# Design, Assessment, and Comparison of Antagonistic, Cable-Driven, Variable Stiffness Actuators

Ryan P. Moore
*Marquette University*

# DESIGN, ASSESSMENT, AND COMPARISON OF ANTAGONISTIC, CABLE-DRIVEN, VARIABLE STIFFNESS ACTUATORS

by

Ryan P. Moore, 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, Wisconsin

May 2020

ABSTRACT
DESIGN, ASSESSMENT, AND COMPARISON OF ANTAGONISTIC, CABLE-DRIVEN, VARIABLE STIFFNESS ACTUATORS

Ryan P. Moore, B.S.

Marquette University, 2020

This thesis presents the designs and test results for two antagonistic, cable-driven, variable stiffness actuator designs. Each of these variable stiffness actuators is compact, has a large range of controllable stiffness, and limits the inertia at the robotic link it is controlling. Each design consists of a cable running through a set of three pulleys. Tension on the cable displaces a linear spring, which moves along a path designed to achieve quadratic spring behavior. One design uses a variable radius path to achieve the nonlinear elastic behavior while the other uses a fixed radius (lever) path.

A quasi-static model of each mechanism was developed to assess the performance of each design in matching the desired nonlinear (quadratic) elastic behavior of the ideal system. Eight geometric parameters of each design were optimized to match the desired behavior. Prototypes of the optimized designs were built and tested to evaluate performance.

While the results of the parametric optimization predicted that the variable radius design would more closely match the desired elastic behavior, the added complexity of this design resulted in inadequate performance. Test results for the fixed radius design matched the desired behavior well and ultimately proved to be better for achieving controllable linear stiffness at a robotic joint.

# ACKNOWLEDGEMENTS

Ryan P. Moore, B.S.

I would like to thank my teachers and professors throughout my educational journey especially my advisor, Dr. Schimmels, for his guidance and support throughout my graduate school experience. I would like to thank Jacob Rice for his mentorship as a lab mate, colleague, and friend.

I would like to thank my partner, Sarah, for her unwavering support and sacrifices throughout my university education and I would like to thank my siblings, Andrew and Megan Moore, for their love, support, and advice which has helped me grow to where I am today.

Finally, I would like to give a special thank you to my parents, Teresa and Stanley Moore, not only for the love and support they have provided to me my entire life, but also for fostering a love of learning and education that has propelled me through these past nineteen straight years of schooling and will undoubtedly sustain me through what I hope to be a lifetime of learning and growth.

TABLE OF CONTENTS

**Contents**

## LIST OF TABLES

# LIST OF FIGURES

CHAPTER 1

# INTRODUCTION

Modern robotic manipulators perform well in highly structured environments where positioning uncertainty is low. Manipulation in less structured environments, however, is more challenging. In 2006 [1], "the development of good hardware to make [robotic] arms and hands that can perform anything but the simplest of pick-and-place operations that are prevalent in industry," was listed as a fundamental challenge in robotics research. Manipulation tasks more complex than pick-and-place operations are difficult for conventional robotic manipulators to complete due to the high level of relative positioning accuracy required. Small variation in the location of the manipulator, held object, or its environment can result in undesirable, high-force contact when the manipulator and environment are both stiff. While progress has been made in the pursuit of reliable compliance in robotic manipulators, as recently as 2016 the Roadmap for US Robotics [2] identified that "a major limitation in the adoption of robot manipulation systems is lack of access to flexible gripping mechanisms that allow not only pick up but also dexterous manipulation of everyday objects." Added compliance in the system compensates for small errors in the placement of manipulator held objects. Manipulator compliance can be obtained actively or passively. Each approach brings with it a variety of advantages and disadvantages, but neither has proven to be best in every scenario.

In the sections below, an overview of the active and passive approaches to achieve compliance in a robot system are presented.

### 1.1 Passive Compliance

Custom end of arm tooling is often used in industry to achieve a desired compliance for a given task. The tooling for each is designed specifically for a given task and must be remodeled, rebuilt, and replaced any time a new task is to be performed by a manipulator. For traditional industrial manipulation scenarios in which the manipulator is set to do a single action repeatedly for long periods of time, this approach provides a simple and easy to implement solution. Most sources of error in such a system are well understood and can be accounted for in the end of arm tooling design.

The “Remote Center Compliance System” [3] established one of the first frameworks for compliant end of arm tooling. The device achieves the passive compliance needed in insertion or assembly tasks without using any motors, sensors, or energy sources (other than the motion of the arm). The device allows for translational and rotational motion of the held part using passive elements in order to compensate for relative positional variability in each cycle of the single task it is designed to accomplish.

The major drawback of this kind of solution is the limited application of each custom end of arm tooling. In small scale manufacturing environments, more versatile solutions are preferred to reduce cost and increase flexibility of robotic arms in performing multiple manipulation tasks.

### 1.2 Active Control

Active control of stiffness can be attained by sensing the forces arising in the manipulator and controlling the actuators to compensate for these contact forces by moving the manipulator in such a way as to mimic the behavior of having physical

springs built into the system. This strategy can be advantageous as it can have a large range of control for how the manipulator will behave. Forces that are sensed in the joints of the manipulator, or on the end effector, or on the fixturing of the component can be compared to the expected load and position data of the configuration. Corrective action can be taken, if necessary, to reduce the undesired load on the system.

Many of the collaborative robots used in industry today rely on active control to create a safe, collaborative working environment shared with humans. Collaborative robots such as the Franka Emika Panda [4], Universal Robots UR10 [5], and the KUKA LBR iiwa [6] use some level of active control in order to simulate mechanical compliance in the system and operate more safely around human workers.

There are limitations to this approach, however. The motor response is delayed by mechanical bandwidth and feedback delay (the time it takes for the sensors to read and send the information to the central controller and for the controller to then send its desired motion to the motors). These delays limit the speed at which a manipulator can perform a task because an excessively high speed would not give the controller enough time to compensate for the contact forces that this method is supposed to prevent.

Feedforward control strategies might be able to reduce or eliminate these feedback delays if the disturbances were understood and modeled well beforehand. However, the unknown disturbances that might happen to a robotic manipulator, such as a foreign object impeding the desired motion or the manipulated object being in the wrong position or orientation, can not be reliably modeled for a feedforward control strategy to be effective.

To eliminate the issues that arise from active control strategies and custom passive compliance tooling, work has been done in developing several kinds of variable stiffness actuators (VSAs). VSAs control the passive stiffness of mechanisms that are linked in series with the links of a robotic manipulator. By controlling and varying the stiffness of a spring-like mechanism, a manipulator can achieve controllable passive elastic behavior without the feedback delays in active control or the need to create new tooling for each task in custom end of arm tooling.

### 1.3 Variable Stiffness Actuators

A review of the various types of VSAs is needed to understand the range of capabilities and limitations that currently exist in the field. Fundamentally, VSAs are devices consisting of a motor (or actuator) and an elastic element that connect to a robotic link allowing for controlled variation of the stiffness of the element. Wolf et al. [7] identified the five major use-cases for VSAs: “shock absorbing, stiffness variation with constant load, stiffness variation at constant position, cyclic movements, and explosive movements.”

Each of these use-cases may require different designs and VSA characteristics. Many VSAs are effectively nonlinear springs (with the most basic example being an extension, compression, or torsion spring manufactured to have a nonlinear stiffness profile such as variable pitch progressive springs) that are placed in series between the mechanism motor and the robotic link. Methods [8] to achieve nonlinear spring behavior from otherwise linear springs include triangle mechanisms, cam mechanisms, four-bar mechanisms, and pneumatic muscles to achieve nonlinear behavior from otherwise linear springs.

One type of VSA design controls position and stiffness independently. It uses one motor to control the stiffness of the joint and another to control the position of the joint. This independent control of the stiffness and position simplifies the control approach. The example schematic, in Figure 1 below, shows the makeup of an independent control VSA similar to that of the DLR's Floating Spring Joint [9].

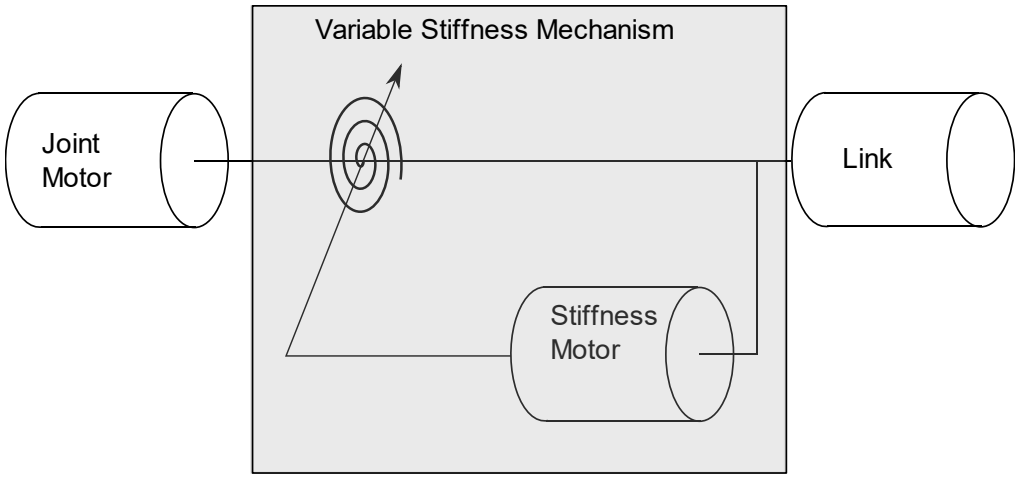


Figure 1: Independent Control VSA Schematic

The stiffness motor in an independent control method typically is located at the joint. The effect of this independent control means that changes to the stiffness setting of the mechanism do not impact the neutral position of the link and therefore the control of each is decoupled. The downside to this approach is that the mass of the variable stiffness mechanism adds additional gravitational load on the joint motor resulting in the need for more expensive, higher torque motors. This added mass also increases the risk of injury or damage due to impact.

Another type of VSA mechanism is inspired by human kinesiology in their utilization of an agonist-antagonist configuration of muscles to control the movement of the body.

Most human skeletal muscles, such as the biceps and triceps in the upper arm, work in agonist-antagonistic pairs where a muscle on one side of the joint contracts while the other relaxes producing movement. The human musculoskeletal system is able to regulate the elastic behavior of joints well and, as a result, many roboticists have developed similar systems to control the position and elastic behavior of robotic joints through similar mechanisms.

Using an antagonistic setup, similar to that seen in Figure 2, allows control of both the stiffness of the link and position of the link semi-independently. Use of two motors, positioned away from the moving links of the mechanism, reduces the moving inertia of the mechanism, making this setup more viable for small mechanisms such as robotic hands and fingers. Drawbacks of this methodology include a more complex mechanism schematic due to both motors being placed away from the joint they are controlling, as well as, the need for synchronization between motors in order to control the position and stiffness as the action of one motor affects both position and stiffness.

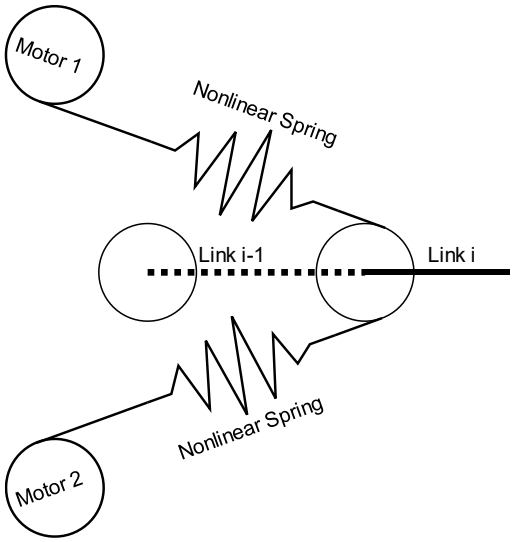


Figure 2: Antagonistic VSA Schematic

For this thesis work, three major design objectives are established to achieve the kind of performance desired. These three objectives are to minimize the inertia of the controlled robotic joint, minimize the overall size of the VSA mechanism, and create a large range of controllable linear stiffness at the robotic joint.

### 1.4 Design Objectives

Low inertia at the joint is one of the fundamental needs of a robotic finger joint. Usually, these robotic hands and fingers are attached to robotic arms to provide additional reach and mobility. Lowering the mass that the attached robotic arm must move around improves safety and reduces cost by requiring smaller, less expensive motors to be used in the robotic arm. By relocating the heavy motors and stiffness controlling mechanisms away from the joint itself, the mass and mass moment of inertia of the joint will be reduced resulting in lower torque requirements for the previous joints in the serial chain. Additionally, the design of the joint can be smaller as there is no need to design supports for the motors and stiffness controlling mechanisms on the finger itself.

The compactness of the VSA mechanism design is another objective of the design required to make it feasible to implement in real-world systems. Creating a joint design that can mimic the compactness of a human finger joint allows for robotic fingers and hands to be made by combining multiple mechanisms together.

The final objective of the mechanism is to create a large range of controllable linear stiffness at the joint. A joint motion, at least as much as a human finger, is needed

for the robotic hand applications. If the force-deflection relationship at the joint is linear, then the joint stiffness will be constant when it is deflected from its equilibrium position.

The intended use-case of the proposed designs in this project is variable stiffness finger joints. Therefore, the independent control setup will not be suitable for these designs and an antagonistic setup will be a better option so that the motors and VSA mechanisms can be placed a distance away from the finger joint. In the following subsections, two state of the art VSA designs are presented.

### 1.4.1 DLR – Flexible Antagonistic Spring Element

The German Aerospace Center (DLR) has developed a Flexible Antagonistic Spring element (FAS) for use in the DLR Hand Arm System [10] [11]. This mechanism utilizes antagonistic nonlinear spring stiffness effects by incorporating a lever and pulley-cable mechanism that creates nonlinear stiffness behavior from a linear extension spring as seen in Figure 3.

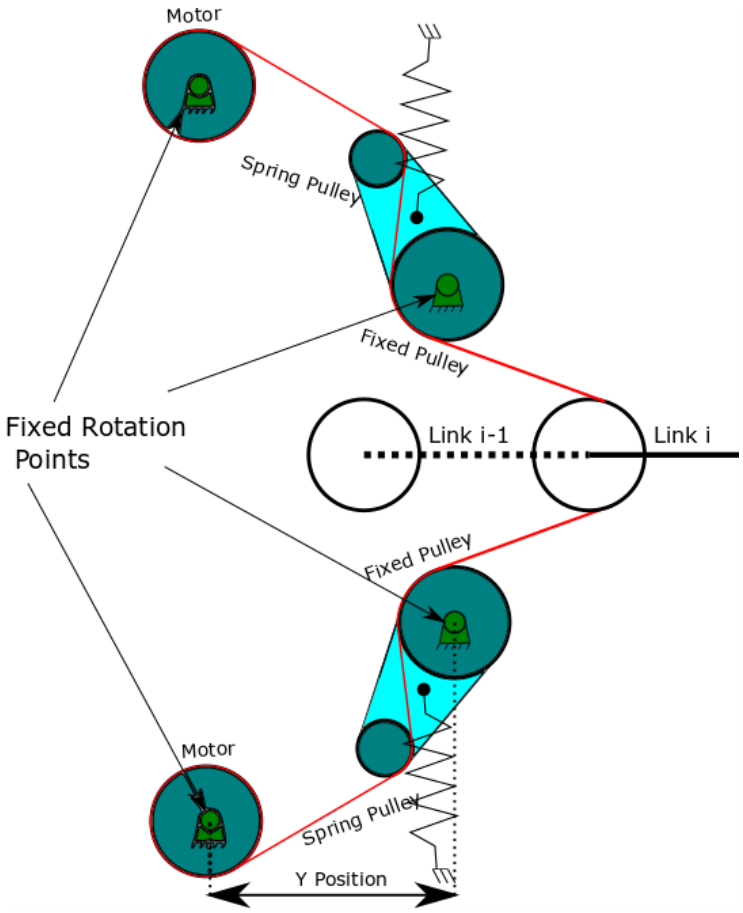


Figure 3: DLR's FAS Mechanism

As the cable within the mechanism is shortened by the motor, the rotation of the lever extends the spring in a nonlinear manner. As stated in Friedl et al [12], “To obtain the required stiffness characteristics, the initial position of the spring, the spring rate and the y position of the lever can be selected. The rest of the parameters are imposed because the tendon routing and motor positions are given. The resulting force-deflection curve of the mechanism can be tuned by optimization or trial and error to achieve a desired behavior.”

No mention is made regarding the exact stiffness characteristics desired, only that a highly nonlinear stiffness profile is desired and that each joint in each finger may have a different desired stiffness profile. Because the stiffness profile of each lever/spring mechanism is not quadratic, the resulting antagonistic stiffness at the joint will not be linear, one of the design criteria of this VSA.

### 1.4.2 Migliore – Biologically Inspired Joint Stiffness Device

The biologically inspired joint stiffness device designed by Migliore et al. [13] utilizes the same antagonistic nonlinear spring theory as the DLR's FAS mechanism to achieve a linearly variable stiffness at a joint using a cable and spring system as seen in Figure 4. The cam-like design of this mechanism allows for more precise control of the resulting mechanism stiffness characteristics by controlling the shape of the cam-like path that the spring rollers follow.

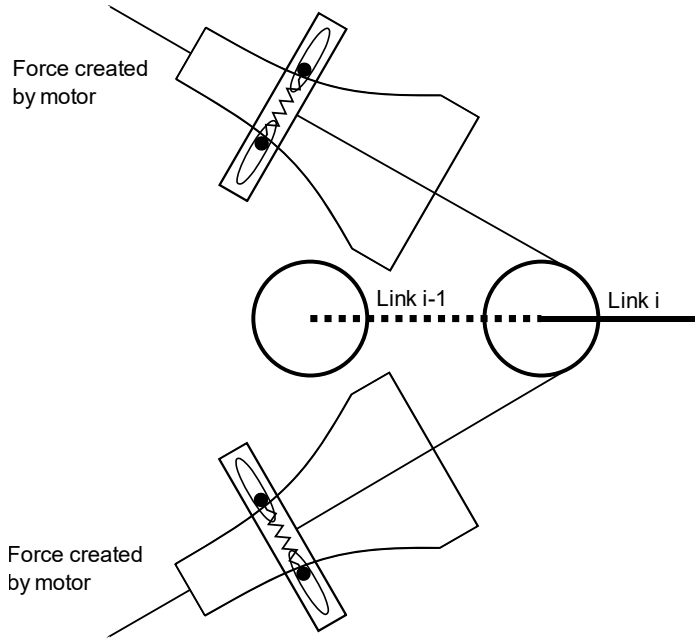


Figure 4: Migliore's Biologically Inspired Joint Stiffness Device

Migliore's mechanism requires a large amount of space to house the VSA mechanism leading to a bulky design of the overall robotic arm that this type of robotic finger may be placed into. The accuracy of the device at low stiffnesses is also non-ideal. At low stiffness configurations, the friction between the rollers and the contoured track can hinder the joint from moving to its expected position causing an error between the expected position and desired position.

Due to these drawbacks of each of the two discussed examples, a new cable-driven antagonistic mechanism should be designed in order to maximize performance in the key objectives discussed above.

### 1.5 Approach Overview

This section identifies the approach to achieving the expected design performance. An overview of two alternative designs is presented below. Each design incorporates a quadratic force-deflection (stiffness) profile. Building from the current state of the art, while keeping in mind the objectives and design goals (low inertia of the link, compact design, and large range of linear controllable stiffness at the link), two cable-driven variable stiffness mechanisms capable of achieving controllable linear stiffness and constant stiffness at the joint given modest deflection are described in this section.

Additionally, a constant stiffness is desired through a large range of deflection from the free length position. This constant stiffness at the joint can be achieved by attaching two opposing springs with quadratic force-deflection characteristics in series with the joint.

### 1.5.1 Opposing Quadratic Spring Behavior

When two elastic mechanisms with quadratic force-displacement relationships are place in opposition to one another, as seen in Figure 5, the resulting stiffness is based on the difference between spring equilibrium positions $(x_R - x_L)$ and the resulting equilibrium position is proportional to the sum of the individual equilibrium positions $(x_R + x_L)$. Therefore, for any desired $x_R$ & $x_L$, a large region of linear stiffness response exists for deflections at the joint. The net force from the two quadratic springs acting on the body ($F_1$ and $F_2$) has a linear relationship with the deflection of the body from its equilibrium position (x). The linear force-deflection relationship is calculated using the following constitutive equations.

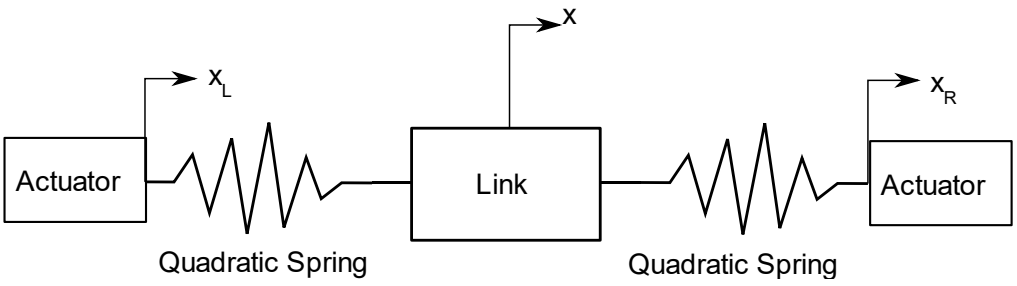


Figure 5: Opposing Quadratic Spring Configuration

$$F_1 = K(x - x_L)^2 \tag{1}$$

$$F_2 = -K(x - x_R)^2 \tag{2}$$

$$F_1 + F_2 = K(x^2 - 2xx_L + {x_L}^2 - x^2 + 2xx_R - {x_R}^2) \tag{3}$$

$$F_1 + F_2 = 2K(x_R - x_L)\left[x - \frac{1}{2}(x_L + x_R)\right] \tag{4}$$

For any desired joint and joint position, control inputs $(x_R - x_L)$ & $(x_R + x_L)$ can be used to achieve any combination of joint stiffness and joint position desired [14].

The relationship between force and deflection from equilibrium is linear as shown in Eq. 4. This behavior allows simple control of both the position and stiffness of the joint without coupling between the positional and elastic behavior of the joint.

### 1.5.2 Description of Design Alternatives

Two alternative approaches to achieving a compact, lightweight, linear stiffness VSA are considered. Both designs, through the control of two antagonistic motors, function by retracting a cable running through the set of spring-pulley mechanisms in order to alter the stiffness and angular position of a 1 DOF finger. Additionally, each design will attempt to produce a quadratic force-deflection behavior on one half of the mechanism that, when attached antagonistically, yield an easily controllable linear stiffness at the joint by utilizing the mathematics provided above. The two designs are described throughout this paper as the Lever Mechanism and the Slot Mechanism.

#### 1.5.2.1 Lever Mechanism Design

The Lever Mechanism design closely mimics the DLR's FAS mechanism described in Section 1.4.1 in utilizing the same style of cable-driven spring-loaded lever system as the FAS mechanism to introduce nonlinearities into the force-deflection curve of the mechanism. The mechanism's match of a desired quadratic force-deflection curve will be improved by geometric parameter optimization (as compared to the DLR's design which attempted to achieve other, unspecified, types of nonlinear behavior). Eight

physical parameters will be optimized to find a set of parameters that yield the desired quadratic force-deflection curve.

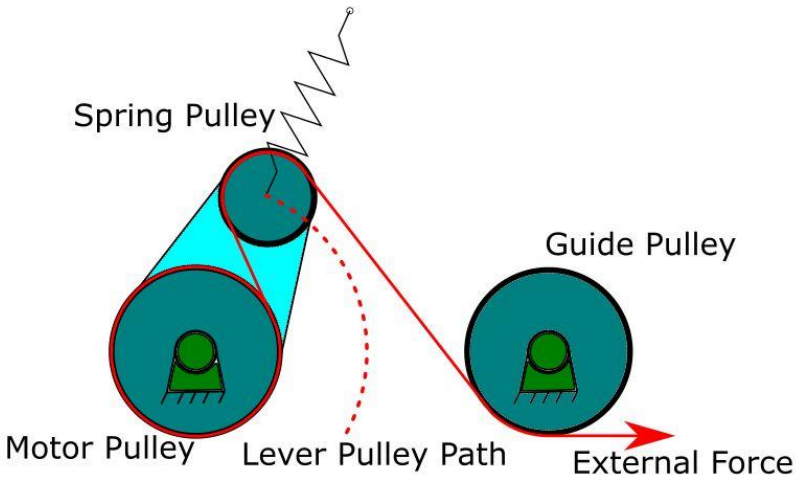


Figure 6: Proposed Lever Mechanism Design

#### 1.5.2.2 Lever Mechanism Design

The basis of the Lever Mechanism design draws from the compact pulley design of the DLR's FAS mechanism and the highly controllable rail design of Migliore et al. in order to create a compact variable stiffness cable mechanism that can be optimized to match the quadratic force-deflection curve needed to achieve linear stiffness at the joint.

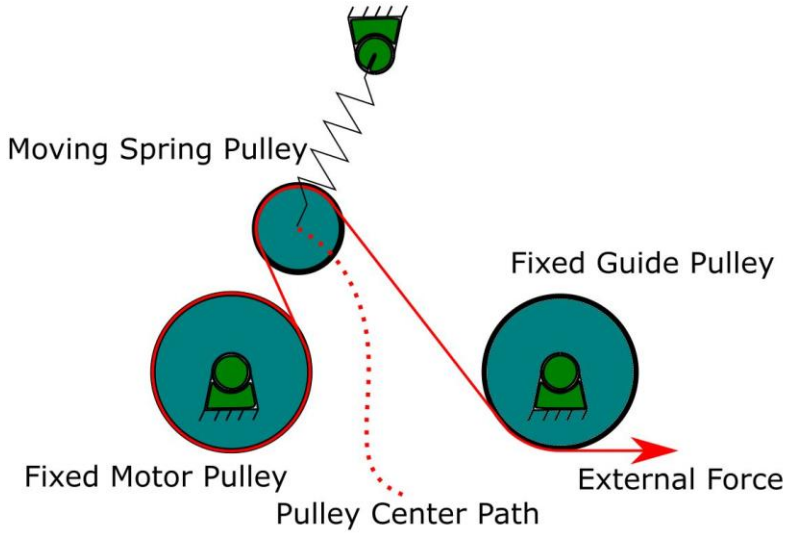


Figure 7: Proposed Slot Mechanism Design

**1.6 Thesis Overview**

This thesis demonstrates a design for a cable driven VSA that achieves controllable linear stiffness across a large range of stiffness values. Chapter 2 presents the mathematical modeling and optimization of the two alternative antagonistic VSA mechanism designs. Chapter 3 describes the prototype design and component selection for the two VSA mechanism prototypes. Chapter 4 provides the testing methodology and results for each of the two VSA mechanisms tested. Finally, Chapter 5 summarizes the findings of the thesis, draws conclusions about the two alternative VSA mechanisms, and makes recommendations for future work.

CHAPTER 2

# QUADRATIC NONLINEAR SPRING MECHANISM DESIGN

This chapter describes the models of the two quadratic antagonistic spring mechanism designs and the optimization of the geometric and physical parameters to achieve the designed force-deflection relationship. The functional design of both mechanisms is reviewed along with a static analysis of the force-deflection behavior of each functional design. The modeling and optimization strategy of each design and the differences between the approaches is investigated. Optimal geometric and physical parameter sets are detailed and discussed.

## 2.1 Functional Design

One of the objectives of the mechanism designs is to produce a quadratic force-deflection behavior on the cable that attaches on one end to the driving motor and on the other to the controlled joint as seen in Figure 8.

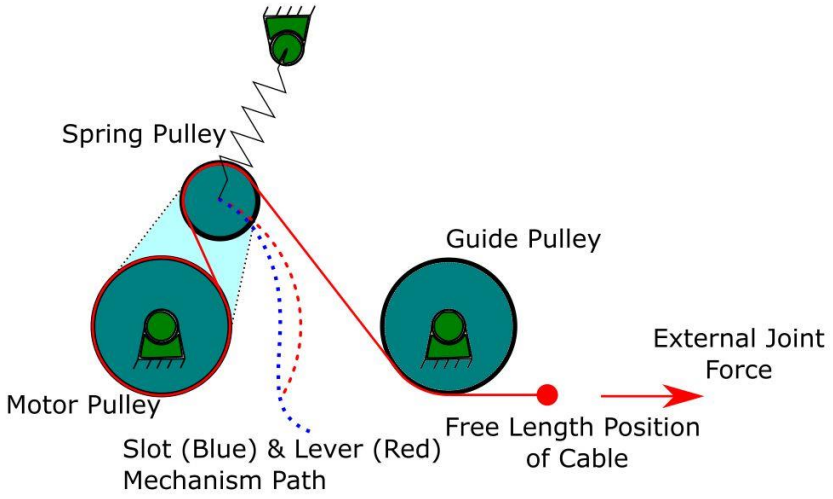


Figure 8: Functional Design of Both Mechanisms

As described in Chapter 1, both designs are made up of three pulleys, a cable, and a linear spring. The motor pulley and guide pulley are in fixed positions while the spring pulley is free to move along a specified path. In the Lever Mechanism, the spring pulley is connected to a lever with a center of rotation at the center of the motor pulley and thus the spring pulley is constrained to move in a constant radius arc denoted by the red dotted line in Figure 8. In the Slot Mechanism, the spring pulley is free to move along a slot with a non-constant radius arc such as the one depicted with the blue dotted line in Figure 8.

In both mechanisms, the spring pulley moves due to the relative length of the cable within the mechanism either due to the position of the attached link or due to the angular position of the motor. Due to the rotation of the linear spring about its fixed end as the spring pulley moves, a nonlinear relationship between the force imposed on the cable end and the deflection of the cable end from its free length is obtained.

Functionally, both mechanisms work using the same agonist-antagonist principle that governs human muscle control. A muscle, or in this case a cable, is attached to either side of a link across a joint and applies torque to the joint in opposite rotational directions. A motor controls the positioning of each end of the cable on both the upper and lower mechanism.

In Figure 9, the four forces acting at the spring pulley center are depicted along with the direction of the available motion. This motion is perpendicular to the force acting on the pulley provided by the lever or provided by the slot constraint. The torque produced by the cables (tendons) can be ignored due to the pulley being able to spin freely, therefore the forces from the cables can be relocated to the pulley's center.

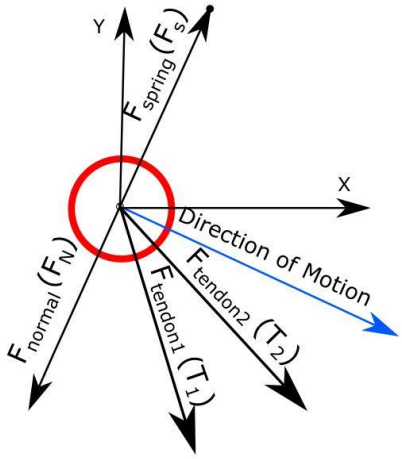


Figure 9: Free Body Diagram of the Pin Joint on the Spring Pulley at the Starting Configuration

Given the known physical and geometric information of each mechanism, including the locations of the pulleys, the fixed point of the linear spring, and the stiffness of the linear spring, and the assumption that the force of the tendon on either side of the spring pulley are equal, the forces in the free body diagram can be resolved because the only two unknown values are the magnitudes of the normal and tendon forces. Because the pulley can spin, friction between the tendon and the pulley is ignored and the quasistatic nature of this analysis assumes the pulley is not accelerating, therefore, the assumption that the tension in the cable on either side of the pulley can be used.

$$|\boldsymbol{T}| = |\boldsymbol{T_1}| = |\boldsymbol{T_2}| \tag{5}$$

$$\boldsymbol{F_N} + \boldsymbol{T_1} + \boldsymbol{T_2} + \boldsymbol{F_s} = 0 \tag{6}$$

$$\begin{bmatrix} (\widehat{\boldsymbol{T}}_{\boldsymbol{1x}} + \widehat{\boldsymbol{T}}_{\boldsymbol{2x}}) & \widehat{\boldsymbol{F}}_{\boldsymbol{Nx}} \\ (\widehat{\boldsymbol{T}}_{\boldsymbol{1y}} + \widehat{\boldsymbol{T}}_{\boldsymbol{2y}}) & \widehat{\boldsymbol{F}}_{\boldsymbol{Ny}} \end{bmatrix} \begin{bmatrix} |\boldsymbol{T}| \\ |\boldsymbol{F_N}| \end{bmatrix} = \begin{bmatrix} -|\boldsymbol{F_s}|\widehat{\boldsymbol{F}}_{\boldsymbol{sx}} \\ -|\boldsymbol{F_s}|\widehat{\boldsymbol{F}}_{\boldsymbol{sy}} \end{bmatrix} \tag{7}$$

Where the subscript “x” indicates the x-direction unit vector of the specified force and the subscript “y” indicates the y-direction unit vector of the specified force.

The relative change in length of the tendon within the mechanism is determined by first calculating the length of the cable within the mechanism from point A to point F, shown in Figure 10, in the mechanism's initial configuration, then calculating the same length for each finite movement taken by the mechanism and finding the difference in lengths. Segments AB, CD, and EF are calculated as arclengths given the radii of the respective pulleys and the tangent points of the lines between the pulleys. Segments BC and DE are calculated as the distance between the two tangent point locations using a developed MATLAB code, Crosstan.m, found in Appendix A.

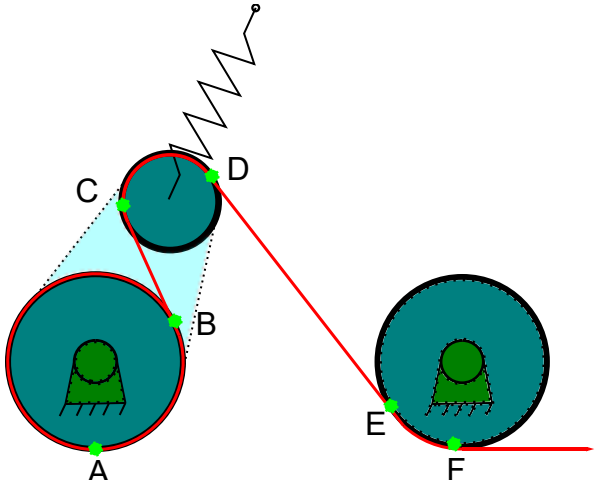


Figure 10: Cable Length Segments

The geometric optimization for each dimensionless design resulted in a required spring constant for the linear spring. The spring constant value is needed to match the specific desired quadratic path with a quadratic coefficient of one. Doubling the spring constant of the linear spring would result in a quadratic path with a coefficient of approximately two for example. This relationship assumes that the spring constant of the cable used in the mechanism is significantly higher than that of the linear spring (in order

for it to not stretch significantly during operation and effectively resulting in an additional linear spring in series). The implication of this relationship between the linear spring constant and the quadratic coefficient of the resulting force-deflection curve is that a variety of springs can be swapped in and out depending on the requirements of the application and the availability of the springs without sacrificing performance.

A requirement was placed on the initial configuration of the mechanism for each iteration such that the mechanism would always start in a zero-stiffness configuration. This requires that the center of the motor pulley, the center of the slot pulley, and the fixed point of the spring are collinear. Because the spring force is perpendicular to the motion, the motion is instantaneously unconstrained. Figure 9 shows the free body diagram of the moving spring pulley in the initial configuration where the spring force and normal force are collinear.

### 2.2 Parametric Modeling Strategy

Along with the general model of the mechanisms provided above, geometric and physical parameters determine the overall elastic behavior of each mechanism. For each mechanism type, these parameters are optimized to identify a mechanism geometry that yields a force-deflection behavior that closely matches the desired behavior. The optimization of each mechanism type maximizes the range of mechanism cable deflection that falls within the user defined acceptable region near the desired nonlinear force-deflection curve. The objective function for both optimizations quantifies the length of deflection each mechanism has within that acceptable region. However, due to the added degree of freedom of the Slot Mechanism, slightly different optimization methods are used in the optimization procedures.

### 2.2.1 Mechanism Design Parameters

The eight geometric mechanism parameters that contribute to the overall elastic behavior of the mechanism as illustrated in Figure 11.

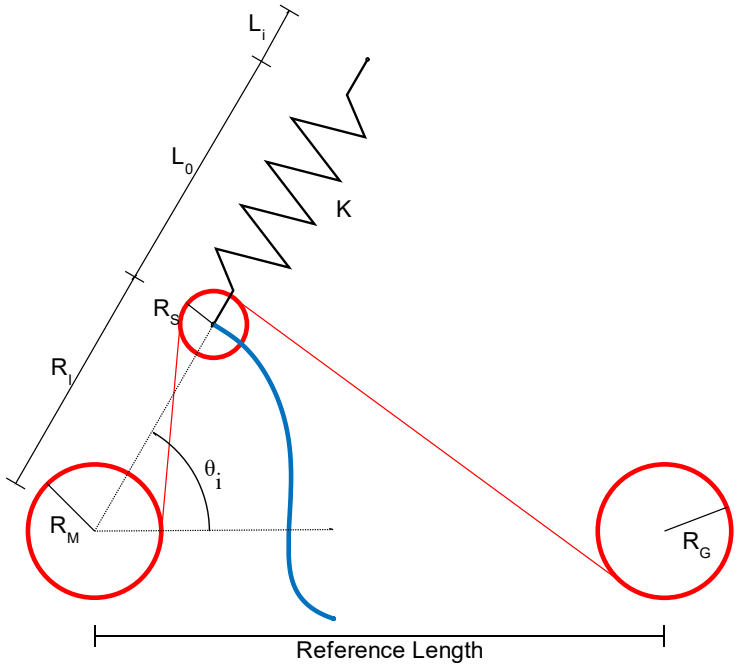


Figure 11: Mechanism Design Parameters

The optimization parameters shown in Figure 11 are defined as:

- $R_M$: radius of motor pulley that is attached to the motor and one end of the cable
- $R_S$: radius of spring pulley that follows the path of the optimized slot or follows the constant radius of the lever
- $R_G$: radius of the guide pulley directing the cable towards the link at which the other end of the cable is attached
- $R_I$: initial distance between the center of the motor pulley and spring pulley

$L_0$: free length of the spring

$\theta_i$: initial angle between the x-axis and the line connecting the centers of the motor and spring pulleys

$L_i$: initial extension length of the spring from its free length

$K_s$: spring constant of the linear spring

All model distance parameters are normalized to a 1-unit reference length relative to the distance between the center of the motor pulley and the center of the guide pulley.

The stiffness of the cable was assumed to be significantly higher than that of the springs in the system and therefore not considered in the analysis. Friction was also neglected in the mathematical model to simplify the analysis. The Lever Mechanism design is ultimately a constrained case of the Slot Mechanism design limited to a constant radius due to the constraint of the physical lever as opposed to a variable radius slot.

### 2.2.2 Lever Mechanism Optimization Methodology

In the design optimization of the Lever Mechanism, the eight parameters were optimized to best match the desired quadratic force-deflection behavior. The built-in MATLAB nonlinear optimization function, fmincon, was utilized as the optimization method to find the maximum value of the optimization objective function.

**Lever Mechanism Optimization Standard Form:**

**Max** $L^*$

**s.t.** $R_M + R_S - R_I + 0.1 < 0$

$R_S + R_G + R_I \cos(\theta_i) - 0.95 < 0$

$10(K_s) - K_C < 0$

$LB \leq R_M, R_S, R_G, R_I, L_0, \theta_i, L_i, K_S \leq UB$

where $L^*$ is the change in length of cable from initial to final position and is a function of geometric and elastic properties ($R_M$, $R_S$, $R_G$, $R_I$, $L_0$, $\theta_i$, $L_i$, $K_S$). LB and UB are the upper and lower bounds based on physical limitation estimates.

The lever path is determined by the set of geometric parameter values. The path of the moving lever pulley will sweep from the zero-stiffness configuration towards the fully taut configuration with the cable being fully straightened as seen in Figure 12.

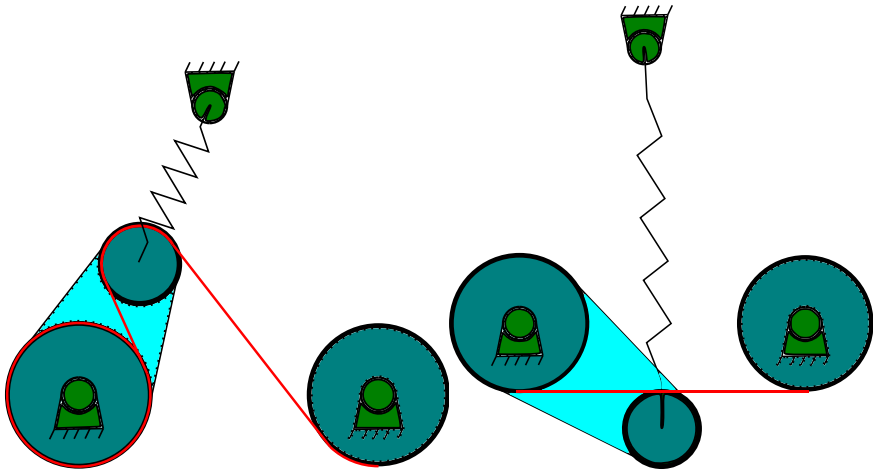

Figure 12: Zero Stiffness Configuration (Left) and Cable Taut Configuration (Right)

The full range of angular positions of the Lever Mechanism is divided into 1000 equal segments and the change in cable length within the mechanism from the zero-stiffness configuration and the tension on the cable is calculated. This force-deflection

data is then differentiated to achieve a stiffness-deflection curve to be compared to the desired linear stiffness-deflection curve that results from an exactly quadratic force-deflection curve and the optimization objective value is calculated for each set of parameters.

An error band of ±0.05 is set around the desired linear stiffness-deflection line and each of the 1000 calculated values of the mechanism stiffness-deflection curve is determined to be within or not within the error tolerance band. The longest deflection length in which the calculated stiffness-deflection curve is within the error tolerance band is output as the objective function value for the candidate mechanism which is being maximized by the optimization program. The optimization's objective function was evaluated using the following logic.

**Lever Mechanism Optimization Pseudocode:**
**Set** $L^*$ = 0 (Current best deflection range & objective function value)
**Set** $L^*_{new}$ = 0 (Current deflection range measure)
**For** i = 1 to 1000 (the number of segments of mechanism positions)
    **Evaluate** $T_i$ (Tension in cable in current position i)
    **If** $T_i$ > minimum error threshold & $T_i$ < maximum error threshold
        **Set** $L^*_{new} = L^*_{new}$ + deflection from segment i
    **Else**
        **If** $L^*_{new} > L^*$
            **Set** $L^* = L^*_{new}$
        **End**
        **Set** $L^*_{new}$ = 0
    **End**
**End**
**Output:** $L^*$ as objective function value

Note that the portion of the deflection range that falls within the error bounds does not necessarily have to start from the zero-stiffness configuration. The acceptable deflection range that is output by the objective function can begin at a non-zero stiffness

configuration as long as it is the longest deflection range in which the calculated stiffness-deflection values fall within the error tolerance band.

The reasoning for creating an error tolerance band for the stiffness-deflection curve is because, in early tests, the Lever Mechanism designs were not able to maintain an exactly linear stiffness-deflection curve for any finite deflection therefore a tolerance band was introduced in order for the optimization program to function correctly. An error band of ±0.05 was selected as it was large enough to allow for significantly large objective function values while being narrow enough to prevent poor stiffness-deflection curves from being considered good.

Three constraints are placed on the geometry to ensure that the mechanism does not collide with itself. First, the sum of the radius of the motor pulley and the radius of the spring pulley must be less than the length of the lever to ensure the two pulleys do not touch each other. Second, a similar condition is imposed to prevent the spring pulley from contacting the guide pulley throughout the motion of the spring pulley. Finally, the spring constant of the cable, assumed to be very stiff, must be at least an order of magnitude greater than the spring constant of the linear spring in order to ensure real springs could be purchased with spring constants similar to values optimized for. The constraint equations used in the optimization can be seen below using the parameter variables discussed in Section 2.2.1.

$$R_M + R_S - R_I + 0.1 < 0 \tag{8}$$

$$R_S + R_G + R_I \cos(\theta_i) - 0.95 < 0 \tag{9}$$

$$10 * K_S - K_C < 0 \tag{10}$$

### 2.2.3 Lever Mechanism Optimization Results

The optimized Lever Mechanism geometric parameters and force-deflection results can be seen in

Figure 15.

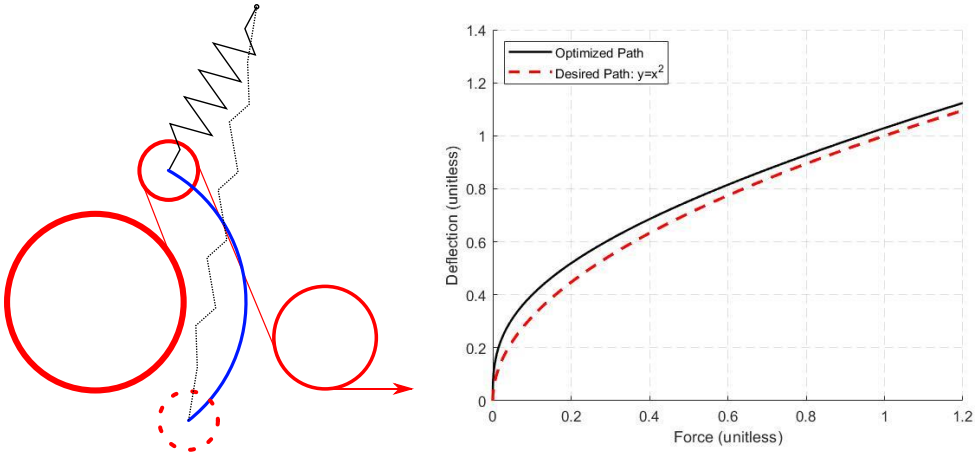


Figure 13: Optimized Lever Mechanism Geometry and Performance

The optimized Lever Mechanism resulted in an objective function value of 0.7942. This indicated that, for a displacement of 0.7942 of the refence length, the optimal set of parameters was able to fall within the ± 0.05 error bounds of the objective function limits. The stiffness-deflection curve can be seen in Figure 14 along with the desired linear stiffness-deflection curve and the ± 0.05 error bounds. Note that the acceptable region of the mechanism does not have to begin at the zero-deflection position at the origin of the graph in Figure 14. In this optimized mechanism, the acceptable region falls between a deflection of approximately 0.5 and 1.3 units of deflection, resulting in the objective function value of 0.7942.

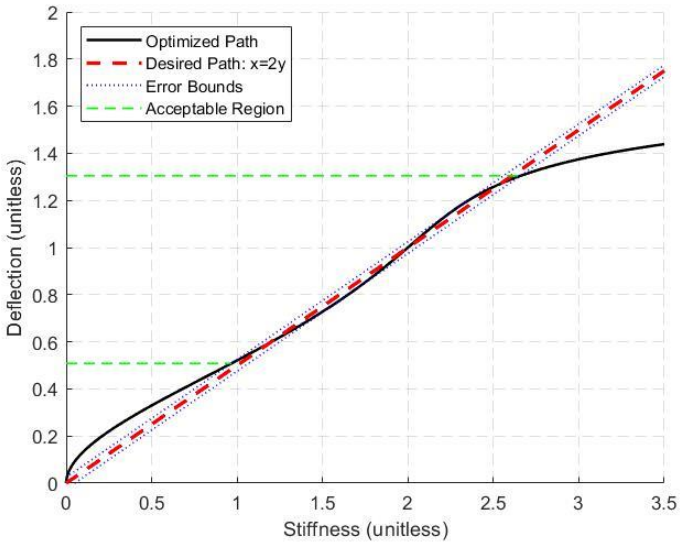


Figure 14: Optimized Lever Mechanism Stiffness-Deflection Curve

In the prototype implementation of this mechanism, a non-optimal configuration, with an objective function value of 0.5064, was selected in order to reduce the number of total parts requiring fabrication for the set of prototype mechanisms in order to reduce cost and lead time. The radii of the motor pulley and guide pulley were selected to be the same size as the motor and guide pulley radii of the Slot Mechanism and the optimization was re-run to find the optimal configuration given these new constraints. In Table 1 below are the optimization values for both the constrained and unconstrained Lever Mechanism optimizations.

Table 1: Optimized Lever Mechanism Parameters

| Mechanism Parameters | Optimization Values | Constrained Optimization Values |
|---|---|---|
| $R_M$ | 0.3997 | 0.2394 |
| $R_S$ | 0.1612 | 0.1498 |
| $R_G$ | 0.0660 | 0.2320 |
| $R_i$ | 0.6609 | 0.4958 |
| $L_0$ | 1.4881 | 0.57977 |
| $\theta_i$ | 60.703 | 64.663 |
| K | 6.5383 | 3.7411 |
| $L_i$ | 0 | 0 |

The final constrained optimization appears to qualitatively match the desired force-deflection curve well. This non-optimal solution does not look as if it will perform much worse than the original optimized version and thus the compromises in the prototype configuration was deemed good enough to test a physical prototype. The final Lever Mechanism parameter design and force-deflection curve can be seen in Figure 15.

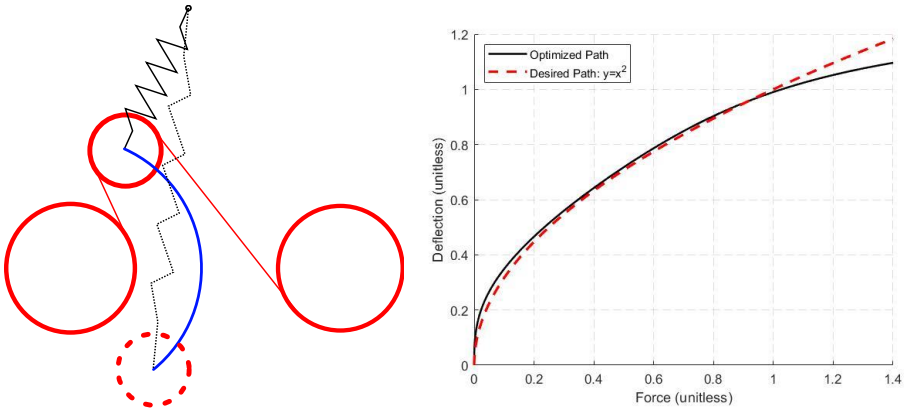


Figure 15: Final Lever Mechanism Configuration Mathematical Results

The stiffness-deflection curve of the optimization with additional constraints can be seen in Figure 16 along with the desired linear stiffness-deflection curve and the ±0.05 error bounds. The acceptable region for this non-optimal mechanism falls between a deflection of approximately 0.25 and 0.75 units from the initial configuration resulting in

the objective function value of 0.5064. At a deflection of 0.75, the optimized path exits the error bounds for a short distance before re-entering the acceptable region until a deflection of approximately 0.86.

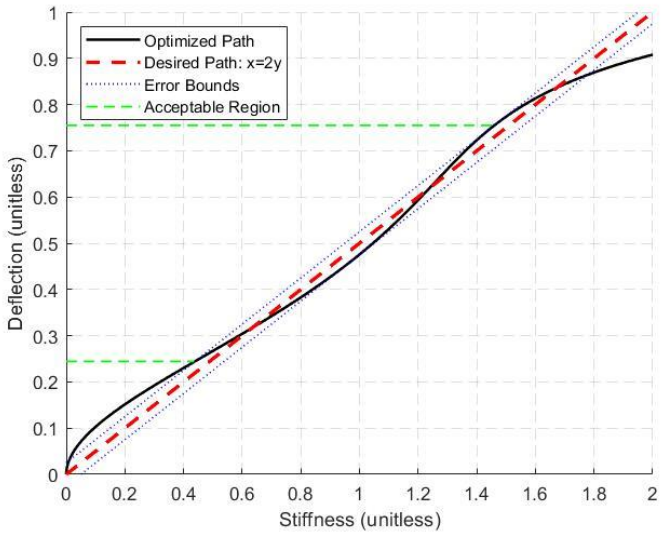


Figure 16: Final Lever Mechanism Stiffness-Deflection Curve

Qualitatively, the mechanism design with additional constraints, to match the pulley size of the Slot Mechanism design, looks to match the desired path better than the original optimal design. This discrepancy between the qualitative and quantitative results indicates the level of sensitivity this optimization approach has when choosing an error limit for the mechanism and how the selection of an error limit may affect one's ability to find the optimal mechanism design.

### 2.2.4 Slot Mechanism Optimization Methodology

For the Slot Mechanism Design, a nested optimization was developed to find the design parameters and slot shape that best matched the desired quadratic force-deflection characteristics.

**Optimization Standard Form:**

**Max** $L^*$

**s.t.** $R_M + R_S - R_I + 0.1 < 0$

$R_S + R_G + R_I \cos(\theta_i) - 0.95 < 0$

$10(K_s) - K_C < 0$

$LB \leq R_M, R_S, R_G, R_I, L_0, \theta_i, L_i, K_S \leq UB$

where $L^*$ is the change in length of cable from initial to final position and is a function of geometric and elastic properties ($R_M$, $R_S$, $R_G$, $R_I$, $L_0$, $\theta_i$, $L_i$, $K_S$, $\boldsymbol{\theta}$, $\mathbf{R}$). LB and UB are the upper and lower bounds based on physical limitation estimates and $\boldsymbol{\theta}$ and $\mathbf{R}$ are vectors of polar coordinate values creating the shape of the optimized slot.

The outer loop of the program varied the eight design parameters within the limits imposed for manufacturability and compactness purposes. For each set of design parameters selected by this outer optimization loop, an inner loop would track the path of the slot pulley such that the desired quadratic force-deflection curve would be matched for as long as possible. Once the tracked path could no longer match the path of the desired path, the inner loop would terminate, and an optimization objective value would be sent to the outer loop. The objective value represented the length of cable deflection that would be achieved by that set of design parameters while still maintaining the desired quadratic force-deflection characteristics. The added degree of freedom of the Slot Mechanism design allows for exact tracking of the desired quadratic force-deflection

curve. Therefore, there is no need for an error bound around the desired curve similar to the error bound required in the Lever Mechanism case. The Slot Mechanism objective function for each parameter set in the optimization was evaluated using the following logic.

**Slot Mechanism Optimization Pseudocode:**
**Set** $L^* = 0$ (Current best deflection range & objective function value)
**Set** step_size = 0.001 (Initial change in deflection of each step)
**While** step_size > $10^{-10}$
    **Optimize** $\theta$ and $R_i$ (the angle and radius from origin to next point on slot curve)
        **s.t.** $T_i = T_{desired}$ at $L_i$ (if possible)
    **If** $T_i = T_{desired}$ at $L_i$ (desired curve still matches calculated curve)
        **Set** $L^* = L_i$
        **Set** $L_i = L_i$ + step_size
    **Else**
        **Set** step_size = step_size/2 (take a smaller step)
    **End**
**End**
**Output:** $L^*$ as objective function value

This more complex optimization strategy is required due to the extra degree of freedom provided by the variable radius slot path. Given only the initial configuration of the mechanism, the entire path cannot be resolved as the slot path is not yet known. This added complexity requires the inner optimization of the program to be created to find the best slot path shape for any given set of geometric parameters.

The original three constraints were placed on the mechanism that were placed on the Lever Mechanism geometries. These constraints prevent the motor and spring pulleys from colliding, prevent the spring and guide pulley from colliding, and ensure that the ratio of the spring constants of the linear spring and the cable are realistically achievable with stock linear extension springs. The fourth constraint from the Lever Mechanism optimization, requiring a match between the motor pulley radius and guide pulley radius

of the Lever Mechanism and Slot Mechanism can be ignored because the results of the Slot Mechanism optimization constrains the Lever Mechanism but the opposite was not true during implementation.

### 2.2.5 Slot Mechanism Optimization Results

The optimized geometric parameters and slot path shape can be seen in Figure 17 along with the force-deflection results.

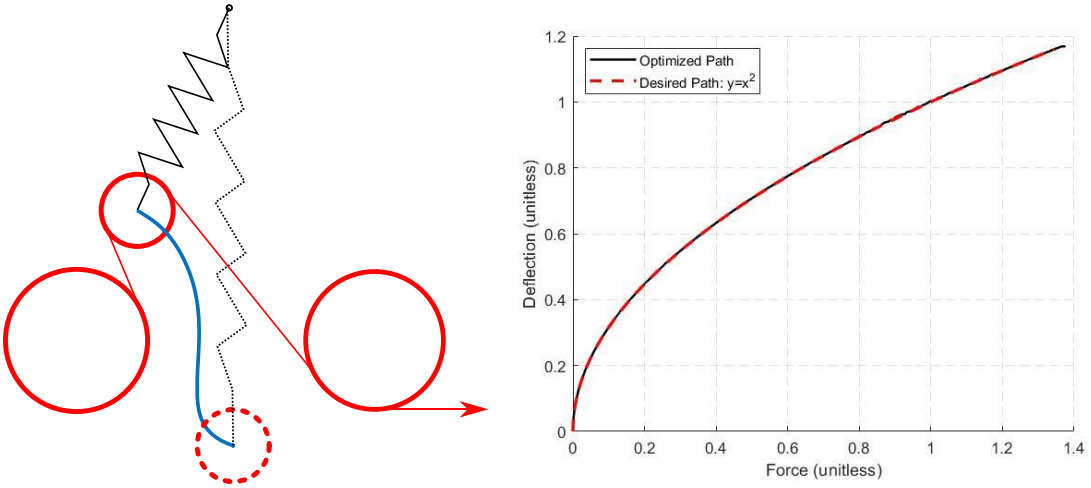


Figure 17: Final Slot Mechanism Configuration and Mathematical Results

The optimized slot design yielded an objective function value of 1.1685, meaning the mechanism can draw the cable a reference length of 1.1685 while still maintaining the desired quadratic relationship.

Table 2: Optimized Slot Mechanism Parameters

| Mechanism Parameters | Optimization Values |
|---|---|
| $R_M$ | 0.2394 |
| $R_S$ | 0.1371 |
| $R_G$ | 0.2321 |
| $R_i$ | 0.5291 |
| $L_0$ | 0.7537 |
| $\theta_i$ | 62.640 |
| K | 3.5121 |
| $L_i$ | 0.0221 |

## 2.3 Conclusion

The Lever Mechanism is effectively a special case of the Slot Mechanism design in which the radius of the "slot" in the Lever Mechanism's case is fixed. Given a constraint of a constant radius to the optimization of the Slot Mechanism, the slot would provide a normal force collinear to the line between the motor and spring pulley identical to the normal force create by the lever in the Lever Mechanism calculations. Therefore, this investigation into the two alternate mechanism designs is a look into how important that constraint of a constant radius slot is on the ability to optimize the mechanism to match the desired nonlinear behavior.

While both the Slot Mechanism and Lever Mechanism optimizations yield theoretical performances that adequately approximate a quadratic force-deflection curve (needed to achieve antagonistic linear stiffness at the joint), the Slot Mechanism design provides the better match. The Slot Mechanism is able to do this because of the additional degree of freedom in the variation of the slot path.

# CHAPTER 3

# PHYSICAL DESIGN

The physical prototypes of both mechanisms were designed and fabricated to evaluate the quality of the match between the theoretical and experimental performance. Both styles of mechanism were built into a single test apparatus. Descriptions of the Lever Mechanism and Slot Mechanism incorporated into the test apparatus are provided below.

## 3.1 Design Overview

Both antagonistic cable-driven finger mechanisms' physical implementations were designed based from the optimizations discussed in Chapter 2. A single prototype with two antagonistic mechanisms controlling a single joint was fabricated with interchangeable parts to allow for swapping between the Slot Mechanism and Lever Mechanism without the need for duplicate parts that would add to the overall cost of the prototypes. In Figure 18, the final fabricated prototype can be seen in its Slot Mechanism on the left and Lever Mechanism on the right. The test apparatus was scaled to size using a 50mm length to substitute for the 1-unit reference length used in the mathematical optimization. This size allowed for easily obtainable, off the shelf bearings and shafts to be used.

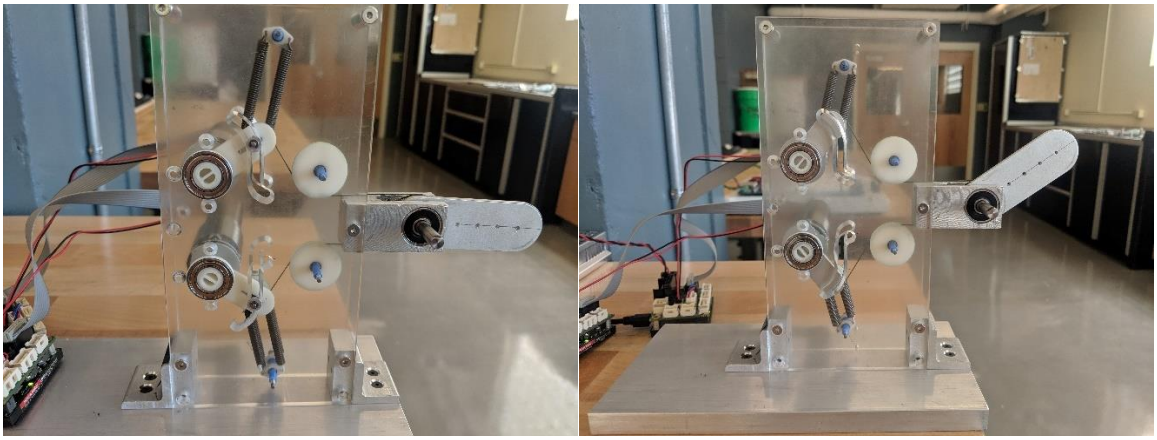

Figure 18: Test Apparatus for the Slot Mechanism (Left) and Lever Mechanism (Right)

Functionally, both mechanisms work using the same agonist-antagonist principle that governs human muscle control. A muscle, or in this case a cable, is attached to either side of a link across a joint and applies torque to the joint in opposite rotational directions. A motor controls the positioning of each end of the cable on both the upper and lower mechanism. For example, when the effective length of the cable on top is shortened, a torque is applied to the finger joint causing an upward deflection along with a stiffening of the mechanism due to force in the cable causing a displacement in the linear springs.

Due to part availability and budgetary constraints some deviations from the optimal designs were used. The stiffness values of the linear springs did not match the optimized models due to a limited supply of off-the-shelf springs. Additionally, in order to reduce the number of parts needed for the overall testing, the motor pulley and guide pulley radii for the Lever Mechanism were selected to be the same as the pulley sizes for the Slot Mechanism. This reduced the number of pulleys that needed to be fabricated and the time it takes to swap the prototype apparatus between mechanism types. To compensate for

these deviations, the models were re-run after the physical prototypes were made and tested using the measured dimensions of each mechanism in order to compare the physical prototypes performances with the computer models.

As described in Chapter 1, having mechanisms in opposition to each other, each with quadratic force-deflection curves, allows for control of both the stiffness and angular position of the joint using the two motors and a mathematical model of the system.

Given the constraints on the Lever Mechanism design discussed in Chapter 2 regarding limiting the motor and guide pulleys to be the same size as those of the Slot Mechanism design, only a few components needed to be swapped out between the two designs. The springs in the two optimizations are different lengths requiring a swap of the springs as well as their attachment point location. Additionally, the lever and lever pulley must be swapped out for the slot pulley and slot shaft that are needed for the Slot Mechanism to follow the path of the optimized slot.

## 3.2 Detailed Design Features

Various aspects of the design for both the Slot Mechanism and Lever Mechanism will be discussed in detail in this section regarding the design choice reasonings as well as the implications to the physical implementation and results.

### 3.2.1 Motor Selection and Motor Pulley Design

Two Maxon DC motors with 150:1 ratio planetary gearheads and built-in rotary encoders were selected along with Maxon’s EPOS4 positional control drives to act as the control motors in this system. These motors were selected for their compact design and high torque to size ratio allowing for enough torque to drive a robotic link 50 mm in

length to lift at least 20 lbs. With the final configuration of the mechanism's pulleys and joint lengths, the motors were designed to handle loads of approximately 21 lbs. at the end of the robotic joint.

Pulleys slotted onto the shaft of the Maxon motors acted as the "motor pulley" as described in the mathematical optimizations discussed in Chapter 2. These pulleys rotate with the motor shaft and are designed to allow for the mechanism cable to attach to a standard screw attached to the top of the pulley and wrap around the pulley to allow for minimal inadvertent compliance to be added to the system. As seen in Figure 19, spacers were also integrated into the design of the motor pulleys to properly position the pulleys along the motor shaft in order to align it properly with the rest of the mechanism pulleys and finger joint.

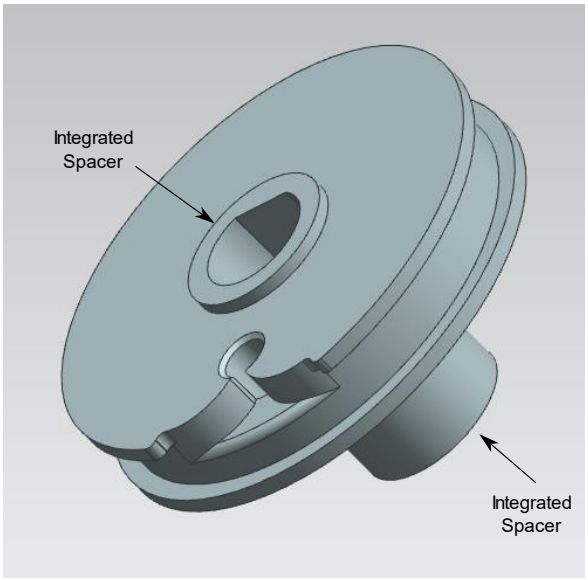


Figure 19: Motor Pulley Design

### 3.2.2 Lever Piece Design

Different lever pieces are used for each style mechanism and perform different functions in the two different mechanism styles. Lever pieces for each style are shown in Figure 20. The lever piece for the Slot Mechanism is used solely to measure the angle of the moving pulley. While the lever piece for the Lever Mechanism is also used to measure the angle of the moving pulley, it also acts as the guide for the circular path of the pulley. In order to measure the angle of the moving pulley, potentiometers are inserted into the top portion of each lever. This angle measurement can then be used to estimate the position of the moving pulley and therefore the state of the mechanism and resulting force being applied to the finger joint through the cable.

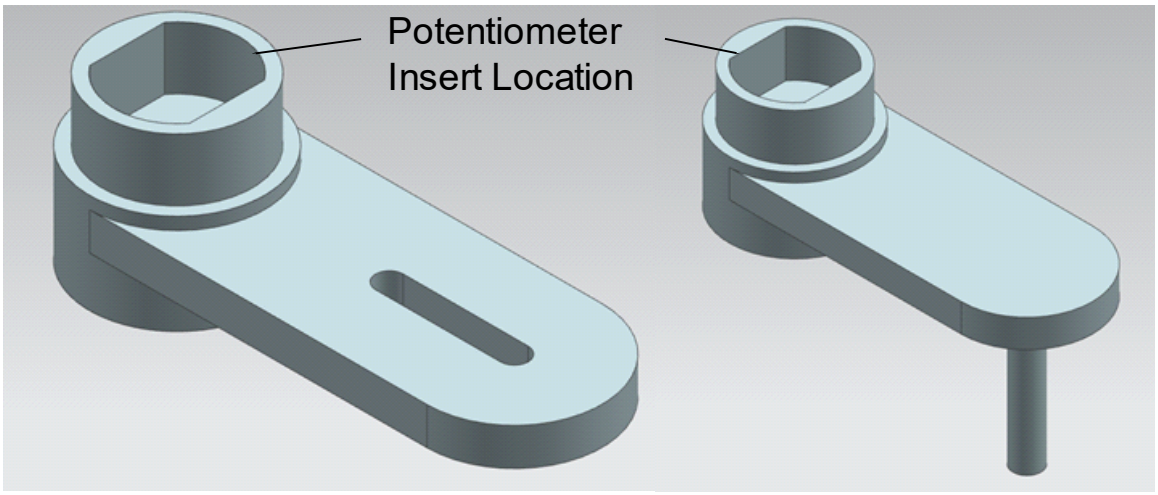


Figure 20: Lever Piece for Slot Mechanism (Left) and Lever Mechanism (Right)

In the lever piece design for the Lever Mechanism seen in Figure 21, the moving lever pulley and the spring cradles are attached directly to the shaft on the lever piece. This shaft is spaced away from the rotational center of the lever at a distance specified by the optimized lever design. The moving lever pulley and spring cradles rotate freely from

the lever allowing for reduced friction in the design. Due to this lever piece being load bearing, the piece was machined from aluminum to prevent significant deflection of the attached shaft. Significant deflection in the shaft would result in errors between the expected mechanism configuration and the true mechanism configuration resulting in discrepancy between the measured and mathematical models of the mechanism.

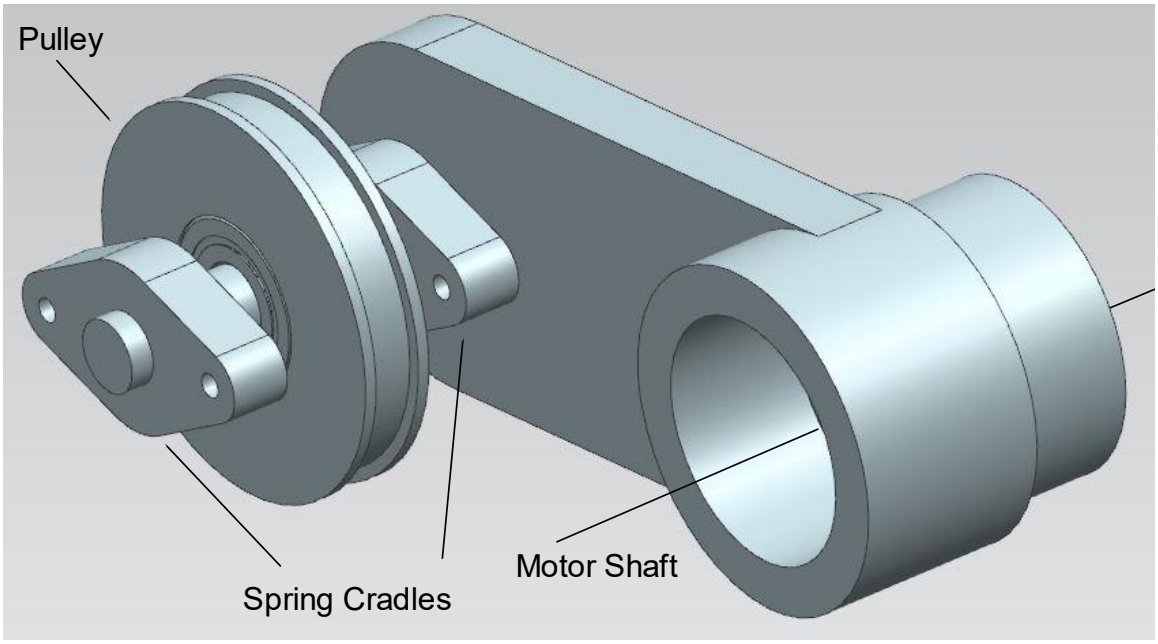


Figure 21: Lever Mechanism Lever Piece & Shaft Assembly

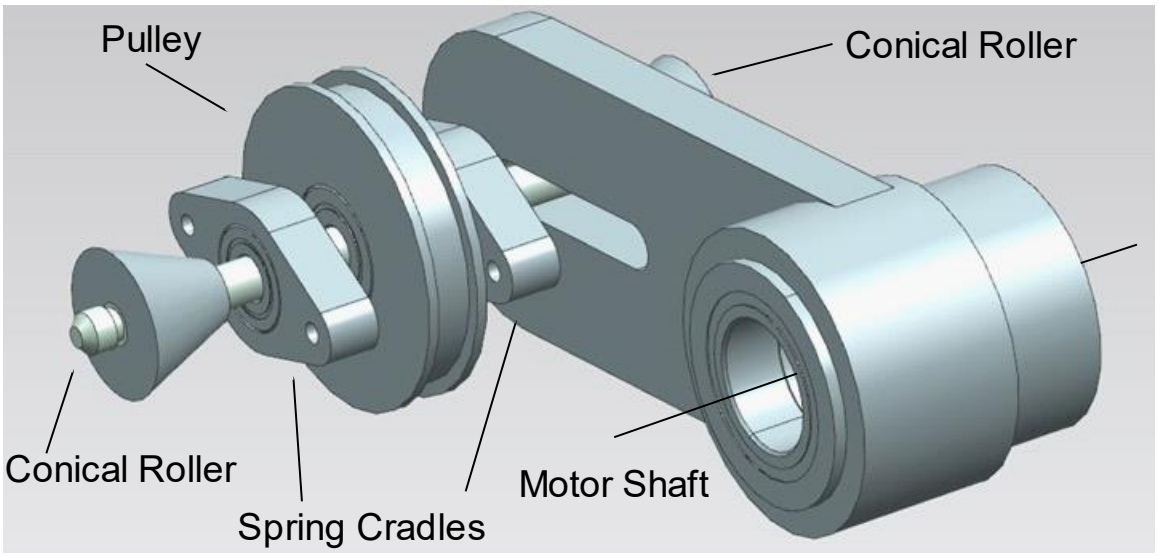


Figure 22: Slot Mechanism Lever Piece & Shaft Assembly

In the lever piece design for the Slot Mechanism seen in Figure 22, the moving slot pulley that follows the path of the optimized slot is affixed to a shaft that is inserted through the linear slot on the lever piece. This linear slot allows for the position of the moving pulley to measured. As opposed to the Lever Mechanism design, the lever piece in the Slot Mechanism is not load bearing and is only used for angle measuring purposes. A high stiffness 3D printed plastic was used for the piece because of the minimal loading requirements for the part. Which reduced cost while ensuring minimal off-axis deflection or rotation of the lever piece that would result in measurement error.

### 3.2.3 Slot and Roller Shaft Design

From its inception, binding and sticking of the rollers in the optimized slots was a concern. The conical shape of the rollers rolling in matching cone shapes slots proved to be qualitatively better than cylindrical rollers in the early mock-ups of the design. The cone shapes, seen in Figure 23, acted as a centering mechanism to prevent the shaft from obtaining a skewed orientation with respect to the slots on either side of the shaft.

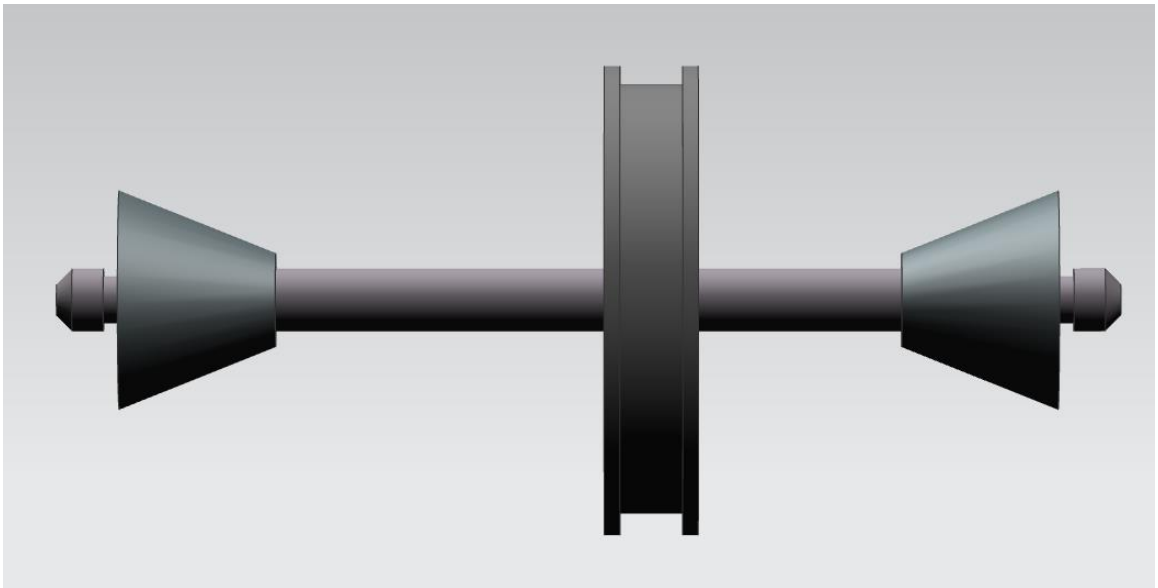

Figure 23: Slot Shaft, Conical Rollers, and Slot Pulley Assembly

The slot pulley is off-center as a result of an attempt to reduce the overall size of the mechanism framework, as well as, to re-use some of the same mechanism components, such as the base frame, joint attachment, and motor placement, in both the Lever Mechanism and Slot Mechanism. However, this design choice, in addition to the slot lever and spring cradle pieces being attached to this same shaft, causes a variety of forces and torques to arise that have the potential to lead to twisting and binding of the rollers in the optimized wall slots. Ultimately, function is more important than compactness and in future design attempt, this slot pulley should be designed in a way to ensure symmetric loading of the shaft to prevent twisting and binding in the slot.

### 3.2.4 Spring Selection

Off-the-shelf linear extension springs were selected for each mechanism. Selection of both spring stiffness and spring length were limited thus springs were

selected based on the following criteria. The allowable spring deflection for the spring must meet or exceed the expected deflection that would happen in the system in order to prevent unwanted plastic deformation of the spring. The selected spring lengths were close to but not quite as long as the desired spring length resulting and some undesired pretension that would need to accounted for in the mathematical comparison, and the spring rates would be selected to be as large as possible in order to achieve the largest possible range of stiffnesses. In Table 3 & Table 4 below, the desired characteristics of the springs for both designs are compared to the characteristics of the best stock springs from spring distributors.

Table 3: Desired and Actual Slot Spring Characteristics

| Characteristic | Desired Slot Spring | Actual Slot Spring |
|---|---|---|
| Spring Length (mm) | 37.68 | 35.10 |
| Allowable Deflection (mm) | 19.25 | 25.15 |
| Spring Rate (N/mm) | 3.51 | 3.012* |

* Four springs with spring constant of 0.753 N/mm were used in combination

Table 4: Desired and Actual Lever Spring Characteristics

| Characteristic | Desired Lever Spring | Actual Lever Spring |
|---|---|---|
| Spring Length (mm) | 28.83 | 25.40 |
| Allowable Deflection (mm) | 27.84 | 27.94 |
| Spring Rate (N/mm) | 3.74 | 1.892* |

* Four springs with spring constant of 0.473 N/mm were used in combination

Spring cradles that allow for 4 springs to be used in parallel for each mechanism were created in order to achieve acceptably high stiffness levels in the mechanism. These cradles are distributed equidistant on either side of the slot pulley in order to offset the torque created by the offset loading as seen in Figure 24.

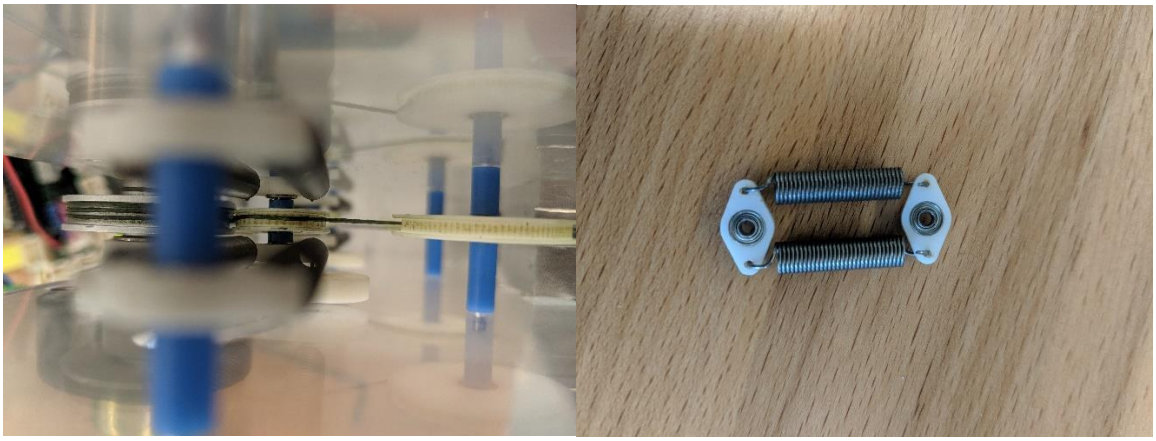

Figure 24: Top View of Spring Cradles in Use (Right) and Spring Cradles (Left)

These cradles allowed the actual stiffness of the system to more closely match the desired stiffness in the system, however, the achieved stiffness are still lower than desired. For larger designs, custom high stiffness springs or additional cradles could be added to the system to allow for increased loading or high stiffness requirements.

### 3.2.5 Cable Material Selection

Selection of the cable material was also important in the implementation. Along with the high strength requirement to ensure the cable can stand up to the applied loads on the system, high stiffness in the cable material is crucial to reduce unaccounted for compliance in the system. If the cable material allowed for significant stretching during loading, the actual stiffness of the mechanism would be reduced compared to the expected stiffness from the mathematical model.

Spiderwire Ultracast Ultimate Braid fishing wire with an 80 lb. breaking strength was selected for its high stiffness, sufficiently high break strength, and low friction. The reduced friction of the fishing line is due to a polytetrafluoroethylene coating on the

surface similar to Teflon™ coatings. The fishing line is made from braided ultra-high molecular weight polyethylene (Dyneema®), which exhibits high stiffness at least one order of magnitude higher than the stiffness of the selected springs used in testing.

### 3.2.6 Finger Joint Design

The finger link was designed to accomplish several goals. As with other components, the material selection of aluminum was to reduce unwanted bending and deflection of the piece that would cause unaccounted for compliance in the system. The cable routing method, illustrated in Figure 25, allowed for the cables to properly wrap around the joint to ensure the desired joint range was possible. Finally, some method of torque application needed to be designed into the finger to achieve reliable and repeatable static torque application for testing purposes.

Figure 25, shows the design for the finger link. A shaft mounted onto a bearing runs through the pivot point of the joint allowing for smooth rotation. Four small holes are drilled at known 10mm increments to allow for various static loads to be hung from the joint providing a known applied torque.

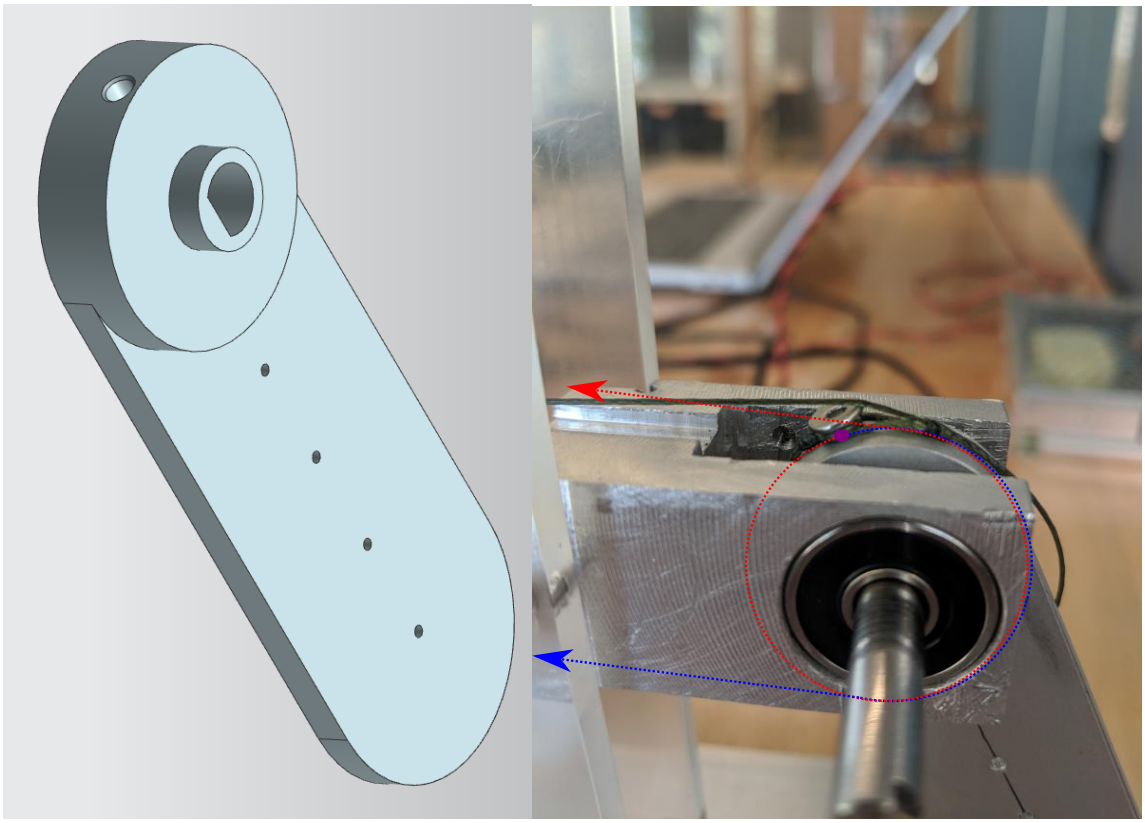

Figure 25: Finger Link Design (left) & Cable Attachment Method (right)

Similar to the cable attachment method of the motor pulley, a screw is inserted into the back of the finger joint and the cable is wrapped around the joint and back into the mechanism. This method allows for 180 degrees of rotational range, much larger than that of a real human finger joint. One drawback of this design is at a link deflection of approximately ±75°, the cable will contact the screw as seen in Figure 25. This interaction results in a larger lever arm than expected for the cable acting on the joint limiting the quality of the data at high angles of deflection. For future, more specific applications, especially applications that only require large deflections in one direction, the location of the attachment screw could be placed in a location such that interactions like this do not occur.

# CHAPTER 4

# TESTING & RESULTS

The two different mechanism designs 1were fabricated and tested to determine whether actual performance was consistent with theoretical (simulated) performance. The tests were designed to confirm that the added complexity of the Slot Mechanism outweighs the potential issues that the Slot Mechanism may cause, and to determine what effect the minor errors between the desired quadratic path and the optimized path have on the overall performance of Lever Mechanism.

The mechanism prototypes were subjected to a series of static loading tests to determine the quality of match between the physical testing and the mathematical model, the reliability and quality of the two prototypes compared to each other, and the level of quality of each implementation to drive a single degree of freedom robotic finger.

## 4.1 Single Complaint Actuator Mechanism Testing

Both the Slot Mechanism and Lever Mechanism were first subjected to a set of single sided mechanism testing to determine the how close to matching the desired quadratic force-deflection curve that a single side of the antagonistic mechanism should produce. This result was then compared to the predicted curve from the mathematical model with the actual design parameters input into the mathematical model. By applying incrementally increasing, known loads to the end of the mechanism cable and measuring the deflection of the cable, the force-deflection curve of the mechanism can be determined.

### 4.1.1 Single Compliant Actuator Testing Procedure

The cable of the upper half of the prototype mechanism was detached from the finger joint and a bag to hold the applied load was attached to a cable hanging off the end of the measurement table as depicted in Figure 26.

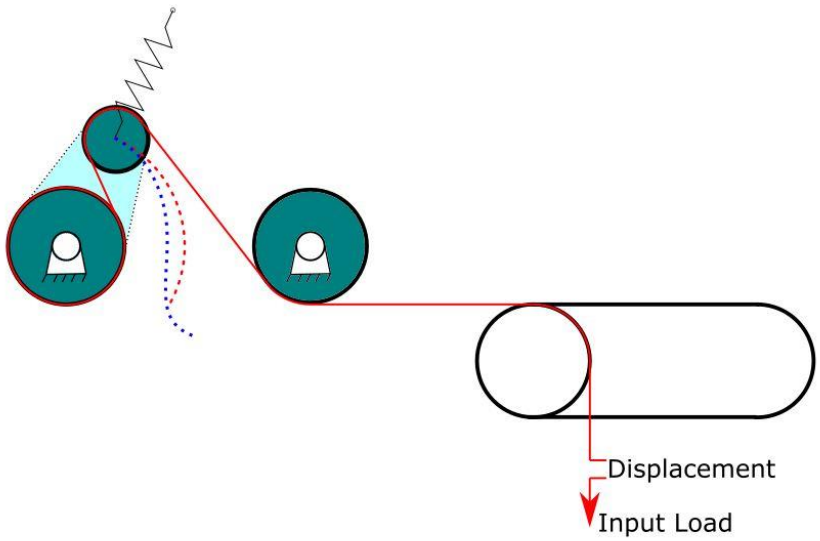


Figure 26: Single Sided Mechanism Testing Configuration

50g masses were incrementally added to the bag attached to the end of the cable and for each 50g added, the displacement of the cable was measured. The displacement of the cable was measured using a visual inspection of a meter stick with resolution of 0.5 mm (half the size of its 1 mm markings). Mass was added to the bag until a maximum deflection position was reached or a mechanism maximum load was reached. From the known mass increments and the measured displacement of the mechanism, a graph of the displacement vs force curve of the mechanism is obtained. This methodology was repeated several times for both the Slot Mechanism and the Lever Mechanism to evaluate the repeatability and reliability of the mechanism and the test methodology.

### 4.1.2 Single Sided Testing Results

Results for the single sided testing were mixed. The numerical model of the Slot Mechanism did not match the testing data well while the test data itself was also rather inconsistent and sporadic. Figure 27, shows the results of the two tests of the Slot Mechanism and compares these to the expected results from the mathematical model.

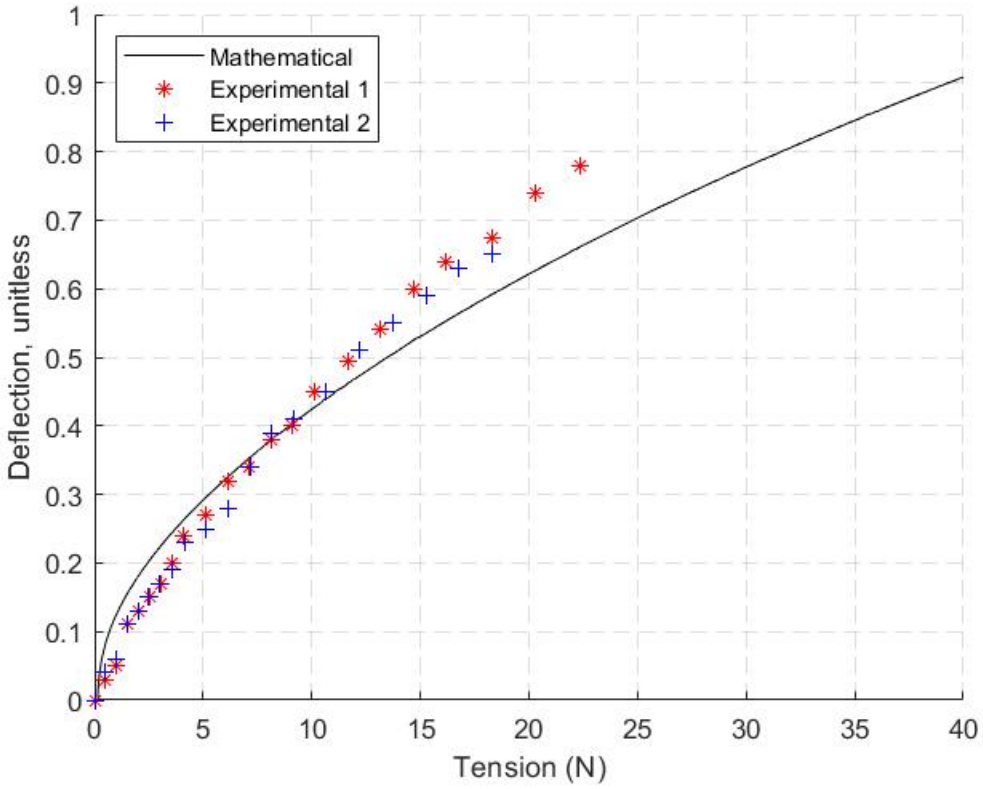


Figure 27: Testing Results for Single Sided Slot Mechanism

The experimental data appears more linear than the expected results, probably due to higher than modeled friction and binding happening in the slot. This added friction

results in a higher required load to be applied to achieve the same level of deflection as a frictionless system.

The testing had to be stopped short of the maximum deflection of the Slot Mechanism due to the slot pulley mechanism binding causing the measured deflection of the mechanism to no longer be related to the applied load. This binding is likely due to the off-centered loading of the slot pulley shaft causing an off-axis moment. The cocking of the mechanism due to this off-centered loading can be plainly seen in Figure 28.

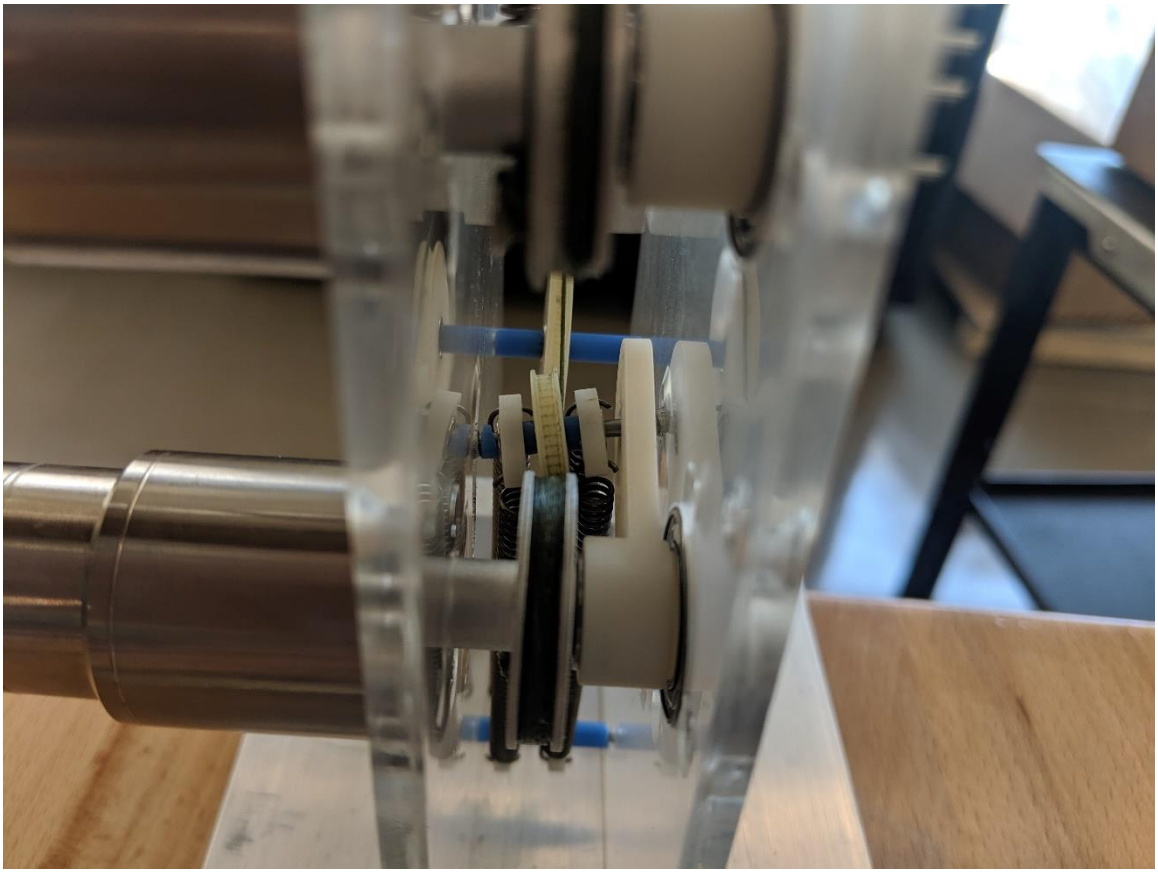

Figure 28: Off-Centered Loading Results in Binding of Mechanism in Slot

In contrast, the single sided testing of the Lever Mechanism appeared relatively consistent with the mathematical models and repeatable between tests. The lever design

allowed for the off-centered loading of the lever shaft to be counteracted by the bearings holding the lever in place. The Lever Mechanism design prevented any sticking or binding of the mechanism and a relatively frictionless system. Figure 29, shows the Lever Mechanism's single sided test results compared to the mathematical estimation of the system using the actual parameters of the prototype system.

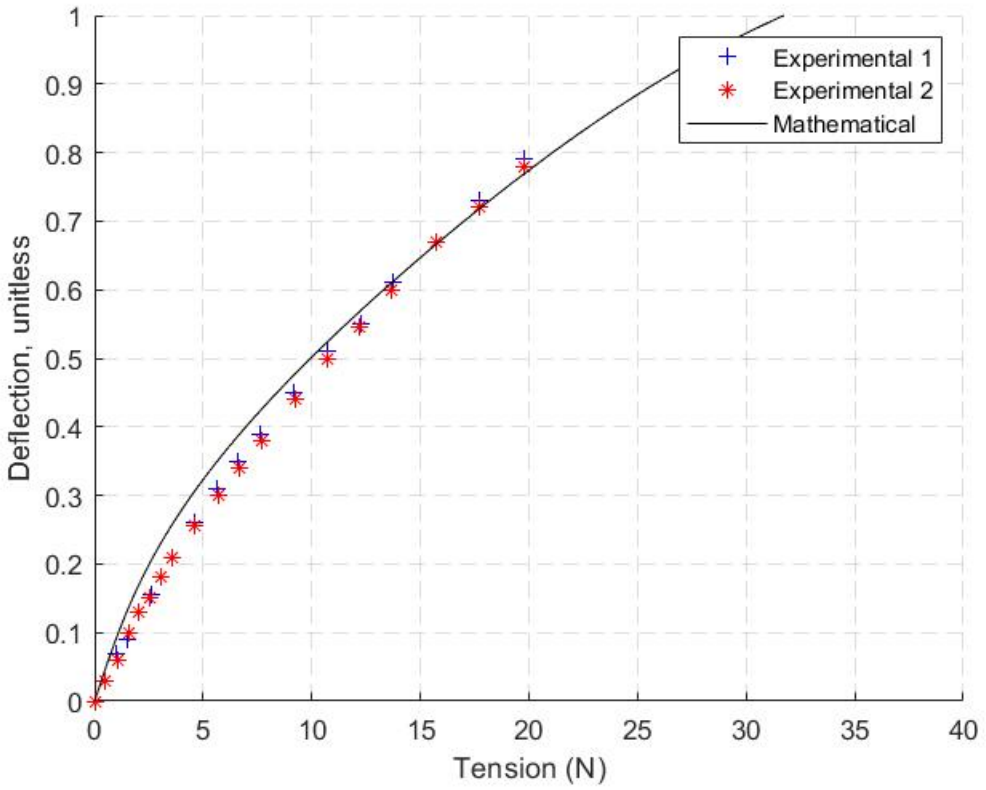


Figure 29: Testing Results for Single Sided Lever Mechanism

Similar to the Slot Mechanism results, the experimental results of the Lever Mechanism design are more linear than the predicted curve. This linearity could be due, in part, to friction in the system however, the measurement of a few key design parameters may also be to blame. Mismeasurements of the stiffness of the linear springs, the linearity of the springs, the stiffness of the cable, or the level of pretension in the

springs could result in this sort of mismatch between experimental and mathematical results. In Figure 29, the spring stiffness is estimated to be approximately 105 N/mm and the pretension of the spring is measured to be 3.72mm. Figure 30 shows the error bounds of the mathematical model resulting from an estimated parameter error of ±1mm in the pretension of the springs and ±5% error in the measured linear spring rate. Figure 30, it appears that the misalignment between the mathematical model and experimental results stems from these two highly sensitive parameters and perhaps some unaccounted-for friction resulting in more linear than expected behavior.

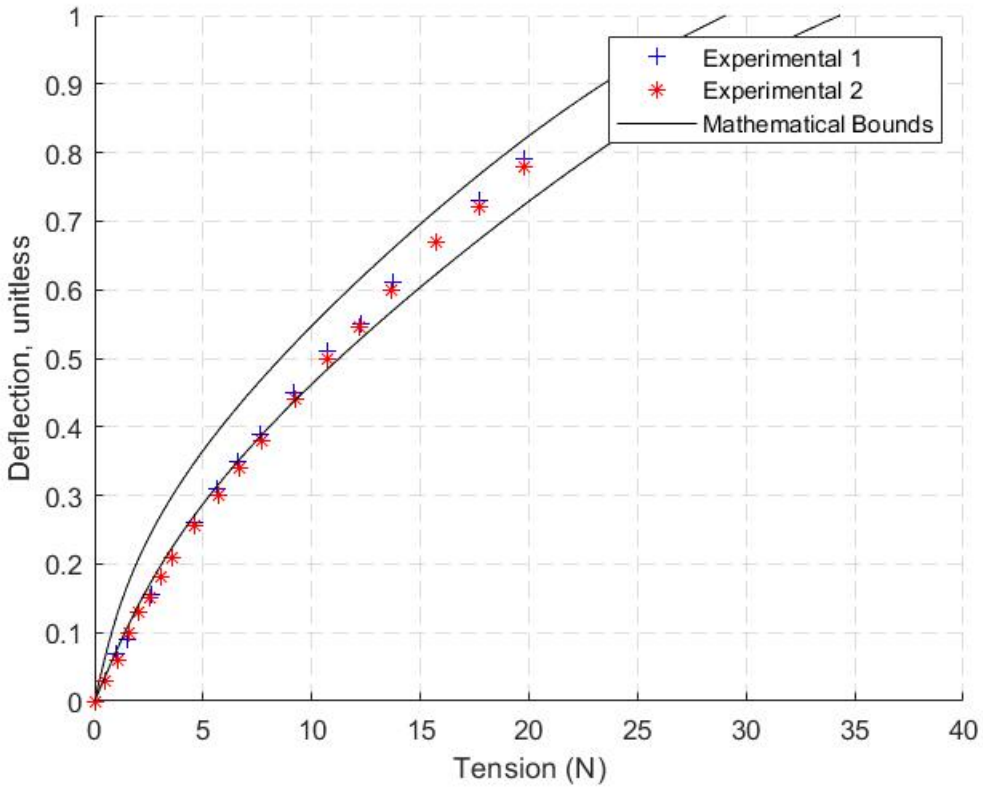


Figure 30: Error Bounds for Single Sided Lever Mechanism

## 4.2 Two Compliant Actuator (Antagonistic) Mechanism Testing

Each prototype mechanism was also subjected to a set of testing for the full finger joint mechanism to determine the reliability of the design, the repeatability of the results, the stiffness profile at the finger joint, and the controllability of the stiffness. The experiments are designed to obtain the relationship between torque applied to the finger joint by the applied loading versus the angular deflection of the finger joint.

### 4.2.1 Measurement Devices

Potentiometers were used to measure the angles of the lever pieces in the mechanism as well as the angle of the finger joint. The potentiometers can be seen in Figure 31. Linear potentiometers were selected over more costly but accurate options such as rotary encoders due to budgetary limitations. The potentiometers also added some additional friction to the system but were determined to not be detrimental to the quality of the data collection.

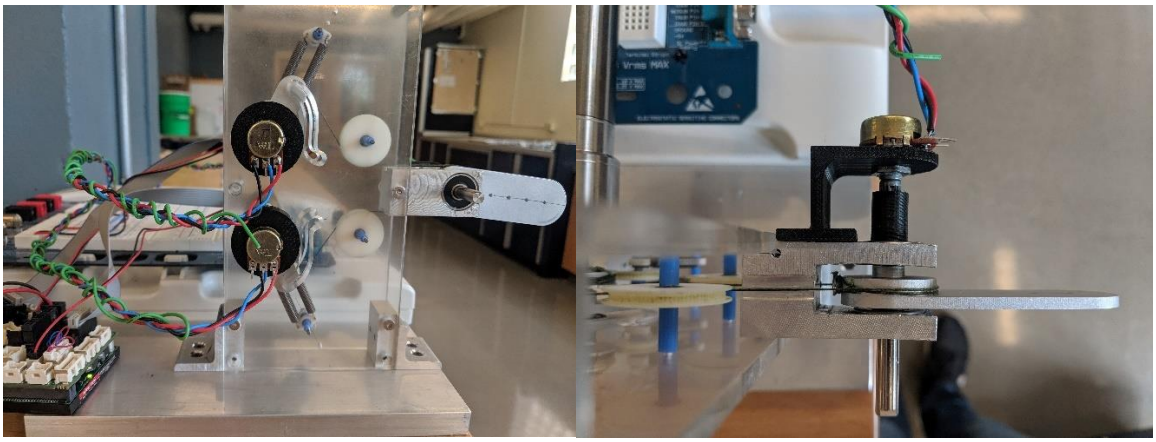

Figure 31: Potentiometers Attached to Mechanism Levers (Left) and Finger Joint (Right)

### 4.2.2 Two Complaint Actuator (Antagonistic) Testing Procedure

A series of tests were conducted at various levels of stiffness for the joint. For each set of tests, both actuators were set to an initial configuration. The motors were wound to contract the cable so that the lever arms for each side of the mechanism were at approximately the same angle and the finger joint was horizontal. Once the mechanism was set into position, a known mass was hung from the finger joint to provide a load torque to the mechanism. As mass was incrementally added, the finger joint angle was recorded, joint torque was calculated using the known mass, known distance between the center of rotation and the attachment point of the mass, and the measured joint angle.

A sweep of various mechanism stiffnesses were tested in approximately 10º increments of the lever arm angles and the finger joint was positioned in a zero degree from horizontal position as well as a +20º and -20º angle configuration. Examples of various mechanism configurations can be seen in Figure 32 and Figure 33.

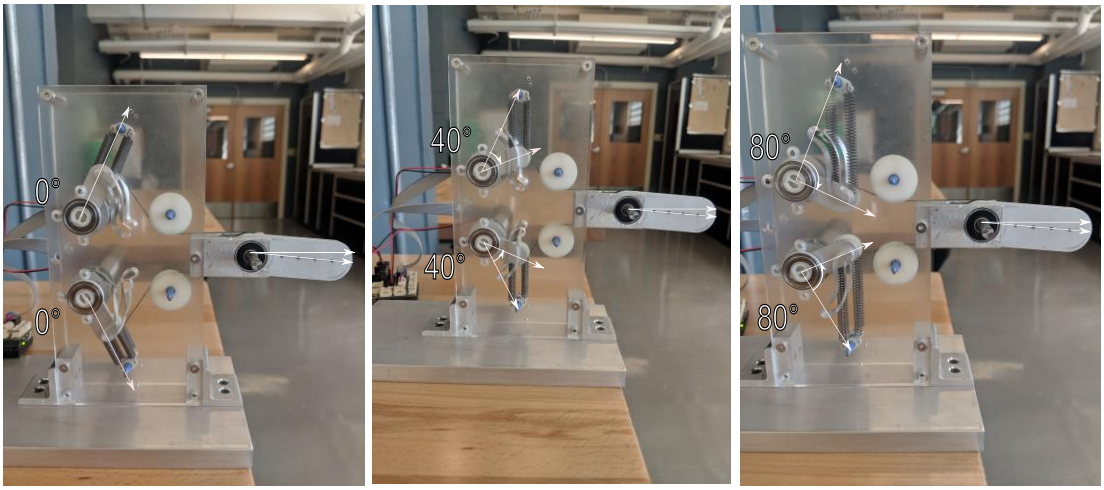


Figure 32: Various stiffness configurations of the Antagonistic Mechanism

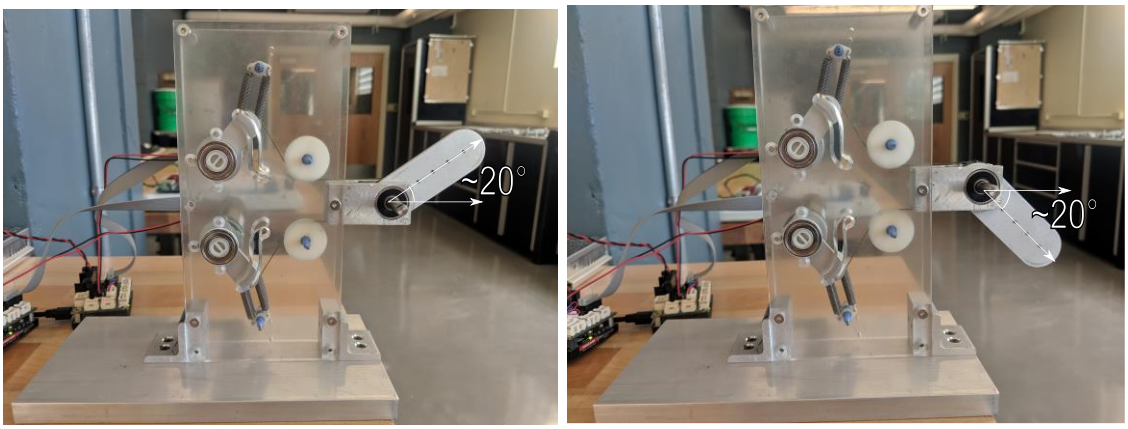


Figure 33: Antagonistic Mechanism with +20° and -20° Finger Joint Angles

When possible, the finger joint was loaded until a deflection of approximately 40° was achieved; however in high stiffness cases, only a small amount of deflection was obtained. Both mechanisms were tested in a variety of configurations. For each set of tests the initial position of the finger joints were set to either a 0°, -20°, or 20° angle as shown in Figure 33, and the motor positions were set such that the mechanism lever angles started at 10° from the zero-stiffness configuration. Each subsequent test incremented the mechanism lever angles by 10° up to 50° (i.e., 10°, 20°, …, 50°) as shown in Figure 32. When the finger joint position of the Lever Mechanism was set to 0°, the tests were conducted up to an initial lever angle position of 80° to evaluate the highest stiffness settings.

The results of each test were then evaluated individually to check for the quality of the linear behavior, and collectively to evaluate the repeatability of the mechanism to achieve a specific position and stiffness multiple times. Theoretically, tests with the same top and bottom lever angles but different joint angles should result in the same stiffness at the joint; this was verified by compared the tests of different finger joint angles with each

other. However, it should be noted that exact angles of the levers and joint were difficult to replicate exactly from test to test, especially at low stiffnesses, due to the different torque applied to the system due to the weight of the finger joint itself at different angles.

### 4.2.3 Antagonistic Mechanism Results

Similar to the results found in the single-sided mechanism testing, the results for the antagonistic mechanism testing showed much higher levels of consistency and linearity for the Lever Mechanism compared to the Slot Mechanism. A representative sampling of the testing results is provided in this section, while the rest of the test data is found in Appendix C. The Slot Mechanism antagonistic testing results were both qualitatively and quantitatively poor. The mechanism did not display reliable consistency between duplicate tests nor did individual tests show acceptably linear behavior.

Figure 34, shows a suite of tests ranging from a relatively low stiffness of 3.78 N/mm to a higher stiffness of 9.18 N/mm using a sweep of slot angles between 10° and 50°. While there is a relative trend of increasing stiffness at the joint as the slot angle increases, the linearity of each line is poor and the range of achievable stiffnesses with the mechanism is low. Higher stiffness values at the mechanism joint are theoretically achievable with the design. However, due to wedging of the slot shaft in the slot, higher stiffness tests were not able to be completed. Additionally, lower stiffness values would theoretically be achievable in this setup; however, the high friction of the slot rollers in the slot resulted in poor results at extremely low stiffness levels as the torque applied could not overcome the static friction of the system.

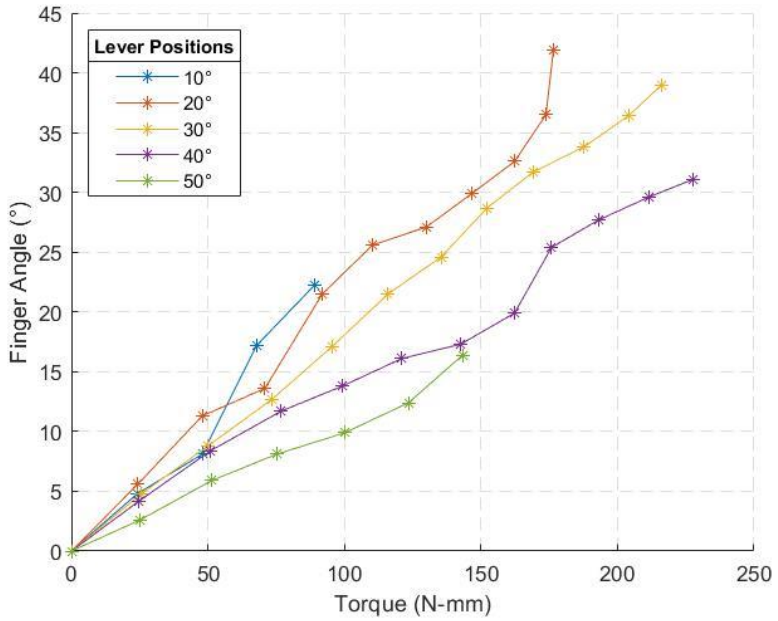


Figure 34: Slot Mechanism Antagonistic Results for 0° Joint Angle

The consistency of the Slot Mechanism antagonistic testing was also poor. Duplicate tests rarely matched and the reliability of the mechanism to achieve the same configuration and stiffness given the same motor positions was inconsistent. Figure 35 shows several examples of the kinds of matches the mechanism achieved in duplicate tests with A and B being examples of relatively successful matches, and C and D being more common poor matches. Figure 35 A-C show results when the motor angles start at 40° from the zero-stiffness position while Figure 35 D shows a result while the motor angle starts at 20° from the zero stiffness position. The starting motor angle controls the stiffness of the mechanism with a higher angle resulting in a higher mechanism stiffness.

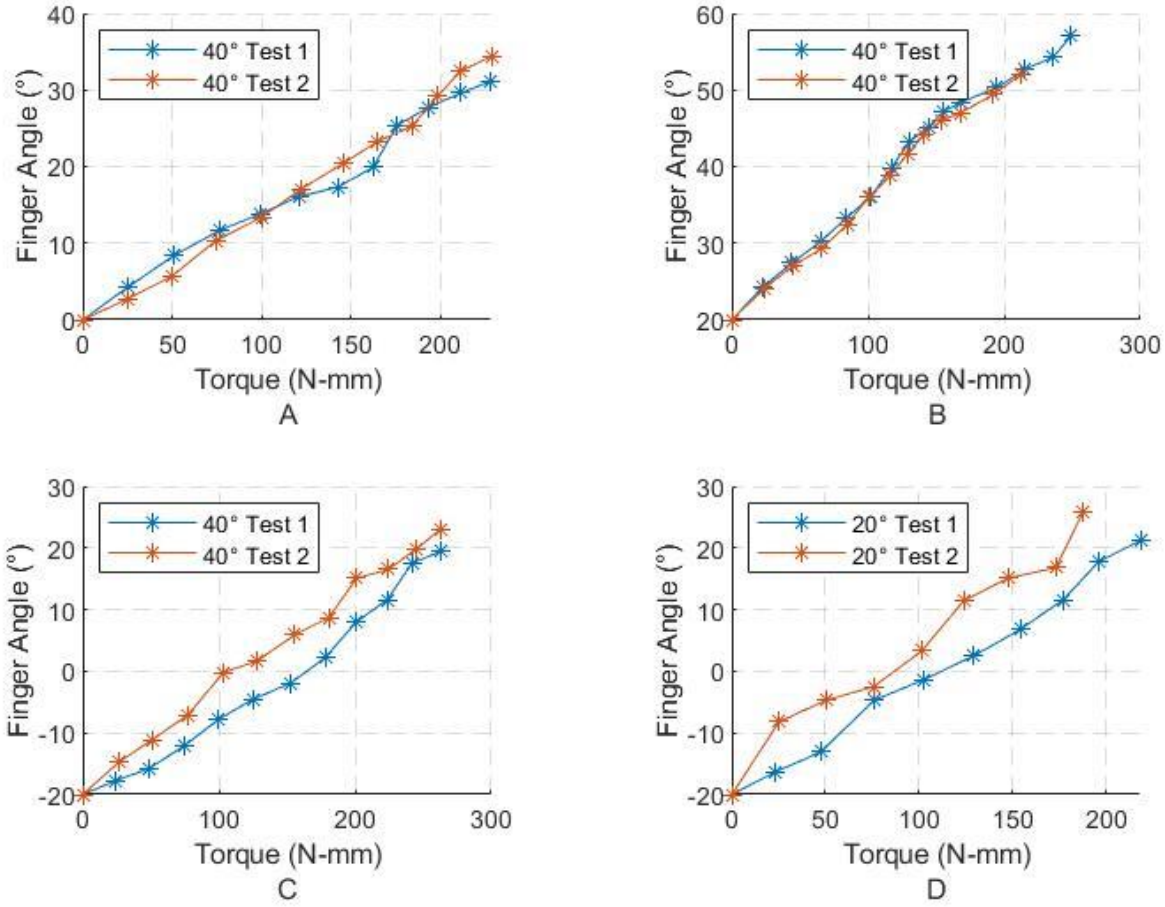


Figure 35: Slot Mechanism Results from (A, B, & C) 40°, (D) 20° Motor Angles

Additionally, as discussed in the single sided mechanism results, the Slot Mechanism had qualitative performance issues where one or both sides of the mechanism shaft would wedge itself in the slot of the mechanism and jam the mechanism. This jamming limited the ability to test high stiffness configurations of the mechanism because the mechanism would often jam in the slot and cut the testing short as accurate measurements of the joint torque required to attain a certain joint angle were extremely inaccurate.

The Lever Mechanism displayed significantly better and more consistent results. The testing data showed consistency between duplicate tests and excellent linearity for each individual antagonistic test.

The results, as seen in Figure 36, show the consistent linearity of the antagonistic Lever Mechanism design as well as a sampling of the stiffness range possible with such a mechanism. With the setup shown, the lowest stiffness tested was approximately 2.7 N/mm when the Lever Mechanism angles were at approximately 10° and 38 N/mm when the Lever Mechanism angles were at approximately 80°. The minimum stiffness of the mechanism is theoretically 0 N/mm when the Lever Mechanism angles at are 0°; however, this stiffness would only be valid for an infinitesimal deflection and 10° was the lowest angle tested that allowed for consistent joint angle deflection. Additionally, the maximum stiffness should approach the stiffness of the cable as the Lever Mechanism angle approached approximately 85°; however, as is apparent from Figure 36, the sensitivity of the mechanism stiffness increases as the mechanism stiffness increases and around approximately 80° the stiffness became too sensitive to consistently control and reliable data could not be collected.

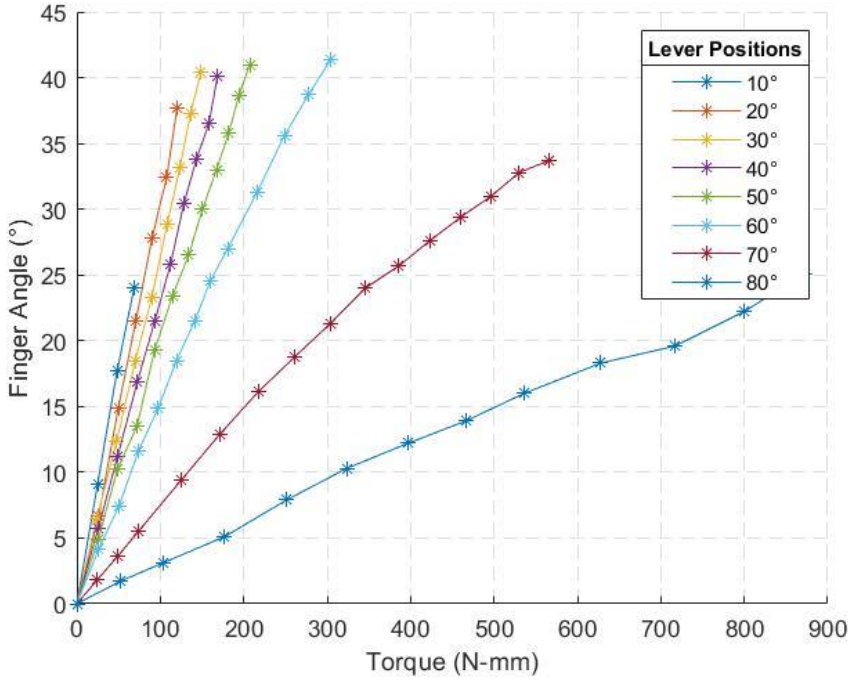


Figure 36: Lever Mechanism Antagonistic Results for 0° Joint Angle

The consistency of the measurements between duplicate tests is also excellent with the Lever Mechanism as seen in Figure 37 A, B, & C below with Figure 37 D showing the least consistent duplicate tests of the testing suite. Figure 37 A-C show results when the motor angles start at 50° from the zero-stiffness position while Figure 37 D shows a result while the motor angle starts at 10° from the zero stiffness position. The starting motor angle controls the stiffness of the mechanism with a higher angle resulting in a higher mechanism stiffness.

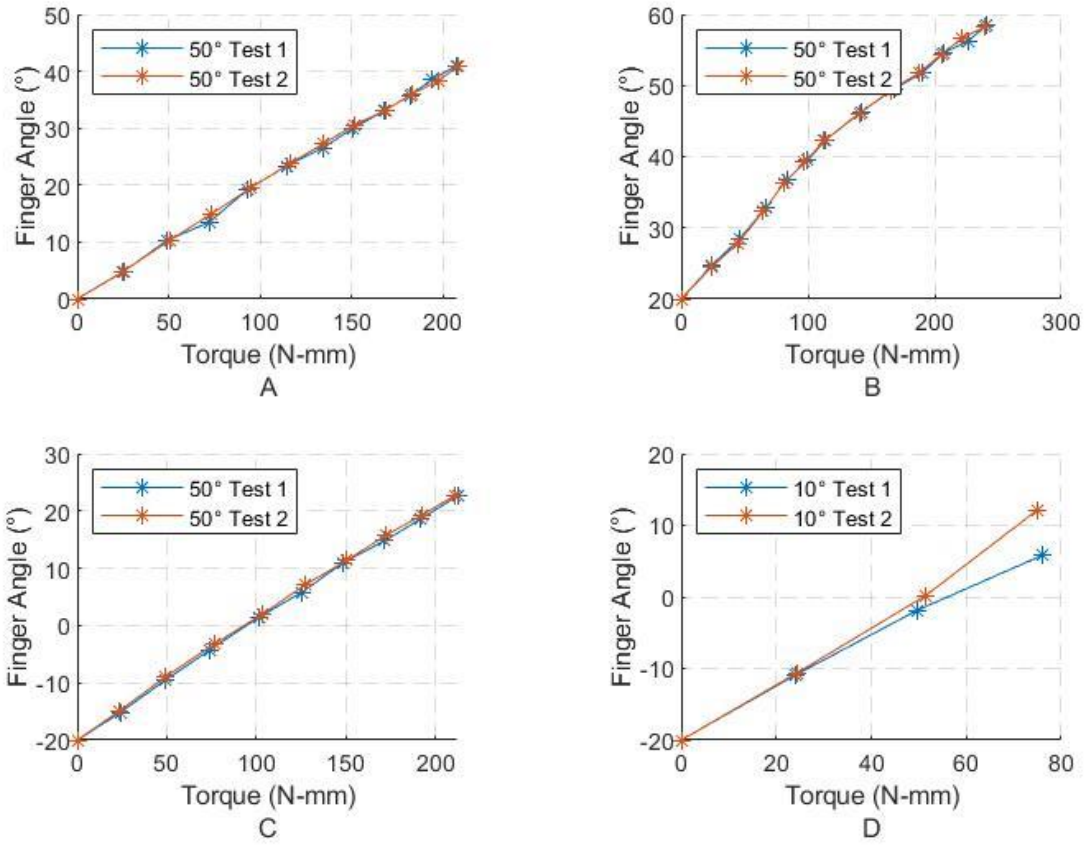


Figure 37: Lever Mechanism Results from (A, B, & C) 50°, (D) 10° Motor Angles

Most of the testing showed strong consistency and linearity as seen in Figure 37 with few anomalous results, usually at extremely low stiffnesses, likely due to errors in measurement equipment or possibly friction in the system.

# CHAPTER 5

# DISCUSSION & CONCLUSIONS

The objectives of this project were to design a variable stiffness actuator that has low inertia at the joint, is compact, and has a large range of controllable linear stiffness. The Lever Mechanism design presented in this paper achieves all three objectives.

## 5.1 Discussion of Results

The utilization of parametric optimizations to design each mechanism was critical in both understanding the effects of each parameter in the mechanism design as well as developing the specifications of the final prototype designs in order to achieve the desired high range of variable linear stiffness at the robotic joint. This approach along with the scalability of the mechanism in both physical size and stiffness ranges allows for this work to be used as a foundation for other unique applications and use-case scenarios.

While the Lever Mechanism was unable to exactly match the desired quadratic force-deflection curve in the mathematical simulations, the optimizations were able to match the desired paths closely enough for a high quality variable linear stiffness to be achieved at the joint. The less complex mechanism design of the lever system also allowed for less friction, more reliability, and more repeatability of the mechanism in testing.

Ultimately, the Slot Mechanism initially hypothesized to be the more effective mechanism fell short of achieving the desired linear stiffness at the joint or the necessary reliability due to its more complex design. The added degree of freedom for the path of the spring-roller mechanism did allow for theoretically better tracking of the desired

quadratic force-deflection path and may prove useful in other applications requiring more complex force-deflection paths that a Lever Mechanism would be unable to achieve. Serious reworking of the physical implementation, however, would be needed in order to reduce the friction of the Slot Mechanism and the unreliable nature of the slot shaft.

### 5.2 Work Contributions

The major success of this work is from the optimization and design of the Lever Mechanism as described in this thesis. Utilizing the optimization strategy laid out in Chapter 2, the Lever Mechanism was able to closely approximate the desired quadratic force-deflection characteristics for a limited range of joint deflection. This elastic behavior translated to controllable linear elastic behavior at the joint. Additionally, the mechanism design strategy is scalable to meet high or low stiffness needs by replacing the linear spring of the mechanism with a spring with a higher or lower spring constant to adjust for various applications.

The ability for the Lever Mechanism to match the desired behaviors so well and its inherently more reliable and less complex design mean that it is difficult to recommend the Slot Mechanism for an application requiring the specific elastic behavior laid out in this thesis even if the testing performance matched the Lever Mechanism as the manufacturing costs and reliability would almost definitely be lower. However, as stated before if a developer desired some more complex force-deflection behavior and was able to increase the reliability of the mechanism, the Slot Mechanism may be suitable for some scenarios.

## 5.3 Future Work

With the successful design and testing of the Lever Mechanism in a single joint complete. The next steps to take for additional work would be to extend this design to a multi-joint finger and eventually a multi-finger hand. A variety of work and possible modifications may need to be made in order to successfully create a multi-finger hand using this design.

For example, a three-finger robotic hand with each finger having three joints per hand would require nine antagonistic mechanisms and 18 motors to control. This may require a rework of the cable routing design in the finger mechanism in order to prevent joints from interacting with each other in ways that would affect the linear stiffness that these mechanisms create. Additionally, a more complex motor control strategy would be required to achieve smooth motion of the fingers. The work in this thesis only tested static loading scenarios therefore motor positions could be manually entered and checked prior to any loading, however, the dynamic loading and motion control that would be tested using a robotic hand would require more sophisticated programming.

Additionally, improvements to the current designs might include reducing the friction in the system to improve the match between the mathematics used in the design phase and the physical implementation. Alternatively, the mathematics could be extended to include friction and damping effects of the system that were not considered in this work.

# BIBLIOGRAPHY


[1] G. Bekey, R. Ambrose, V. Kumar, A. Sanderson, B. Wilcox and Y. Zheng, "International Assessment of Research and Development in Robotics," World Technology Evaluation Center, Inc, 2006.

[2] "A Roadmap for U.S. Robotics: From Internet to Robotics," National Science Foundation, 2016.

[3] P. C. Watson, "Remote Center Compliance System," *US4098001A*, 1978.

[4] Franka Emika, "Panda," [Online]. Available: https://www.franka.de/technology. [Accessed 2019].

[5] Universal Robots, "UR10," [Online]. Available: https://www.universal-robots.com/products/ur10-robot/. [Accessed 2019].

[6] KUKA, "LBR iiwa," [Online]. Available: https://www.kuka.com/en-us/products/robotics-systems/industrial-robots/lbr-iiwa. [Accessed 2019].

[7] S. Wolf, G. Grioli, O. Eiberger, W. Friedl, M. Grebenstein, H. Hoppner and E. Burdet, "Variable Stiffness Actuators: Review on Design and Components," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 5, pp. 2418-2430, 2016.

[8] Vanderbought *et al.*, "Variable Impedance Actuators: A Review," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1601-1614, 2013.

[9] S. Wolf, O. Eiberger and G. Hirzinger, "The DLR FSJ: Energy based design of a variable stiffness joint," in *IEEE International Conference on Robotics and Automation*, Shanghai, 2011.

[10] M. Grebenstein *et al.*, "The DLR Hand Arm System," in *IEEE International Confrence on Robotics and Automation*, Shanghai, 2011.

[11] M. Grebenstein, M. Chalon, W. Friedl, S. Haddadin, T. Wimbock, G. Hirzinger and R. Siegwart, "The hand of the DLR Hand Arm System: Designed for Interaction," *The International Journal of Robotics Research*, vol. 31, no. 13, pp. 1532-1555, 2012.

[12] W. Friedl, M. Chalon, J. Reinecke and M. Grebenstein, "FAS A Flexible Antaongistic Spring Element for a High Performance Over Actuated Hand," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, 2011.

[13] S. Migliore, E. Brown and S. DeWeerth, "Biologically Inspired Joint Stiffness Control," in *Internation Conference on Robotics and Automation*, Barcelona, 2005.

[14] K. F. Laurin-Kovitz, J. E. Colgate and S. D. R. Carnes, "Design of Components for Programmable Passive Impedance," in *International Conference on Robotics and Automation*, Sacramento, 1991.

# APPENDIX A

This appendix shows the MATLAB code used to create the Slot Mechanism optimization of the mechanism parameters. The main script (Mechanism Geometry Optimization) is the executed script to control the parameter bounds and the initial guesses of the optimization. The additional functions run inside the main loop at calculate the kinematics of the mechanism and to optimize the slot path for each iteration of mechanism parameters.

**Mechanism Geometry Optimization**
Author: Ryan Moore  Date: 6/12/18

```
% Description: Optimization of Quadratic Spring Mechanism Parameters to
% follow a desired quadratic force/displacement path of a spring pulley
% system. Optimizing for maximum displacement while still matching
% desired quadratic path.

% Inputs:   motorRadius - (scalar) Radius of fixed motor pulley

%           slotRadius - (scalar) Radius of moving slot pulley

%           guideRadius - (scalar) Radius of fixed guide pulley

%           R_i - (scalar) Initial distance from center of motor
%           pulley to center of moving slot pulley

%           springLength - (scalar) Distance from center of slot pulley
%           to spring termination point.

%           theta_i - (scalar) Initial angle between the vector from origin to the center
of the slot pulley and x-axis

%           springConstant - (scalar) Spring constant of linear spring

%           Pretension - (scalar) Pretension distance of the spring
%           (springLength-SpringFreeLength)=Pretension

%           cableConstant - (scalar) Spring constant of the cable

% Outputs:  cableTravelLength - (scalar) Change in cable length of the system from
initial configuation to current

%           rArray - (Vector) sequence of polar R coordinates of optimal slot pulley path
```

```
%           thetaArray - (Vector) sequence of polar theta coordinates of optimal slot
pulley path

%           tensionArray - (Vector) sequence of Cable Tension Values along slot pulley
path

%           lengthArray - (Vector) sequence of cable Lengths along slot pulley path

%           initialTension - (scalar) Tension magnitude at inital configuration

%           initialCableLength - (scalar) Length of cable in mechanism from its initial
geometry

%           errorArray - (Vector) sequence of deviation values from desired along slot
path

%           errorArray = (actual-desired tension)^2 + (actual-desired length)^2

clear all
close all
clc
```

*Initialization of Parameters*

```
motorRadius = 0.20;

slotRadius = 0.10;

guideRadius = 0.10;

R_i = 0.50;

springLength = 0.75;

theta_i = 68.5;

springConstant = 4;

Pretension = 0;

cableConstant = 45000;

x0 = [motorRadius; slotRadius; guideRadius; R_i; springLength; theta_i; springConstant;
Pretension; cableConstant];
```

*Optimization Constraints*

A & b matrices prevent various inadmissible scenarios from happening

```
% 1st: motorRadius + slotRadius - R_i < -0.1 This prevents the motor and slot pulleys
% from contacting.

% 2nd: slotRadius + guideRadius +R_i*cosd(theta_i) < 0.95 This prevents the slot and
fixed guide
% pulleys from contacting.

% 3rd: 10*springConstant - cableConstant < 0 This ensure the cable stiffness is at least
10 times
% as stiff as the spring stiffness.

A = [1 1 0 -1 0 0 0 0 0; 0 1 1 cosd(theta_i) 0 0 0 0 0;...      % Coefficients for
inequality constraints (A*x0<b)
    0 0 0 0 0 0 10 0 -1];

b = [-0.1; 0.95; 0];                                            % Right side of
inequality constraints (A*x0<b)

lowerBounds = [0.1 0.1 0.1 0.1 0.10 45 0.01 0.001 1]';          % Lower bounds of
parameters

upperBounds = [0.40 0.40 0.40 0.9 1.5 120 100 0.1 100000]';     % Upper bounds of
parameters
```

*Single Point Start*

This runs a single point gradient search using the @FixedGeometryObjective function file as an objective function.

```
 options = optimset('PlotFcn',@optimplotfval,'MaxFunEvals',200000 );                % 
Options for single start optimization
[x,fval] = 
fmincon(@FixedGeometryObjective,x0,A,b,[],[],lowerBounds,upperBounds,[],options);
```

**FixedGeometryObjective.m**

```
% Description: Calls SlotPathGeneration.m function to generate the
% optimal slot path given the initial parameter estimates, x0. Optimal slot path
% tracks the desired quadratic force-deflection curve for as long of a
% distance as possible. This function is called by MechanismGeometryOptimization.m

%   Inputs: x0 - (vector) Initial Parameter estimates from
MechanismGeometryOptimization.m

%   Outputs: fval - (scalar) negative of the change in cable length of the system.

function fval = FixedGeometryObjective(x0)

[cableTravelLength,rArray,thetaArray,tensionArray,lengthArray] = SlotPathGeneration(x0);
```

```
fval = -cableTravelLength                    % Objective function reverses to a
maximization problem
```

**SlotPathGeneration.m**

```
function
[cableTravelLength,rArray,thetaArray,tensionArray,lengthArray,initialTension,initialCable
Length,errorArray] = SlotPathGeneration(x0)
```

---

```
% Description: SlotPathGeneration calculates the inital mechanism configuration, initial
tension, and inital cable length
% Then optimizes the slot pulley path by incrementing the desired cable length and
desired cable tension and using
% StaticEquilibriumPointOptimization.m to find the next acceptable location along the
slot pulley path.

% Inputs:   x0 - (vector) Initial parameter values from MechanismGeometryOptimization.m

% Outputs:  cableTravelLength - (scalar) Change in cable length of the system from
initial configuration

%           rArray - (Vector) sequence of polar radial coordinates of optimal slot pulley
path

%           thetaArray - (Vector) sequence of polar theta coordinates of optimal slot
pulley  path

%           tensionArray - (Vector) sequence of cable tension values along slot pulley
path

%           lengthArray - (Vector) sequence of cable lengths along slot pulley path

%           initialTension - (scalar) Tension magnitude at inital configuration

%           initialCableLength - (scalar) Length of cable in mechanism from its initial
geometry

%           errorArray - (Vector) sequence of values of deviation from desired along slot
path

%           errorArray = (actual-desired tension)^2 + (actual-desired length)^2
```

*Initialization*

```
motorRadius = x0(1);    % (scalar) Radius of fixed motor pulley

slotRadius = x0(2);     % (scalar) Radius of moving slot pulley

guideRadius = x0(3);    % (scalar) Radius of fixed guide pulley
```

```
R_i = x0(4);            % (scalar) Distance from center of motor pulley to center of
moving slot pulley

springLength = x0(5);   % (scalar) Distance from center of slot pulley to spring
termination point.

theta_i = x0(6);        % (scalar) Initial angle between the vector from origin to the
center of the slot pulley and x-axis

springConstant = x0(7); % (scalar) Spring constant of linear spring

Pretension = x0(8);     % (scalar) Pretension distance of the spring (springLength-
SpringFreeLength)=Pretension

cableConstant = x0(9);  % (scalar) Spring constant of the cable
```

*Find Initial Configuration*

```
% This section calculates the initial configuration of the mechanism in its
% zero stiffness orientation.

motorCenter = [0 0 0]';

motorBottom = [0 -motorRadius 0];

guideCenter = [1 -(motorRadius-guideRadius) 0]';

guideBottom = [1 guideCenter(2)-guideRadius 0];

slotCenter = [cosd(theta_i)*R_i sind(theta_i)*R_i 0]';

[xMotorTangent, yMotorTangent, xSlotTangentL, ySlotTangentL, tangentMotorToSlot,
motorToSlotAlpha] =... % see crosstan.m
    crosstan(motorCenter,motorRadius,slotRadius,slotCenter,1);

tangentMotorToSlot=-tangentMotorToSlot;

cableLengthMotorToSlot = norm(tangentMotorToSlot);

springCenter = [slotCenter(1)+(Pretension+springLength)*cosd(theta_i)...
    slotCenter(2)+sind(theta_i)*(springLength+Pretension) 0]';

[xSlotTangentR, ySlotTangentR, xGuideTangent, yGuideTangent, tangentSlotToGuide,
slotToGuideAlpha] =... % see crosstan.m
    crosstan(slotCenter,slotRadius,guideRadius,guideCenter-slotCenter,0);

cableLengthSlotToGuide = norm(tangentSlotToGuide);

slotArcangle = motorToSlotAlpha-slotToGuideAlpha+180;

slotCableArclength = 2*pi*slotRadius*(slotArcangle/360);
```

```
motorArcangle = motorToSlotAlpha+90;

motorCableArclength = 2*pi*motorRadius*(motorArcangle/360);

guideArcangle = 90-slotToGuideAlpha;

guideCableArclength = 2*pi*guideRadius*(guideArcangle/360);

initialCableLength =
motorCableArclength+cableLengthMotorToSlot+slotCableArclength+cableLengthSlotToGuide+guid
eCableArclength;

unitTangentMotorToSlot = tangentMotorToSlot/norm(tangentMotorToSlot);

unitTangentSlotToGuide = tangentSlotToGuide/norm(tangentSlotToGuide);

springVector = springCenter-slotCenter;

unitSpringVector = springVector/norm(springVector);

slotReactionForce = motorCenter-slotCenter;

unitSlotReactionForce = slotReactionForce/norm(slotReactionForce);

springForce = springConstant*(norm(springVector)-springLength);


forceMatrix = [(unitTangentMotorToSlot(1)+unitTangentSlotToGuide(1))
unitSlotReactionForce(1); (unitTangentMotorToSlot(2)+unitTangentSlotToGuide(2))
unitSlotReactionForce(2)];

[forceMagnitudes] = forceMatrix\[(-springForce*unitSpringVector(1)); (-
springForce*unitSpringVector(2))];

initialTension = forceMagnitudes(1);
```

*Path Finding Optimization*

```
% This section runs a optimization to determine the best slot path to match
% the desired quadratic tension-deflection curve for the longest
% possible length.

% The desired length is reduced by an increment of
% "stepsize" then the function StaticEquilibriumPointOptimization.m attempts to find the
% optimal new Ll and theta to achieve the new desired length and tension.

% This while loop will continue to run until no further steps can be taken
% and the stepsize is sufficiently small.

thetaArray=[];tensionArray=[];lengthArray=[];errorArray=[];rArray=[];

desiredTensionArray = []; desiredLengthArray = []; stepsizeArray=[];
```

```
theta = theta_i;

R = R_i;

stepsize = 0.001;

desiredLength = initialCableLength+stepsize;

while stepsize > 1e-10

    desiredLength=desiredLength-stepsize;

    desiredTension = 1*(initialCableLength-desiredLength)^2;

        try

        previousTheta = thetaArray(end);

        previousR = rArray(end);

        catch

        previousTheta = theta_i;

        previousR = R_i;

        end

    [xMotorTangent, fval, Tension, Length, flag] = StaticEquilibriumPointOptimization...
        (x0,desiredTension,desiredLength,theta,R,previousTheta,previousR);

    if flag == 1

        tensionArray = [tensionArray Tension];

        lengthArray = [lengthArray Length];

        errorArray = [errorArray double(fval)];

        R = xMotorTangent(4);

        rArray = [rArray R];

        theta = xMotorTangent(10);

        desiredTensionArray = [desiredTensionArray desiredTension];

        desiredLengthArray = [desiredLengthArray desiredLength];

        thetaArray = [thetaArray theta];

        desiredLength = Length;

        stepsizeArray = [stepsizeArray stepsize];
```

```
    else

        desiredLength = desiredLength+stepsize;

        stepsize = stepsize/2;

    end
end

cableTravelLength = initialCableLength-desiredLength
```

---

```
end
```

### Crosstan.m

```
% Description: Calculates the crossed tangent points between two circles.
% This is used to find the tangent points of the cable and the pulleys as
% well as the force vector of the cable tensions. This function is called in
% SlotPathGeneration.m and StaticAnalysis.m.

% Inputs:   C1 - (Vector) of center point of the first circle (pulley)

%           r1 - (scalar) radius of first circle (pulley)

%           r2 - (scalar) radius of second circle (pulley)

%           L  - (vector) of length between the two circles

%           tb - (scalar) "top or bottom" crossed tangent. 1 is used from the cable
%           between the motor and slot pulley and 0 is used from the cable
%           between the slot and guide pulley.

% Outputs:  x1 - (scalar) x location of the tangent point on first circle

%           y1 - (scalar) y location of the tangent point on first circle

%           x2 - (scalar)x location of the tangent point on second circle

%           y2 - (scalar) y location of the tangent point on second circle

%           T  - (Vector) of crossed tangent from first circle to second
%                circle from (x1,y1) to (x2,y2)

function [x1 y1 x2 y2 T alpha angle] = crosstan(C1,r1,r2,L,tb)
```

*cross tangents to two circles*

```
x = [1 0 0]';                       %  Establishes global x direction
C2 = L+C1;                          %  Center of second circle
```

```
d2 = L/(1+r1/r2);                       %  Distance between intersection point of tangent
line and x axis to the center of...
                                        %  the second circle
d1 = L-d2;                              %  Distance between the center of the first circle
to the intersection point of...
                                        %  tangent line and x axis.
cros = cross(x,L);
angle = atan2d(cros(3), dot(x,L));      %  Angle between x axis and actual vector from one
circle center to the other
alpha = acosd((r1+r2)/norm(L));         %  Angle between vector from one circle center to
the other and tangent point...
                                        %  of cable on circle.
if tb > 0
    alpha = -alpha;                     %  Flips angle if using the other crossed tangent
line
end
alpha = alpha+angle;                    %  adds angles together to achieve true angle of
vector from global x-axis
x1=r1*cosd(alpha)+C1(1);                %  x location of tangent point on first circle
y1=r1*sind(alpha)+C1(2);                %  y location of tangent point on first circle
beta=180+alpha;                         %  rotates alpha by 180 degrees to calculate other
circle points
x2=L(1)+r2*cosd(beta)+C1(1);            %  x location of tangent point on second circle
y2=L(2)+r2*sind(beta)+C1(2);            %  y location of tangent point on second circle
T = [x2-x1; y2-y1; 0];                  %  Vector of crossed tangent from first circle to
second circle.
```

### StaticEquilibriumPointOptimization.m

```
function [x1, fval, cableTension, cableLength, flag] =
StaticEquilibriumPointOptimization(x0,desiredTension,desiredLength,theta,R,previousTheta,
previousR)
```

```
%   Description: This function sets up the opimization for evaluting the next point in
%   the moving pulley path. The gradient search optimization varies the parameters Ll
%   (the distance between the motor pulley center and slot pulley center) and theta
(angle between the
%   x axis and the line connecting the motor and slot pulley centers) in order to match
the next step
%   of the desired force-deflection curve of the mechanism. This function is called in
SlotPathGeneration.m

% Inputs:   x0 - (vector) of Initial parameter values from
MechanismGeometryOptimization.m

%           desiredTension - (scalar) Desired cable tension value for current
configuration

%           desiredLength - (scalar) Desired cable length in mechanism for current
configuration

%           theta - (scalar) Current angle between global x-axis and the vector from the
```

```
center of motor pulley through the
%           center of slot pulley.

%           R - (scalar) Current distance between center of motor pulley and center of
slot pulley

%           previousTheta - (scalar) Previous angle between global x-axis and the vector
from the center of the motor pulley
%           through the center of the slot pulley. (this is used to estimate the vector
normal to the slot at its
%           current position.)

%           previousR - (scalar) Previous distance between cneter of motor pulley and
center of slot pulley. (this is used
%           to estimate the vector normal to the slot at its current position.

% Outputs:  x1 - (vector) of mechanism parameters output StaticEquilibriumObjective.m

%           fval - (scalar) objective function value of StaticEquilibriumObjective.m
related to the error between the desired
%           and calculated tension and deflection of the mechanism.

%           Tension - (scalar) cable tension value at the current configuration

%           Length - (scalar) cable length value at the current configuration

%           flag - (scalar) flag indicating whether the optimization output is good
enough to be included.
```

*Initialization*

```
motorRadius = x0(1);                                % radius of motor pulley
slotRadius = x0(2);                                % radius of slot pulley
guideRadius = x0(3);                                % radius of fixed guide pulley
R_i = x0(4);                                % Distance between motor pulley center and slot
pulley center
springLength = x0(5);                                   % initial spring length
theta_i = x0(6)-3;                           % initial polar slot pulley angle
springConstant = x0(7);                                   % spring constant of spring
Pretension = x0(8);                          % Pretension of spring
cableConstant = x0(9);                                   % Spring constant of cable
x0 = [motorRadius; slotRadius; guideRadius; R; springLength; theta_i; springConstant;...
% Vector of inital parameters
    Pretension; cableConstant; theta; desiredTension;...
    desiredLength; previousTheta; previousR; R_i];

% A & b matrices prevent various inadmissable scenarios from happening

% 1st: Rmp + Rsp - Ll_i < -0.1 This prevents the motor and slot pulleys
% from contacting.

% 2nd: Rsp + Rfp +Ll_i*cosd(theta) < 0.95 This prevents the slot and fixed guide
```

```
% pulleys from contacting.

% 3rd: 10*Ks - Kc < 0 This ensure the cable stiffness is at least 10 times
% as stiff as the spring stiffness.

A = [1 1 0 -1 0 0 0 0 0 0 0 0 0 0 0;...
    0 1 1 cosd(theta) 0 0 0 0 0 0 0 0 0 0 0;...
    0 0 0 0 0 0 10 0 -1 0 0 0 0 0 0];          % Coefficients for inequality constraints
b = [-0.1; 0.95; 0];                           % Right side of inequality constraints
(A*x>b)

% Note: only the Ll and theta can change making this a 2 variable
% optimization.

lowerBounds = [motorRadius slotRadius guideRadius R-0.02 springLength theta_i
springConstant...      % Lower Bounds of Parameters (only slot pulley location (R &
theta) can change)
    Pretension cableConstant -50 desiredTension...
    desiredLength previousTheta previousR R_i]';
upperBounds = [motorRadius slotRadius guideRadius R+0.02 springLength theta_i
springConstant...      % Upper bounds of parameters (only slot pulley location (R &
theta) can change)
    Pretension cableConstant theta  desiredTension...
    desiredLength previousTheta previousR R_i]';
```

*Single Start fmincon*

```
options = optimset('MaxFunEvals',200000,...
    'TolFun',1e-10,'TolX',1e-10,'Display','off');
[x1,fval] =
fmincon(@StaticEquilibriumObjective,x0,A,b,[],[],lowerBounds,upperBounds,[],options); %
Optimization of the path objective to...
                                                                                % achieve
desired slot path
[cableTension, cableLength, error] = StaticAnalysis(x1);
checkerror = abs(fval-error);                                                   %
checkerror is for debugging purposes.
if checkerror > 0.01
   checkerror
end
if fval > 0.0001                                                        % If the
objective function is bad
    flag = 0;                                                           % Flag it as bad
else                                                                    % else
    flag = 1;                                                           % flag it as good
end

end
```

**StaticEquilibriumObjective.m**

Description: This objective function pulls the error value of Pathsim.m as the objective function of the path finding tool. This function is called in StaticEquilibriumPointOptimization.m

```
% Inputs: x0 - (vector) Current mechanism parameters

% Outputs: e - (scalar) squared error of actual vs desired tension and length.

function e = StaticEquilibriumObjective(x0)

[cableTension, cableLength, e] = StaticAnalysis(x0);
```

**StaticAnalysis.m**

```
%  Description: This code evaluates the pulley mechanism at a given configuration and
%  outputs the tension and length values as well as the error compared to
%  the desired values from SlotPathGeneration.m.
%  This function is called in StaticEquilibriumPointOptimization.m and
StaticEquilibriumObjective.m

%   Inputs: x0 - (Vector) of 15 parameter values of mechanism. See
%   Initialization section below

%   Outputs: cableTension - (Scalar) Tension value of cable in current configuration

%           cableLength - (Scalar) Length value of cable in mechanism in current
%           configuration

%           error - (scalar) squared difference between actual and desired length
%           and tension.

function [cableTension, cableLength, error] = StaticAnalysis(x0)
```

*Initialization*

```
motorRadius = x0(1);            % Radius of motor pulley
slotRadius = x0(2);            % Radius of slot pulley
guideRadius = x0(3);            % Radius of fixed guide pulley
R = x0(4);             % Current distance between center of motor pulley and slot pulley
springLength = x0(5);             % Spring Length: Distance from center of slot pulley to
spring termination point.
theta_i = x0(6);        % Initial angle between global x-axis to vector from center of
motor pulley through center of slot...
                        % pulley to the spring
```

```
springConstant = x0(7);             % Spring Constant of Linear Spring
Pretension = x0(8);     % Pretension distance of the spring
cableConstant = x0(9);             % Stiffness Constant of the cable
theta = x0(10);         % Current angle between global x-axis to vector from center of
motor pulley through center of slot...
                        % pulley to the spring
desiredTension = x0(11);     % Desired cable Tension value from SlotPathGeneration.m
desiredLength = x0(12);     % Desired cable Length value from SlotPathGeneration.m
previousTheta = x0(13);    % Previous theta value needed to calculate normal vector
previousR = x0(14);       % Previous distance between center of motor pulley and slot
pulley to calculate normal vector
R_i = x0(15);          % Initial distance between center of motor pulley and slot pulley
```

*Find Initial Orientation*

```
% This section finds the inital orientation of the mechanism.

try

motorCenter = [0 0 0]';                                          % Center point of
motor pulley
slotCenter = [cosd(theta_i)*R_i sind(theta_i)*R_i 0]';           % Initial position of
center point ofslot pulley
guideCenter = [1 -(motorRadius-guideRadius) 0]';                                  %
Center point of fixed guide pulley
springCenter = [slotCenter(1)+(Pretension+springLength)*cosd(theta_i)...              %
Position of fixed end of spring
    slotCenter(2)+sind(theta_i)*(Pretension+springLength) 0]';
```

*Evaluation of Cable Tension*

```
% This section uses the current mechanism information to evaluate the
% current cable tension and the current length of the cable in the
% mechanism.

slotCenter = [cosd(theta)*(R) sind(theta)*(R) 0]';           % Position of center point
of slot pulley
[xMotorTangent, yMotorTangent, xSlotTangentL, ySlotTangentL, tangentMotorToSlot] =
crosstan(motorCenter,motorRadius,slotRadius,slotCenter,1);     % Calculates the contact
points (x1,y1) & (x2,y2) as well as the...
                                                    % vector between them
realCableCheck = isreal(tangentMotorToSlot);                                  %
Ensures that the cable tension is real
if realCableCheck == 1                                          % If it's real it will
continue as normal
else                                                % If it's not real,
    tangentMotorToSlot                                              % Output the
cable tension for analysis
    tangentMotorToSlot = real(tangentMotorToSlot);                                %
Take only the real part (this shouldn't be an issue anymore)
end
xMotorBottom = 0; yMotorBottom = -motorRadius;
```

```
xGuideBottom = 1; yGuideBottom = guideCenter(2)-guideRadius;             % Specifies the
locations for the bottom of the motor pulley & bottom of guide pulley from centers
cableLengthMotorToSlot = norm(tangentMotorToSlot);
% Calculates the length of cable between the fixed pulley and...
                                                        % moving pulley
tangentMotorToSlot=-tangentMotorToSlot;                                          %
Flips direction of cable
[xSlotTangentR, ySlotTangentR, xGuideTangent, yGuideTangent, tangentSlotToGuide] =
crosstan(slotCenter,slotRadius,guideRadius,guideCenter-slotCenter,0); % Calculates the
contact points (x1,y1) & (x2,y2) as well as the...
                                                        % vector between them
cableLengthSlotToGuide = norm(tangentSlotToGuide);
% Calculates the length of cable between the moving pulley and...
                                                        % the guide pulley
slotRayLeft = [xSlotTangentL-slotCenter(1) ySlotTangentL-slotCenter(2) 0];
% cable Vector between Contact point of motor pulley and slot pulley
slotRayRight = [xSlotTangentR-slotCenter(1) ySlotTangentR-slotCenter(2) 0];
% cable Vector between contact point of slot pulley and guide pulley
slotAngleRight = atan2d(slotRayRight(2),slotRayRight(1));                        %
Angle from horizontal to the right contact point on the slot pulley
slotAngleLeft = atan2d(slotRayLeft(2),slotRayLeft(1));                          % Angle
from horizontal to the left contact point on the slot pulley
if slotAngleLeft<0
    slotAngleLeft = slotAngleLeft+360;
end
slotArcangle = slotAngleLeft-slotAngleRight;                                % Arcangle of
contact on the slot pulley
slotCableArcLength = 2*pi*slotRadius*(slotArcangle/360);                     % Converting
arcangle into a cable length
motorRayLeft = [motorCenter(1)-xMotorBottom motorCenter(2)-yMotorBottom 0];
% Calculates right contact point of fixed pulley
motorRayRight = [motorCenter(1)-xMotorTangent motorCenter(2)-yMotorTangent 0];
% Calculates left contact point of fixed pulley
motorArcangle = atan2d(norm(cross(motorRayLeft,motorRayRight)),
dot(motorRayLeft,motorRayRight));         % Calculates the arc angle between two contact
points
motorCableArcLength = 2*pi*motorRadius*(motorArcangle/360);                      %
Calculates the arc length of contact on fixed pulley
guideRayRight = [guideCenter(1)-xGuideBottom guideCenter(2)-yGuideBottom 0];
% Calculates right contact point of fixed pulley
guideRayLeft = [guideCenter(1)-xGuideTangent guideCenter(2)-yGuideTangent 0];
% Calculates left contact point of fixed pulley
guideArcangle = atan2d(norm(cross(guideRayRight,guideRayLeft)),
dot(guideRayRight,guideRayLeft));         % Calculates the arc angle between two contact
points
guideCableArcLength = 2*pi*guideRadius*(guideArcangle/360);                      %
Calculates the arc length between two contact points
cableLength =
(cableLengthMotorToSlot+slotCableArcLength+cableLengthSlotToGuide+motorCableArcLength+gui
deCableArcLength);                    % Calculates full length of cable in mechanism
unitTangentMotorToSlot = tangentMotorToSlot/norm(tangentMotorToSlot);
% Unit vector of left cable direction
unitTangentSlotToGuide = tangentSlotToGuide/norm(tangentSlotToGuide);
% Unit vector of right cable direction
```

```
springVector = springCenter-slotCenter;                                                    %
Calculates spring vector
unitSpringVector = springVector/norm(springVector);
% Unit vector of spring vector
[unitSlotReactionForce] = NormalVector(previousR,previousTheta,slotCenter);           %
Normal unit vector of moving pulley against path
springForce = springConstant*(norm(springVector)-springLength);
% Calculates spring force
forceMatrix = [(unitTangentMotorToSlot(1)+unitTangentSlotToGuide(1))
unitSlotReactionForce(1); (unitTangentMotorToSlot(2)+unitTangentSlotToGuide(2))
unitSlotReactionForce(2)];   % Matrix to calculate magnitude of tension and normal forces
[forceMagnitudes] = forceMatrix\[(-springForce*unitSpringVector(1)); (-
springForce*unitSpringVector(2))];       % Calculation of magnitudes
cableTension = forceMagnitudes(1);                                                          %
Magnitude of tension
cableLength = cableLength-(cableTension/cableConstant);
% Accounts for the change in length of the cable
tensionError=cableTension-desiredTension;                                        % Calculates
the difference between the tension calculated and...
                                                          % tension desired
lengthError=cableLength-desiredLength;                                    % Calculates
the difference between the length calculated and...
                                                          % length desired
error = tensionError^2+lengthError^2;                            % Calculates the error
used as an objective function

catch                                                     % If something bad happens
    error = NaN;                                          % Outputs become NaN and lets the
program move onto the next attempt
    cableTension = NaN;                                             % Instead of
terminating
    cableLength = NaN;                                             % Instead of
terminating
end
```

## NormalVector.m

```
% Description: Calculated Normal Vector for the normal force of the slot
% pulley against the path wall. This function is called in StaticAnalysis.m

% Inputs:   R - (scalar) Current Distance between center of motor pulley and center
%               of slot pulley.

%           theta - (scalar) angle between global x-axis to vector from center of
%                 motor pulley through center of slot pulley to the spring.

%           slotCenter - (Vector) location of the center of the slot pulley

% Outputs:  unitNormal - (Unit vector) of the normal force of the slot pulley acting
%                 against the path wall.
```

```
function [unitNormal] = NormalVector(R,theta,slotCenter)

Last_Point = [R*(cosd(theta)); R*(sind(theta)); 0]; % vector coordinate of last moving
pulley locaion
PathVec = slotCenter-Last_Point;                    % vector from last point to current
point of moving pulley
RotationMatrix = [cosd(-90) -sind(-90) 0;...        % -90 degree rotation matrix
    sind(-90) cosd(-90) 0; 0 0 1];
NormalVector = RotationMatrix*PathVec;                % normal force vector of moving
pulley against path wall
unitNormal = NormalVector/norm(NormalVector);            % normal unit force vector
end
```

# APPENDIX B

This appendix shows the MATLAB code used to create the Lever Mechanism optimization of the mechanism parameters. The main script (Lever Mechanism Optimization) is the executed script to control the parameter bounds and the initial guesses of the optimization. The additional functions run inside the main loop at calculate the kinematics of the mechanism for each optimization iteration.

**LeverMechanismOptimization.m**

```
% Author: Ryan Moore
% Description:  Optimization of Lever Mechanism Quadratic Spring Mechanism Parameters to
% follow a desired quadratic force/displacement path of a spring pulley
% system. Optimizing for maximum displacement that falls within specified
% error bounds using fmincon optimization method

% Inputs:   motorRadius - (scalar) Radius of fixed motor pulley

%           springRadius - (scalar) Radius of moving spring pulley

%           guideRadius - (scalar) Radius of fixed guide pulley

%           R_i - (scalar) Initial distance from center of motor
%           pulley to center of moving slot pulley

%           springLength - (scalar) Distance from center of slot pulley
%           to spring termination point.

%           theta_i - (scalar) Initial angle between the vector from origin to the center
of the spring pulley and x-axis

%           springConstant - (scalar) Spring constant of linear spring

%           Pretension - (scalar) Pretension distance of the spring
%           (springLength-SpringFreeLength)=Pretension

%           cableConstant - (scalar) Spring constant of the cable

% Outputs:  cableTravelLength - (scalar) Change in cable length of the system from
initial configuation to current

%           tensionArray - (Vector) sequence of Cable Tension Values along slot pulley
path

%           lengthArray - (Vector) sequence of cable Lengths along slot pulley path
```

```
clear all
close all
clc
```

*Initialization of Parameters*

```
    motorRadius = 0.23944; % radius of fixed lever pulley
    springRadius = 0.1498; % radius of moving lever pulley
    guideRadius = 0.232; % radius of fixed guide pulley
    R_i = 0.4958; % lever length
    springLength = 0.57977; % initial spring length
    theta_i =64.663; %initial lever position
    springConstant = 3.7411;
    Pretension = 0.0; %Pretension of spring
    cableConstant = 485.67;
x0 = [motorRadius; springRadius; guideRadius; R_i; springLength; theta_i; springConstant;
Pretension; cableConstant];
```

*Optimization Constraints*

A & b matrices prevent various inadmissible scenarios from happening

```
% 1st: motorRadius + springRadius - R_i < -0.1 This prevents the motor and slot pulleys
% from contacting.

% 2nd: springRadius + guideRadius +R_i*cosd(theta_i) < 1 This prevents the slot and fixed
guide
% pulleys from contacting.

% 3rd: 10*springConstant - cableConstant < 0 This ensure the cable stiffness is at least
10 times
% as stiff as the spring stiffness.

A = [1 1 0 -1 0 0 0 0 0; 0 1 1 1 0 0 0 0 0; 0 0 0 0 0 0 100 0 -1];
b = [-0.1; 1; 0];
Aeq = [];
beq = [];
lowerBounds = [motorRadius 0.05 guideRadius 0.2 0.2 30  0.01 0 10]';
upperBounds = [motorRadius  0.15  guideRadius  0.8 1   120 100  0 10000]';
```

*Single Point Start*

This runs a single point gradient search using the @LeverObjective function file as an objective function.

```
options = optimset('PlotFcn',@optimplotfval,'MaxFunEvals',2000);
x = fmincon(@LeverObjective,x0,A,b,Aeq,beq,lowerBounds,upperBounds,[],options)
```

*Output Results*

This portion reruns the results of the optimization in order to obtain the results in the workspace.

```
[tensionArray, cableTravelLength, lengthArray] = LeverStaticAnalysis(x);


close all

plot(lengthArray,tensionArray,lengthArray,lengthArray.^2)
title('Tension vs Deflection')
xlabel('\DeltaL')
ylabel('Tension')
legend('Calculated','Desired','Location','Northwest')
axis([0 1 0 1])

dT = diff(tensionArray);
ddL = diff(lengthArray);
dTdL = dT./ddL;
figure(2)
plot(lengthArray(1:end-1),dTdL,[0 1 2],[0 2 4])
axis([0 1 0 2])
title('Stiffness vs Deflection')
xlabel('\DeltaL')
ylabel('Stiffness')
legend('Calculated','Desired','Location','Northwest')
figure(1)
cableTravelLength
```

**LeverObjective.m**

Description: Relays the objective function from the main loop LeverMechanismOptimization.m to the internal kinematics script (LeverStaticAnalysis.m)

```
function e = LeverObjective(x0)

[magTs, e, xT] = LeverStaticAnalysis(x0);
```

**LeverStaticAnalysis.m**

```
%  Description:

%   Inputs: x0 - (Vector) of 9 parameter values of mechanism. See
%   Initialization section below

% Outputs:  cableTravelLength - (scalar) Change in cable length of the system from
initial configuation to current
```

```
%           tensionArray - (Vector) sequence of Cable Tension Values along slot pulley
path

%           lengthArray - (Vector) sequence of cable Lengths along slot pulley path

function [tensionArray, cableTravelLength, lengthArray] = LeverStaticAnalysis(x0)
```

```
xVector = [1 0 0]';
yVector = [0 1 0]';
```

*Initialization*

```
motorRadius = x0(1);   % Radius of motor pulley
springRadius = x0(2);  % Radius of spring pulley
guideRadius = x0(3);   % Radius of fixed guide pulley
R_i = x0(4);             % Current distance between center of motor pulley and slot
pulley
springLength = x0(5);  % Spring Length: Distance from center of slot pulley to spring
termination point.
theta_i = x0(6);       % Initial angle between global x-axis to vector from center of
motor pulley through center of slot...
                       % pulley to the spring
springConstant = x0(7);% Spring Constant of Linear Spring
Pretension = x0(8);    % Pretension distance of the spring
cableConstant = x0(9); % Stiffness Constant of the cable
```

*Finding Initial Orientation*

```
xMotorBottom = 0; yMotorBottom = -motorRadius; xGuideBottom = 1;
motorCenter = [0 0 0]'; % center point of motor pulley
springPulleyCenter = [cosd(theta_i)*R_i sind(theta_i)*R_i 0]'; % initial position of
center point of spring pulley
guideCenter = [1 -(motorRadius-guideRadius) 0]'; % center point of fixed guide pulley
yGuideBottom = guideCenter(2)-guideRadius;
[xMotorTangent, yMotorTangent, xSpringTangentL, ySpringTangentL, tangentMotorToSpring] =
crosstan(motorCenter,motorRadius,springRadius,springPulleyCenter,1); %
tangentMotorToSpring=-tangentMotorToSpring;
cableLengthMotorToSpring = norm(tangentMotorToSpring);
springPoint = [springPulleyCenter(1)+(Pretension+springLength)*cosd(theta_i)
springPulleyCenter(2)+sind(theta_i)*(Pretension+springLength) 0]'; % position of fixed
end of spring
[xSpringTangentR, ySpringTangentR, xGuideTangent, yGuideTangent, tangentSpringToGuide] =
crosstan(springPulleyCenter,springRadius,guideRadius,guideCenter-springPulleyCenter,0); %
cableLengthSpringToGuide = norm(tangentSpringToGuide);
springPulleyRayLeft = [xSpringTangentL-springPulleyCenter(1) ySpringTangentL-
springPulleyCenter(2) 0];
springPulleyRayRight = [xSpringTangentR-springPulleyCenter(1) ySpringTangentR-
springPulleyCenter(2) 0];
springPulleyAngleRight = atan2d(springPulleyRayRight(2),springPulleyRayRight(1));
springPulleyAngleLeft = atan2d(springPulleyRayLeft(2),springPulleyRayLeft(1));
if springPulleyAngleLeft<0
```

```
    springPulleyAngleLeft = springPulleyAngleLeft+360;
end
springPulleyArcangle = springPulleyAngleLeft-springPulleyAngleRight;
springPulleyArclength = 2*pi*springRadius*(springPulleyArcangle/360);
    motorRayLeft = [motorCenter(1)-xMotorBottom motorCenter(2)-yMotorBottom 0];
    motorRayRight = [motorCenter(1)-xMotorTangent motorCenter(2)-yMotorTangent 0];
    motorArcangle = atan2d(norm(cross(motorRayLeft,motorRayRight)),
dot(motorRayLeft,motorRayRight));
    motorArclength = 2*pi*motorRadius*(motorArcangle/360);
    guideRayRight = [guideCenter(1)-xGuideBottom guideCenter(2)-yGuideBottom 0];
    guideRayLeft = [guideCenter(1)-xGuideTangent guideCenter(2)-yGuideTangent 0];
    guideArcangle = atan2d(norm(cross(guideRayRight,guideRayLeft)),
dot(guideRayRight,guideRayLeft));
    guideArclength = 2*pi*guideRadius*(guideArcangle/360);
lengthArray = [];
cableLength =
springPulleyArclength+cableLengthMotorToSpring+cableLengthSpringToGuide+motorArclength+gu
ideArclength;
deflectionArray = [];
unitTension1 = tangentMotorToSpring/norm(tangentMotorToSpring);
unitTension2 = tangentSpringToGuide/norm(tangentSpringToGuide);
currentSpringLength = springPoint-springPulleyCenter;
unitSpringLength = currentSpringLength/norm(currentSpringLength);
normalVector = motorCenter-springPulleyCenter;
unitNormalVector = normalVector/norm(normalVector);
% Find Final Orientation %

finalSpringCenterY = -(motorRadius+springRadius);
theta_f = asind(finalSpringCenterY/R_i);
theta_f = theta_f + 2;
theta = linspace(theta_i,theta_f,1000);
% Evaluation of Cable Tension

for i = 1:length(theta)
    springPulleyCenter = [cosd(theta(i))*R_i sind(theta(i))*R_i 0]'; %  position of
center point of moving lever pulley
    [xMotorTangent, yMotorTangent, xSpringTangentL, ySpringTangentL,
tangentMotorToSpring, ~, ~] =
crosstan(motorCenter,motorRadius,springRadius,springPulleyCenter,1); %
    xMotorBottom = 0; yMotorBottom = -motorRadius; x6 = 1; yGuideBottom = guideCenter(2)-
guideRadius;
    cableLengthMotorToSpring = norm(tangentMotorToSpring);
    tangentMotorToSpring=-tangentMotorToSpring;
    [xSpringTangentR, ySpringTangentR, xGuideTangent, yGuideTangent,
tangentSpringToGuide, ~, ~] =
crosstan(springPulleyCenter,springRadius,guideRadius,guideCenter-springPulleyCenter,0);
    cableLengthSpringToGuide = norm(tangentSpringToGuide);
    springPulleyRayLeft = [xSpringTangentL-springPulleyCenter(1) ySpringTangentL-
springPulleyCenter(2) 0];
    springPulleyRayRight = [xSpringTangentR-springPulleyCenter(1) ySpringTangentR-
springPulleyCenter(2) 0];
    springPulleyAngleRight = atan2d(springPulleyRayRight(2),springPulleyRayRight(1));
    springPulleyAngleLeft = atan2d(springPulleyRayLeft(2),springPulleyRayLeft(1));
    if springPulleyAngleLeft<0
```

```
        springPulleyAngleLeft = springPulleyAngleLeft+360;
    end
    springPulleyArcangle = springPulleyAngleLeft-springPulleyAngleRight;
    springPulleyArclength = 2*pi*springRadius*(springPulleyArcangle/360);
    lengthArray = [lengthArray; springPulleyArclength];
    motorRayLeft = [motorCenter(1)-xMotorBottom motorCenter(2)-yMotorBottom 0];
    motorRayRight = [motorCenter(1)-xMotorTangent motorCenter(2)-yMotorTangent 0];
    motorArcangle = atan2d(norm(cross(motorRayLeft,motorRayRight)),
dot(motorRayLeft,motorRayRight));
    motorArclength = 2*pi*motorRadius*(motorArcangle/360);
    guideRayRight = [guideCenter(1)-x6 guideCenter(2)-yGuideBottom 0];
    guideRayLeft = [guideCenter(1)-xGuideTangent guideCenter(2)-yGuideTangent 0];
    guideArcangle = atan2d(norm(cross(guideRayRight,guideRayLeft)),
dot(guideRayRight,guideRayLeft));
    guideArclength = 2*pi*guideRadius*(guideArcangle/360);
    deflectionArray = [deflectionArray cableLength-
(cableLengthMotorToSpring+springPulleyArclength+cableLengthSpringToGuide+motorArclength+g
uideArclength)];
    unitTension1 = tangentMotorToSpring/norm(tangentMotorToSpring);
    unitTension2 = tangentSpringToGuide/norm(tangentSpringToGuide);
    currentSpringLength = springPoint-springPulleyCenter;
    unitSpringLength = currentSpringLength/norm(currentSpringLength);
    normalVector = motorCenter-springPulleyCenter;
    unitNormalVector = normalVector/norm(normalVector);
    springForce = springConstant*(norm(currentSpringLength)-springLength);
    magnitudeSpringForce = norm(springForce);
    Amatrix = [(unitTension1(1)+unitTension2(1)) unitNormalVector(1);
(unitTension1(2)+unitTension2(2)) unitNormalVector(2)];
    [forceMagnitudes] = inv(Amatrix)*[(-springForce*unitSpringLength(1)); (-
springForce*unitSpringLength(2))];
    tensionMagnitude = forceMagnitudes(1);
    normalMagnitude = forceMagnitudes(2);
    tensionArray(i) = tensionMagnitude;
    T1s(i,:) = tensionMagnitude*unitTension1;
    T2s(i,:) = tensionMagnitude*unitTension2;
    Fss(i,:) = magnitudeSpringForce*unitSpringLength;
    magFss(i,:) = magnitudeSpringForce;
    Fls(i) = normalMagnitude;
end
xc = tensionArray/cableConstant;
lengthArray = deflectionArray+xc;

cableTravelLength=100;
tol = 0.05;
dT = diff(tensionArray);
ddL = diff(lengthArray);
dTdL = dT./ddL;
for i = 1:length(dTdL)
    dif(i) = (abs(dTdL(i)-((2*lengthArray(i)))))-tol;
end
signdif=sign(dif);
swtch=find(diff(sign(signdif)))+1;
difference = NaN;
if length(swtch)>2
```

```
    oops = 1;
end
for i = 1:length(swtch)-1
    difference(i) = swtch(i+1)-swtch(i);
end
try
[B I] = sort(difference,'descend');
if signdif(1)>0
    if mod(I(1),2) == 1
        cableTravelLength = -(lengthArray(swtch(I(1)+1))-lengthArray(swtch(I(1))));
        top = lengthArray(swtch(I(1)+1));
        bottom = lengthArray(swtch(I(1)));
    elseif mod(I(2),2) == 1
        cableTravelLength = -(lengthArray(swtch(I(2)+1))-lengthArray(swtch(I(2))));
        top = lengthArray(swtch(I(2)+1));
        bottom = lengthArray(swtch(I(2)));
    elseif mod(I(3),2) == 1
        cableTravelLength = -(lengthArray(swtch(I(3)+1))-lengthArray(swtch(I(3))));
        top = lengthArray(swtch(I(3)+1));
        bottom = lengthArray(swtch(I(3)));
    elseif mod(I(4),2) == 1
        cableTravelLength = -(lengthArray(swtch(I(4)+1))-lengthArray(swtch(I(4))));
        top = lengthArray(swtch(I(4)+1));
        bottom = lengthArray(swtch(I(4)));
    elseif mod(I(5),2) == 1
        cableTravelLength = -(lengthArray(swtch(I(5)+1))-lengthArray(swtch(I(5))));
        top = lengthArray(swtch(I(5)+1));
        bottom = lengthArray(swtch(I(5)));
    end
else
    if mod(I(1),2) == 0
        cableTravelLength = -(lengthArray(swtch(I(1)+1))-lengthArray(swtch(I(1))));
        top = lengthArray(swtch(I(1)+1));
        bottom = lengthArray(swtch(I(1)));
    elseif mod(I(2),2) == 0
        cableTravelLength = -(lengthArray(swtch(I(2)+1))-lengthArray(swtch(I(2))));
        top = lengthArray(swtch(I(2)+1));
        bottom = lengthArray(swtch(I(2)));
    elseif mod(I(3),2) == 0
        cableTravelLength = -(lengthArray(swtch(I(3)+1))-lengthArray(swtch(I(3))));
        top = lengthArray(swtch(I(3)+1));
        bottom = lengthArray(swtch(I(3)));
    elseif  mod(I(4),2) == 0
        cableTravelLength = -(lengthArray(swtch(I(4)+1))-lengthArray(swtch(I(4))));
        top = lengthArray(swtch(I(4)+1));
        bottom = lengthArray(swtch(I(4)));
    elseif  mod(I(5),2) == 0
        cableTravelLength = -(lengthArray(swtch(I(5)+1))-lengthArray(swtch(I(5))));
        top = lengthArray(swtch(I(5)+1));
        bottom = lengthArray(swtch(I(5)));
    end
end

catch
```

```
    cableTravelLength=0;
    bottom = NaN;
    top = NaN;
end
```

### Crosstan.m

```
% Description: Calculates the crossed tangent points between two circles.
% This is used to find the tangent points of the cable and the pulleys as
% well as the force vector of the cable tensions. This function is called in
% LeverStaticAnalysis.m.

% Inputs:   C1 - (Vector) of center point of the first circle (pulley)

%           r1 - (scalar) radius of first circle (pulley)

%           r2 - (scalar) radius of second circle (pulley)

%           L  - (vector) of length between the two circles

%           tb - (scalar) "top or bottom" crossed tangent. 1 is used from the cable
%           between the motor and slot pulley and 0 is used from the cable
%           between the slot and guide pulley.

% Outputs:  x1 - (scalar) x location of the tangent point on first circle

%           y1 - (scalar) y location of the tangent point on first circle

%           x2 - (scalar)x location of the tangent point on second circle

%           y2 - (scalar) y location of the tangent point on second circle

%           T  - (Vector) of crossed tangent from first circle to second
%                circle from (x1,y1) to (x2,y2)

function [x1 y1 x2 y2 T alpha angle] = crosstan(C1,r1,r2,L,tb)
```

*cross tangents to two circles*

```
x = [1 0 0]';                       %  Establishes global x direction
C2 = L+C1;                          %  Center of second circle
d2 = L/(1+r1/r2);                   %  Distance between intersection point of tangent
line and x axis to the center of...
                                    %  the second circle
d1 = L-d2;                          %  Distance between the center of the first circle
to the intersection point of...
                                    %  tangent line and x axis.
cros = cross(x,L);
angle = atan2d(cros(3), dot(x,L));  %  Angle between x axis and actual vector from one
circle center to the other
alpha = acosd((r1+r2)/norm(L));     %  Angle between vector from one circle center to
the other and tangent point...
```

```
                                    %  of cable on circle.
if tb > 0
    alpha = -alpha;                 %  Flips angle if using the other crossed tangent
line
end
alpha = alpha+angle;                %  adds angles together to achieve true angle of
vector from global x-axis
x1=r1*cosd(alpha)+C1(1);            %  x location of tangent point on first circle
y1=r1*sind(alpha)+C1(2);            %  y location of tangent point on first circle
beta=180+alpha;                     %  rotates alpha by 180 degrees to calculate other
circle points
x2=L(1)+r2*cosd(beta)+C1(1);        %  x location of tangent point on second circle
y2=L(2)+r2*sind(beta)+C1(2);        %  y location of tangent point on second circle
T = [x2-x1; y2-y1; 0];              %  Vector of crossed tangent from first circle to
second circle.
```

# APPENDIX C

This appendix contains the graphs for all of the antagonistic testing of the mechanisms that is not included in the main body of this thesis. The graphs will be grouped based on the mechanism the tests were done on, i.e. slot or lever mechanism, and the initial angle of the robot link that the mechanisms are controlling. Each mechanism was tested with an initial link angle of 0 degrees, -20 degrees, and 20 degrees. Each graph within each section represents a set of tests where the motor positions are set to 10 degree increments starting at a 10 degree angle (low stiffness) up to a 50 degree angle (high stiffness) except for the 0 degree tests of the lever mechanism which range from 10 degrees to 80 degrees. Each section then ends with a combination of one test from each of the other graphs to show the stiffness progression as the motor angles change.

## SLOT MECHANISM 0 DEGREE LINK ANGLE

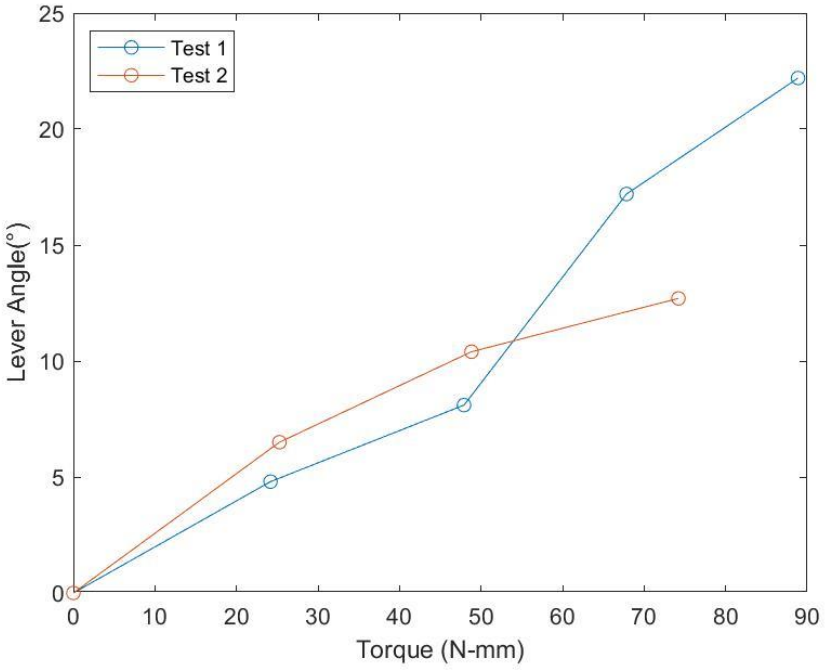


Figure 38: 0 Degree Slot Mechanism 10 Degree Motor Angles

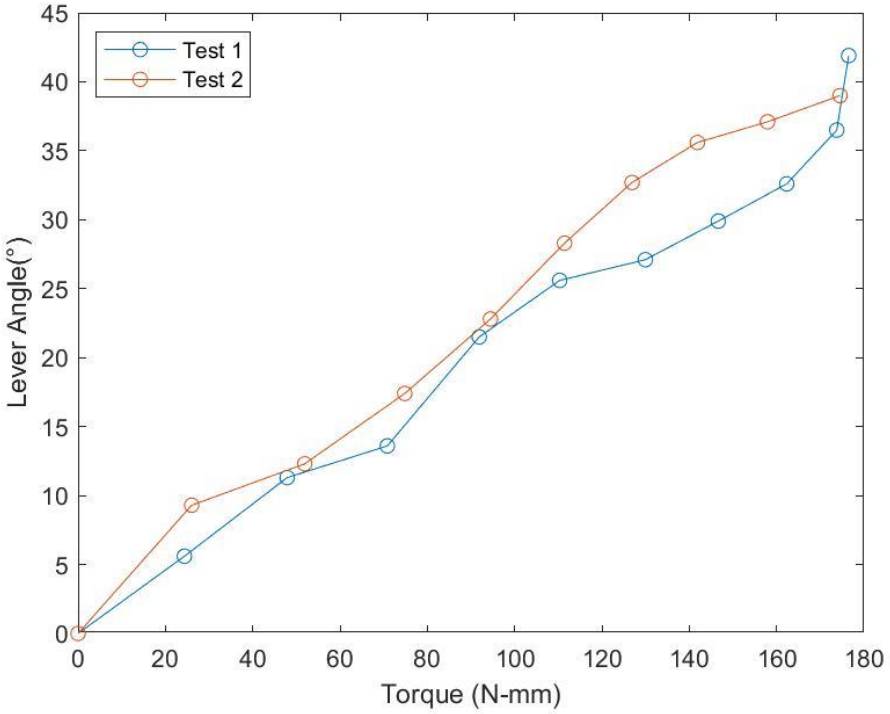


Figure 39: 0 Degree Slot Mechanism 20 Degree Motor Angles

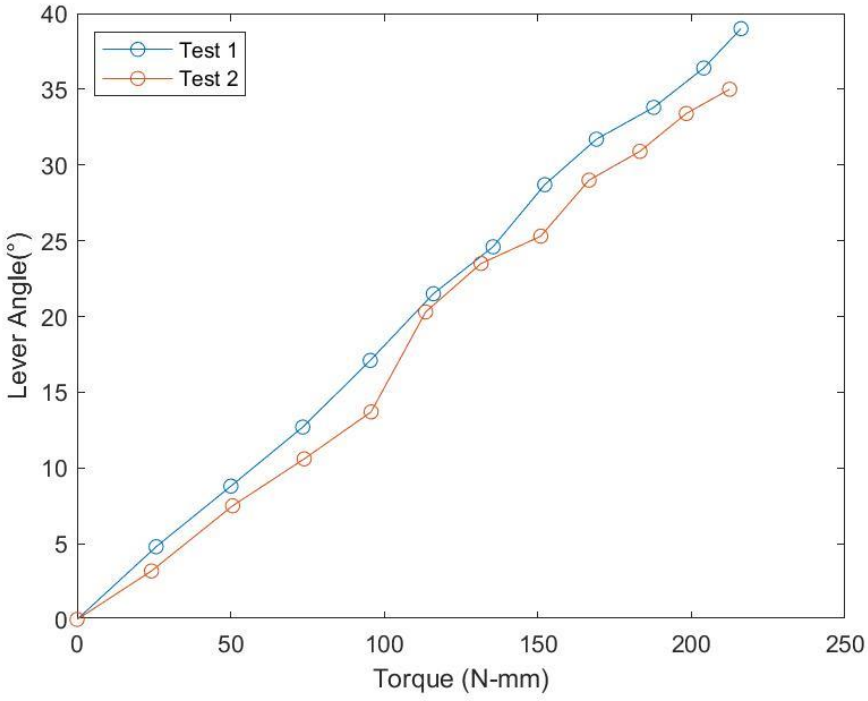


Figure 40: 0 Degree Slot Mechanism 30 Degree Motor Angles

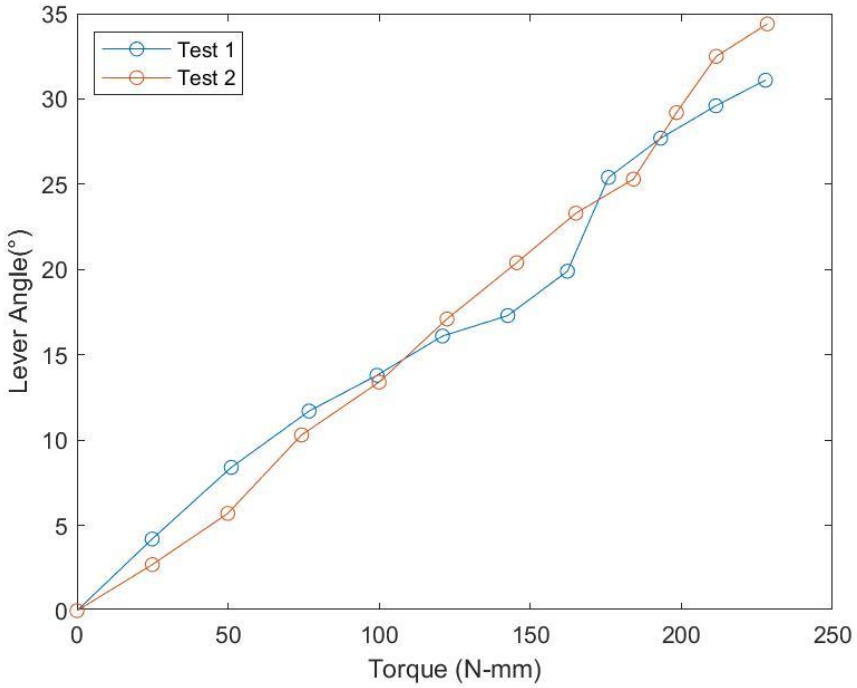


Figure 41: 0 Degree Slot Mechanism 40 Degree Motor Angles

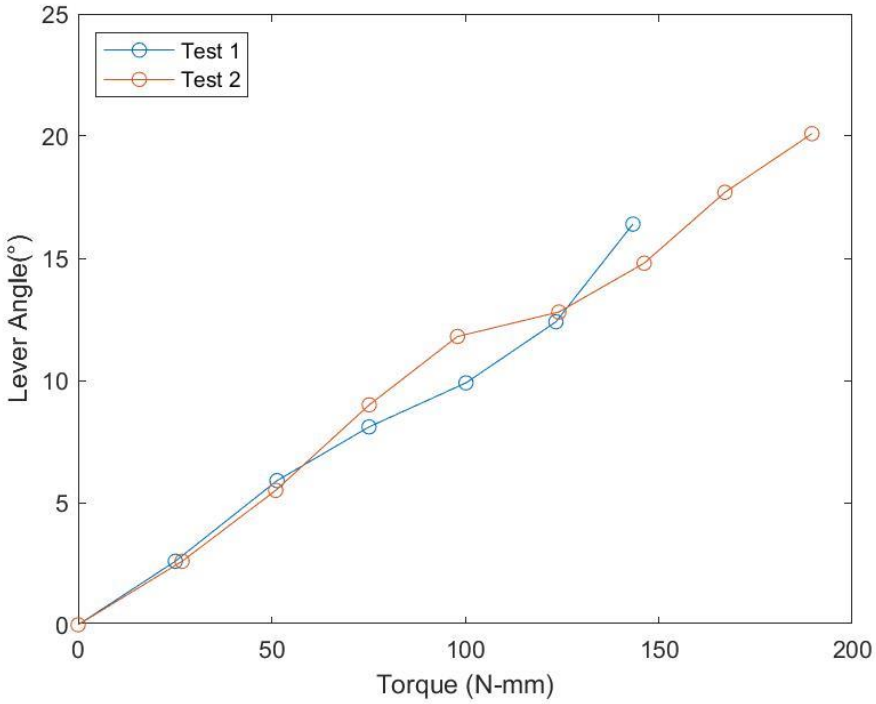


Figure 42: 0 Degree Slot Mechanism 50 Degree Motor Angles

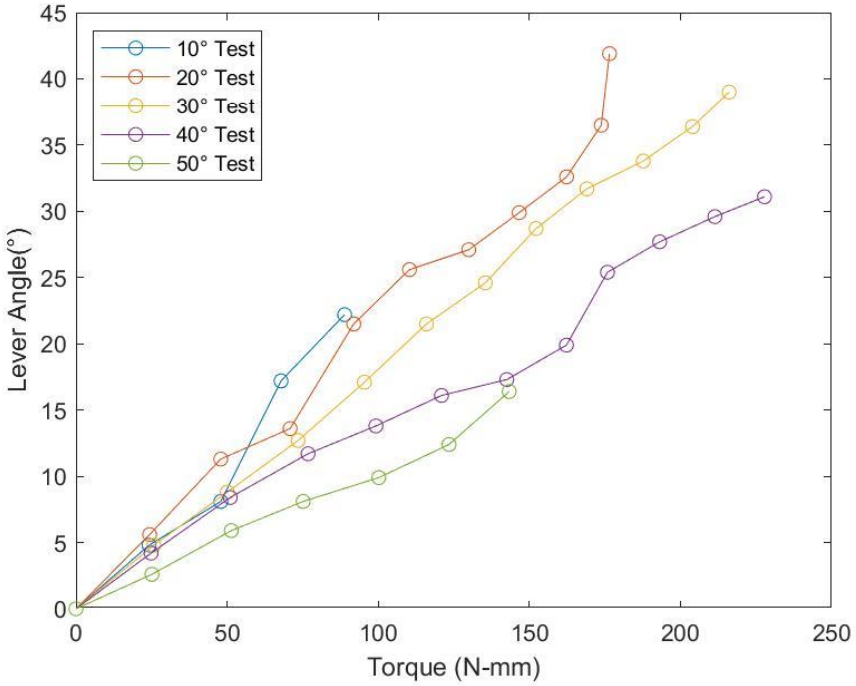


Figure 43: 0 Degree Slot Mechanism Combined Data

**SLOT MECHANISM -20 DEGREE LINK ANGLE**

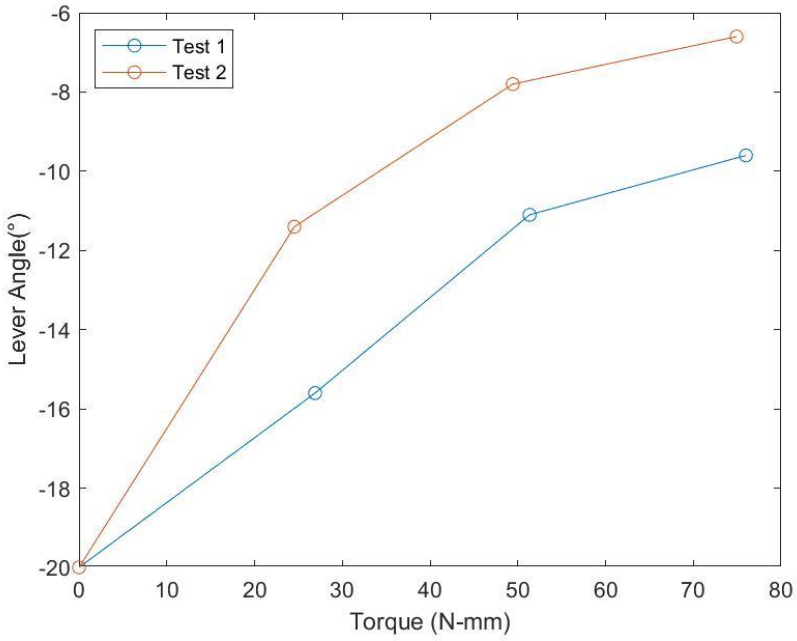


Figure 44: -20 Degree Slot Mechanism 10 Degree Motor Angles

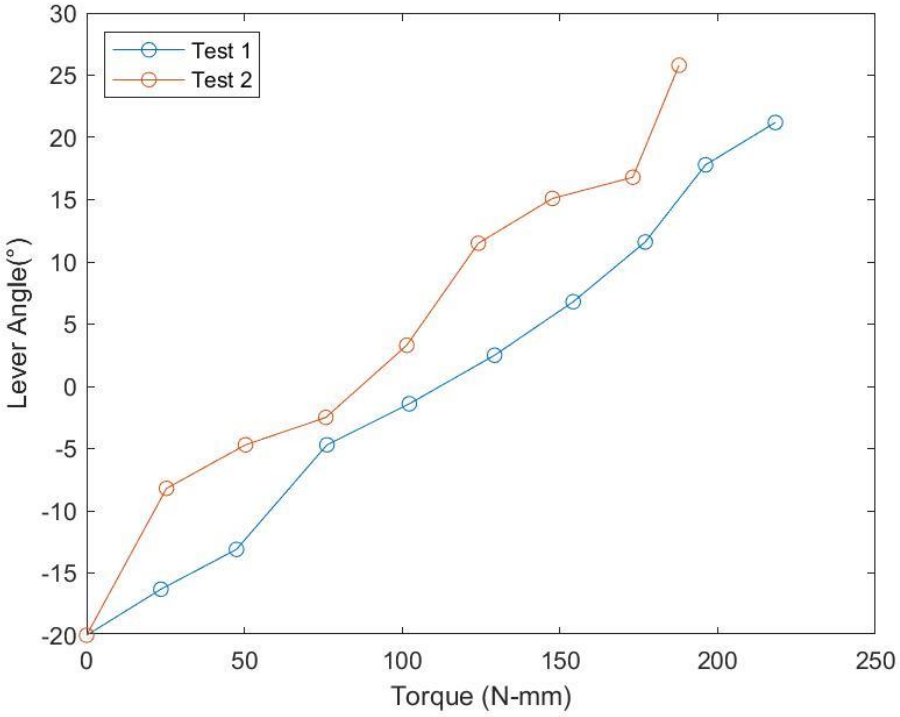


Figure 45: -20 Degree Slot Mechanism 20 Degree Motor Angles

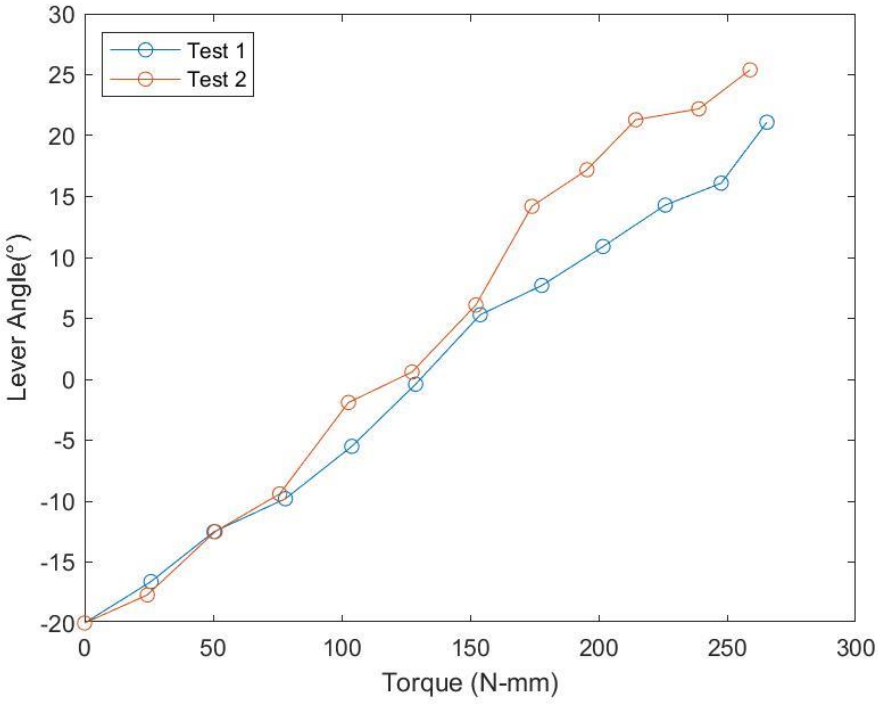


Figure 46: -20 Degree Slot Mechanism 30 Degree Motor Angles

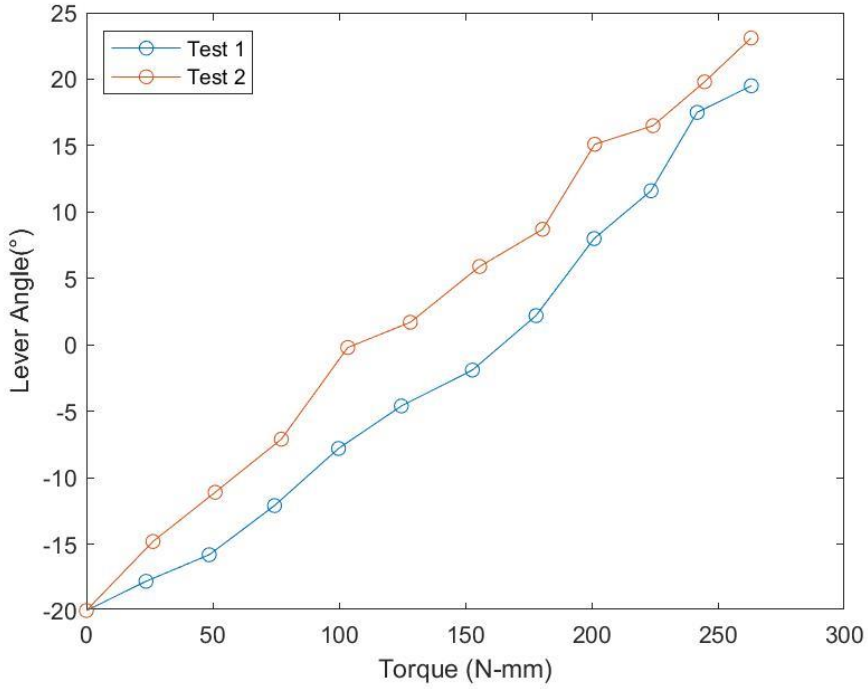


Figure 47: -20 Degree Slot Mechanism 40 Degree Motor Angles

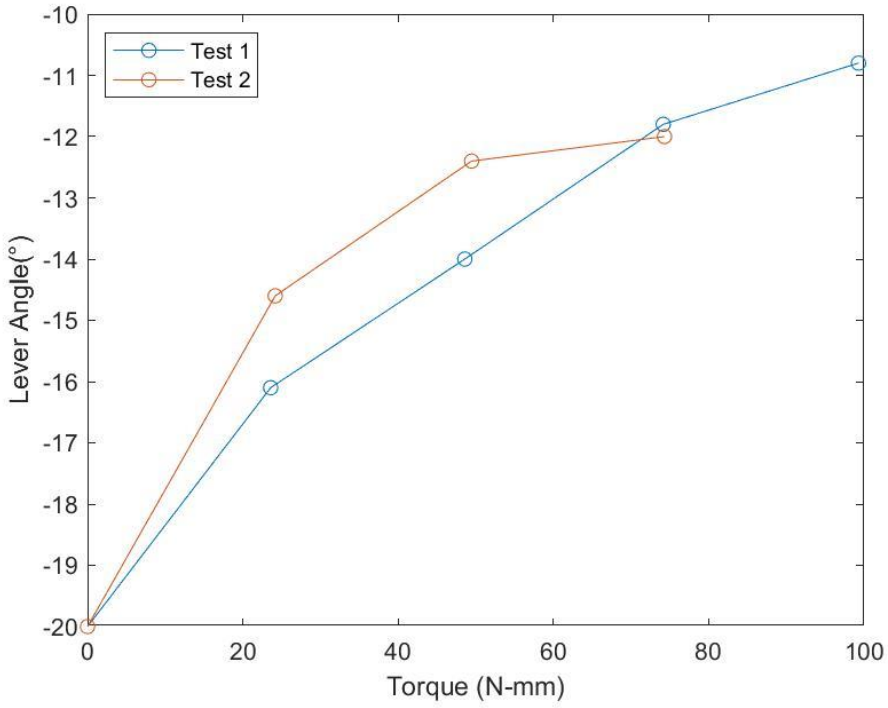


Figure 48: -20 Degree Slot Mechanism 50 Degree Motor Angles

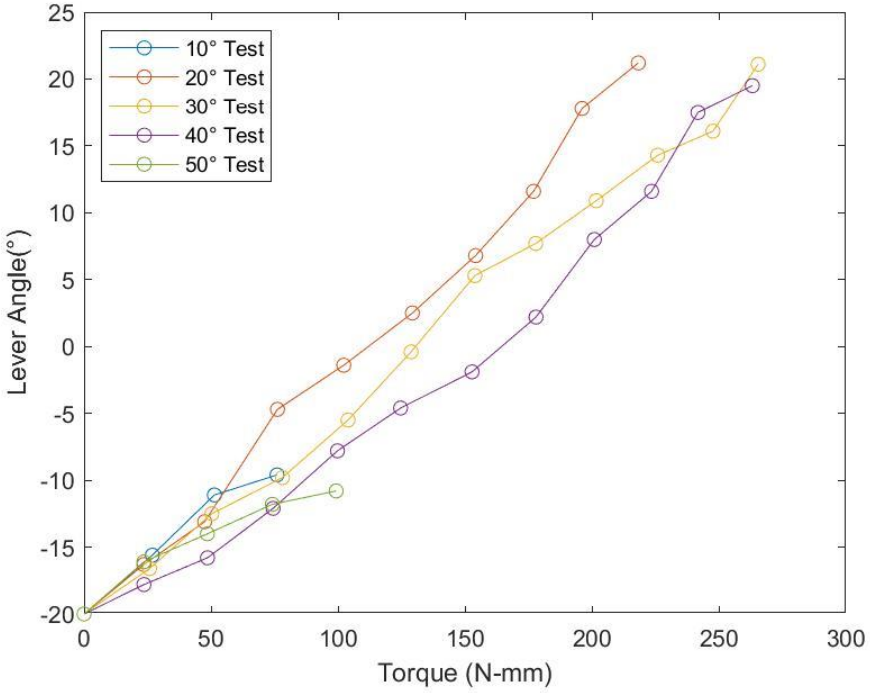


Figure 49: -20 Degree Slot Mechanism Combined Data

**SLOT MECHANISM 20 DEGREE LINK ANGLE**

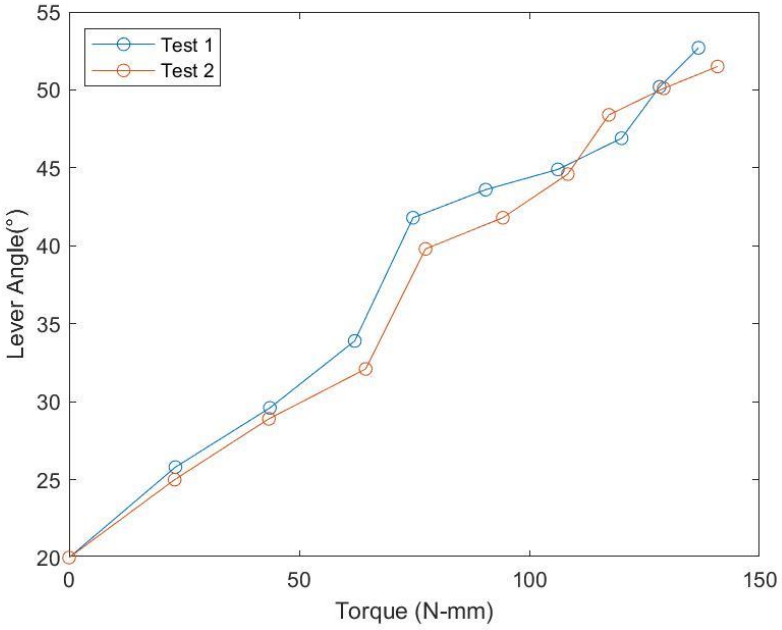


Figure 50: 20 Degree Slot Mechanism 20 Degree Motor Angles

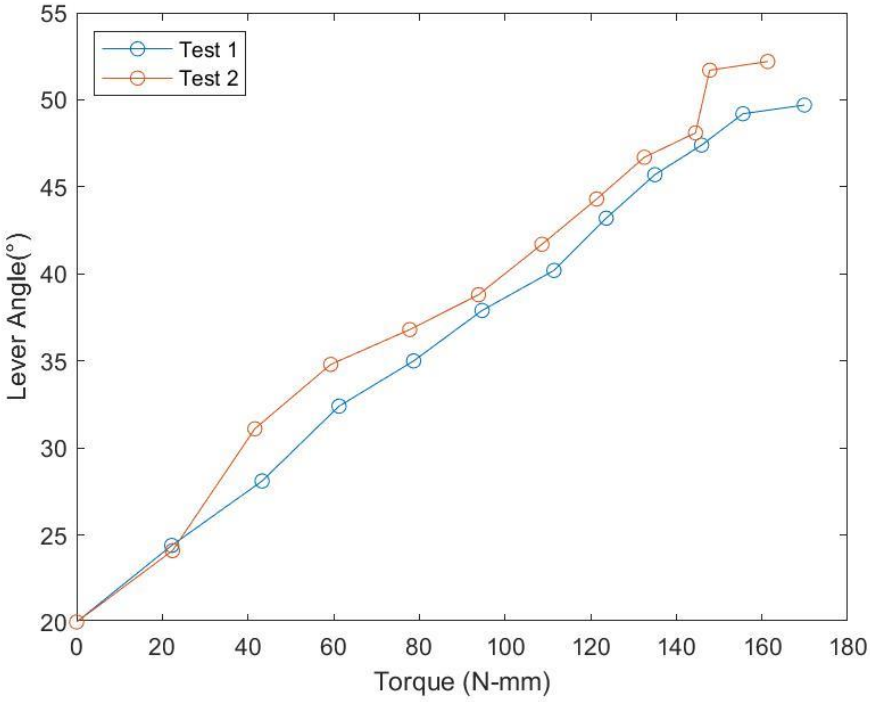


Figure 51: 20 Degree Slot Mechanism 30 Degree Motor Angles

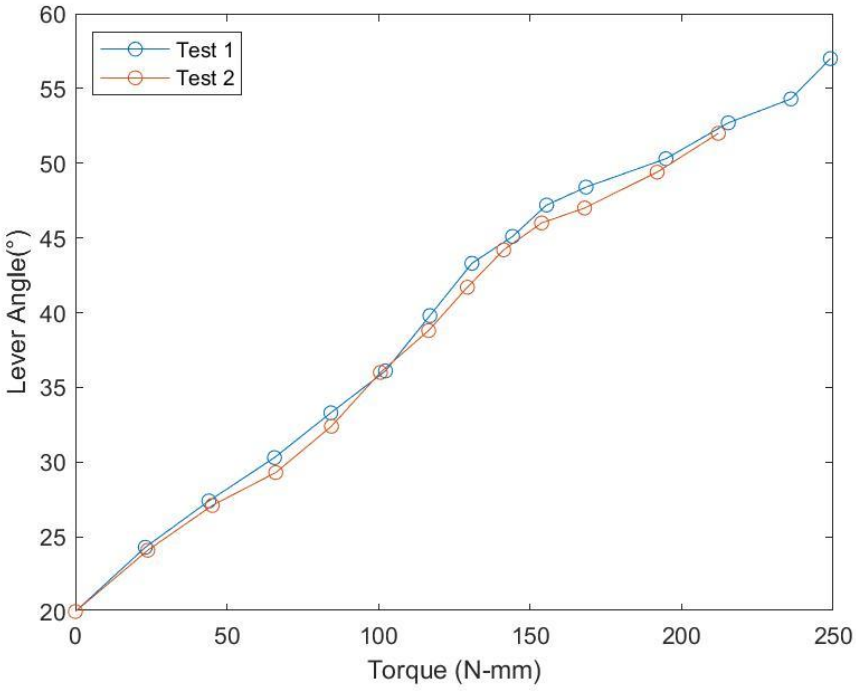


Figure 52: 20 Degree Slot Mechanism 40 Degree Motor Angles

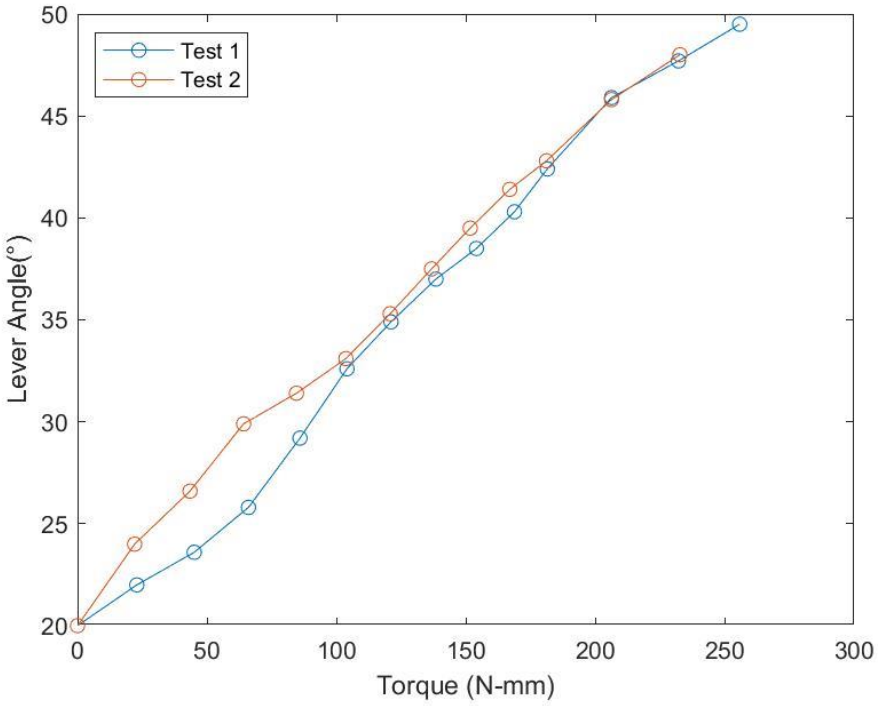


Figure 53: 20 Degree Slot Mechanism 50 Degree Motor Angles

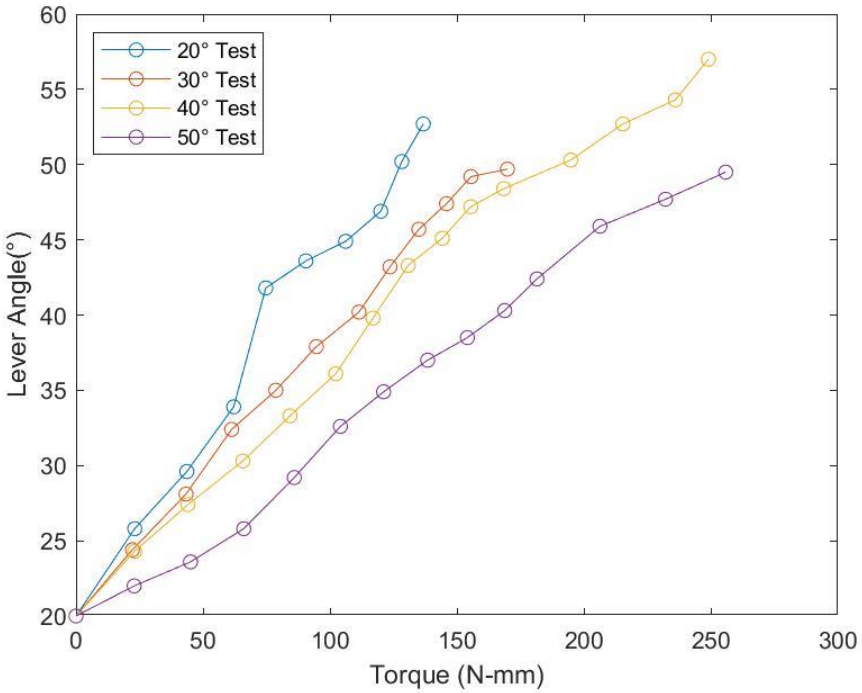


Figure 54: 20 Degree Slot Mechanism Combined Data

**LEVER MECHANISM 0 DEGREE LINK ANGLE**

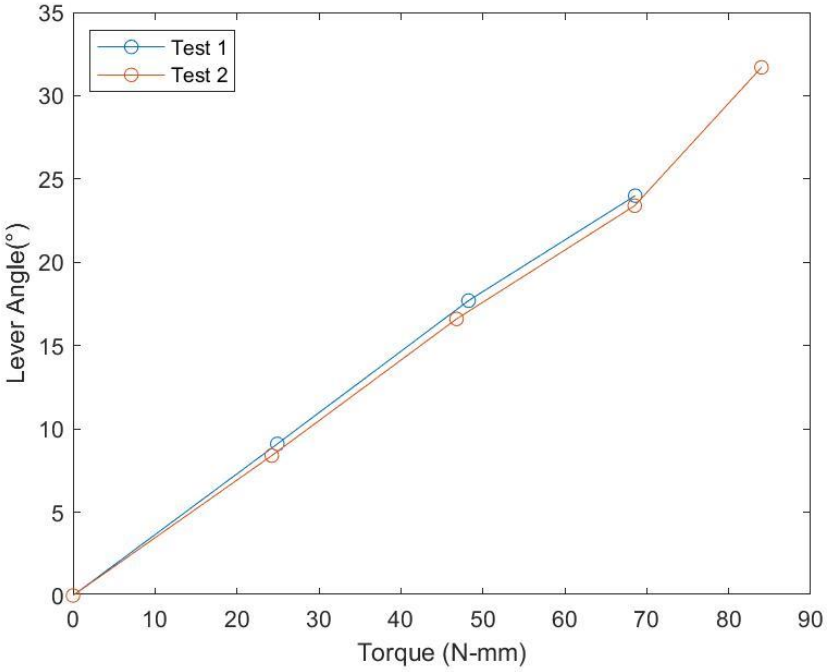


Figure 55: 0 Degree Lever Mechanism 10 Degree Motor Angles

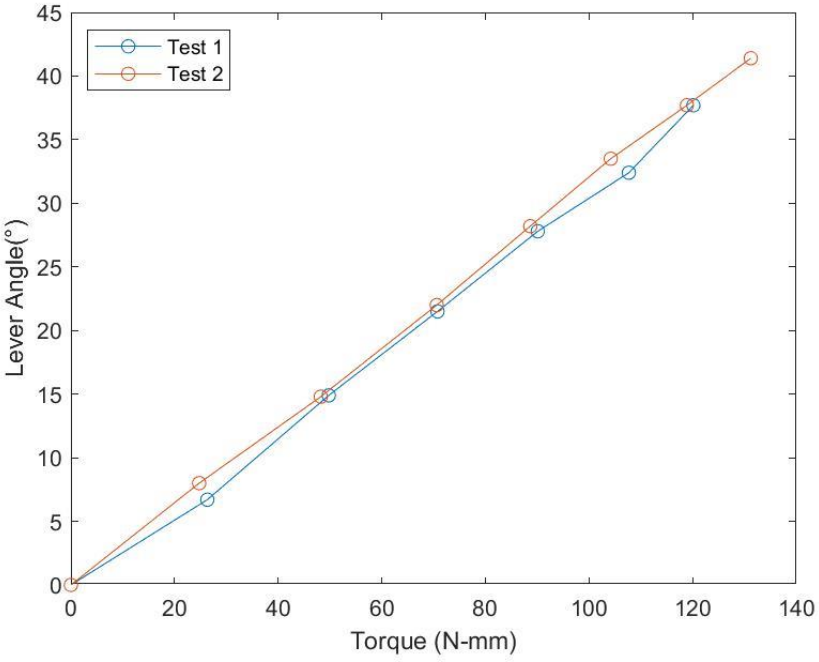


Figure 56: 0 Degree Lever Mechanism 20 Degree Motor Angles

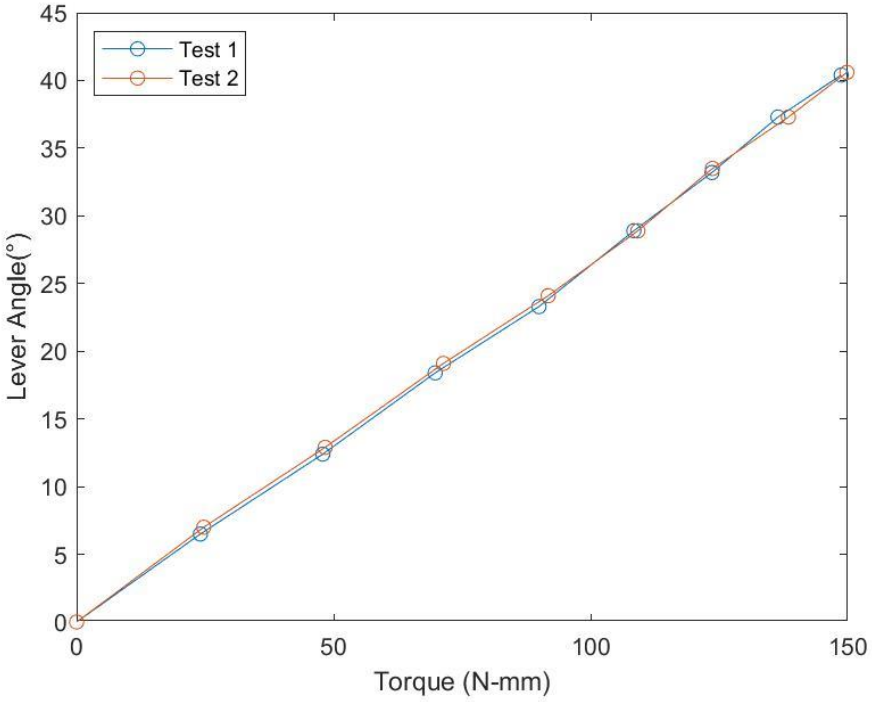


Figure 57: 0 Degree Lever Mechanism 30 Degree Motor Angles

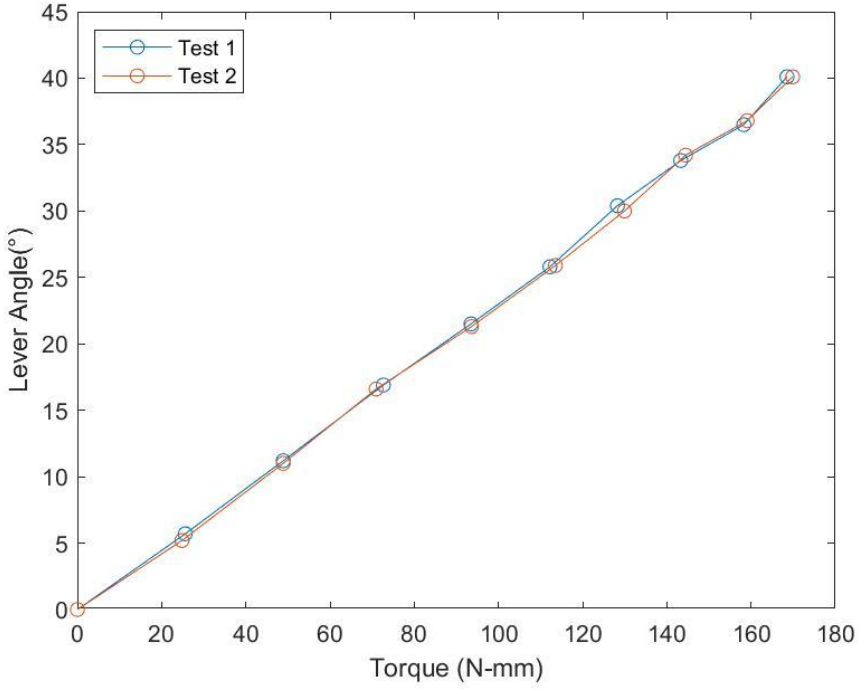


Figure 58: 0 Degree Lever Mechanism 40 Degree Motor Angles

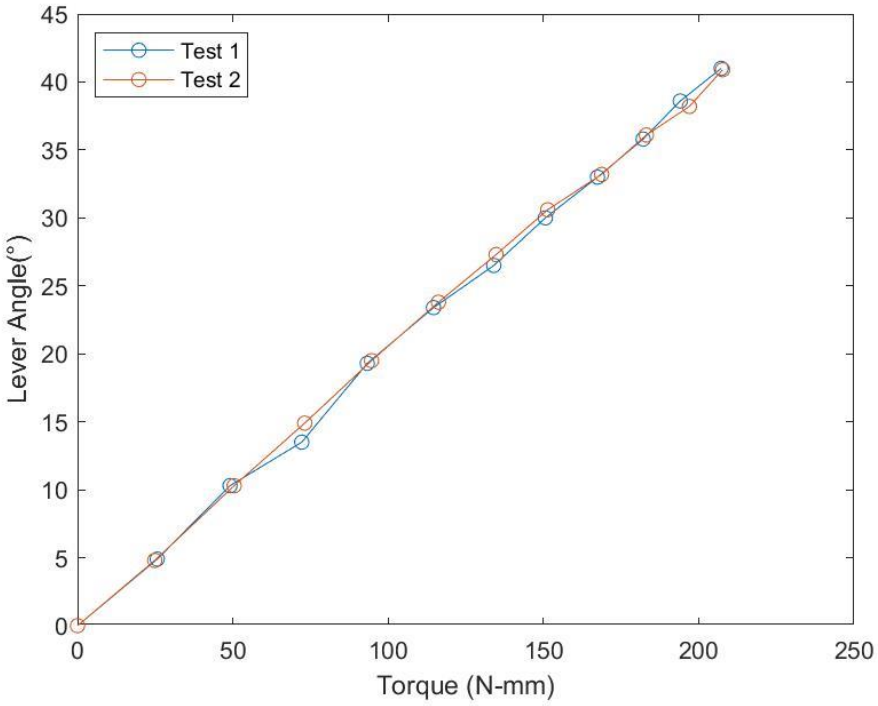


Figure 59: 0 Degree Lever Mechanism 50 Degree Motor Angles

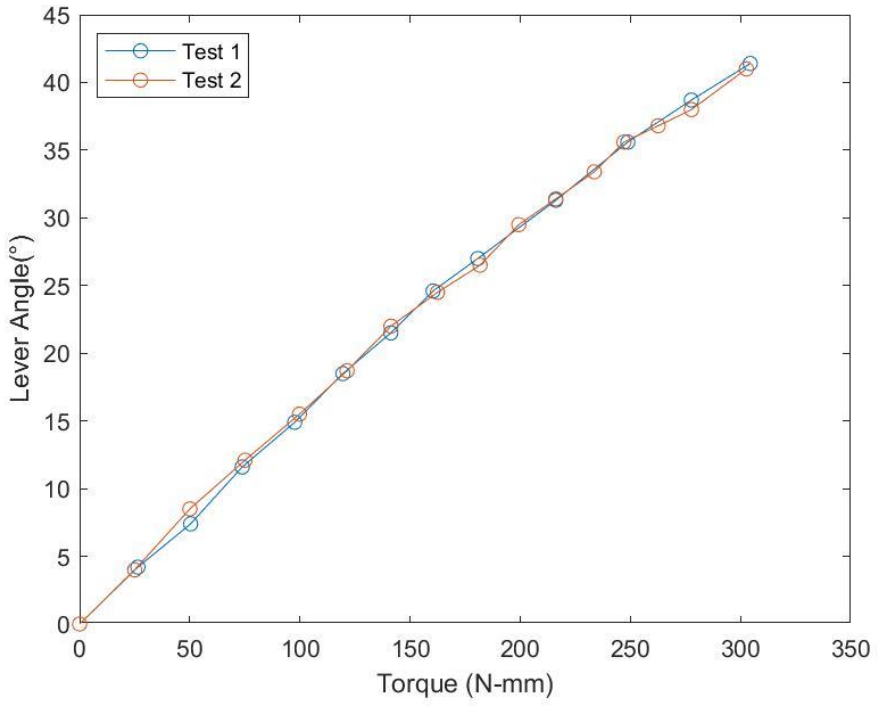


Figure 60: 0 Degree Lever Mechanism 60 Degree Motor Angles

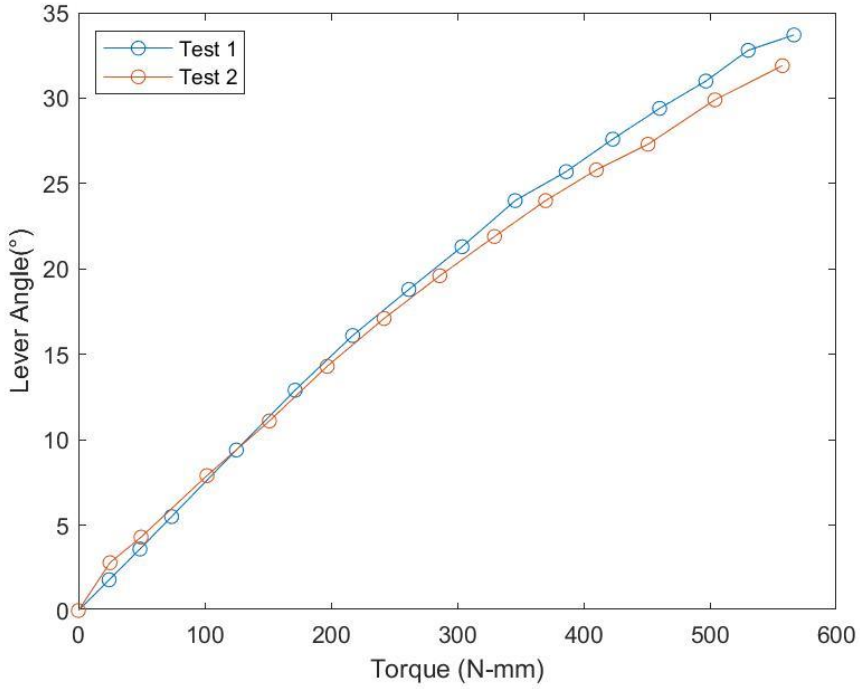


Figure 61: 0 Degree Lever Mechanism 70 Degree Motor Angles

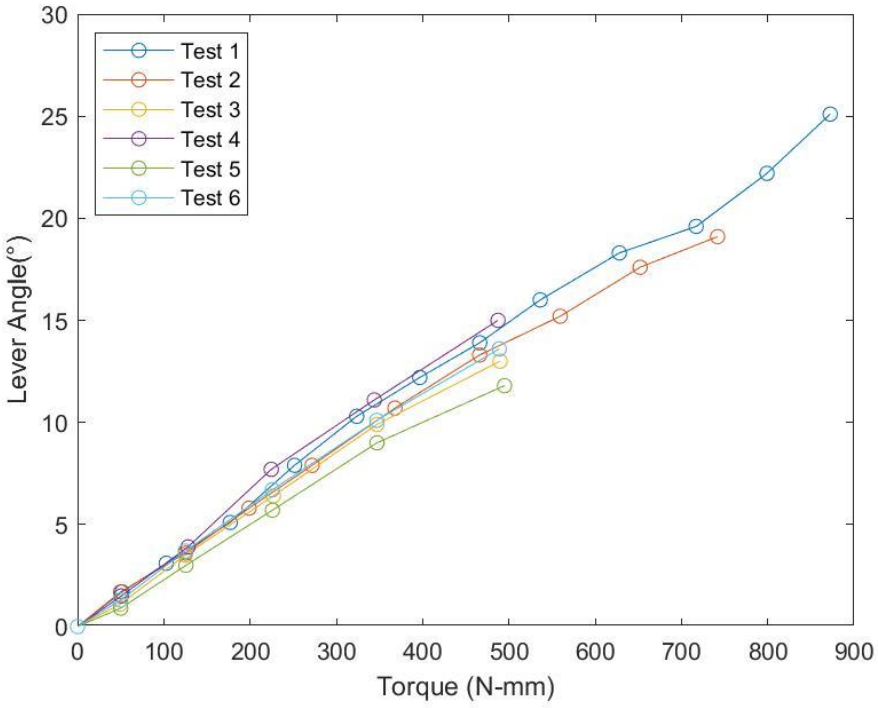


Figure 62: 0 Degree Lever Mechanism 80 Degree Motor Angles

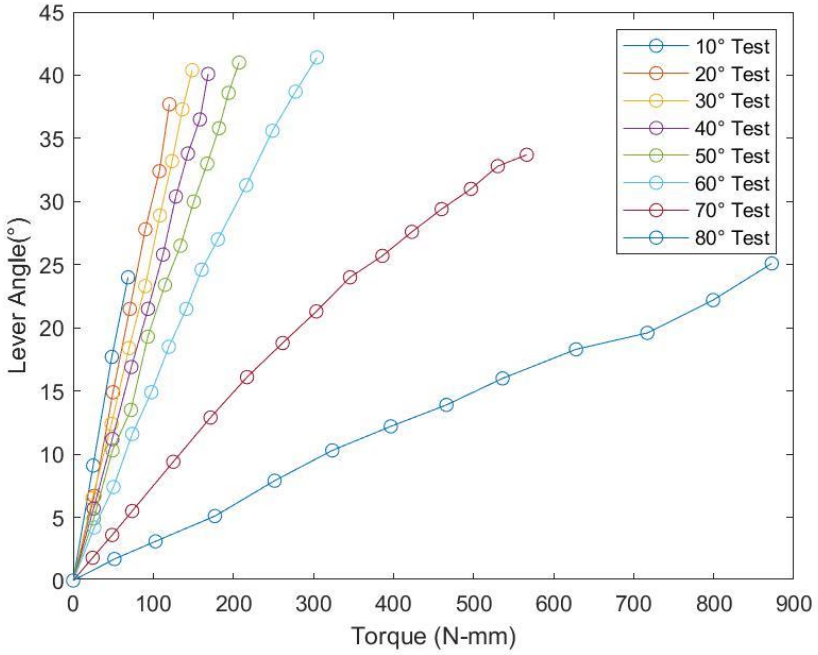


Figure 63: 0 Degree Slot Mechanism Combined Data

**LEVER MECHANISM -20 DEGREE LINK ANGLE**

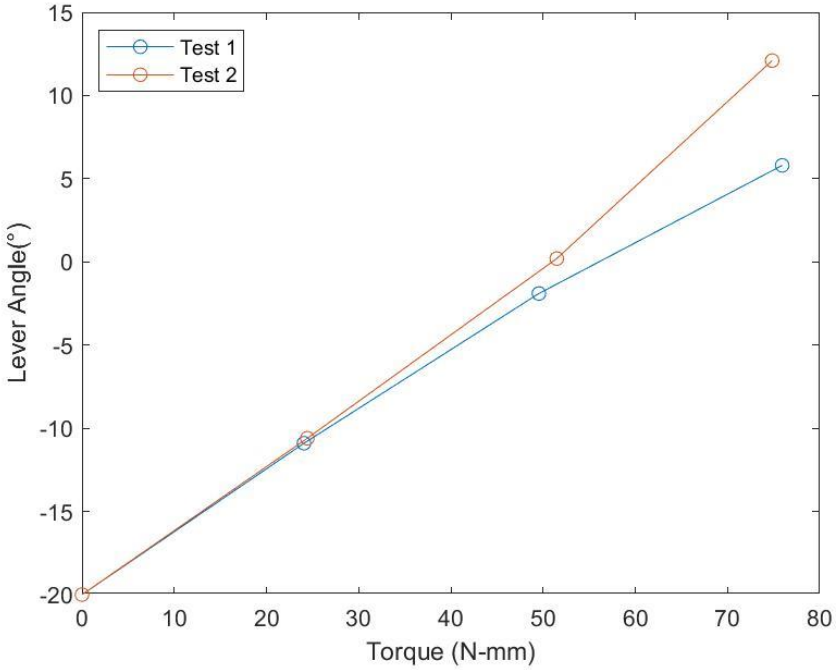


Figure 64: -20 Degree Lever Mechanism 10 Degree Motor Angles

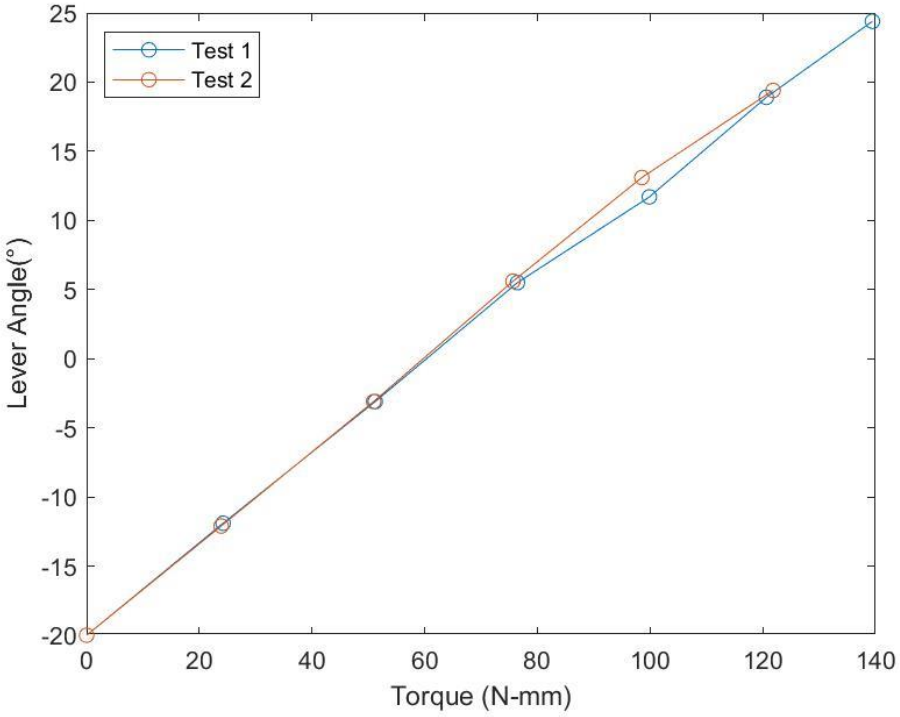


Figure 65: -20 Degree Lever Mechanism 20 Degree Motor Angles

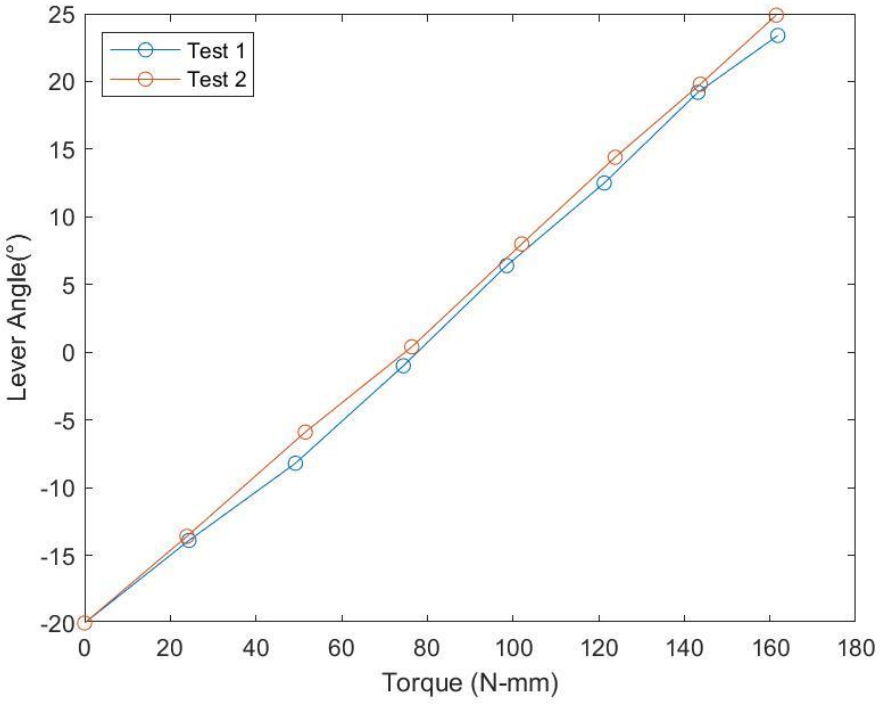


Figure 66: -20 Degree Lever Mechanism 30 Degree Motor Angles

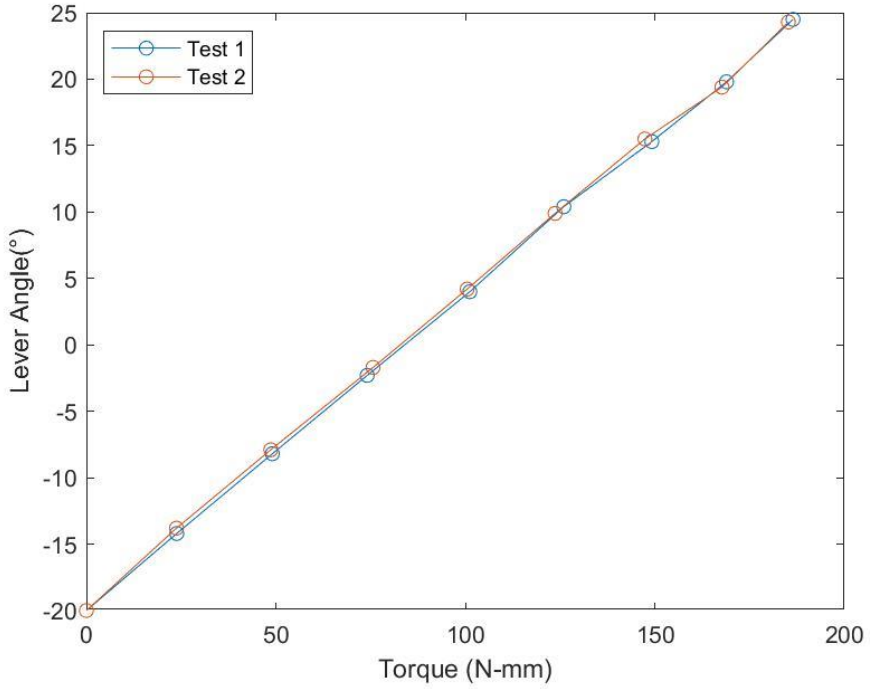


Figure 67: -20 Degree Lever Mechanism 40 Degree Motor Angles

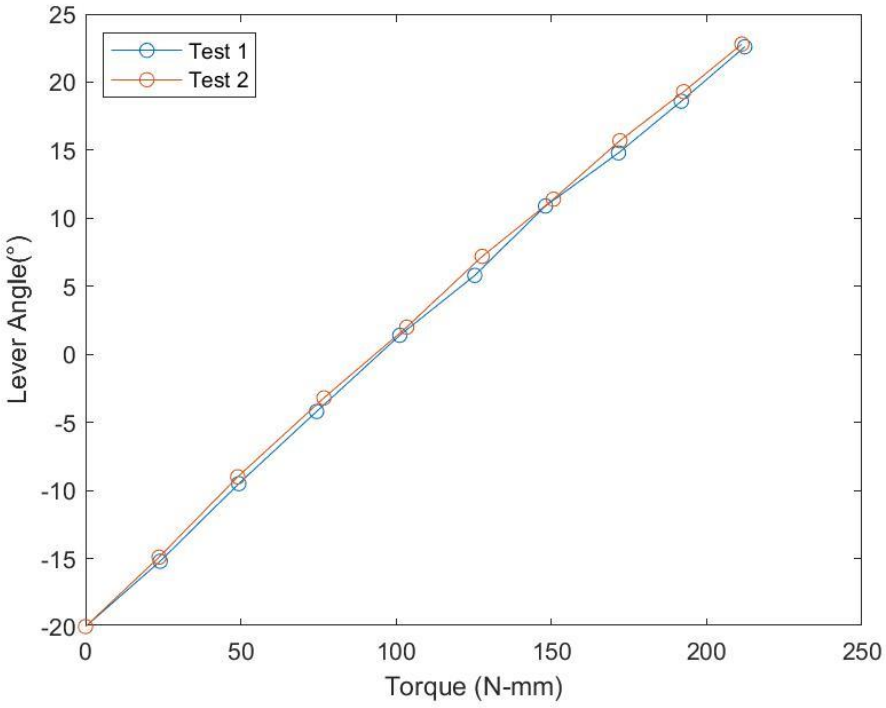


Figure 68: -20 Degree Lever Mechanism 50 Degree Motor Angles

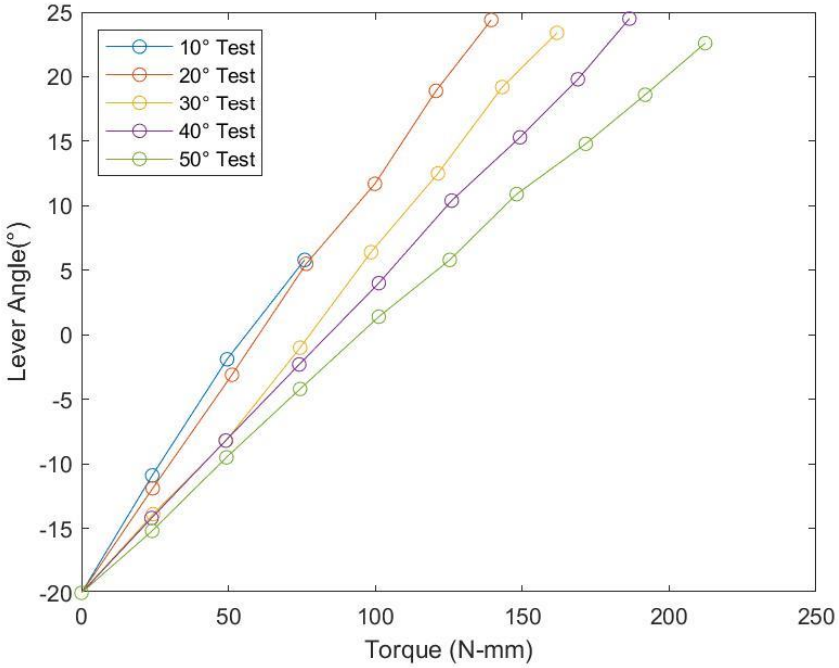


Figure 69: -20 Degree Lever Mechanism Combined Data

**LEVER MECHANISM 20 DEGREE LINK ANGLE**

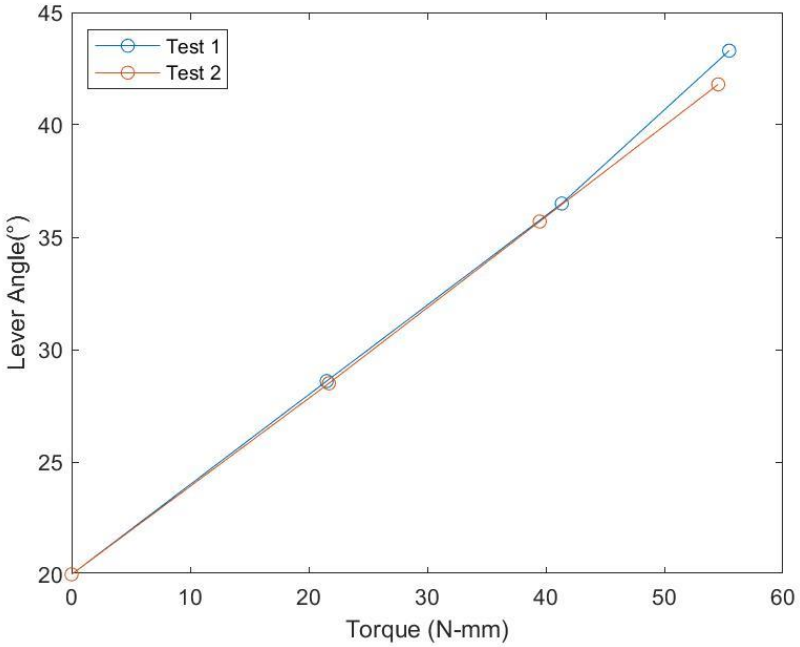


Figure 70: 20 Degree Lever Mechanism 10 Degree Motor Angles

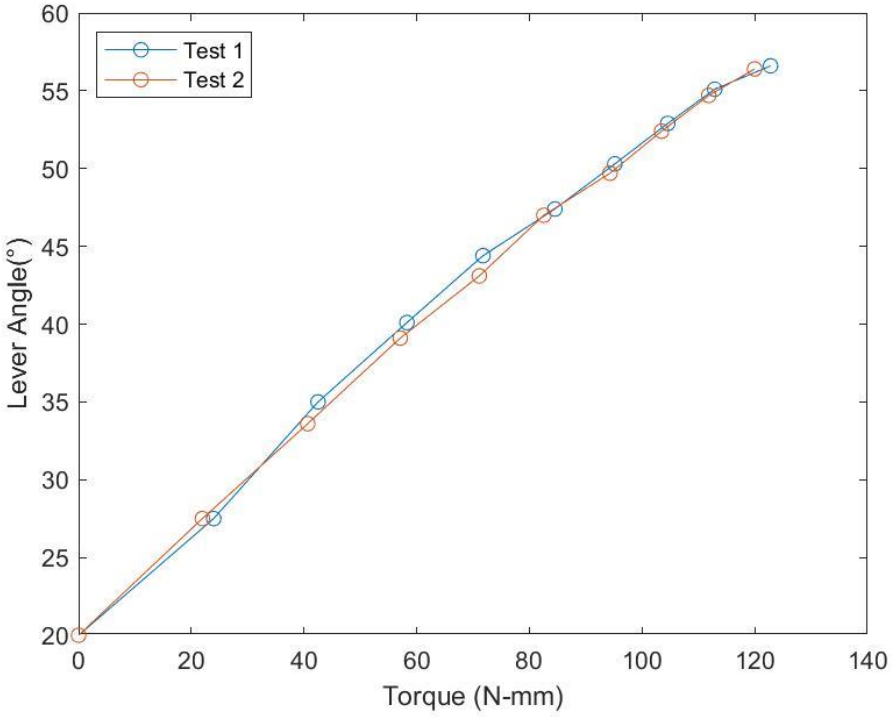


Figure 71: 20 Degree Lever Mechanism 20 Degree Motor Angles

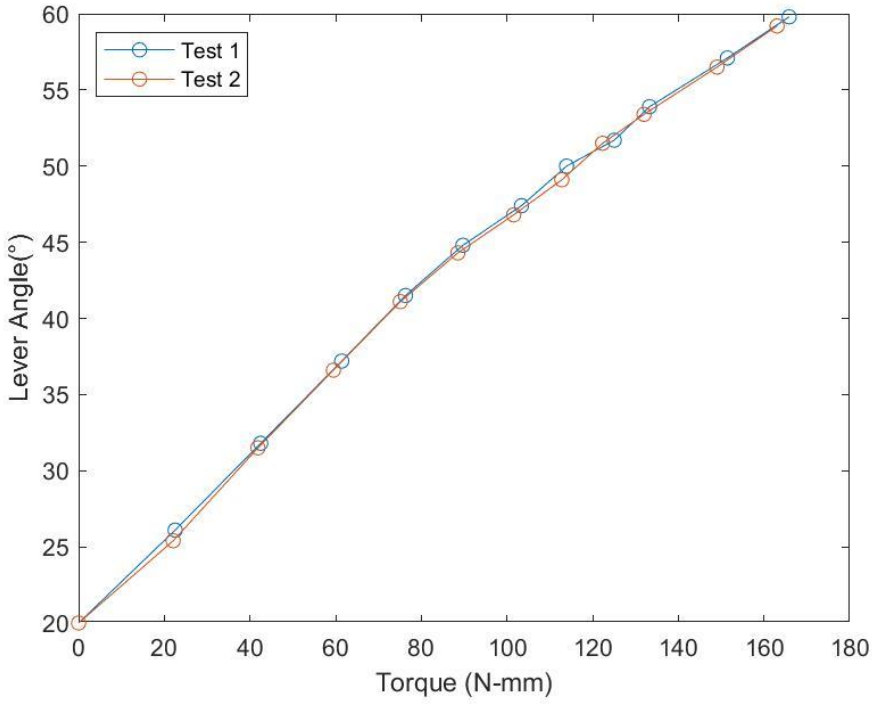


Figure 72: 20 Degree Lever Mechanism 30 Degree Motor Angles

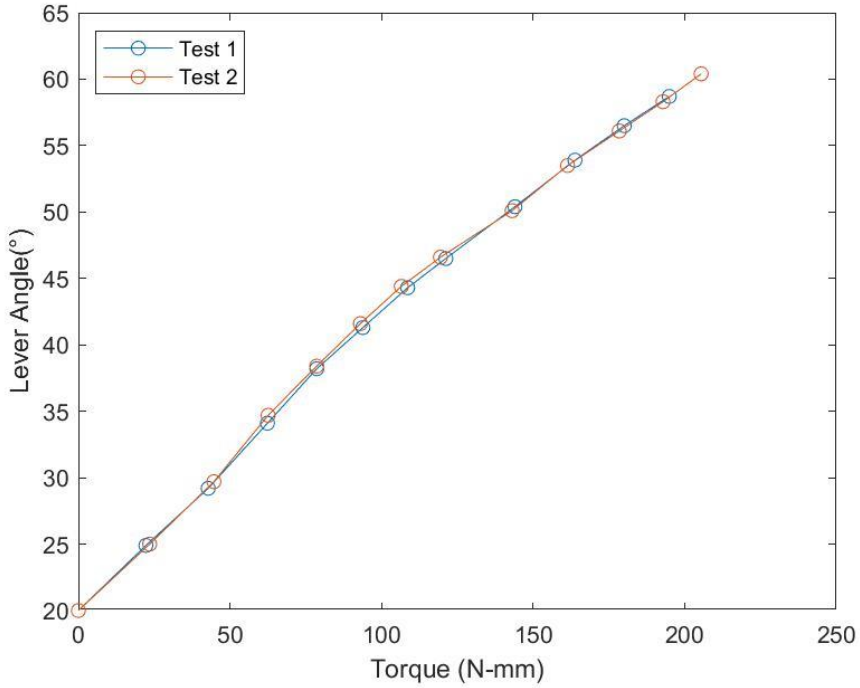


Figure 73: 20 Degree Lever Mechanism 40 Degree Motor Angles

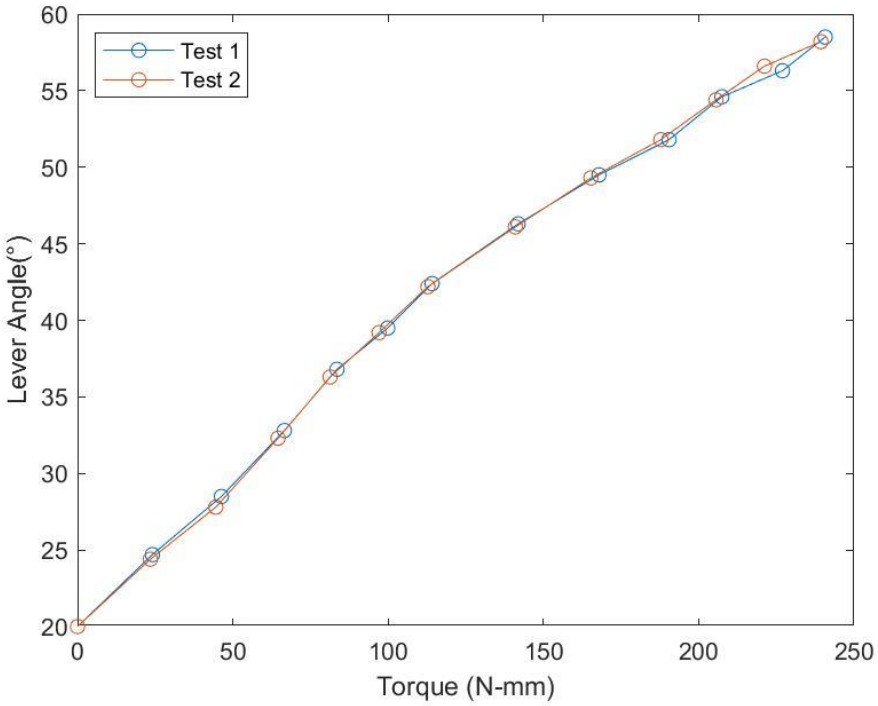


Figure 74: 20 Degree Lever Mechanism 50 Degree Motor Angles

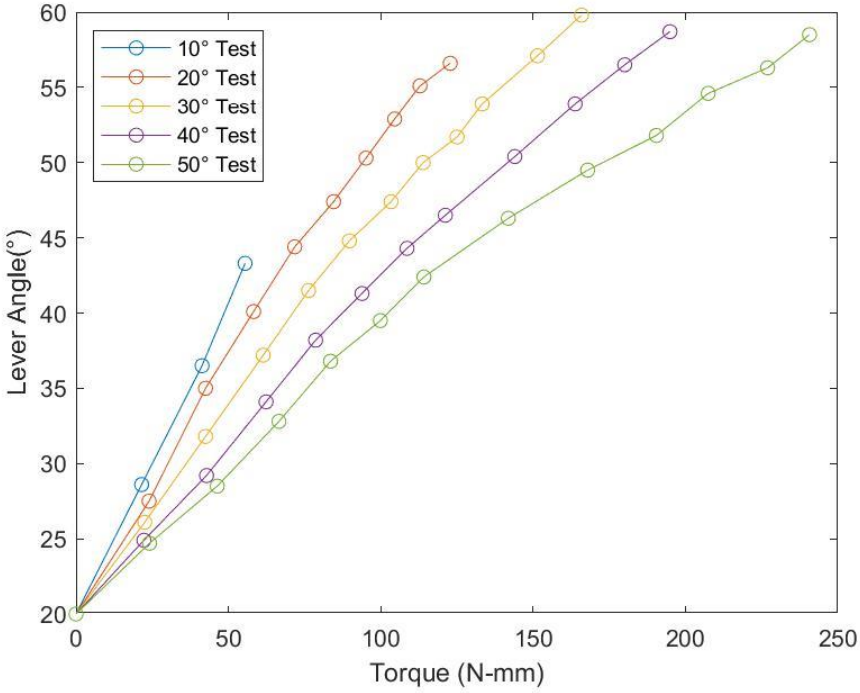


Figure 75: 20 Degree Lever Mechanism Combined Data