Separating into real (x) and imaginary (y) components respectively:

$$\begin{aligned} &- a(\alpha_2\sin\theta_2 + \omega_2^2\cos\theta_2) - b(\alpha_3\sin\theta_3 + \omega_3^2\cos\theta_3) \\ &\quad + c(\alpha_4\sin\theta_4 + \omega_4^2\cos\theta_4) - \ddot{d}\cos\theta_1 = 0 \end{aligned} \quad (5.26)$$

$$\begin{aligned} &ja\left(\alpha_2\cos\theta_2 - \omega_2^2\sin\theta_2\right) + jb\left(\alpha_3\cos\theta_3 - \omega_3^2\sin\theta_3\right) \\ &\quad - jc\left(\alpha_4\cos\theta_4 - \omega_4^2\sin\theta_4\right) - j\ddot{d}\cos\theta_1 = 0 \end{aligned} \quad (5.27)$$

Solving Equation 5.26 for α_3 and Equation 5.27 for α_4:

$$\alpha_3 = \frac{-a\alpha_2\sin\theta_2 - a\omega_2^2\cos\theta_2 - b\omega_3^2\cos\theta_3 + c\alpha_4\sin\theta_4 + c\omega_4^2\cos\theta_4 - \ddot{d}\cos\theta_1}{b\sin\theta_3} \quad (5.28)$$

$$\alpha_4 = \frac{a\alpha_2\cos\theta_2 - a\omega_2^2\sin\theta_2 + b\alpha_3\cos\theta_3 - b\omega_3^2\sin\theta_3 + c\omega_4^2\sin\theta_4 - \ddot{d}\sin\theta_1}{c\cos\theta_4} \quad (5.29)$$

Substituting Equation 5.28 into Equation 5.27 and Equation 5.29 into Equation 5.26 and simplifying leads to:

$$\alpha_4 = \frac{a\alpha_2 \sin(\theta_2 + \theta_3) + a\omega_2^2 \cos(\theta_2 - \theta_3) + b\omega_3^2 - c\omega_4^2 \cos(\theta_4 - \theta_3) + \ddot{d}\cos(\theta_1 - \theta_3)}{c \sin(\theta_4 - \theta_3)} \tag{5.30}$$

and

$$\alpha_3 = \frac{a\alpha_2 \sin(\theta_2 - \theta_4) + a\omega_2^2 \cos(\theta_2 - \theta_4) + b\omega_3^2 \cos(\theta_3 - \theta_4) - c\omega_4^2 + \ddot{d}\cos(\theta_1 - \theta_4)}{b \sin(\theta_4 - \theta_3)} \tag{5.31}$$

Like the position and velocity analysis, Equation 5.30 and 5.31 were also checked against [31]. Setting $\ddot{d}$ equal to zero does not instantly cause the equations to degrade to those given in [31]. However, with the application of some basic trigonometric identities the equations can be shown to be the same.

Using just the kinematic analysis (position, velocity and acceleration) a kinematic model could be made. However, because the block is not actually attached to the other links this would not be sufficient. The forces at the joints must be known to ensure that the block remains in contact with the end effector at all times, as well as, to guarantee a minimal normal force can be maintained.

5.3.4 Force Analysis

It does not matter if a forward or inverse analysis is being performed, the derivation of the force equations is the same. The difference comes in what values are known and what values are not. This derivation follows closely what is shown in Norton [31] but is not the same. In Norton there is a force applied to link b that is not present in the end effector.

Using the standard equations for planar motion:

$$\Sigma F = ma \tag{5.32}$$

and

$$\Sigma T = I_G \alpha \tag{5.33}$$

the dynamic equations of motion can be developed. Applying Equation 5.32 and 5.33 to links a, b, and c (link d remains in the same orientation at all times and is attached to ground) results in:

$$\begin{aligned} F_{12_x} + F_{32_x} &= m_2 a_{G_{2_x}} \\ F_{12_y} + F_{32_y} &= m_2 a_{G_{2_y}} \\ T_{12} + \left(R_{12_x} F_{12_y} - R_{12_y} F_{12_x}\right) + \left(R_{32_x} F_{32_y} - R_{32_y} F_{32_x}\right) &= I_{G_2} \alpha_2 \end{aligned} \tag{5.34}$$

$$\begin{aligned} F_{43_x} - F_{32_x} &= m_3 a_{G_{3_x}} \\ F_{43_y} - F_{32_y} &= m_3 a_{G_{3_y}} \\ \left(R_{43_x} F_{43_y} - R_{43_y} F_{43_x}\right) - \left(R_{23_x} F_{23_y} - R_{23_y} F_{23_x}\right) &= I_{G_3} \alpha_3 \end{aligned} \tag{5.35}$$

$$\begin{aligned} F_{14_x} - F_{43_x} &= m_4 a_{G_{4_x}} \\ F_{14_y} - F_{43_y} &= m_4 a_{G_{14_y}} \\ \left(R_{14_x} F_{14_y} - R_{14_y} F_{14_x}\right) - \left(R_{34_x} F_{43_y} - R_{34_y} F_{43_x}\right) + T_4 &= I_{G_4} \alpha_4 \end{aligned} \tag{5.36}$$

where the R values are defined in Figure 5.8. Reformatting Equation 5.34, 5.35 and

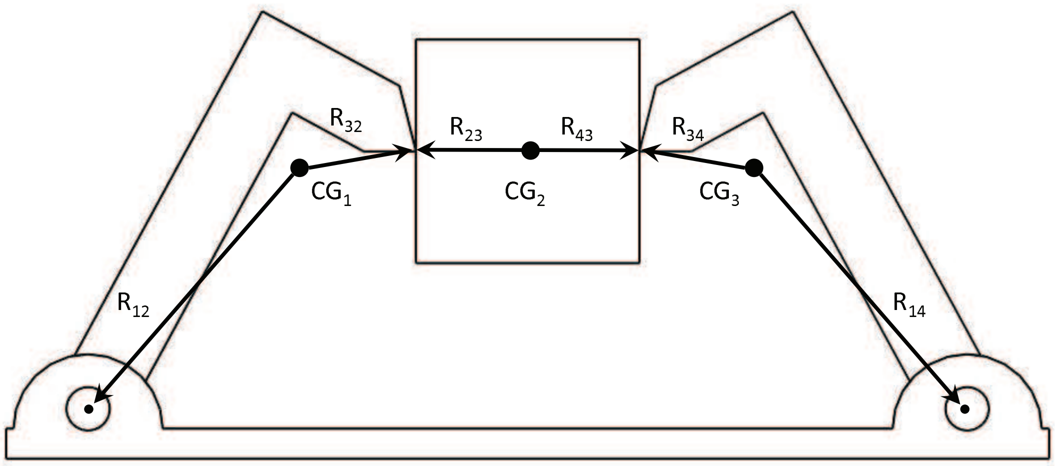


Figure 5.8: End Effector With R Vectors Displayed

5.36 into matrix form allows them to be solved simultaneously.

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -R_{12_y} & R_{12_x} & -R_{32_y} & R_{32_x} & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & R_{32_y} & -R_{32_x} & -R_{43_y} & R_{43_x} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & R_{32_y} & -R_{32_x} & -R_{43_y} & R_{43_x} & 0 \end{bmatrix} \begin{bmatrix} F_{12_x} \\ F_{12_y} \\ F_{32_x} \\ F_{32_y} \\ F_{43_x} \\ F_{43_y} \\ F_{14_x} \\ F_{14_y} \\ T_{12} \end{bmatrix} = \begin{bmatrix} m_2 a_{G_{2_x}} \\ m_2 a_{G_{2_y}} \\ I_{G_2}\alpha_2 \\ m_3 a_{G_{3_x}} \\ m_3 a_{G_{3_y}} \\ I_{G_3}\alpha_3 \\ m_4 a_{G_{4_x}} \\ m_4 a_{G_{4_y}} \\ I_{G_4}\alpha_4 - T_4 \end{bmatrix} \tag{5.37}$$

Combining the kinematic model with the forces creates a dynamic model. The dynamic model was verified using Simulink in Matlab and delivered corroborating results as seen in Figure 5.9. Both Simulink and the model presented here indicate that the mechanism, as currently designed, is in an unstable

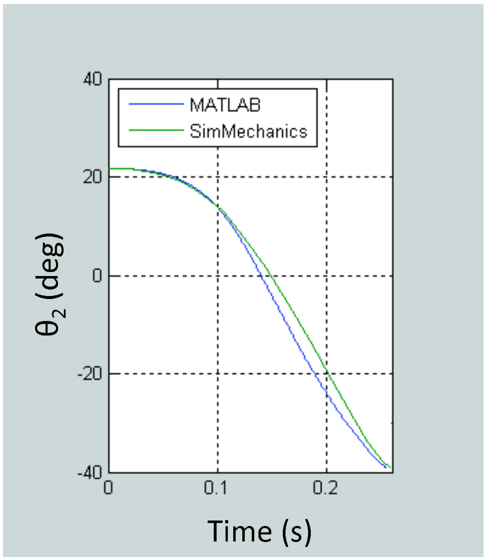


Figure 5.9: Simulink and Analytical Model Output for θ_2

equilibrium position. Given the opportunity the mechanism much prefers the lower energy state of the crossed configuration as can be seen in in the Simulink model shown in Figure 5.10.

The geometry of the end effector was not optimized prior to constructing the prototype as only a proof of concept was desired. Once proof of concept had been established the equations developed for the dynamic model could be implemented

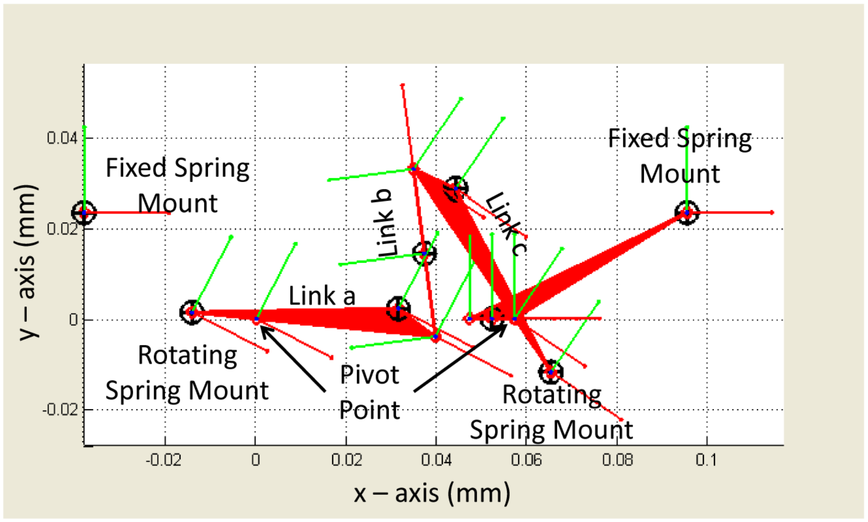


Figure 5.10: Simulink Output

into an optimization routine to improve upon the undesirable characteristics of the mechanism.

CHAPTER 6

Optimization

The end effector is currently in an unstable equilibrium position in Poses 2 and 3. To determine if there is a stable equilibrium position possible within the current design parameters that will allow the desired motion, the analytical model was optimized. The model is not immensely complex as it is not intended to give the ultimate final design and is limited to the open configuration of a fourbar. The hopes of this optimization, as with all optimizations, is to find some set of conditions that would cause the mechanism to work while balancing the compromises that exist. The results from this optimization will assist in better understanding what would be required for a mechanism of this design paradigm to be feasible. Having a better understanding will assist in not only the next design but also possibly in synthesizing more mechanisms for other tasks.

Due to the correlation found between the Simulink model and the analytical model, there was no need to use the Simulink model for the optimization. Although the Simulink model is more robust as it can allow for the crossed configuration, the increase in required computational time for each iteration was counter productive for the desired results.

6.1 Optimization Methodology

The optimization presented is minimizing the angle of rotation for the block with respect to x where x is defined as:

$$x = \begin{bmatrix} aa \\ cc \\ k\_spring1 \\ k\_spring2 \\ a \\ c \\ l\_s10 \\ x\_s1f \\ y\_s1f \\ l\_s20 \\ x\_s2f \\ y\_s2f \end{bmatrix} \tag{6.1}$$

such that:

- aa = Length Of Leaver Arm On Link a
- cc = Length Of Leaver Arm On Link c
- $k\_spring1$ = Spring Rate for Spring 1
- $k\_spring2$ = Spring Rate for Spring 2
- a = Length Of Link a

- c = Length Of Link c
- l_s10 = Free Length Of Spring 1
- x_s1f = x Position For The Fixed End Of Spring 1
- y_s1f = y Position For The Fixed End Of Spring 1
- l_s20 = Free Length Of Spring 2
- x_s2f = x Position For The Fixed End Of Spring 2
- y_s2f = y Position For The Fixed End Of Spring 2

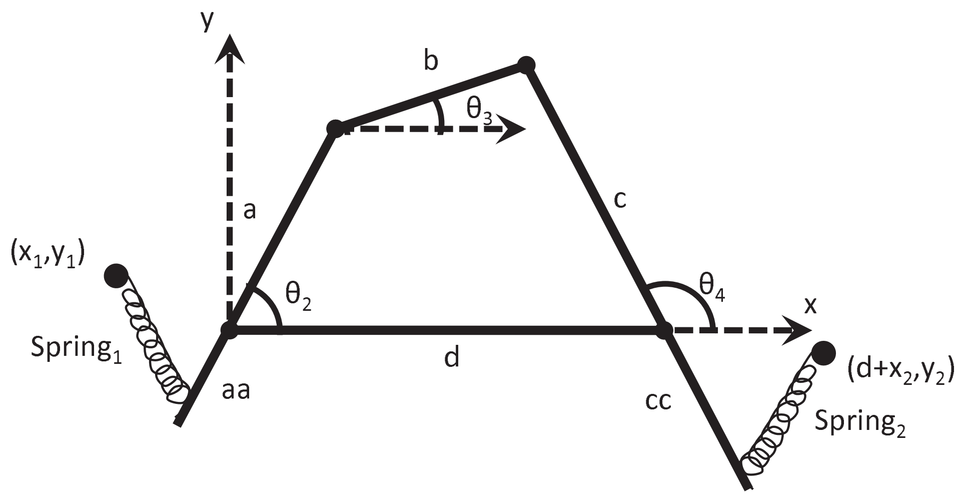


Figure 6.1: Simulink Output

as shown in Figure 6.1 by use of Matlab's fmincon function. fmincon is Matlab's built in function used to minimize continuous functions with constraints. With the ultimate goal as a feasible functioning end effector, minimizing θ_3^2, the rotation angle of the block, is thought to keep the block as level and stable as possible, resulting in easier assembly and a more predictable final orientation. The parameters seen in x were selected to give a large enough work space to allow some freedom but not so large as to become computationally objectionable or to allow configurations that cannot possibly be manufactured.

6.1.1 Constraints

There are two types of constraints that can be placed on the minimization, an equality constraint and an inequality constraint. An inequality constraint requires that the value returned must be less than or equal to zero. This type was used to ensure the normal forces at the contact points with the block are greater than zero. Because a positive force is required, the opposite value of the normal forces returned by the inverse dynamics function plus the desired minimum force is placed in the inequality constraint. The lengths of the links are also constrained in the inequality constraint such that $a + b + c - d - 10 \leq 0$. This insures that $a + b + c$ is a minimum of 10mm longer than d to maintain a safe distance from the pose where all the links fall on the same line. The values for the upper and lower bounds of the initial guesses, a special form of inequality constraint, were selected with the intent of limiting the design space to a useable solution.

If the mechanism encounters a singularity or some other position that cannot be handled, that iteration will stop prematurely. This causes the contact force vectors returned to be shorter than desired. Along with the force vectors the variable time_out, the time the simulation actually ran, is also returned. The short force vectors are the input for the inequality constraints which are returned to fmincon. fmincon attempts to take the gradient between the newly returned constraints and the previous values to determine the next set of guesses. Because the sets of values are not the same length, fmincon mistakenly reads the change as good. To prevent this, values are appended to the short force vectors to create a nonzero gradient sending fmincon back in the correct direction.

The other form of constraint, an equality constraint, was also created requiring time_out to be equal to the desired runtime, referred to as time, assisting in sending fmincon in the correct direction. Putting all of this into standard form can be seen in Equation 6.2 such that, g_1 and g_2 represent the complex calculations required to determine the normal forces at the interfaces between the block and the end effector.

$$\begin{aligned}
\min_{x} \quad & error = \sum_{i=1}^{length(time)} (\theta_{3i})^2 && (6.2)\\
\text{subject to} \quad & length(time) - length(time\_out) = 0\\
& g_1(x) \leq 0\\
& g_2(x) \leq 0\\
& -(a+b+c) + d + 10 \leq 0\\
& 2mm \leq aa \leq 50mm\\
& 2mm \leq cc \leq 50mm\\
& 0 \leq k\_spring1 \leq 50\\
& 0 \leq k\_spring2 \leq 50\\
& 42.5mm \leq a \leq 100mm\\
& 42.5mm \leq c \leq 100mm\\
& 2mm \leq l\_s10 \leq 10mm\\
& -50mm \leq x\_s1f \leq 0\\
& -30 \leq y\_s1f \leq 80\\
& 2mm \leq l\_s20 \leq 10mm\\
& 5mm \leq x\_s2f \leq 50\\
& -30 \leq y\_s2f \leq 80
\end{aligned} \tag{6.3}$$

6.1.2 Assumptions

Along with the assumptions made in the derivation of the equations used to develop the analytical model implemented in the optimization, the following assumptions were also made:

- The springs are ideal

- No friction
- No gravity
- The lever arms fall in line with a and c
- The end effector is always in contact with the block
- The linear velocity, $\dot{d}$, is constant
- The linear acceleration, $\ddot{d}$, is zero

6.2 Program Flow

There are several layers to the optimization program, as seen in Figure 6.2 and 6.3. In these diagrams the program flows from top to bottom and from left to right. The initial guesses go into an umbrella program along with the initial conditions and other constants and constraints. The umbrella program then calls the minimization function fmincon and passes the needed values. As can be seen in Figure 6.3, fmincon then calls the cost function. The cost function then calls the forward dynamics function which returns a value to cost function. Cost function then returns a value to fimincon which calls the non-linear constraints function. The non-linear constraints function then calls the forward dynamics function which returns a value to the non-linear constraints function. Then the non-linear constraints function calls the inverse dynamics function which again returns a value to the non-linear constraints function. The non-linear constraints function then calculates the non-linear constraints, hence the function's name, and returns them to fmincon. Based on the results, fmincon then chooses the next iteration step and starts the process again. This process iterates until a local minimum is found. Each step will be discussed in further detail in Sections 6.2.1 to 6.2.6.

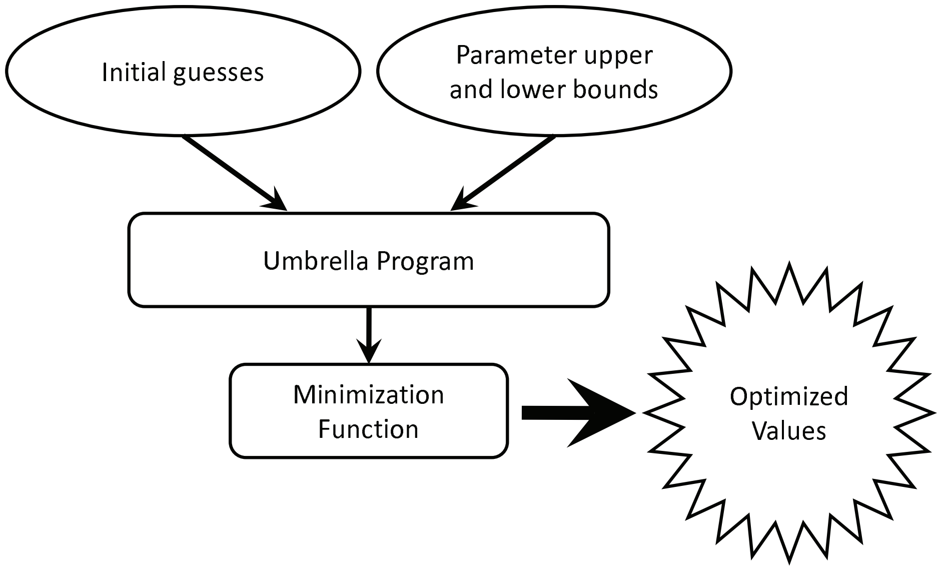


Figure 6.2: Program Overview

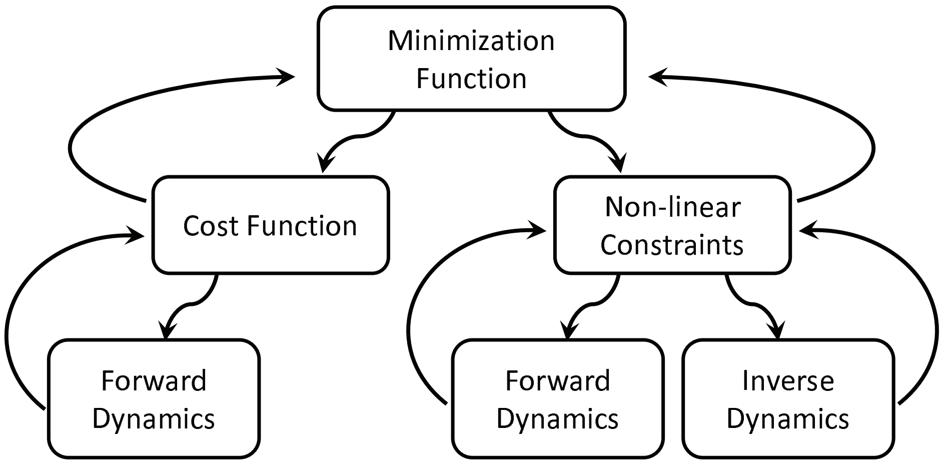


Figure 6.3: Detailed Program Flowchart

6.2.1 Umbrella Program

The umbrella program is the top layer of the program and contains all the fixed parameters and initial guesses. It also takes the final optimized values returned from the minimizing function, fmincon, and outputs them in a graphical format as well as in text. The code for this program can be found in Appendix C.1.

6.2.2 Minimization Function

The minimization function (fmincon) requires an initial guess on the inputs, upper and lower bounds, the cost function (cf) and the non-linear constraint function (nlcf). To avoid the use of global variables an extra vector is also passed to fmincon containing constants and initial conditions.

The minimizing function takes the initial guesses and runs them through the cf and then the nlcf. It then changes the value of the guesses depending on the feedback results returned from the cf and the nlcf. This cycle is repeated until a local minimum is found.

6.2.3 Cost Function

The first function called by the minimization function is the cf and varies from one application to the next. Ultimately the cost function needs to return a measure of the value being minimized, the cost.

Because several of the variables that fmincon changes for every iteration affect the torque placed on the fingers by the springs, θ_2 must be solved using the forward dynamics. To do this the cf makes use of Matlab's ode15i solver which solves implicit differential equations. One of the required inputs for ode15i is the forward dynamics function. This setup returns a time vector along with the corresponding values of θ_2 and ω_2 for each time step.

Using the previously defined position analysis (Section 5.3.1) θ_3 can be calculated for each time step. The cost is then calculated as:

$$cost = \sum_{i=0}^{n} {\theta_{3i}}^2 \tag{6.4}$$

such that n is the number of time steps. The cost is then passed back to fmincon. The code for the cost function can be found in Appendix C.2.

6.2.4 Forward Dynamics Function

A common approach to solving the second order ordinary differential equations is to make use of the state space representation. Generally all of the equations are stated explicitly allowing them to be solved simultaneously relatively easily.

The forward dynamics function is used to solve for the forward dynamics of the mechanism. This is a common application for the state space representation. Due to the complexity of the system of equations for this mechanism, nine coupled second order ordinary differential equations, it is not feasible to solve for everything explicitly. ode15i requires the forward dynamics function to return the differential equations in the state space format in implicit form, as opposed to the standard explicit form. The implicit form can be seen in Equation 6.5:

$$\begin{aligned} f(1) &= {y_1}' - y_2 = 0 \\ f(2) &= g(t, y_1, {y_1}', y_2, {y_2}') = 0 \end{aligned} \tag{6.5}$$

such that ${y_1}' = \omega_2$, $y_2 = \omega_2$ and $g(t, y_1, {y_1}', y_2, {y_2}')$ is a function of θ_2, ω_2 and α_2. In this case these equations are Equation 5.37 found in Section 5.3.4 stated in implicit form. The physical code can be found in Appendix C.3.

6.2.5 Non-linear Constraint Function

The nlcf is the second function called by the minimizing function, fmincon. The nlcf in turn calls ode15i which uses forward dynamics function and again returns θ_2 and ω_2. α_2 is then calculated by taking the numerical derivative of ω_2 using the forward difference method for the first point, the backwards difference method for the last point and the center difference method for all points between. These three vectors θ_2, ω_2 and α_2 are then passed to the inverse dynamics which calculates and returns the forces at each joint. The nlcf returns two vectors, one containing the inequality constraints and one containing the equality constraints. This code can be found in Appendix C.4.

6.2.6 Inverse Dynamics

Once α_2 is calculated in the nlcf it can be passed into the inverse dynamics function along with θ_2 and ω_2. These values are then used to calculate $\theta_3, \theta_4, \omega_3, \omega_4, \alpha_3$ and α_4. These values are then used directly in Equation 5.37 to calculate the forces in component form at each joint, as well as, the torques required. The forces are then returned to the nlcf in component form. This code can be seen in Appendix C.5.

6.2.7 Optimization Results

Due to the runtime required to do an exhaustive search, it was not feasible. Therefore, many different initial guess combinations were run varying all of the values, including $\dot{d}$ and run time, for extended periods (over 12 hours) without the optimization coming to a conclusion, the maximum acceptable change for $f(x)$ was changed and value time was shortened from .054 seconds to .02 seconds, with $\dot{d}$ of -507.963 mm/s and 101 time steps resulting in Figure 6.4 and returning the following optimized values:

$$x_{Optimized} = \begin{bmatrix} aa \\ cc \\ k\_spring1 \\ k\_spring2 \\ a \\ c \\ l\_s10 \\ x\_s1f \\ y\_s1f \\ l\_s20 \\ x\_s2f \\ y\_s2f \end{bmatrix} = \begin{bmatrix} 3.3829mm \\ 3.9267mm \\ 29.7689N/mm \\ 34.2676N/mm \\ 87.3532mm \\ 56.4365mm \\ 2.0000mm \\ -41.4991mm \\ 44.6810mm \\ 2.6123mm \\ 41.2402mm \\ 70.6307mm \end{bmatrix} \tag{6.6}$$

When this problem was approached it was assumed to be a symmetrical problem. Therefore the mechanism was designed with symmetry in mind. As can be seen from the optimized solution the problem is *not* symmetrical. One cause for this is only one side of the mechanism is actuated at a time instead of both actuating simultaneously as originally intended.

Figure 6.5 shows θ_3 with respect to time. In this figure it can be seen that the longer it is allowed to run the further from zero θ_3 becomes. If the finger tips on the end effector can be designed to allow the change from planar to point contact with the minimal change in d the value of θ_3 will also be minimized.

The torque applied at Joints 1 and 4 from the springs can be seen in Figures

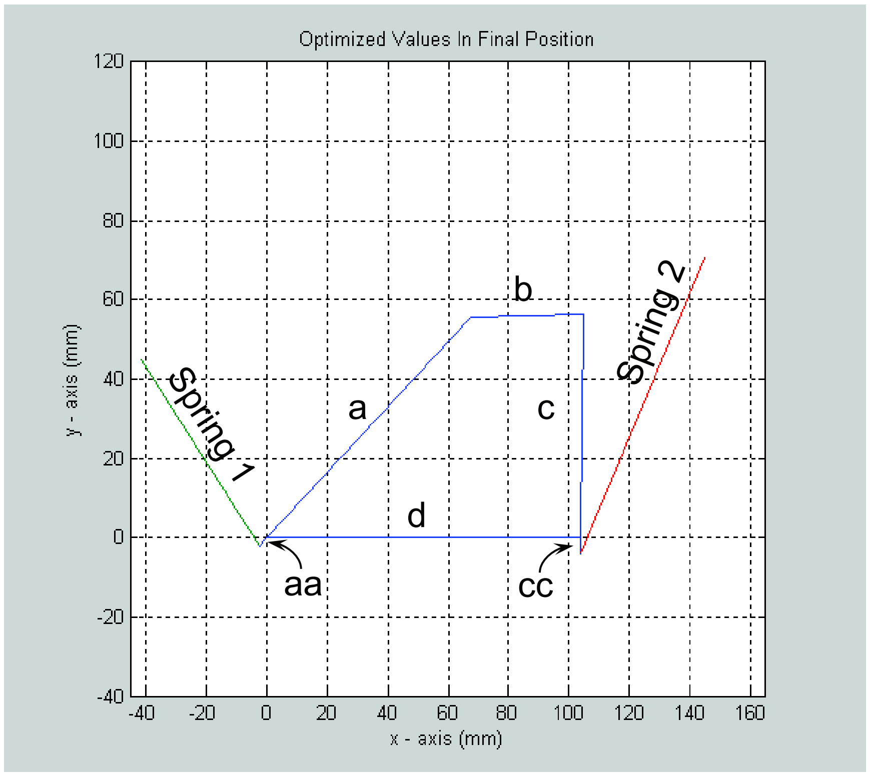


Figure 6.4: End Effector With Optimized Values In Final Position

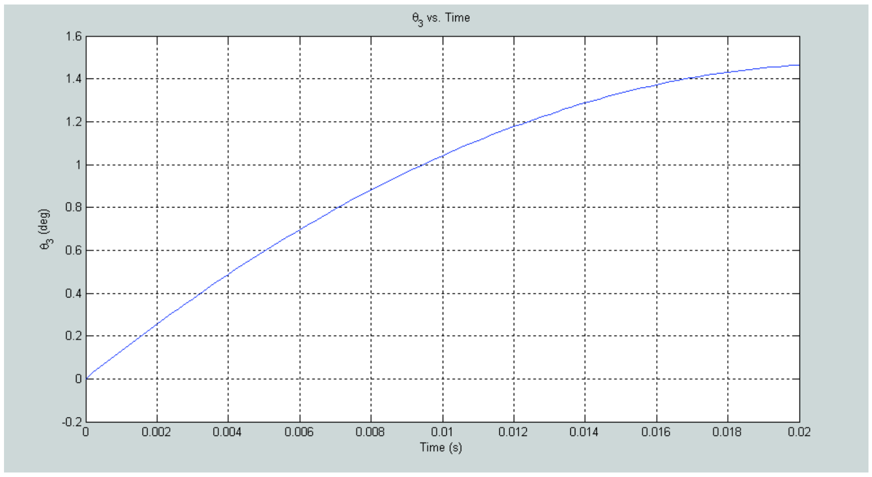


Figure 6.5: θ_3 vs. Time

6.6 and 6.7 respectively. The torque at Joint 1 can be seen to start with a parabolic form and becomes nearly linear while the torque in Joint 4 is nearly linear for the entire run. Also, it appears as if the torque on Joint 1 is increasing while the torque on Joint 4 is decreasing. Due to the sign convention, the torque on Joint 1 is negative resulting in the magnitude of the torque actually decreasing with time. When looking at the actual values, the torque on Joint 1 only varies by about 0.9 N/mm while the torque on Joint 4 varies by 1400 N/mm. Due to the geometry of the solution, torsional springs may be a better choice than the currently used linear springs. Torsional springs would more easily create the required forces and fit within the size needed to make this mechanism feasible.

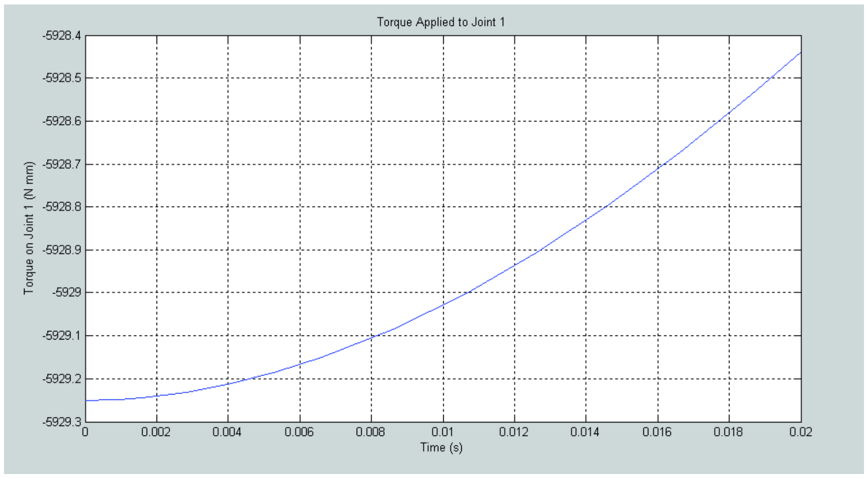


Figure 6.6: Torque Applied To Joint 1 In Optimized Configuration

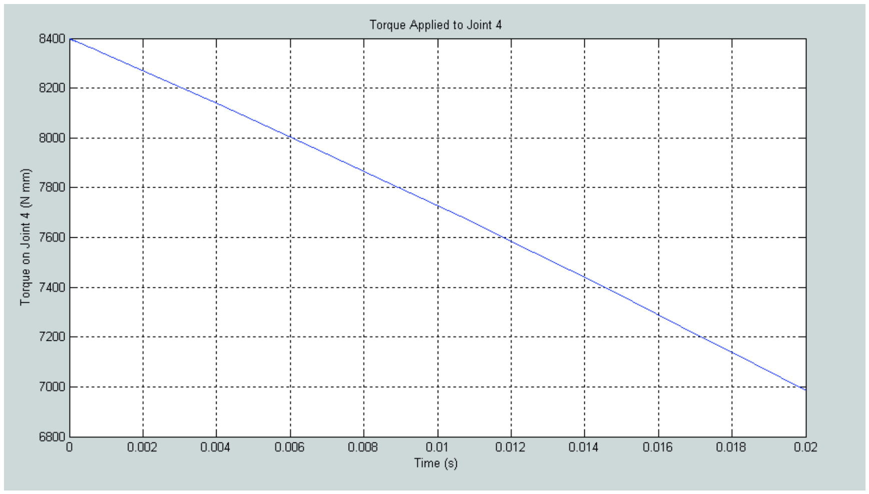


Figure 6.7: Torque Applied To Joint 4 In Optimized Configuration

Because this solution to the optimization was found, analytically there is a feasible solution. This solution is based on one set of initial guesses and an increase in the acceptable variance in $f(x)$ between steps in a twelve dimensional space, the solution shown here is not likely to be a global minimum. There is no way of guaranteeing a global minimum, within the set constraints, will ever be found. Even an exhaustive search can not guarantee a global minimum. If this set of optimized values is allowed to run for longer than 0.02 seconds, θ_3 reaches a maximum and then returns to zero before continuing along that path and becoming negative. If the actuation can be set to take advantage of this it may be a better design as θ_3 does return to zero, but because of the design of the cost function this would return a large error and would not be and acceptable solution by this optimization. An optimization routine will only answer the question asked, which is not necessarily the answer desired. This one was asked to minimize the sum of θ_3^2. If as previously mentioned the other design paradigm is actually better it would at least require a different cost function if not an entirely different optimization program. Just adding or subtracting parameters to the x vector does not assure a better answer in the selected paradigm either. If the workspace is made to large finding a local minimum *close* to the initial guesses may take much longer than expected and will limit the usefulness of the program but if the space is to small the solution may never be found.

CHAPTER 7

Conclusions

This thesis looked at the bin picking problem focusing solely on the end effector, where most research has been done for the vision systems. Design of the end effector was approached with the concept that the end effector and part were one set of links instead of two separate objects. Using this approach made it possible for the part to be considered a link in the mechanism that needed to be manipulated. This lead to the use of reconfigurable mechanisms and ultimately a metamorphic mechanism of type c. A mechanism was designed that was thought to be able to allow the desired manipulation of the part. During testing it was shown to successfully complete the task but not as reliably as desired. The geometry of the mechanism had not been optimized prior to the prototype. To allow for the optimization a dynamic analysis was completed. Because the part and end effector were considered as one mechanism it allowed the analysis to be conducted assuming a closed loop chain and the application of classical dynamics. Using the dynamic model developed, an optimization program was created. The results of the optimization showed that a numerical solution exists that will maintain the contact between the block and the end effector allowing for the assumption that they are linked to be acceptable.

Due to the geometry of the optimized model, primarily the location of the spring mounts and the values of the springs, it is infeasible. The spring mounts are such that a part could not be picked up on a flat surface, let a lone out of a bin of parts. It may be possible to use torsion springs in place of the extension springs which may allow for a more reasonable layout as well as more attainable springs rates. It also seems that minimizing the amount of rotation of the arms would reduce the dynamic effects within the mechanism. Both of these issues could be dealt with by redesigning the fingers so that the degrees the fingers rotate is minimized when changing between phases 2 and 3 and allowing the fingers to be closer to parallel with the sides of the part. This would also help minimize the amount of space required around the part to grasp it. If the actuation could be done such that both sides of the end effector actuate simultaneously, it is believed that a more symmetrical solution could be found which would lead to a more elegant and simpler design.

It was seen that the dynamics involved, caused by the actuation of a single side rather than the simultaneous actuation of both sides of the end effector, had undesirable effects. Since it is not feasible to have perfectly simultaneous actuation of both sides of the end effector another design may be more effective. If the change in contact (point to planar or planar to point) could be made without the need for rotation of the arms, allowing the part to remain stationary and avoiding the dynamic problems, it may be a better option. This is a completely different design paradigm and was not explored here. The problem of needing access to opposing parallel sides still needs to be addressed as well as it is a core assumption in this research.

REFERENCES

[1] M.T. Mason and J.K. Salisbury. *Robot Hands and the Mechanics of Manipulation.* MIT Press, Cambridge, MA, 1985.

[2] H. Saldner. PalletPicker 3D, the solution for picking of randomly placed parts. *Assembly Automation*, 23(1):29–31, 2003.

[3] B. Carlisle, K. Goldberg, A. Rao, and J. Wiegley. A pivoting gripper for feeding industrial parts. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, volume 2, pages 1650–1655, San Diego, CA, May 1994.

[4] G. Boothroyd, P. Dewhurst, and W. Knight. *Product Design for Manufacture and Assembly.* Marcel Dekker, Inc., New York, 1994.

[5] J.S. Dai and J. Rees Jones. Mobility in metamorphic mechanisms of foldable/ erectable kinds. *Journal of Mechanical Design*, 121:375–382, September 1999.

[6] Jian S. Dai and J. Rees Jones. Matrix representation of topological changes in metamorphic mechanisms. *Journal of Mechanical Design*, 127:837–840, July 2005.

[7] I. Ebert-Uphoff and P.A. Voglewede. On the connections between cable-driven robots, parallel manipulators and grasping. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pages 4521–4526, New Orleans, April 2004.

[8] Tao Zhang, Gordon Smith, and Ken Goldberg. Compensatory grasping with the parallel jaw gripper. In *4th Workshop on Algorithmic Foundations of Robotics*, 1999.

[9] B. Yeung and J.K. Mills. Design of a six dof reconfigurable gripper for flexible fixtureless assembly. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 34(2):226–235, May 2004.

[10] K. Wohlhart. Kinematotropic linkages. In J. Lenarcic and V. Parenti-Castelli, editors, *Recent Advances in Robot Kinematics*, pages 359–368. Kluwer Academic, Dordrecht, The Netherlands, 1996.

[11] C. Galletti and P. Fanghella. Single-loop kinematotropic mechanisms. *Mechanism and Machine Theory*, 36(6):743–761, June 2001.

[12] C. Galletti and P. Fanghella. Multiloop kinematotropic mechanisms. In *Proceedings of the ASME DETC*, Montreal, CA, September 2002.

[13] P.A. Voglewede. Novel design of a robotic gripper allowing for in-hand manipulation. In *Proceedings of the 2007 ASME Design Engineering Technical Conference*, Las Vegas, NV, September 2007. Paper Number: DETC2007-34797.

[14] Jiansheng Dai and Zhang Qixian. Metamorphic mechanisms and their configuration models. *Chinese Journal of Mechanical Engineering*, 13(3):212, 2000.

[15] Z.H. Lan and R. Du. Representation of topological changes in metamorphic mechanisms with matrices of the same dimension. *Journal of Mechanical Design*, 130, July 2008.

[16] H.-S. Yan and C.-H. Kuo. Topological representations and characteristics of variable kinematic joints. *ASME Journal of Mechanical Design*, 128:384–391, March 2006.

[17] Jian S. Dai and Delun Wang. Geometric analysis and synthesis of the metamorphic robotic hand. *Journal of Mechanical Design*, 129:1191–1197, November 2007.

[18] John J. Parise, Larry L. Howell, and Spencer P. Megleby. Ortho-planar mechanisms. In *26th Biennial Mechanisms and Robotics Congerences, DETC2000/MECH-14193*, pages 1279–1286, Baltimore, Maryland, September 2000. ASME.

[19] Craig P. Lusk and Larry L. Howell. Design space of single-loop planar folded micro mechanisms with out-of-plane motion. *Journal of Mechanical Design*, 128:1092–1100, September 2006.

[20] Daniel W. Carroll, Spencer P. Magleby, Lary L. Howell, Robert H. Todd, and Craig P. Lusk. Simplified manufacturing through a metamorphic process for compliant ortho-planar mechanisms. In *2005 ASME International Mechanical Engineering Congress and Exposition*, pages 389–399, Orlando, FL, November 2005.

[21] Liping Zang, Delun Wang, and Jian S. Dai. Biological modeling and evolution based synthesis of metamophic mechanisms. *Journal of Mechanical Design*, 130, July 2008.

[22] Jennifer E. Walter, Jennifer L. Welch, and Nancy M. Amato. Concurrent metamorphosis of hexagonal robot chains into simple connected configurations. *IEEE Transactions on Robotics ;and Automation*, 18(6):945–956, December 2002.

[23] G. Gogu. Mobility of mechanisms: A critical review. *Mechanism and Machine Theory*, 40(9):1068–1097, September 2005.

[24] Jian S. Dai, Zhen Huang, and Harvy Lipkin. Screw system analysis of parallel mechanisms and applications to constraint and mobility study. In *Procedings of DETC'04*, pages 1569–1582, Salt Lake City, Utah, September 2004. ASME DETC.

[25] Jian S. Dai, Duanling Li, Qixian Zhang, and Guoguang Jin. Mobility analysis of a complex structured ball based on mechanism decomposition and equivalent

screw system analysis. *Mechanism and Machine Theory*, 39:445–458, 2004.

[26] C.-H. Kuo and H.-S. Yan. On the mobility and configuration singularity of mechanisms with variable topologies. *ASME Journal of Mechanical Design*, 129:617–624, June 2007.

[27] V.-D. Nguyen. Constructing force-closure grasps. *The International Journal of Robotics Research*, 7(3):3–16, June 1988.

[28] T.M. Mason. *Mechanics of Robotic Manipulation.* MIT Press, August 2001.

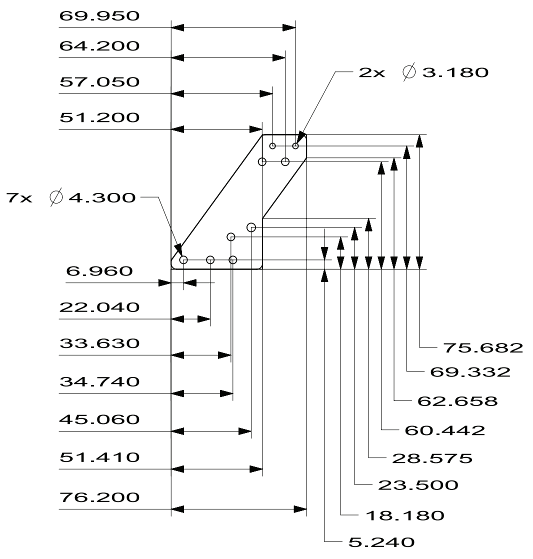
[29] Liping Zhang, Delun Wang, and Jian S. Dai. Biological modeling and evolution based synthesis of metamorphic mechanisms. *Journal of Mechanical Design*, 130:072303–1–072302–11, July 2008.

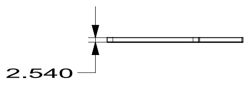
[30] L.-W. Tsai. *Mechanism Design: Enumeration of Kinematic Structures According to Function.* CRC Press LLC, Boca Raton, FL, 2001.

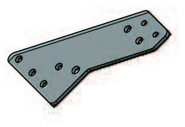
[31] R.L. Norton. *Design of Machinery: An Introduction to the synthesis and analysis of mechanisms and machines.* McGraw-Hill, New York, NY, 3rd edition, 2004.

APPENDIX A

Gripper Body Components

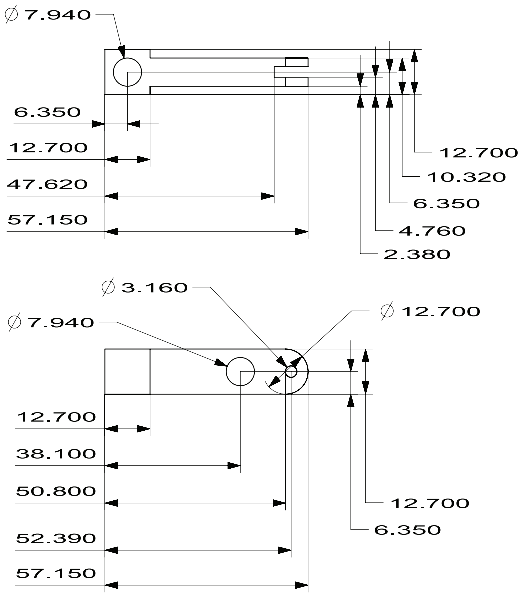


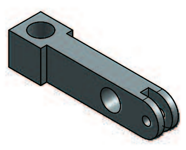




Marquette University	Dept. of Mech. Eng.	Name:	Jacob A. Ziesmer	
Project:	End Effector	Scale:	1:2	
Part:	Main Plate	Tolerance:	0.0	+/-0.2
Quantity:	4		0.00	+/-0.02
Material:	Aluminum		0.000	+/-0.005

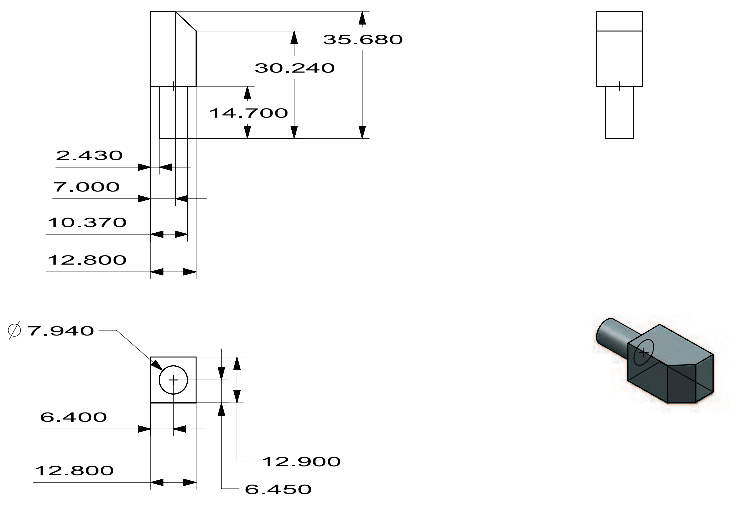
Figure A.1: Side Plate





Marquette University	Dept. of Mech. Eng.	Name:	Jacob A. Ziesmer	
Project:	End Effector	Scale:	1:1	
Part:	Rotating Arm	Tolerance:	0.0	+/-0.2
Quantity:	2		0.00	+/-0.02
Material:	Aluminum		0.000	+/-0.005

Figure A.2: Finger



Marquette University	Dept. of Mech. Eng.	Name:	Jacob A. Ziesmer	
Project:	End Effector	Scale:	1:1	
Part:	Arm Finger	Tolerance:	0.0	+/–0.2
Quantity:	2		0.00	+/–0.02
Material:	Steel		0.000	+/–0.005

Figure A.3: Finger Tip

APPENDIX B

Actuator Assembly Components

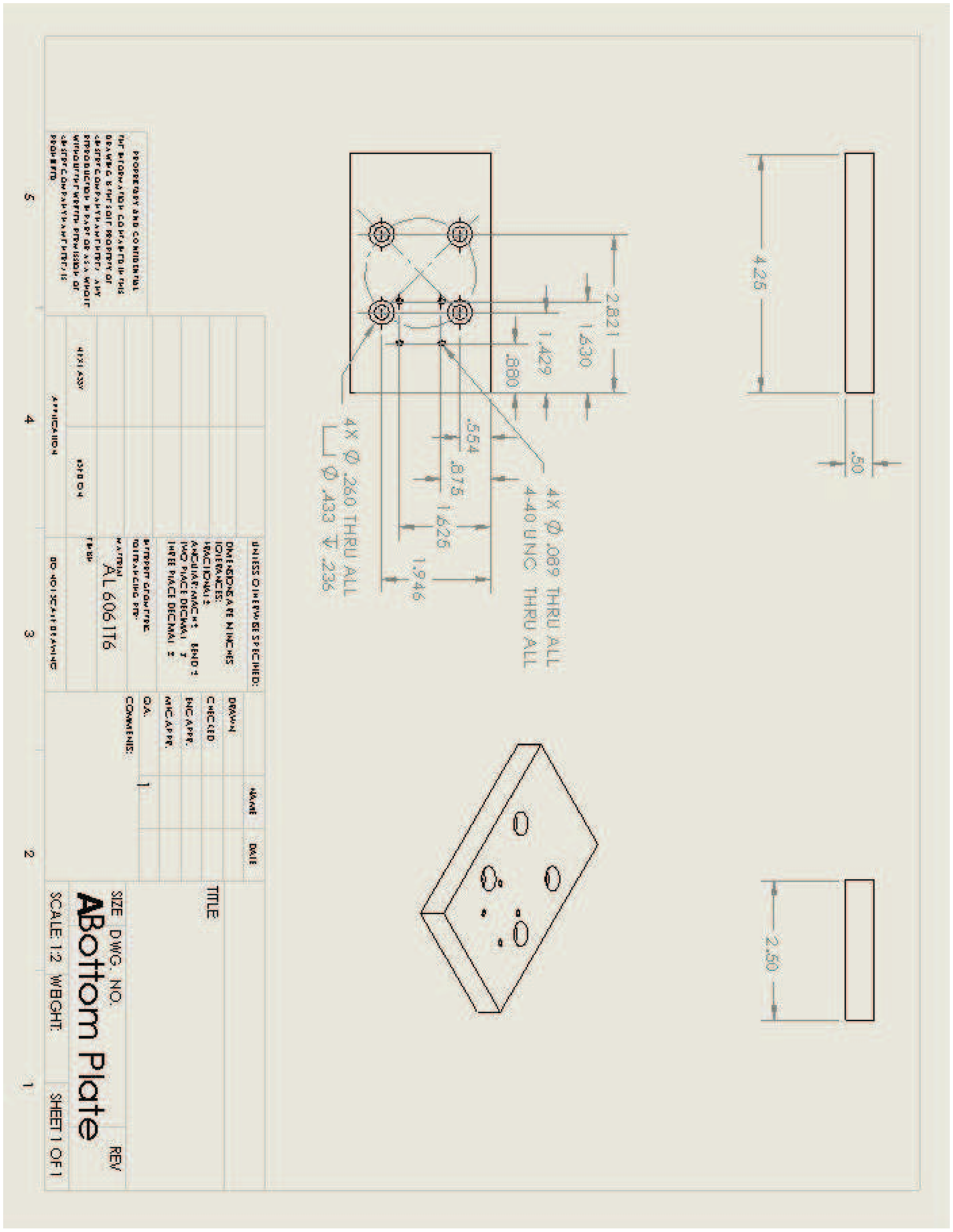


Figure B.1: The base plate adaptor between the end effector and the robot.

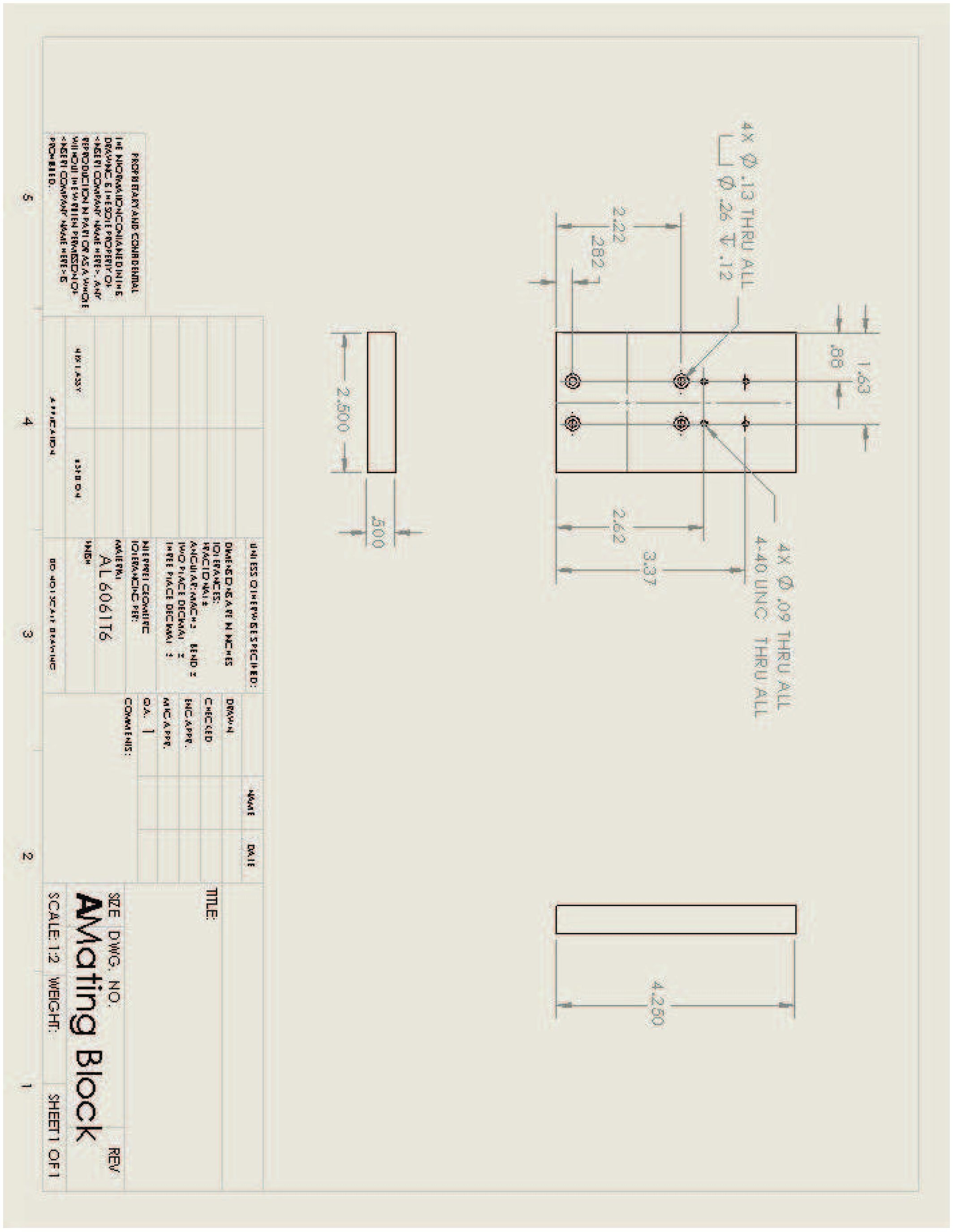


Figure B.2: The mating block mates the two linear actuators.

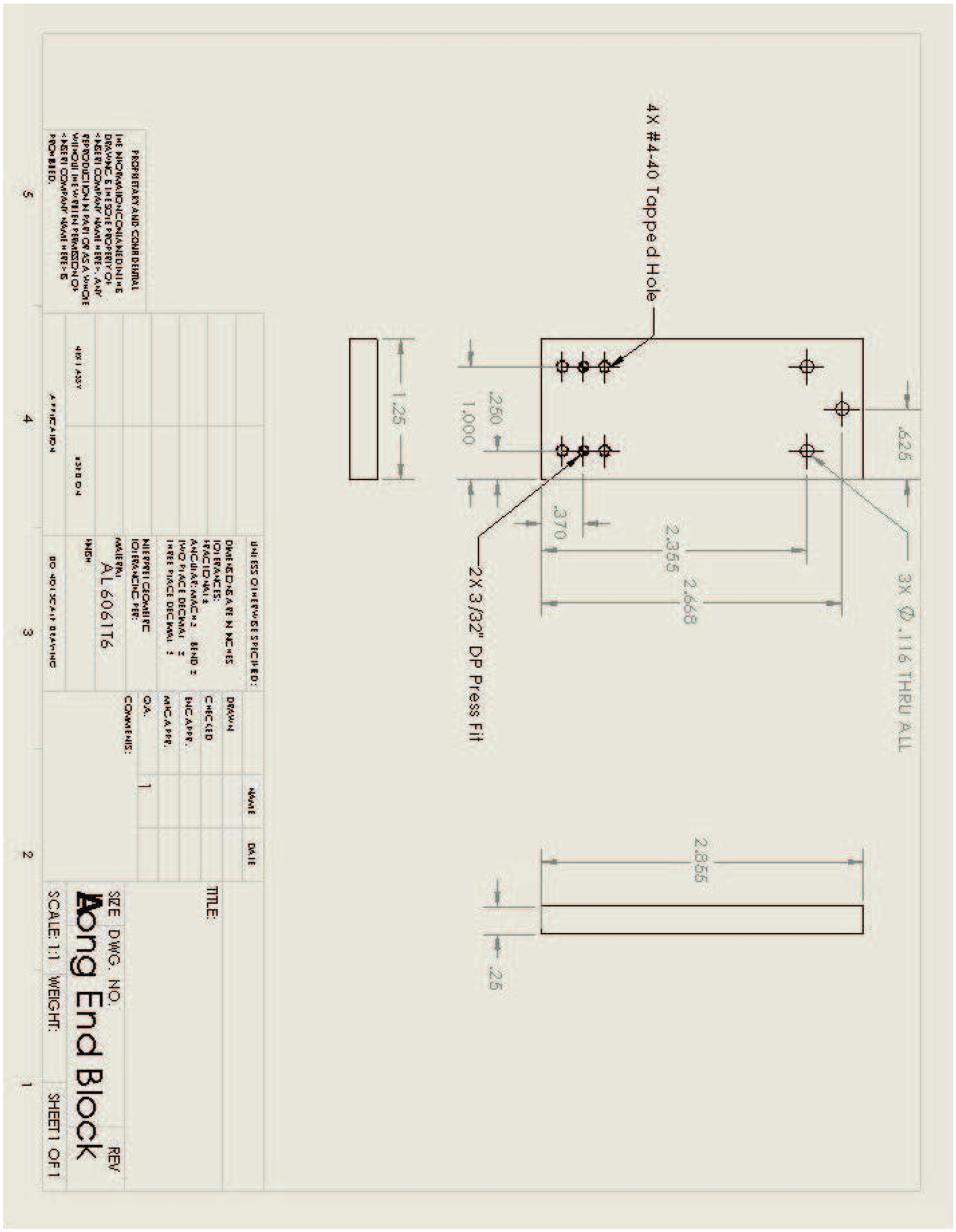


Figure B.3: The long end block attaches to the linear actuator closest to the robot.

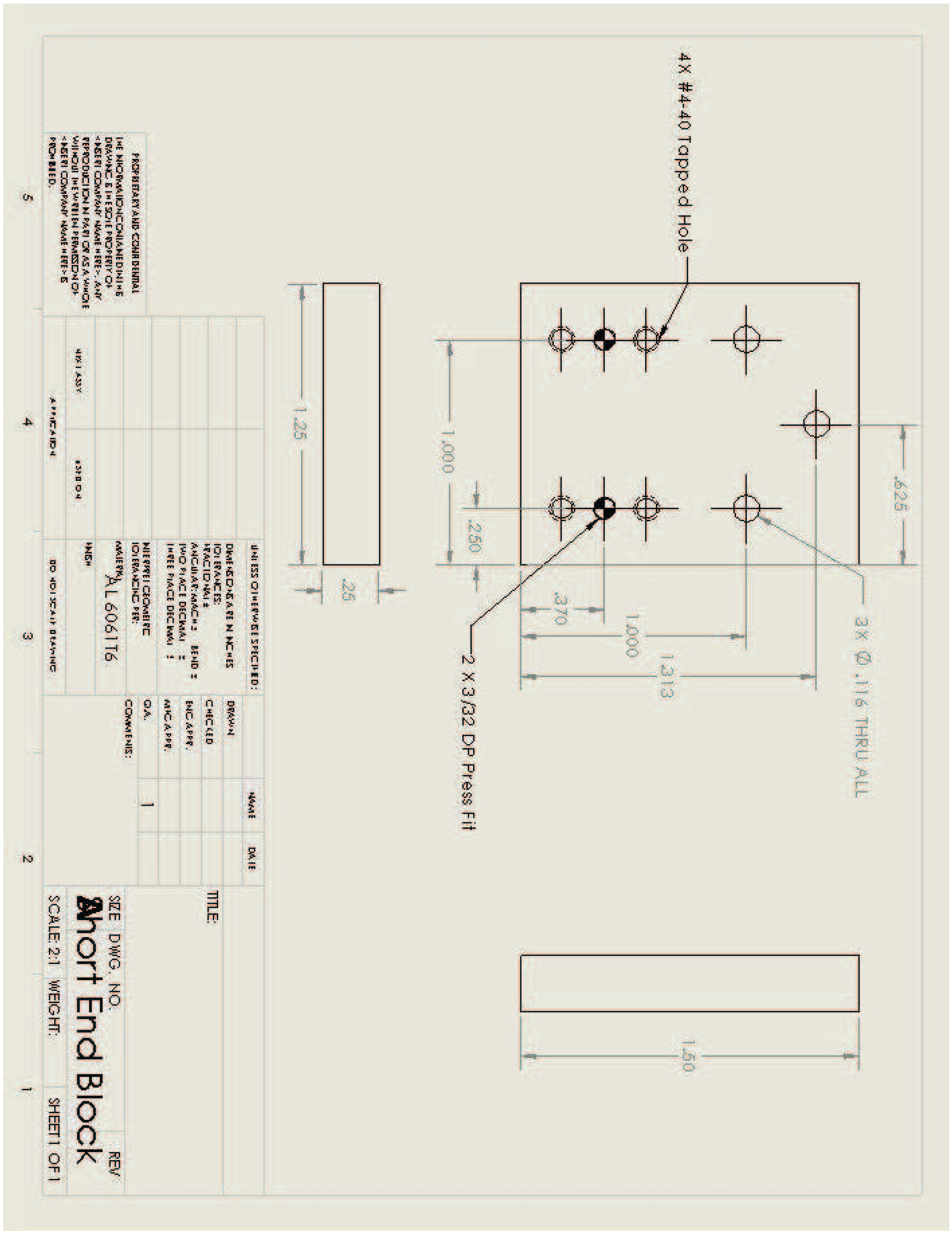


Figure B.4: The short end block attaches to the linear actuator furthest from the robot.

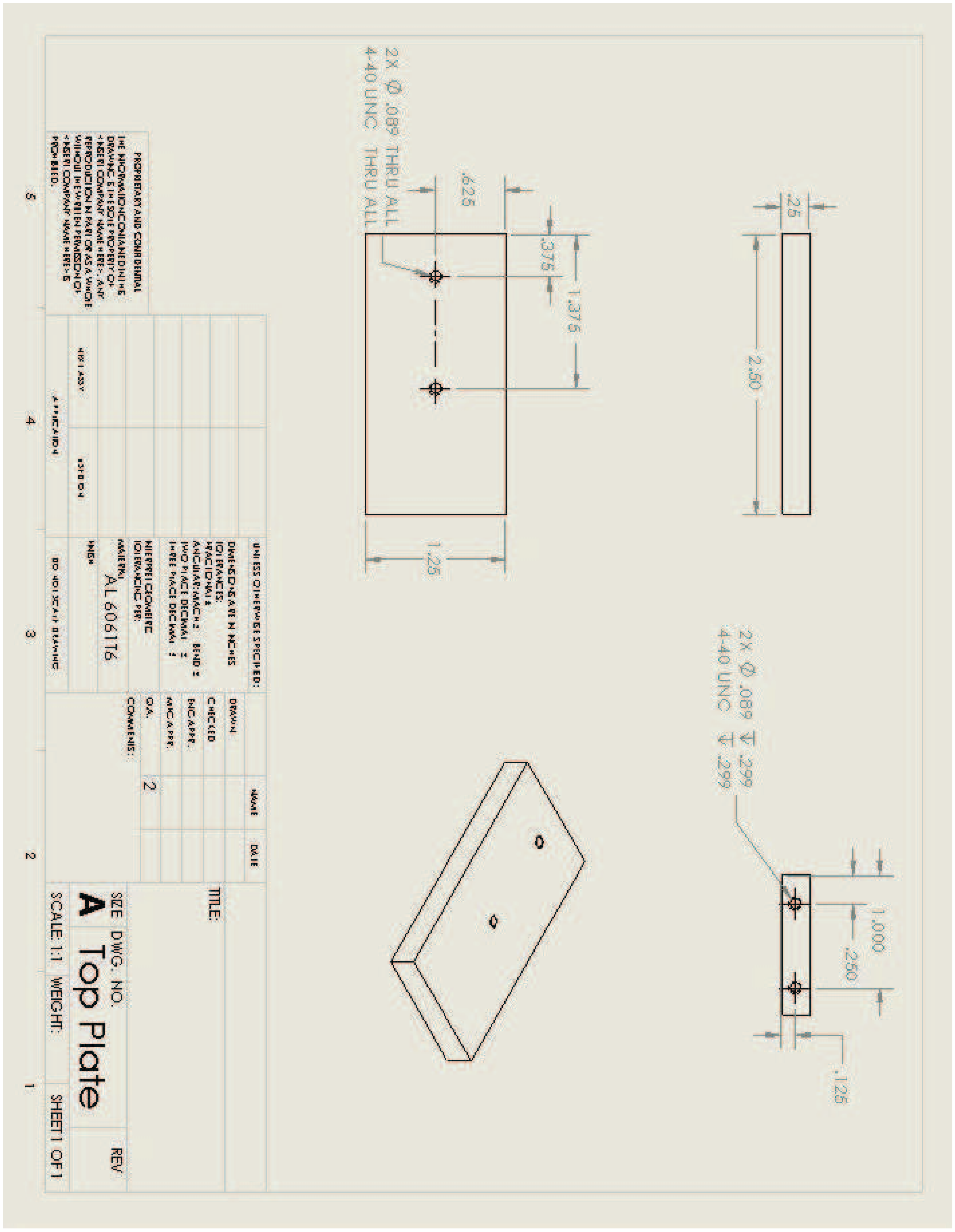


Figure B.5: The top plate connects to the end plate and comes just under halfway back to the center of the end effector.

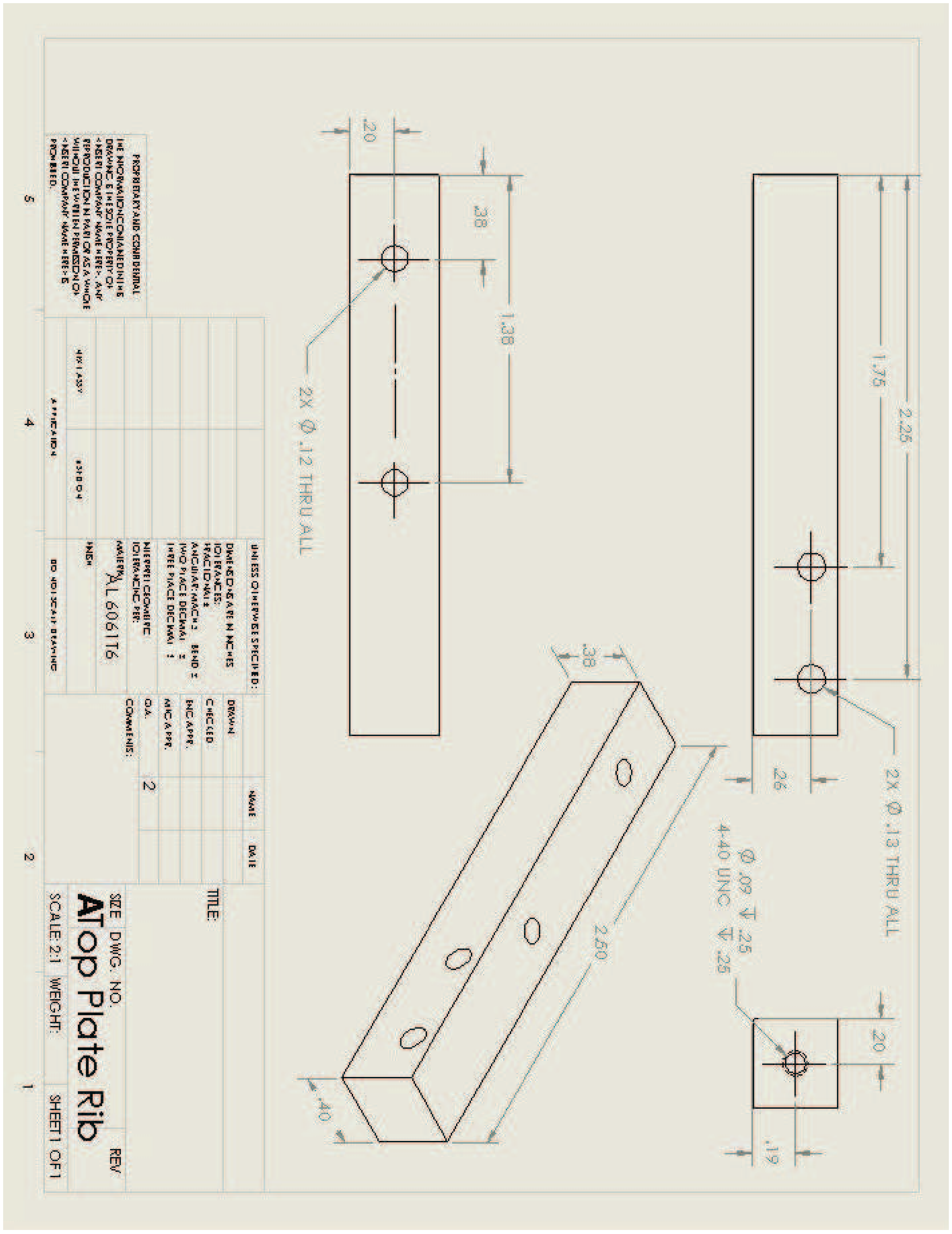


Figure B.6: The top plate rib runs down the center of the top plate providing additional rigidity as well as a place to mount the side plates.

APPENDIX C

Optimization Code

C.1 Umbrella Program Code

```
%% Program to optimize the four bar mechanism spring rates
%
%  Phil Voglewede
%  6/22/09

%% Clear the register and screen for debugging purposes

clear all
clc

%% Global Variables
global Counter

%% Initialize Counter
    Counter = 1;

%% Define the constants
    % Link Lengths (taken from measurements on v2 of end effector).
    % Units are in millimeters
    b = 37.16;
    initial_length_of_d = 111.10;

% Angles.
    t1 = 0;

    Block_Size_And_Theta1 = [b; t1;0;0;0;0;0;0;0;0;0;0];

    % centroid values - these axes are not the same as the rest of the
    % problem.
        %x - axis runs along the length of the arm from the
        %finger towards the bolt
        %y-axis runs width of the arm from the threaded side to the
        %conical side
        %z-axis runs up the finger starting at the base of the arm

        Xcbar=12.686;
        Zcbar=15.374;

        p=12.7+25.4-Xcbar;
        q=Zcbar-(12.7/2);
        r=Xcbar-(12.7-3.5);
        s=(12.7/2)+20.98;
        t=s-q;

        Centroid_Calc_Vector = [p; q; r; s; t;0;0;0;0;0;0;0];



% Link Mass
```

```
    % Units are in kilograms
         m1 = 10;  %arbitrary number
         m2 = .045473777;
         m3 = .02335;
         m4 = .045473777;

         Mass_Vector = [m1;m2;m3;m4;0;0;0;0;0;0;0;0];
    % Inertia
         %Parallel axis theorem for arm/finger assembly
         Iyc=12.1537;        %Inertia from NX6
         Izz=Iyc+m2*((((12.7/2)-Zcbar)^2)+(12.7+25.4-Xcbar)^2);

         I2 = Izz;       %arm/finger assembly
         I3 = 5.3739;    %wood block
         I4 = Izz;       %arm/finger assembly

         Inertia_Vector = [0; I2; I3; I4; 0;0;0;0;0;0;0;0];

     % Time steps
         Start_Time = 0;
         End_Time = .01;     %.054 sec was calculated from high speed camera
         Step_Size = End_Time/1000;



         Time_Vector = [Start_Time; Step_Size; End_Time; 0;0;0;0;0;0;0;0;0];
         time = (Start_Time:Step_Size:End_Time);

%% Define Initail positions, velocities and accelerations

% theta2
```

```
    Initial_Theta2_Dot = 0;

    Initial_Theta2_Double_Dot = 0;

    Theta_Vector = [0; Initial_Theta2_Dot; ...
        Initial_Theta2_Double_Dot;0;0;0;0;0;0;0;0;0];
    %The extra 0's makes the vector the same length as x.
    %This is needed to create Passing_Vector in costfunction

% d
    Initial_d = initial_length_of_d;        %set value above

    Initial_d_Dot = -800;         % Units are in millimeters per second
               %value was determined from average of high speed video data

    Initial_d_Double_Dot = 0;

    d_Vector = [Initial_d; Initial_d_Dot; Initial_d_Double_Dot; ...
        0;0;0;0;0;0;0;0;0];
    %The extra 0's makes the vector the same length as x0.  This is needed
    %to create Passing_Vector in costfunction


Initial_Conditions = [Theta_Vector d_Vector Time_Vector Mass_Vector ...
    Inertia_Vector Centroid_Calc_Vector Block_Size_And_Theta1];

%% Define initial guesses for fmincon

% Spring lever arm
```

```
aa = -42;
cc = -42;


% Spring rate
k_spring1 = 30;
k_spring2 = 45;


% Link Lengths
a = 30;
c = 45;


% Free lengths for the spring on the left (spring 1)
l_s10 = -2;
% Fixed end of spring 1
% (x position, spring1, fixed)
x_s1f = 200;
% (y position, spring1, fixed)
y_s1f = 60;
% Free length of spring 2
l_s20 = 0;
% Fixed end of spring 2
% (x position, spring2, fixed)
x_s2f = -90;               %Initial_d - x_s1f;
% (y position, spring2, fixed)
y_s2f = 60;
```

```
% Put into a vector
x0 = [aa; cc; k_spring1; k_spring2; a; c; l_s10; x_s1f; ...
        y_s1f; l_s20; x_s2f; y_s2f];

%% fmincon Options

options = optimset('display','iter','MaxFunEvals',1e1000, ...
    'MaxIter',1e100,'TolX',1e-012);
%% Define the optimization routine

xxx = fmincon(@(x0) costfunction(x0,Initial_Conditions),x0,[],[],[],[], ...
    [-42.5;-42.5;  0;0;   41.97;41.97;   -5;-50;-30; -5;-50;-30], ...
    [50;50;        50;50; 100;100;     10;250;70;    10;161.1;70], ...
    @(x0)nonlinear_constraints(x0,Initial_Conditions),options);

%% Plot the results

aa = xxx(1)
cc = xxx(2)
k_spring1 = xxx(3)
k_spring2 = xxx(4)
a = xxx(5)
c = xxx(6)
l_s10 = xxx(7)
x_s1f = xxx(8)
y_s1f = xxx(9)
l_s20 = xxx(10)
x_s2f_From_End_of_d = xxx(11)
y_s2f = xxx(12)
```

```
%% Check the results

% Rerun the solver
%Set the initial values for the optimization
    % Theta 2
    y0(1) = acos((c^2-a^2-(initial_length_of_d-b)^2)/ ...
        (-2*a*(initial_length_of_d-b)));
    % Theta 2 dot
    y0(2) = Theta_Vector(2);
    % Theta 2 dot
    yprime0(1) = Theta_Vector(2);
    % Theta 2 ddot
    yprime0(2) = Theta_Vector(3);

    % Creating the vector to pass

    Passing_Vector = [xxx, Initial_Conditions];

% Set up routine
[time_out, y] = ode15i(@(input_time,input_y,input_yprime) ...
    implicit_derivative_optimization(input_time,input_y,input_yprime, ...
    Passing_Vector),time,y0',yprime0',Passing_Vector);
% NOTE:  Time_out should equal time.  This is done here to avoid confusion
% and to allow for future changes if desired

% Theta 2
t2 = y(:,1);

% Loop for the video - Uses same code as the positional kinematics.
for i=1:length(time_out),
    % Units are millimeters
    d(i) = Initial_d_Dot*time_out(i)+initial_length_of_d;

%% Calculatingin actual position of x_s2f

    x_s2f = d + x_s2f_From_End_of_d;



    %% Position analysis - Derived from Norton
    % Theta 4 (output link)
    % Parameters from Norton - page 176.
    k1=d(i)/a;
    k2=d(i)/c;
    k3=((a^2)-(b^2)+(c^2)+(d(i)^2))/(2*a*c);

    A=cos(t2(i))-k1-k2*cos(t2(i))+k3;
    B=(-2)*sin(t2(i));
    C=k1-(k2+1)*cos(t2(i))+k3;

    % Solution for theta 4 from Norton - page 176
        % Solution 1
        T4(i) = 2*atan((-B+sqrt((B^2)-4*A*C))/(2*A));
        % Solution 2
        t4(i) = 2*atan((-B-sqrt((B^2)-4*A*C))/(2*A));

    % Theta 3 (coupler link)
    % Parameters from Norton - page 176.
    k4 = d(i)/b;
    k5 = ((c^2)-(d(i)^2)-(a^2)-(b^2))/(2*a*b);
```

```
D = cos(t2(i))-k1+k4*cos(t2(i))+k5;
E = (-2)*sin(t2(i));
F = k1+(k4-1)*cos(t2(i))+k5;

% Solution for theta 4 from Norton - page 176
    % Solution 1
    T3(i) = 2*atan((-E+sqrt((E^2)-4*D*F))/(2*D));
    % Solution 2
    t3(i) = 2*atan((-E-sqrt((E^2)-4*D*F))/(2*D));

figure(2)
clf
x_axis = [-aa*cos(t2(i))
     0
     a*cos(t2(i))
     a*cos(t2(i))+b*cos(t3(i))
     a*cos(t2(i))+b*cos(t3(i))-c*cos(t4(i))
     d(i) - cc*cos(t4(i))
     a*cos(t2(i))+b*cos(t3(i))-c*cos(t4(i))
     a*cos(t2(i))+b*cos(t3(i))-c*cos(t4(i))-d(i)];
y_axis = [-aa*sin(t2(i))
     0
     a*sin(t2(i))
     a*sin(t2(i))+b*sin(t3(i))
     a*sin(t2(i))+b*sin(t3(i))-c*sin(t4(i))
     -cc*sin(t4(i))
     a*sin(t2(i))+b*sin(t3(i))-c*sin(t4(i))
     a*sin(t2(i))+b*sin(t3(i))-c*sin(t4(i))];
xs1 = [x_s1f -aa*cos(t2(i))];
ys1 = [y_s1f -aa*sin(t2(i))];
```

```
    xs2 = [x_s2f d(i)-cc*cos(t4(i))];
    ys2 = [y_s2f -cc*sin(t4(i))];
    plot(x_axis,y_axis,xs1,ys1,xs2,ys2)
    axis square
    axis([-20 140 -40 120])

    MyMovie(i) = getframe;

end



movie(MyMovie,3)
```

C.2 Cost Function Code

```
function error = costfunction(x,Initial_Conditions)
%% checking step order for debug
%CostFunction = 1

%% Extracting passed Values

% link lengths
    a = x(5);
    c = x(6);

% Values for d
    d_Vector = Initial_Conditions(:,2);
    initial_length_of_d = d_Vector(1);
    ddot = d_Vector(2);

Block_Size_And_Theta1 = Initial_Conditions(:,7);
    b = Block_Size_And_Theta1(1);
    %t1 = Block_Size_And_Theta1(2);

Theta_Vector = Initial_Conditions(:,1);

    % Theta 2
    y0(1) = acos((c^2-a^2-(initial_length_of_d-b)^2)/ ...
        (-2*a*(initial_length_of_d-b)));
    % Theta 2 dot
    y0(2) = Theta_Vector(2);
    % Theta 2 dot
    yprime0(1) = Theta_Vector(2);
    % Theta 2 ddot
    yprime0(2) = Theta_Vector(3);

Time_Vector = Initial_Conditions(:,3);
    Start_Time = Time_Vector(1);
    Step_Size = Time_Vector(2);
    End_Time = Time_Vector(3);

    time = (Start_Time:Step_Size:End_Time);

%% Creating the vector to pass

    Passing_Vector = [x, Initial_Conditions];
```

```
%% checking step order for debug
%ImplicitDerivative = 1
%% Set up routine
ode15iOptions = odeset('RelTol',1e-003,'AbsTol',1e-006);

[time_out, y] = ode15i(@(input_time,input_y,input_yprime) ...
    ForwardDynamicsFunction(input_time,input_y,input_yprime, ...
    Passing_Vector),time,y0',yprime0',ode15iOptions);



%% Find the theta 3 versus time

t2 = y(:,1);

for i=1:length(time_out),

    % Units are millimeters
    d(i) = ddot*time(i)+initial_length_of_d;

    %% Position analysis - Derived from Norton

    % Parameters from Norton - page 176.
    k1=d(i)/a;

    % Theta 3 (coupler link)
    % Parameters from Norton - page 176.
    k4 = d(i)/b;
    k5 = ((c^2)-(d(i)^2)-(a^2)-(b^2))/(2*a*b);
    D = cos(t2(i))-k1+k4*cos(t2(i))+k5;
    E = (-2)*sin(t2(i));
    F = k1+(k4-1)*cos(t2(i))+k5;

    % Solution for theta 4 from Norton - page 176
        % Solution 2
        t3(i) = 2*atan((-E-sqrt((E^2)-4*D*F))/(2*D));
end

%% Find the cost function

error = sum(t3.^2);
```

C.3 Forward Dynamics Function Code

```
function f = ForwardDynamicsFunction(input_time,input_y, ...
    input_yprime,Passing_Vector)
%% Extracting passed Values
    x = Passing_Vector(:,1);
        %Lever arm lengths
        aa = x(1);
        cc = x(2);

        % Spring rates
        k_spring1 = x(3);
        k_spring2 = x(4);

        % Link Lengths
        a = x(5);
        c = x(6);

        % Free lengths for the spring on the left (spring 1)
        l_s10 = x(7);
        % Fixed end of spring 1
         % (x position, spring1, fixed)
         x_s1f = x(8);
         % (y position, spring1, fixed)
         y_s1f = x(9);

        % Free length of spring 2
        l_s20 = x(10);
        % Fixed end of spring 2
        % (x position, spring2, fixed distance measured from the end of d)
        x_s2f_From_End_of_d = x(11);
        % (y position, spring2, fixed)
        y_s2f = x(12);

    Initial_Conditions = [Passing_Vector(:,2) Passing_Vector(:,3) ...
        Passing_Vector(:,4) Passing_Vector(:,5) Passing_Vector(:,6) ...
        Passing_Vector(:,7) Passing_Vector(:,8)];

        % Initial conditions for d
        d_Vector = Initial_Conditions(:,2);
        initial_length_of_d = d_Vector(1);
        ddot = d_Vector(2);
        ddoubledot = d_Vector(3);

    Mass_Vector = Initial_Conditions(:,4);
        m2 = Mass_Vector(2);
```

```
        m3 = Mass_Vector(3);
        m4 = Mass_Vector(4);

    Inertia_Vector = Initial_Conditions(:,5);
        I2 = Inertia_Vector(2);
        I3 = Inertia_Vector(3);
        I4 = Inertia_Vector(4);

    Centroid_Calc_Vector = Initial_Conditions(:,6);
        p = Centroid_Calc_Vector(1);
        q = Centroid_Calc_Vector(2);
        r = Centroid_Calc_Vector(3);
        s = Centroid_Calc_Vector(4);
        t = Centroid_Calc_Vector(5);

    Block_Size_And_Theta1 = Initial_Conditions(:,7);
        b = Block_Size_And_Theta1(1);
        t1 = Block_Size_And_Theta1(2);

% if a+b+c-9.9<initial_length_of_d  %for debug
%     BAD = 1
%     t2 = input_y(1)
% end

%% Specified accelerations or velocities or positions (depending on run).
% Time
    time = input_time;

% Acceleration
    % Units are radians per second squared
    alpha2 = input_yprime(2);

% Velocity
    % Units are radians per second
    omega2 = input_y(2);

% Positions
    % Units are radians
    t2 = input_y(1);
    % Units are millimeters
    d = ddot*time+initial_length_of_d;

%% Calculatingin actual position of x_s2f

    x_s2f = d + x_s2f_From_End_of_d;

%% Position analysis - Derived from Norton
    % Theta 4 (output link)
    % Parameters from Norton - page 176.
    k1=d/a;
    k2=d/c;
    k3=((a^2)-(b^2)+(c^2)+(d^2))/(2*a*c);

    A=cos(t2)-k1-k2*cos(t2)+k3;
    B=(-2)*sin(t2);
    C=k1-(k2+1)*cos(t2)+k3;

    % Solution for theta 4 from Norton - page 176
    % Solution 1
    T4 = 2*atan((-B+sqrt((B^2)-4*A*C))/(2*A));
    % Solution 2
    t4 = 2*atan((-B-sqrt((B^2)-4*A*C))/(2*A));

    % Theta 3 (coupler link)
    % Parameters from Norton - page 176.
    k4 = d/b;
    k5 = ((c^2)-(d^2)-(a^2)-(b^2))/(2*a*b);

    D = cos(t2)-k1+k4*cos(t2)+k5;
    E = (-2)*sin(t2);
    F = k1+(k4-1)*cos(t2)+k5 ;

    % Solution for theta 4 from Norton - page 176
    % Solution 1
    T3 = 2*atan((-E+sqrt((E^2)-4*D*F))/(2*D));
    % Solution 2
    t3 = 2*atan((-E-sqrt((E^2)-4*D*F))/(2*D));


%% Velocity Analysis - Derived from Norton
    % Solution for omega 3 from Norton - page 297
    omega3 = (a*omega2*sin(t4-t2)-ddot*cos(t1-t4))/(b*sin(t3-t4));
    % Solution for omega 3 from Norton - page 297
    omega4 = (a*omega2*sin(t2-t3)+ddot*cos(t1-t3))/(c*sin(t4-t3));


%% Acceleration Analysis - Derived from Norton
    % Solution for alpha 3
    % The same as Norton - page 339 - but uses a substitution
    % See hand calculations for more information.
    alpha3 = (a*alpha2*sin(t2-t4)+a*omega2^2*cos(t2-t4)+b* ...
        omega3*cos(t3-t4)-c*omega4^2 + ...
        ddoubledot*cos(t1-t4))/(b*sin(t4-t3));
    % Solution for alpha 4
    % The same as Norton - page 339 - but uses a substitution
    % See hand calculations for more information.
    alpha4 = (a*alpha2*sin(t2-t3)+a*(omega2^2)*cos(t2-t3)+b* ...
        (omega3^2)-c*(omega4^2)*cos(t4-t3)+ ...
        ddoubledot*cos(t1-t3))/(c*sin(t4-t3));

%% Calculation of Torque

%Torque - Left Spring

    % Moving end of spring 1
     % (x position, spring1, moving)
     x_s1m = -aa*cos(t2);
     % (y position, spring1, moving)
     y_s1m = -aa*sin(t2);
    % Calculate the force
     % First, find the magnitude
     % Units are milli-Newtons
     F12 = k_spring1*(sqrt((x_s1m - x_s1f)^2 + (y_s1m - y_s1f)^2)-l_s10);
     % Now find the direction
     zeta1 = atan(abs((y_s1f-y_s1m)/(x_s1f-x_s1m)));
    % Calculate the moment
     % Units are milli-Newtons times millimeters
     T12 = - aa*F12*sin(pi-zeta1-t2);

%Torque - Right Spring
```

```
    % Moving end of spring 2
     % (x position, spring2, moving)
     x_s2m = d - cc*cos(t4);
     % (y position, spring2, moving)
     y_s2m = -cc*sin(t4);
    % Calculate the force
     % First, find the magnitude
     % Units are milli-Newtons
     F4 = k_spring2*(sqrt((x_s2m - x_s2f)^2 + (y_s2m - y_s2f)^2)-l_s20);
     % Now find the direction
     zeta2 = atan(abs((y_s1f-y_s1m)/(x_s1f-x_s1m)));
    % Calculate the moment
     % Units are milli-Newtons times millimeters
     % NOTE:  The change in notation (T14 instead of T4) is inconsistent
     %with Norton
     T14 = cc*F4*sin(t4-zeta2);



%% Force Analysis - Derived from Norton


    % Creating alpha vectors
       Alpha2_vector = [0,0,alpha2];
       Alpha3_vector = [0,0,alpha3];
       Alpha4_vector = [0,0,alpha4];

    % Creating omega vectors
       Omega2_vector = [0,0,omega2];
       Omega3_vector = [0,0,omega3];
       Omega4_vector = [0,0,omega4];

    % Solving for R vectors
       beta=atan(s/(p+r));
       psi=atan(q/p);
       phi=atan(r/t);
       gamma=(pi()/2)-beta-phi;
       lambda=pi()-t4;

    % Radius to link with respect to CG
       R12 = [-sqrt(p^2+q^2)*cos(t2+(beta-psi)), ...
           -sqrt(p^2+q^2)*sin(t2+(beta-psi)),0];
       R23 = [-(b/2)*cos(t3),-(b/2)*sin(t3),0];
       R32 = [sqrt(r^2+t^2)*cos(gamma-t2), ...
           -sqrt(r^2+t^2)*sin(gamma-t2),0];
       R34 = [-sqrt(r^2+t^2)*cos(gamma-lambda), ...
           -sqrt(r^2+t^2)*sin(gamma-lambda),0];
       R43 = [(b/2)*cos(t3),(b/2)*sin(t3),0];
       R14 = [sqrt(p^2+q^2)*cos(lambda+(beta-psi)), ...
           -sqrt(p^2+q^2)*sin(lambda+(beta-psi)),0];

    % Radius to CG with respect to CG
       R3_2 = [R32(1,1)-R23(1,1),R32(1,2)-R23(1,2),0];
       R4_3 = [R43(1,1)-R34(1,1),R43(1,2)-R34(1,2),0];

    % Acceleration
       a1 = [0,0,0];
       a2 = a1+cross(Alpha2_vector,-R12)+ ...
           cross(Omega2_vector,cross(Omega2_vector,-R12));
       a3 = a2+cross(Alpha3_vector,R3_2)+ ...
           cross(Omega3_vector,cross(Omega3_vector,R3_2));
       a4 = a3+cross(Alpha4_vector,R4_3)+ ...
           cross(Omega4_vector,cross(Omega4_vector,R4_3));


   % Acceleration Decomposed
       a2x = a2(1,1);
       a2y = a2(1,2);
       a3x = a3(1,1);
       a3y = a3(1,2);
       a4x = a4(1,1);
       a4y = a4(1,2);

   % Distance to the joint from the center of gravity
   % R_jointNumber_linkNumber_Direction as in Norton P.570
       R12y = R12(1,2);
       R12x = R12(1,1);
       R32y = R32(1,2);
       R32x = R32(1,1);
       R23y = R23(1,2);
       R23x = R23(1,1);
       R43y = R43(1,2);
       R43x = R43(1,1);
       R34y = R34(1,2);
       R34x = R34(1,1);
       R14y = R14(1,2);
       R14x = R14(1,1);

   big_matrix = [1 0 1 0 0 0 0 0
```

```
              0 1 0 1 0 0 0 0
              0 0 -1 0 1 0 0 0
              0 0 0 -1 0 1 0 0
              0 0 R23y -R23x -R43y R43x 0 0
              0 0 0 0 -1 0 1 0
              0 0 0 0 0 -1 0 1
              0 0 0 0 R34y -R34x -R14y R14x ];

small_matrix = [ m2*a2x
                 m2*a2y
                 m3*a3x
                 m3*a3y
                 I3*alpha3
                 m4*a4x
                 m4*a4y
                 I4*alpha4-T14];

% Find forces using 8 of the 9 equations from Norton
Forces = inv(big_matrix)*small_matrix;

% Establish output as a vector
f = zeros(2,1);

% Implicit function calculation
f(1) = input_yprime(1) - input_y(2);
f(2) = I2*input_yprime(2) - (T12 + R12x*Forces(2) - R12y*Forces(1) ...
    + R32x*Forces(4) - R32y*Forces(3));
```

C.4 Non-Linear Constraints Code

```
function [nonequality_constraint,equality_constraint] = ...
    nonlinear_constraints(x,Initial_Conditions)
global Counter
%% checking step order
%nonlinearConstraints = 1

%% Extracting passed Values

% link lengths
    a = x(5);
    c = x(6);

% Values for d
    d_Vector = Initial_Conditions(:,2);

    initial_length_of_d = d_Vector(1);
%    ddot = d_Vector(2);

Block_Size_And_Theta1 = Initial_Conditions(:,7);
    b = Block_Size_And_Theta1(1);
    %t1 = Block_Size_And_Theta1(2);

Theta_Vector = Initial_Conditions(:,1);

    % Theta 2
    y0(1) = acos((c^2-a^2-(initial_length_of_d-b)^2)/ ...
        (-2*a*(initial_length_of_d-b)));
    % Theta 2 dot
    y0(2) = Theta_Vector(2);
    % Theta 2 dot
    yprime0(1) = Theta_Vector(2);
    % Theta 2 ddot
    yprime0(2) = Theta_Vector(3);

Time_Vector = Initial_Conditions(:,3);
    Start_Time = Time_Vector(1);
    Step_Size = Time_Vector(2);
    End_Time = Time_Vector(3);

    time = (Start_Time:Step_Size:End_Time);

%% Creating the vector to pass

    Passing_Vector = [x, Initial_Conditions];
%% checking step order
%ImplicitDerivative = 1
%% Set up routine
%creating ode15i options
ode15iOptions = odeset('RelTol',1e-003,'AbsTol',1e-006);

[time_out, y] = ode15i(@(input_time,input_y,input_yprime) ...
    ForwardDynamicsFunction(input_time,input_y,input_yprime, ...
    Passing_Vector),time,y0',yprime0',ode15iOptions);


%% Calculating Alpha2 - Numerical derivative of Omega2 to calculate Alpha2

t2 = y(:,1);
```

```
Omega2 = y(:,2);
N = length(Omega2);

% Forward difference
    Alpha2(1) = (Omega2(2) - Omega2(1)) / (time_out(2) - time_out(1));

% Central Difference
    for i=2:(N - 1)
        Alpha2(i) = (Omega2(i+1)-Omega2(i-1))/(time_out(i+1) ...
            - time_out(i-1));
    end

% Backwards Difference
    Alpha2(N) = (Omega2(N) - Omega2(N-1)) / (time_out(N) - time_out(N-1));

%% checking step order  for debug
%twoDOF_Function = 1



%% Loop for calculating forces
for i=1:length(time_out),

  [dummy,tThree] = inverse_Dynamics(time_out(i),t2(i),Omega2(i), ...
      Alpha2(i),Initial_Conditions,x);

   Forces(:,i)=dummy;
   Theta3(i) = tThree;

end
```

```
Theta3 = Theta3';

force3 = Forces(1,:);       %F12_x
force3 = force3';

force4 = Forces(2,:);       %F12_y
force4 = force4';

force5 = Forces(3,:);       %F32_x
force5 = force5';

force6 = Forces(4,:);       %F32_y
force6 = force6';

%If a singularity occurs time_out is less than time
 if length(time_out)-length(time) < 0

     Time_Difference = length(time)-length(time_out);

     Counter = Counter+1;       %Forces a gradiant in the add values

     %The total length of the nonequality constraint must remain the same
     %as when time_out = time
     add = ones(Time_Difference,1)*500000;

    nonequality_constraint = [(force3.*cos(Theta3) ...
        +force4.*sin(Theta3));add*Counter; ...
        ((force5.*cos(Theta3) + force6.*sin(Theta3))); ...
        add*Counter; (-(a+b+c)+initial_length_of_d+10)];
```

```
   equality_constraint = length(time)-length(time_out);

else

   nonequality_constraint = [(force3.*cos(Theta3) ...
       + force4.*sin(Theta3)); ((force5.*cos(Theta3) ...
       +  force6.*sin(Theta3))); (-(a+b+c)+initial_length_of_d+10)];

   equality_constraint = length(time)-length(time_out);


end
```

C.5 Inverse Dynamics Code

```
function [contact_force,t3] = two_DOF_fourbar_forces_function ...
    (time,t2,omega2,alpha2,Initial_Conditions,x)
%% Program to compute the forces in a 2DOF four bar mechanism
%  Two DOF 4-bar analysis
% Jacob Ziesmer
% 6/19/09
%
% Modified by Phil Voglewede
% 6/22/09
%
% Modified by Jacob Ziesmer
% 7/1/09
%
% Notational conventions:
% ----------------------
% Use of Norton for link lengths
%
% t# - Angle of theta# in radians (one kinematic solution)
% T# - Angle of theta# in radians (other kinematic solution)
% theta# - Angle of theta# in degrees (one kinematic solution)
% Theta# - Angle of theta# in degrees (other kinematic solution)
%
% All angular velocities and accelerations are in radians (per second or
% second squared).  For example, omega2 is the angular velocity of link 2
% in radians per second.  alpha2 is the angular acceleration of link 2 in
% radians per second squared.

%% Extracting passed Values
        %Lever arm lengths
        aa = x(1);
        cc = x(2);

        % Spring rates
        k_spring1 = x(3);
        k_spring2 = x(4);

        % Link Lengths
        a = x(5);
        c = x(6);

        % Free lengths for the spring on the left (spring 1)
        l_s10 = x(7);
        % Fixed end of spring 1
         % (x position, spring1, fixed)
         x_s1f = x(8);
         % (y position, spring1, fixed)
         y_s1f = x(9);

        % Free length of spring 2
        l_s20 = x(10);
        % Fixed end of spring 2
        % (x position, spring2, fixed distance measured from the end of d)
        x_s2f_From_End_of_d = x(11);
        % (y position, spring2, fixed)
        y_s2f = x(12);

%Bringing in initial conditions and other constants

    d_Vector = Initial_Conditions(:,2);



    Mass_Vector = Initial_Conditions(:,4);
        m2 = Mass_Vector(2);
        m3 = Mass_Vector(3);
        m4 = Mass_Vector(4);

    Inertia_Vector = Initial_Conditions(:,5);
        I2 = Inertia_Vector(2);
        I3 = Inertia_Vector(3);
        I4 = Inertia_Vector(4);

    Centroid_Calc_Vector = Initial_Conditions(:,6);
        p = Centroid_Calc_Vector(1);
        q = Centroid_Calc_Vector(2);
        r = Centroid_Calc_Vector(3);
        s = Centroid_Calc_Vector(4);
        t = Centroid_Calc_Vector(5);

    Block_Size_And_Theta1 = Initial_Conditions(:,7);
        b = Block_Size_And_Theta1(1);
        t1 = Block_Size_And_Theta1(2);


%% Specified accelerations and velocities of d.
    % Acceleration
        % Units are millimeters per second squared
        ddoubledot = d_Vector(3);

    % Velocity
        % Units are millimeters per second
        ddot = d_Vector(2); %ddoubledot*time;

    % Positions
        % Units are millimeters
        d = .5*ddoubledot*time^2+ddot*time+d_Vector(1);

%% Calculatingin actual position of x_s2f

    x_s2f = d + x_s2f_From_End_of_d;



%% Position analysis - Derived from Norton
    % Theta 4 (output link)
    % Parameters from Norton - page 176.
```

```
k1=d/a;
k2=d/c;
k3=((a^2)-(b^2)+(c^2)+(d^2))/(2*a*c);

A=cos(t2)-k1-k2*cos(t2)+k3;
B=(-2)*sin(t2);
C=k1-(k2+1)*cos(t2)+k3;

% Solution for theta 4 from Norton - page 176
    % Solution 1
    T4 = 2*atan((-B+sqrt((B^2)-4*A*C))/(2*A));
    % Solution 2
    t4 = 2*atan((-B-sqrt((B^2)-4*A*C))/(2*A));

% Theta 3 (coupler link)
% Parameters from Norton - page 176.
k4 = d/b;
k5 = ((c^2)-(d^2)-(a^2)-(b^2))/(2*a*b);

D = cos(t2)-k1+k4*cos(t2)+k5;
E = (-2)*sin(t2);
F = k1+(k4-1)*cos(t2)+k5;

% Solution for theta 4 from Norton - page 176
    % Solution 1
    T3 = 2*atan((-E+sqrt((E^2)-4*D*F))/(2*D));
    % Solution 2
    t3 = 2*atan((-E-sqrt((E^2)-4*D*F))/(2*D));
```

```
%% Velocity Analysis - Derived from Norton
    % Solution for omega 3 from Norton - page 297
        omega3 = (a*omega2*sin(t4-t2)-ddot*cos(t1-t4))/(b*sin(t3-t4));
    % Solution for omega 3 from Norton - page 297
        omega4 = (a*omega2*sin(t2-t3)+ddot*cos(t1-t3))/(c*sin(t4-t3));


%% Acceleration Analysis - Derived from Norton
    % Solution for alpha 3
    % The same as Norton - page 339 - but uses a substitution
    % See hand calculations for more information.
    alpha3 = (a*alpha2*sin(t2-t4)+a*omega2^2*cos(t2-t4)+b* ...
        omega3*cos(t3-t4)-c*omega4^2+ ...
        ddoubledot*cos(t1-t4))/(b*sin(t4-t3));
    % Solution for alpha 4
    % The same as Norton - page 339 - but uses a substitution
    % See hand calculations for more information.
    alpha4 = (a*alpha2*sin(t2-t3)+a*(omega2^2)*cos(t2-t3)+b* ...
        (omega3^2)-c*(omega4^2)*cos(t4-t3)+ ...
        ddoubledot*cos(t1-t3))/(c*sin(t4-t3));


%% Calculation of Torque

%Torque - Left Spring

    % Moving end of spring 1
     % (x position, spring1, moving)
     x_s1m = -aa*cos(t2);
     % (y position, spring1, moving)
     y_s1m = -aa*sin(t2);
    % Calculate the force
     % First, find the magnitude
     % Units are milli-Newtons
     F12 = k_spring1*(sqrt((x_s1m - x_s1f)^2 + (y_s1m - y_s1f)^2)-l_s10);
     % Now find the direction
     zeta1 = atan(abs((y_s1f-y_s1m)/(x_s1f-x_s1m)));
    % Calculate the moment
     % Units are milli-Newtons times millimeters
     T12 = - aa*F12*sin(pi-zeta1-t2);

%Torque - Right Spring

    % Moving end of spring 2
     % (x position, spring2, moving)
     x_s2m = d - cc*cos(t4);
     % (y position, spring2, moving)
     y_s2m = -cc*sin(t4);
     % Calculate the force
     % First, find the magnitude
     % Units are milli-Newtons
     F4 = k_spring2*(sqrt((x_s2m - x_s2f)^2 + (y_s2m - y_s2f)^2)-l_s20);
     % Now find the direction
     zeta2 = atan(abs((y_s1f-y_s1m)/(x_s1f-x_s1m)));
    % Calculate the moment
     % Units are milli-Newtons times millimeters
     % NOTE:  The change in notation (T14 instead of T4) is inconsistent
     % with Norton
     T14 = cc*F4*sin(t4-zeta2);
```

```
%% Force Analysis - Derived from Norton


% Creating alpha vectors
    Alpha1_vector = [0,0,0];
    Alpha2_vector = [0,0,alpha2];
    Alpha3_vector = [0,0,alpha3];
    Alpha4_vector = [0,0,alpha4];

% Creating omega vectors
    Omega1_vector = [0,0,0];
    Omega2_vector = [0,0,omega2];
    Omega3_vector = [0,0,omega3];
    Omega4_vector = [0,0,omega4];

% Solving for R vectors
    beta=asin(s/a);
    psi=atan(q/p);
    phi=atan(r/t);
    gamma=(pi()/2)-beta-phi;
    lambda=pi()-t4;

% Radius to link with respect to CG
R12 = [-sqrt(p^2+q^2)*cos(t2+(beta-psi)), ...
    -sqrt(p^2+q^2)*sin(t2+(beta-psi)),0];
R32 = [sqrt(r^2+t^2)*cos(gamma-t2),-sqrt(r^2+t^2)*sin(gamma-t2),0];
R23 = [-(b/2)*cos(t3),-(b/2)*sin(t3),0];
R43 = [(b/2)*cos(t3),(b/2)*sin(t3),0];
R34 = [-sqrt(r^2+t^2)*cos(gamma-lambda), ...
```

```
    -sqrt(r^2+t^2)*sin(gamma-lambda),0];
R14 = [sqrt(p^2+q^2)*cos(lambda+(beta-psi)), ...
    -sqrt(p^2+q^2)*sin(lambda+(beta-psi)),0];

    % Radius to CG with respect to CG
        R3_2 = [R32(1,1)-R23(1,1),R32(1,2)-R23(1,2),0];
        R4_3 = [R43(1,1)-R34(1,1),R43(1,2)-R34(1,2),0];

    % Acceleration
        a1 = [0,0,0];
        a2 = a1+cross(Alpha2_vector,-R12)+ ...
            cross(Omega2_vector,cross(Omega2_vector,-R12));
        a3 = a2+cross(Alpha3_vector,R3_2)+ ...
            cross(Omega3_vector,cross(Omega3_vector,R3_2));
        a4 = a3+cross(Alpha4_vector,R4_3)+ ...
            cross(Omega4_vector,cross(Omega4_vector,R4_3));



    % Acceleration Decomposed
        a2x = a2(1,1);
        a2y = a2(1,2);
        a3x = a3(1,1);
        a3y = a3(1,2);
        a4x = a4(1,1);
        a4y = a4(1,2);

    % Distance to the joint from the center of gravity
    % R_jointNumber_linkNumber_Direction as in Norton P.570
        R12y = R12(1,2);
        R12x = R12(1,1);
        R32y = R32(1,2);
        R32x = R32(1,1);
        R23y = R23(1,2);
        R23x = R23(1,1);
        R43y = R43(1,2);
        R43x = R43(1,1);
        R34y = R34(1,2);
        R34x = R34(1,1);
        R14y = R14(1,2);
        R14x = R14(1,1);

    big_matrix = [1 0 1 0 0 0 0 0 0
                  0 1 0 1 0 0 0 0 0
                  -R12y R12x -R32y R32x 0 0 0 0 1
                  0 0 -1 0 1 0 0 0 0
                  0 0 0 -1 0 1 0 0 0
                  0 0 R23y -R23x -R43y R43x 0 0 0
                  0 0 0 0 -1 0 1 0 0
                  0 0 0 0 0 -1 0 1 0
                  0 0 0 0 R34y -R34x -R14y R14x 0];

    small_matrix = [ m2*a2x
                     m2*a2y
                     I2*alpha2
                     m3*a3x
                     m3*a3y
                     I3*alpha3
                     m4*a4x
                     m4*a4y
                     I4*alpha4-T14];
```

```
Forces = inv(big_matrix)*small_matrix;


contact_force = [Forces(3)
                 Forces(4)
                 Forces(5)
                 Forces(6)];
```