Project Number ME-RLN-1102

LINK CONVEYOR MODELING

A Major Qualifying Project Report
Submitted to the Faculty and Staff of
WORCESTER POLYTECHNIC INSTITUTE
for requirements to achieve the
Degree of Bachelor of Science
in Mechanical Engineering Design
by

_____
James Alleva

_____
Kathryn Byorkman

_____
Samantha Dubois

_____
Stephanie Klegraefe

Date: March 2, 2012

Approved:

_____
Professor Norton of the
Mechanical Engineering Department

Executive Summary

The conveyor systems used in manufacturing settings are very complex. These systems need to be analyzed so that their vibrations can be determined and minimized before they are put into production. To do so, it is easier and safer for the system to be tested using a dynamic computer model. To aid the sponsor, the Link Conveyor Modeling Major Qualifying Project was proposed to create this model. The team will use a MATLAB script to create 1, 2, and 85 degree of freedom (DOF) models where the input parameters can be adjusted based on the user's needs. From this model, the sponsor will be able to assess the effects of link geometry, conveyor length, and cam motion profiles. From this, the team will be able to determine the factors that control the dynamics of the conveyor system.

To fully understand the system itself as well as modeling methods and relevant computer programs, the team researched many different topics. Index-dwell conveyor systems are able to index then pause while carrying a payload, allowing changes to be made to the payload. The team will be focusing on a link conveyor system. This conveyor system is comprised of multiple links that travel around a carousel in the manufacturing process. The cams used in precision link conveyor systems are customizable, although generally they are constructed with a 50-50 index-dwell ratio and use a modified sinusoid acceleration curve. This is the most common curve, and provides the lowest practical peak of velocity to the system while still keeping the peak acceleration low. The team researched the following cam profiles: modified sinusoid, modified trapezoid, full cycloid, 3-4-5 polynomial, and 4-5-6-7 polynomial. As a cam is rotated, a dial plate contacts the cam and is moved by it. This movement creates the index and dwell motion of the conveyor. A sprocket is used to transmit rotational motion between two shafts where gears cannot be used. The machine is comprised of 85 links that are joined by pins. The sponsor would like the links to be tested under the assumption that it is constructed from either 6061 T6 Aluminum or Commercially Pure CP-Ti UNS R50400 Titanium. The mass of the nest is assumed to be a constant 0.93 kg while the mass of the links change based on the material chosen for the analysis.

To accurately model the system, it is often convenient to simplify the system into a lumped parameter model where the part is replaced by equivalent point masses connected by massless rods. In the simplest models, all of the masses are "lumped" together and represented by *m*, the stiffness is represented by *k*, and the damping by *c*. Single degree of freedom (SDOF) "lumped" parameter models of cam systems correlate well with experimental data and can be adequate to model most of the dynamic behavior of the system with up to about 10% error. Multiple degree of freedom lumped models serve to take into account more accurately the many components of the actual system.

To create a model of this conveyor system, the team researched the relevant computer programs to aid in the creation of the model. The team used SolidWorks Simulation, Dynacam, and MATLAB. Solidworks Simulation can be used to complete a finite element analysis (FEA) of the link. Dynacam can be used to design cams based on segment conditions and functions of cam motion and will investigate the dynamics of the cam-follower system. There are multiple options for the cam profile functions, including modified sinusoid, modified trapezoid, cycloid, polynomials, and B-splines. Dynacam can export tabulated data of position, velocity, acceleration, and jerk which can be graphed

using another program. In MATLAB, the user can solve mathematical and computational problems, as well as perform modeling, simulation and prototyping, and data analysis.

The problems presented in this project have been faced before by a previous Major Qualifying Project (MQP) group, whose MQP is titled *Analysis of an Index-Dwell Conveyor System* (7). The purpose of the project was to analyze an index-dwell conveyor system and determine its dynamic behavior at various speeds, to create a computer model of the system to look at the behavior of various index-dwell ratios, and to present design recommendations.

The sponsor's report is a study on the accelerations of the sponsor's chain conveyor. Twenty six tests were performed at various locations on the machine. They were performed with an accelerometer mounted on a dummy nest. The tests which are most relevant for the data comparison are taken at four positions: at the drive end, in the middle of the slack side, in the middle of the tight side, and at the idler end. These locations will be analyzed in the computer model as well.

To get an appropriate spring constant for our model, the sponsor provided a CAD model of the link for analysis. In order to ensure the accuracy of our finite element approach, a simple stress problem was set up to get an approximation of the spring constant. The sponsor provided link was then loaded into SolidWorks Simulation for finite element analysis. Using this program, the material could be switched easily to look at both aluminum and titanium links. Switching the material will affect both the mass and the spring constant. Once the material was set, pins were inserted which mimic the bearing constraints. A force was applied pulling the link in the same direction as it moves in the system. Solving the system resulted in a displacement which was used to find an approximate spring constant, for both aluminum and titanium links.

The program Dynacam was used to create the cam profiles of the driving sprocket. The cam profiles were set to index-dwell-index-dwell-fall-dwell in order to simulate two indexes and create a complete cam. The cam data were exported to a text file, which was then read by the MATLAB script. This provides the input for the excitation force into the first link. Using the results from the FEA, the link's mass, spring constant, and estimated damping were set in the MATLAB script. Matrices were created for the mass, spring constant, and damping. The matrices took into account the conveyor connecting the first and last link, as well as how the links affected each other's spring constant. Additionally, the natural frequency for each link is different, which needed to be reflected in the damping matrix.

The MatLab script started analyzing by one and two degree of freedom systems, in order to properly test the input of the Dynacam information. This script was then expanded into the full nDOF model.

After the parameters were set, the script then calculates the excitation force on the first link caused by the sprocket. This is where the input from Dynacam is used, along with the spring constant and the damping from the first link.

Using this data the second order ordinary differential equation for each cam profile was written. The script uses a series of *for* loops and the built-in MATLAB function *ode 45* in order to solve the displacement, velocity, and acceleration for each time step. Also, a frictional value is generated and taken into consideration. In order for *ode 45* to solve the model, the second order system must be converted to an equivalent first order system. The script then outputs a series of graphs which demonstrates all of the cam profiles' effects on the system.

The completed MATLAB scripts were tested to ensure accuracy by running them using the same numbers that were used in generating the Dynacam cam profiles. The 1 DOF script was written to plot the difference in the cam follower and cam motion, displacement of the follower, and velocity of the follower. The plot showed that the MATLAB script was accurate in that the MATLAB results were plotted directly over the Dynacam output. The system was then tested using 2 DOF and the displacement and velocity of both links were plotted. It could be seen that the second link had more vibrations in it than the first link which was expected.

The nDOF MATLAB script was tested for accuracy using 20, 40, and 85 DOF. Each test was done using a frictional coefficient of 0.25 and a damping coefficient of 0.13. The 85 DOF was run using varying friction coefficients of 0.15, 0.25, and 0.35 to determine what effect that would have on the system. The tests were run using Aluminum since that is the material used by the sponsor. An additional test was run at 85 DOF with a friction coefficient of 0.35 using Titanium. This was done as a comparison to see which material produced better results. Each MATLAB script ran calculations for the five different cam profiles simultaneously to ensure that the results were calculated under the same conditions. The results of each test were plotted on multiple figures which showed the displacement, velocity, and acceleration of various links throughout the system. A figure was also created that showed the difference between the calculated MATLAB values for a link in the middle of the tight side of the system and the calculated Dynacam values for each cam profile.

Based on the testing done on the MATLAB model the team determined that the model was accurate. The model outputs were compared to the Dyacam theoretical outputs and were determined to be close and comparable. The model was also tested with RMS values which were taken from Dynacam and the script output. These values were compared using a percent difference and were found to be close and therefore the model could be deemed an accurate portrayal of the system.

The model uses different cam profiles to determine the varying effects that they have on the system. The team wanted to determine which cam profile was the best one for the system. The profiles were ranked based on the difference between the calculated MATLAB values for a link in the middle of the tight side of the system and the calculated Dynacam values. The best cam was determined to be the modified sinusoid cam profile. The modified sinusoid was found to have the smallest change in displacement, velocity, and acceleration as compared to the theoretical outputs from Dynacam.

The displacement is, at maximum, 5mm off from the theoretical. The velocity is at most around 2500mm/s off from theoretical. The acceleration is at maximum 5×10^5 mm/s^2 off from the theoretical. The next three profiles, in order of expanding variance, are the 3-4-5 polynomial, modified trapezoid,

and the full cycloid. Finally, the 4-5-6-7 polynomial showed the most variance, with a maximum displacement of 20mm off from the theoretical.

The friction coefficient was varied to be 0.15, 0.25, and 0.35 for the 85 DOF model to determine if the friction changed the results. It was found that the frictional coefficient had no effect on the model. When the three different frictional coefficient outputs were compared, the graphs for displacement, velocity, and acceleration were the same.

It was found that by changing the material from Aluminum to Titanium, there were less vibrations produced in the system. The maximum differences in displacement, velocity, and acceleration for the modified sinusoid cam profile were found to be 7mm, 2000mm/s, and $5x10^4 mm/s^2$ respectively. Although there was slightly more variance in the displacement, both the velocity and acceleration curves were more stable.

The best cam was determined to be the 85 DOF with a 0.13 damping factor and 0.35 friction coefficient. The 0.35 friction coefficient was used as a base for determining the best cam because it is the aspect ratio that most accurately mimics the tested system. The vibrations are the worst on the middle link but, of all of the cam profiles, the best vibrations are shown with the red color line which is the modified sinusoid cam profile. The modified sinusoid cam had the smallest acceleration curves for 85 DOF. The 4-5-6-7 polynomial cam had the smoothest acceleration curve but also has the highest peak acceleration. The high accelerations that occur using the 4-5-6-7 polynomial cam could be too much for the system; therefore, the team has determined that the modified sinusoid profile is the best cam profile for this system.

When running the script, the DOF was set to 20, 40 and 85. When the script ran at 20 DOF and 40 DOF, the 4-5-6-7 polynomial was the better cam profile. The velocity and acceleration were the smallest on the 4-5-6-7 polynomial for those DOF. Once the system was run with 85 DOF, the better profile turned out to be the modified sinusoid cam.

The team has come up with a few recommendations and possible changes to the project that could improve the performance of the model and other parameters that could be changed. A future study could be the rotational effects of the conveyor system since the current script treats the system as a linear system. An investigation into the effects the temperature may have on the system could be studied to see if the temperature would affect the displacement, velocity, and acceleration of the system.

The spring on the idler pulley should also be taken into consideration in a future script. The current script included the preload force exerted on the links from the spring but does not consider the movement of the spring.

The team determined the best cam to be the modified sinusoid profile. It had the smallest change in the displacement, velocity, and acceleration graphs from the theoretical values from Dynacam. The acceleration is the least smooth on the modified sinusoid but since it is has the smallest accelerations it was chosen as the best profile.

Abstract

The aim of this project is to provide the sponsor with a dynamic computer model that accurately approximates their solid-link, index-dwell conveyor belt and can be adjusted to test alternate scenarios. Using background information including research into index-dwell conveyor systems, mathematical models, relevant computer models, past research, and data provided by the sponsor, the team developed their methodology. The cam profiles were created in Dynacam and the link was tested using finite element analysis. The team then created scripts in MATLAB to solve the differential equations for the one, two, and 85 degree of freedom (DOF) cases. The 1 DOF model was verified using Dynacam and the modified sinusoid cam profile was determined to be the best because the displacement, velocity, and acceleration had the smallest discrepancies from the theoretical for the 85 DOF model.

Authorship

Table of Contents

Table of Figures

Table of Equations

Table of Tables

1 Introduction

The conveyor systems used in manufacturing settings are very complex. These systems need to be analyzed so that their vibrations can be determined and minimized before they are put into production. To do so, it is easier and safer for the system to be tested using a dynamic computer model. To aid the sponsor, the Link Conveyor Modeling Major Qualifying Project was proposed to create this model. The team will use a MATLAB script to create 1, 2, and 85 degree of freedom (DOF) models where the input parameters can be adjusted based on the user's needs. From this model, the sponsor will be able to assess the effects of link geometry, conveyor length, and cam motion profiles. From this, the team will be able to determine the factors that control the dynamics of the conveyor system.

2 Background

To better understand the need for and how to create a dynamic model of an index-dwell conveyor system, information about the system itself as well as modeling methods and relevant computer programs are discussed.

2.1 Index-Dwell Conveyor Systems

Index-dwell conveyor systems are used in many manufacturing plants. They are able to index then pause while carrying a payload, allowing changes to be made to the payload. These machines are popular in many manufacturing plants around the world and are used to make a variety of common items. Because of their popularity, index-dwell conveyor systems come in many versatile designs. The team will be focusing on a link conveyor system. This conveyor system is comprised of multiple links that travel around a carousel in the manufacturing process.

Link conveyor systems are comprised of precision links making a special chain. These links are constructed from a strong metal to reduce vibration and increase precision. They are indexed using a customized cam, based on the buyer's needs. When the cam rotates it engages a roller follower and rotates the chain sprocket through an angle, then dwells, creating the desired index-dwell ratio (see Figure 1) (1).

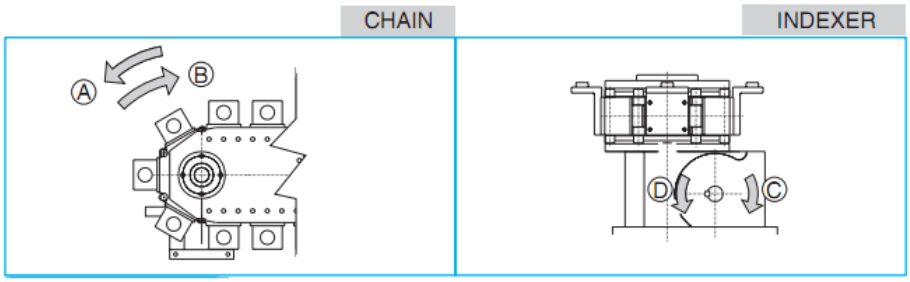

Figure 1: Indexing motion in a precision link conveyor system.

The cams used in precision link conveyor systems are customizable, although generally they are constructed with a 50-50 index-dwell ratio and use a modified sinusoid acceleration curve. This is the most common curve, and provides the lowest practical peak of velocity to the system while still keeping the peak acceleration low. The components of an index-dwell conveyor system are similar in most designs. Descriptions of the components can be seen in this section.

2.1.1 Cams

There are three types of general motions that cams move through, rise-fall (RF), rise-fall-dwell (RFD), and rise-dwell-fall-dwell (RDFD). Rise-fall has no dwell periods; the cam simply rises and falls. Rise-fall-dwell has only a single dwell after the cam has completed its motion. Rise-dwell-fall-dwell has two dwell periods, one after it rises and one after it falls. This is the motion of an indexing cam (2).

A simple harmonic function is able to satisfy the required displacement, velocity, and acceleration profiles. Although it is able to satisfy these conditions, it has infinite jerk because of discontinuities in the acceleration. To solve this, another harmonic function must be added to keep it

continuous. A modified sinusoid wave is made up of pieces of two sinusoidal waves of different frequencies. This wave has no discontinuities in acceleration resulting in finite jerk but higher peak acceleration (Figure 2).

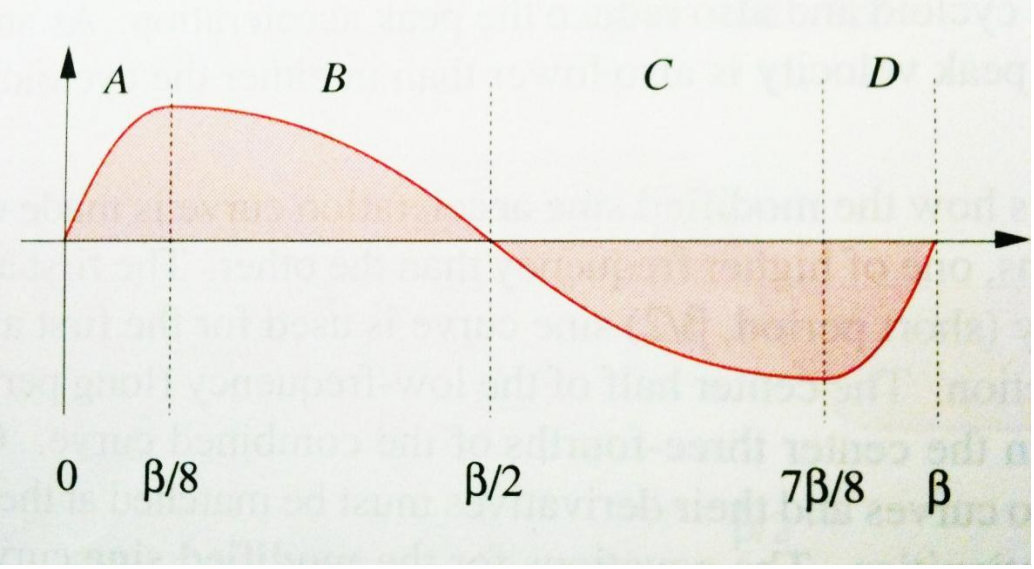

Figure 2: Modified Sine Acceleration (2)

The equations for the displacement, velocity, acceleration, and jerk for each of the three component pieces, section A, sections B and C, and section D are below. In Equation 1, Equation 2, and Equation 3, β is the period and h is the displacement of the function (2).

$$s = h\left(0.43990085\frac{\theta}{\beta} - 0.0350062\sin\left(4\pi\frac{\theta}{\beta}\right)\right)$$

$$v = 0.43990085\frac{h}{\beta}\left(1 - \cos\left(4\pi\frac{\theta}{\beta}\right)\right)$$

$$a = 5.5279571\frac{h}{\beta^2}\sin\left(4\pi\frac{\theta}{\beta}\right)$$

$$j = 69.663577\frac{h}{\beta^3}\cos\left(4\pi\frac{\theta}{\beta}\right)$$

Equation 1: SVAJ for 0≤θ<1/8β

$$s = h\left(0.28004957 + 0.43990085\frac{\theta}{\beta} - 0.31505577\sin\left(\frac{4\pi}{3}\frac{\theta}{\beta} - \frac{\pi}{6}\right)\right)$$

$$v = 0.43990085\frac{h}{\beta}\left(1 + 3\sin(\frac{4\pi}{3}\frac{\theta}{\beta} - \frac{\pi}{6})\right)$$

$$a = 5.5279571\frac{h}{\beta^2}\cos(\frac{4\pi}{3}\frac{\theta}{\beta} - \frac{\pi}{6})$$

$$j = -23.1553\frac{h}{\beta^3}\sin(\frac{4\pi}{3}\frac{\theta}{\beta} - \frac{\pi}{6})$$

Equation 2: SVAJ for 1/8β ≤θ<7/8β

$$s = h\left(0.56009915 + 0.43990085\frac{\theta}{\beta} - 0.0350062\sin\left(2\pi\left(\frac{\theta}{\beta} - 1\right)\right)\right)$$

$$v = 0.43990085\frac{h}{\beta}\left(1 - \cos\left(2\pi\left(\frac{\theta}{\beta} - 1\right)\right)\right)$$

$$a = 5.5279571\frac{h}{\beta^2}\sin\left(2\pi\left(\frac{\theta}{\beta} - 1\right)\right)$$

$$j = 69.663577\frac{h}{\beta^3}\cos\left(2\pi\left(\frac{\theta}{\beta} - 1\right)\right)$$

Equation 3: SVAJ for 7/8β ≤θ≤β

Polynomial functions are another useful family of functions for cam design. The 3-4-5 polynomial is the simplest for the double dwell case. The displacement equation is shown below in Equation 4.

$$s = h(10(\frac{\theta}{\beta})^3 - 15(\frac{\theta}{\beta})^4 + 6(\frac{\theta}{\beta})^5)$$

Equation 4: Displacement Equation for 3-4-5 Polynomial

The jerk is not constrained, resulting in a finite jump in jerk. This is acceptable but not optimal. Another option would be to use a 4-5-6-7 polynomial, whose displacement equation shown below in Equation 5.

$$s = h\left(35\left(\frac{\theta}{\beta}\right)^4 - 84\left(\frac{\theta}{\beta}\right)^5 + 70\left(\frac{\theta}{\beta}\right)^6 - 20\left(\frac{\theta}{\beta}\right)^7\right)$$

Equation 5: Displacement Equation for 4-5-6-7 Polynomial

The 4-5-6-7 polynomial has smoother jerk and better vibration control but has significantly higher peak acceleration than the 3-4-5 polynomial as shown in Figure 3 (2).

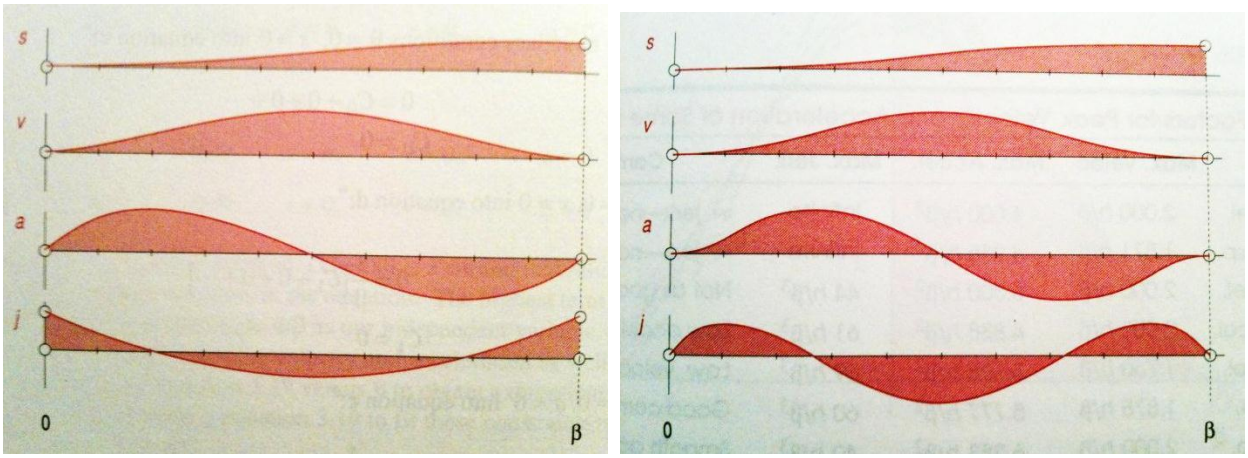

Figure 3: 3-4-5 Polynomial and 4-5-6-7 Polynomial (2)

Another cam profile option would be to use B-splines, a polynomial-like approach to creating cam profiles. A polynomial spline is a motion represented by polynomial pieces blended together at points called knots. Splines are useful because they can form any shape desired, but they are more difficult to comprehend and create because of their complexity.

In precision link conveyor systems, barrel cams are used to index the system. They are ideal because the grooves are cut into the side of a rotating cylinder. These grooves allow for the cam to catch the dial plate and advance it, thus indexing it. Barrel cams are easily customized to perform the desired task. They can be constructed using modified sinusoid curves, polynomial curves, or other curves (3).

2.1.2 Dial Plate

The dial plate is the plate that comes into direct contact with the cam. For the dial plate to be effective, it must be constructed out of a strong metal because of its constant contact with the cam. The size of the plate can also be customized based on its application (3).

2.1.3 Index Drive

A cam-driven indexer is composed of a cam and follower. The cam is attached to an input shaft, more commonly known as a camshaft. Cams are manufactured based on the customer's specifications and are easily configured for individual applications. As a cam is rotated, a dial plate contacts the cam and is moved by it. This movement creates the index and dwell motion of the conveyor.

There are multiple types of indexers used in manufacturing today. The roller gear indexer is designed in such a way that the camshaft is perpendicular to the output shaft. Although the roller gear has a larger cam that may not fit in all packaging, it has more flexibility because of the cam size. Right angle indexers are constructed using cylindrical barrel cams that are placed parallel to the output shaft and perpendicular to the input shaft. This construction allows for a compact design because the cam is partially covered by the indexer wheel. The parallel indexer uses cams that are placed parallel to the output shaft (1).

2.1.4 Sprocket

A sprocket is defined as a “toothed wheel whose teeth engage the links of a chain or a cylinder with teeth around the circumference at either end that project through perforations in something to move it through a mechanism,” (4). A sprocket is used to transmit rotational motion between two shafts where gears cannot be used. The link and sprocket can be chosen by the designer to reduce the amount of destructive vibrations to the system (5). “Sprockets are usually used in bicycles, where the sprocket pulls a linked chain to transform the movement of the rider’s feet into the rotation of the wheels,” (6).

2.1.5 Conveyor Link

The link provided by the sponsor can be seen in Figure 4 below. The machine is comprised of 85 links that are joined by pins. The sponsor would like the links to be tested under the assumption that it is constructed from either 6061 T6 Aluminum or Commercially Pure CP-Ti UNS R50400 Titanium. The mass of the nest is assumed to be a constant 0.93 kg while the mass of the links change based on the material chosen for the analysis. The dimensions used to calculate stiffness are those of the main body, 0.08m x 0.11m x 0.03m.

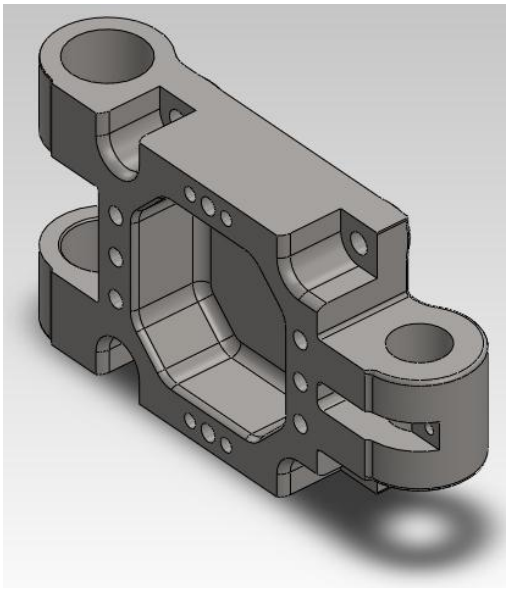

Figure 4: Sponsor Link

2.2 Modeling the System

To accurately model the system, it is often convenient to simplify the system into a lumped parameter model where the part is replaced by equivalent point masses connected by massless rods. In the simplest models, all of the masses are “lumped” together and represented by *m*, the stiffness is represented by *k*, and the damping by *c*. Single degree of freedom (SDOF) "lumped" parameter models of cam systems correlate well with experimental data and can be adequate to model most of the dynamic behavior of the system with up to about 10% error. Multiple degree of freedom lumped

models serve to take into account more accurately the many components of the actual system. Depending on one's needs, different DOF lumped models are appropriate to simulate the dynamic behavior of the system (2).

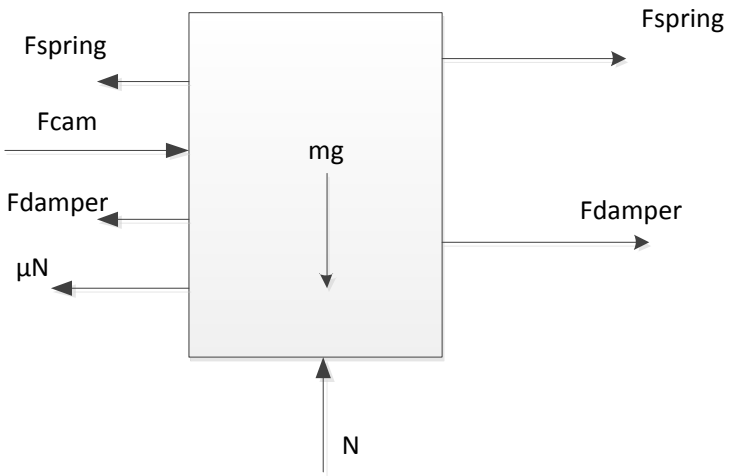

Figure 5: Form Closed Free Body Diagram of a SDOF Cam System

The model is used to determine a simplified equation based on the free-body diagram. For a form-closed cam, pictured above in Figure 5, the equation is shown in Equation 6. A form closed cam was used because it does not allow separation between the cam and the mass.

$$\ddot{x} + \frac{c_1 + c_2}{m}\dot{x} + \frac{k_1}{m}x = \frac{k_1}{m}s + \frac{c_1}{m}\dot{s}$$

Equation 6: Equation of Motion for an SDOF Cam System (2)

This is a second-order, linear, ordinary differential equation (ODE) that can be solved in MATLAB by separating it into two coupled first-order ODEs that can be solved simultaneously. MATLAB uses a 4^{th}-order Runge-Kutta algorithm to solve for x.

The system can be modeled as an nDOF system using the free body diagram seen in Figure 6. A 2224N force is applied to two links, as shown in the free body diagram, to simulate the preload on the idler pulley. The first and last links are also connected by a spring force and a damping force to close the system. This free body diagram is used to calculate the spring constant, mass, and damping matrices that are used to solve the differential equations (Equation 7).

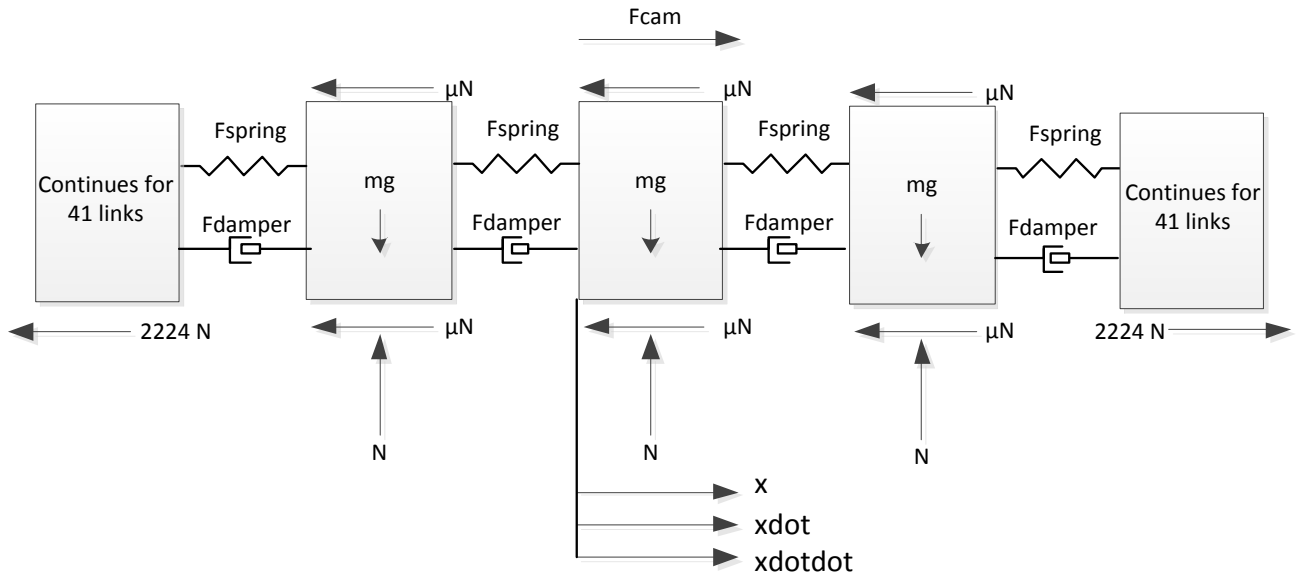

Figure 6: Free Body Diagram of nDOF System

$$[M]\{\ddot{x}\} + [C]\{\dot{x}\} + [K]\{x\} = \{F(t)\}$$

Equation 7: Second Order Differential Equation

2.3 Relevant Computer Programs

To create a model of this conveyor system, the team researched the relevant computer programs to aid in the creation of the model.

2.3.1 SolidWorks

SolidWorks is 3-D Computer-Aided Design (CAD) software that enables to user to create individual parts and assemble them. The parts are generated using mathematical algorithms, but are easily created through a user-friendly interface. The software allows material properties to be applied to each part. With the material selected, SolidWorks can find the mass, density, and other material properties for the given part. The assembly process joins the pieces together, allowing multiple degrees of freedom to be set by the user. This allows the precise control of connecting the assembly and the creation of an animation.

2.3.1.1 SolidWorks Simulation

Finite element analysis (FEA) is a method for solving the resulting deformation and stresses on a loaded part. The basis of the method is to break the system into a mesh of elements and nodes. Generally, FEA approximates a solution but as the number of elements approaches infinity the

approximation converges to the exact result. The method uses partial differential equations and approximates their solutions through ordinary differential equations. The program SolidWorks will be used in order to accomplish this task.

The system needs to be modeled appropriately in the FEA program. For an accurate representation, the material properties, dimensions, and boundary conditions must be specified. With the this information, the program creates a finite element mesh of the system, which is then attached to ground with the appropriate boundary conditions and loaded with forces. Once the program solves the problem, it is able to present the results in a variety of methods which include von Misses stresses, displacement, and temperature.

2.3.2 Dynacam

Dynacam is a program that will design cams based on segment conditions and functions of cam motion and will investigate the dynamics of the cam-follower system. After the input of the segment conditions, Dynacam will solve the kinematic and dynamic equations for a cam-follower system. The followers can be oscillating or translating and be roller or flat-faced. There are multiple options for the cam profile functions, including modified sinusoid, modified trapezoid, cycloidal, polynomials, and B-splines.

Using boundary conditions, the polynomial functions or B-Splines can be calculated by the program. Other variables can be added to the system such as external loads or force closed cams. Pressure angles and radii of curvature are determined by the program, allowing the user to check for possible problems. The program will output the displacement, velocity, acceleration, and jerk functions of the follower. It will also create a one or two mass model of the system for further vibration and dynamic testing.

Dynacam can export tabulated data of position, velocity, acceleration, and jerk which can be graphed using another program. Using the tabulated data and the mass-spring-damper model, the vibrations can be found.

2.3.3 MATLAB

In MATLAB, the user can solve mathematical and computational problems, as well as perform modeling, simulation and prototyping, and data analysis. This program will allow the team to create the model of the system with ordinary differential equations. It will be used to solve each differential equation for the three different degree of freedom models (1, 2, and 85). The use of this program will make the calculations easier for the team and make the model more user friendly. The script written by the team will allow the cam path and dimensions as well as other factors in the system to be changed easily. MATLAB will then solve the problem with the new data and generate the solution much faster than if the entire problem were to be solved by hand. After looking at other options, the team found this to be the best program to use.

2.4 Past Research

The problems presented in this project have been faced before by a previous Major Qualifying Project (MQP) group, whose MQP is titled *Analysis of an Index-Dwell Conveyor System* (7). The purpose

of the project was to analyze an index-dwell conveyor system and determine its dynamic behavior at various speeds, to create a computer model of the system to look at the behavior of various index-dwell ratios, and to present design recommendations.

The conveyor which that team analyzed consists of a horizontal belt made out of polyurethane with steel wire incorporated in it. They also analyzed a part called the dogbone, a piece of plastic that is used to connect the nest to the conveyor belt. The dogbone is designed to break if the nest collides with the tooling carrier (7). To analyze the dogbone, the team loaded it with known weights and measured deflection to calculate an accurate stiffness.

A stiffness test was conducted on the dogbone because the dogbone will fail before the belt does, therefore, it will cause more vibrations to the system. The team calculated the stiffness of the dogbone by hanging weights off of it in order to determine the distance which it moved with each weight. Finite element analysis (FEA) was also used to test the stiffness of the dogbone. To test the stiffness of the belt, the team acquired a section of the belt and placed it in an instron machine (7).

To complete the analysis of the conveyor belt, Pro/ENGINEER was used to model the system and its components. A 61 DOF model was created, which was used to monitor the changes that occurred in the system based on the position of the parts. Using MATLAB, a mathematical representation of the system was derived and then put into Visual Basic and solved (7).

Upon completion of the 61 DOF model, the program “InDwell” was used to solve the differential equations that were created. “InDwell” solved the differential equations at various time steps based on the cam’s displacement in that time frame. Velocity curves for various index-dwell ratios were created using this computer program (7).

After completing the necessary experiments and analyses, it was concluded that the index-dwell ratio of a cam did not make a difference as to how much time the nest was perfectly still during the dwell. A B-Spline cam function was recommended even though it may introduce manufacturing or stress issues. It was also suggested that a stiffer material would result in fewer vibrations within the system (7).

2.5 Sponsor Data

This report is a study on the accelerations of the sponsor’s chain conveyor. Twenty six tests were performed at various locations on the machine. The report breaks the tests into three categories. The first category of tests will be compared to the computer model. They were performed with an accelerometer mounted on a dummy nest. Measurements were then taken at various points during the rotation. These results will be used to validate the computer model.

The tests which are most relevant for the data comparison are taken at four positions: at the drive end, in the middle of the slack side, in the middle of the tight side, and at the idler end. These locations will be analyzed in the computer model as well.

The results of the four tests are shown in Figure 7, Figure 8, Figure 9, and Figure 10. (8)

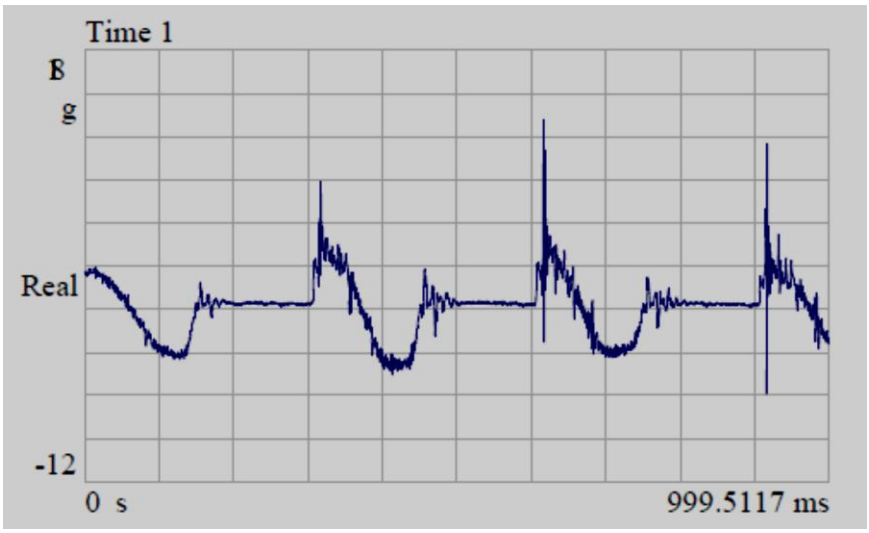

Figure 7 Acceleration around drive end

Figure 8 Acceleration in the middle of tight side

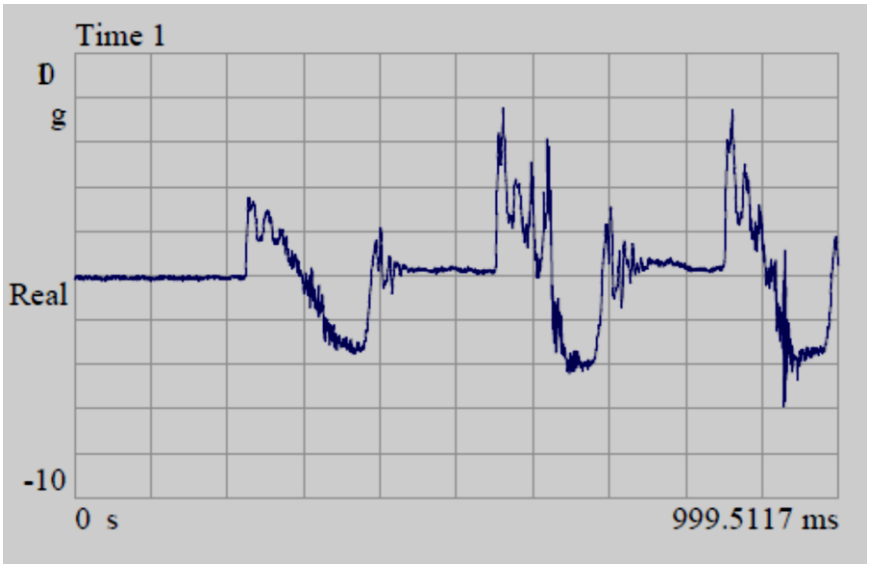

Figure 9 Acceleration around idler end

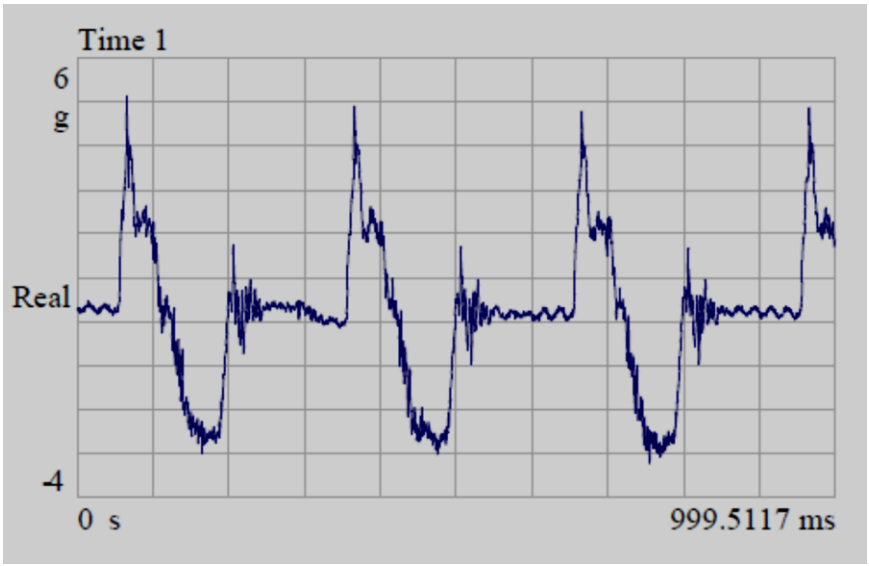

Figure 10 Acceleration in the middle of slack side

Additionally provided with the report was the data used to make the graphs shown. This data will be compared using a root means square method and finding the percent difference from the computer simulation's data.

3 Goal Statement

The goal of this project is to develop a multi-degree of freedom computer model of a link conveyor system that can be used to assess design decisions. This model shall assess link geometries, conveyor lengths, and indexer motion profiles. From this model, the sponsor will be able to determine the factors that control the dynamics of the conveyor system. The model shall also be designed with the ability to change the parameters to test other possible designs.

3.1 Task Specifications

1. Investigate the sponsor's data
2. Find the differential equation that will be used to solve the system
3. Create MATLAB script for mathematical model
    a. Must allow the change of parameters
4. Create a working 1 DOF model of the system
5. Create a working 2 DOF model of the system
6. Create a working n DOF model of the system which will be able to calculate 85 DOF
7. Test various parameters of the model for trends
8. Test various cam profiles to determine the one with the least vibrations
9. Conclude with design recommendations

4 Methodology

To complete the project and create the model there are a number of computer programs that will be used. The first program is MATLAB. MATLAB will be used to create the actual mathematical model. The equations that will be used are found in *Cam Design and Manufacturing Handbook* (2). The model will be created using one degree of freedom, then two, and finally eighty-five.

The equation to model the system is a second-order, linear ordinary differential equation but the MATLAB software cannot solve second order equations. To solve this equation using the program, the script needs to trick the software into thinking that the equations are first order differential equations. By doing this the program believes that it is solving two first order differential equations and will therefore solve the model to provide the state-space solution.

The model involves the cam profile which will be imported into MATLAB from the program Dynacam. An approximation of the cam used in the machine will be created in Dynacam so that an analysis of the cam may be done and the cam data can be imported into the model for analysis.

The team will use SolidWorks to perform an FEA analysis on a computer model of the link that was provided by the sponsor. From the CAD model, the team can find the mass and mass moment of inertia which are the basis for the FEA analysis.

4.1 Finite Element Analysis of the Link

The finite element analysis of the link was done using both hand calculations and computer simulations. This was done to ensure that the computed solutions were correct, and the process can be seen below.

4.1.1 Hand Calculations

SolidWorks simulation will be used in order to find an appropriate spring constant for the link. A quick calculation of the spring constant for a rectangular piece of Aluminum 6061 with the dimensions of the link is shown in Equation 8. The same calculations were done for the rectangular piece of Titanium and are shown below in Equation 9. The cross-sectional area used consists of the thinnest section of the link, the area with the recess. The length used is measured from pin to pin. The modulus of elasticity for Aluminum 6061 is 68.9 GPa while the modulus of elasticity for Titanium is 105 GPa. These calculations will be used to validate our FEA solution for this problem.

$$k = \frac{AE}{l} = \frac{(.006858m * 0.07493m)(68.9 * 10^9 \frac{N}{m^2})}{0.1125m} = 3.147 * 10^8 \ \frac{N}{m}$$

Equation 8: Stiffness Calculation for Aluminum

$$k = \frac{AE}{l} = \frac{(.006858m * 0.07493m)(1.05 * 10^{11} \frac{N}{m^2})}{0.1125m} = 4.796 * 10^8 \ \frac{N}{m}$$

Equation 9: Stiffness Calculation for Titanium

4.1.2 Computer Calculations

The sponsor provided the CAD model of the link which was used for the FEA. Once the link was loaded into SolidWorks Simulation, material properties were specified. The sponsor required both aluminum and titanium links to be analyzed. The two important factors that change with material selection were spring constant and mass. The masses of the aluminum and titanium links are 1.10 kg and 1.35 kg, respectively. In order to determine the spring constant, boundary conditions and loads need to be specified.

The spring constant needs to be measured along the length of the link, so the boundary conditions and loads need to reflect that. The fully constrained and loaded links are shown in Figure 11 and Figure 12, reflecting the aluminum and titanium links respectively. Pins were inserted into the model to simulate the bearing load and constraints. The left pin has a fixed constraint and the right pin has a 100N bearing load placed on it. The simulation software uses a finite element method which turns the link into a mesh of elements connected by nodes. The simulation results shown in Figure 11 and Figure 12 also display the deflection of the link, which will be used to determine the spring constant.

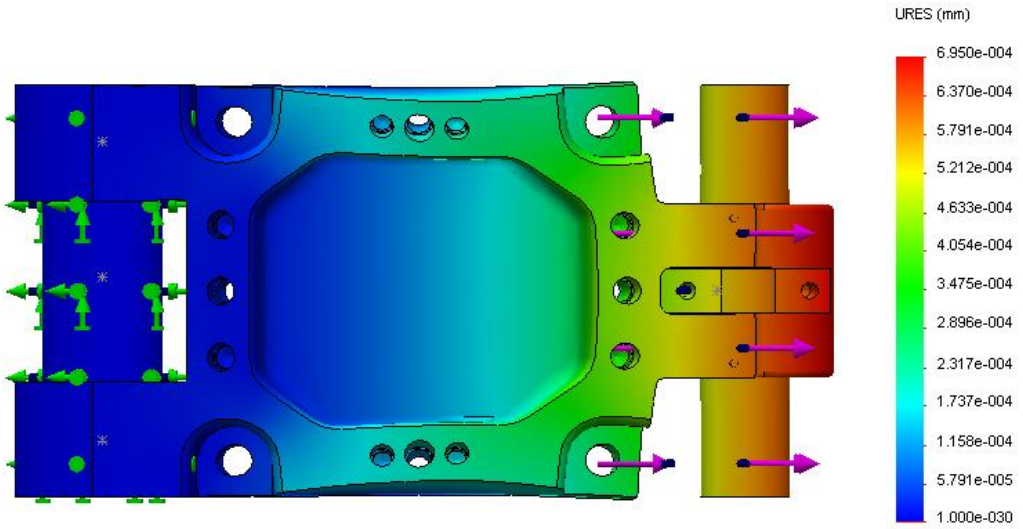

Figure 11: SolidWorks Simulation Aluminum Link Displacement

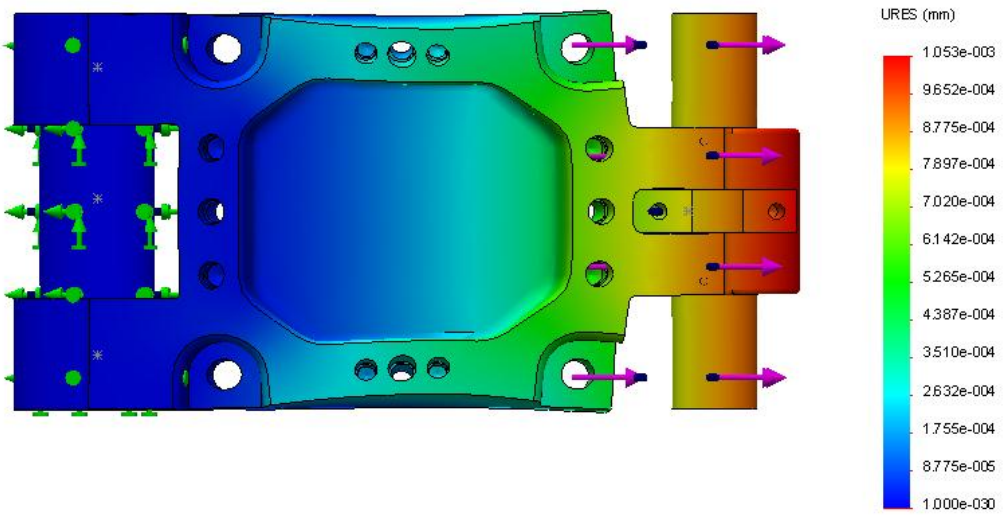

Figure 12: SolidWorks Simulation Titanium Link Displacement

4.2 Creating Cam Profile

After obtaining information from the sponsor, the final cam function was created in Dynacam. In order to imitate the infinite positive displacement cam, the cam function has a rise, dwell, rise, dwell, fall, dwell series. The first 180^{o} shows two cycles of the actual cam. The final 180^{o} is a return to zero so the cam is complete as required by program Dynacam. The team is only using the data from the first 180^{o} because an indexing cam has constant positive displacement. The actual cam gives a repeated set of rises as the cam picks up successive follower notches on the sprocket wheel. On the SVAJ tab, the appropriate displacement, dwell, and rotational speed were entered (see Figure 13). A modified sinusoid function was used to model the displacement. Dynacam's Runge-Kutta method of solving the cam profile requires the function to complete one full cycle before accurate results can be obtained. This is due to the method "guessing" at the appropriate solution until it converges to an exact solution.

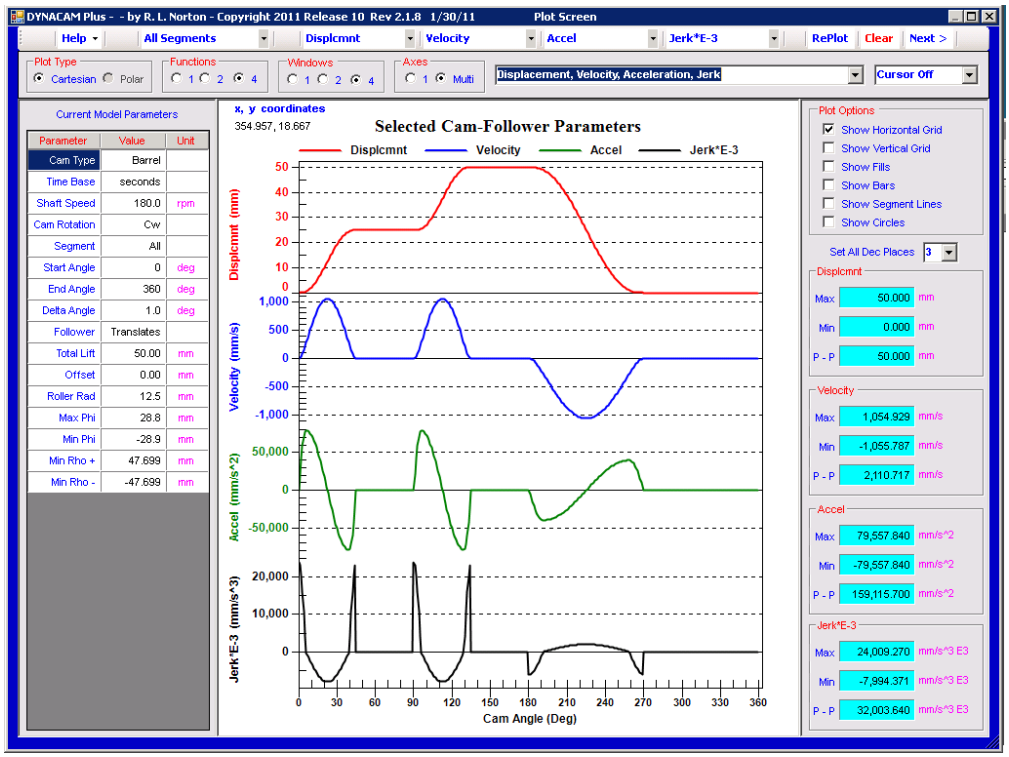

Figure 13: Displacement of test cam.

The dynamics calculation is important in regards to the MATLAB script. In this calculation, the mass, spring constant, damping, and preload were input in order to find the kineostatic force and torque in the system. Along with this data, the natural frequency and critical damping were calculated. These will be important for the vibrational effects of the system. The final step in Dynacam was to calculate the vibrations which tested the validity of the MATLAB code (see Figure 14). Before the vibrations could be calculated, the model of the total system was selected. This model represented the form-closed 1 DOF system. The variables k_1, k_2, c_1, c_2, m_1, and m_2 are the same inputs as those in the MATLAB script. The vibration calculations were done using a spring constant of 1,090,000 N/m. This was done because Dynacam was not able to compute the results based on the actual spring constant of the conveyor link as the value was too stiff and the Runge-Kutta algorithm could not converge.

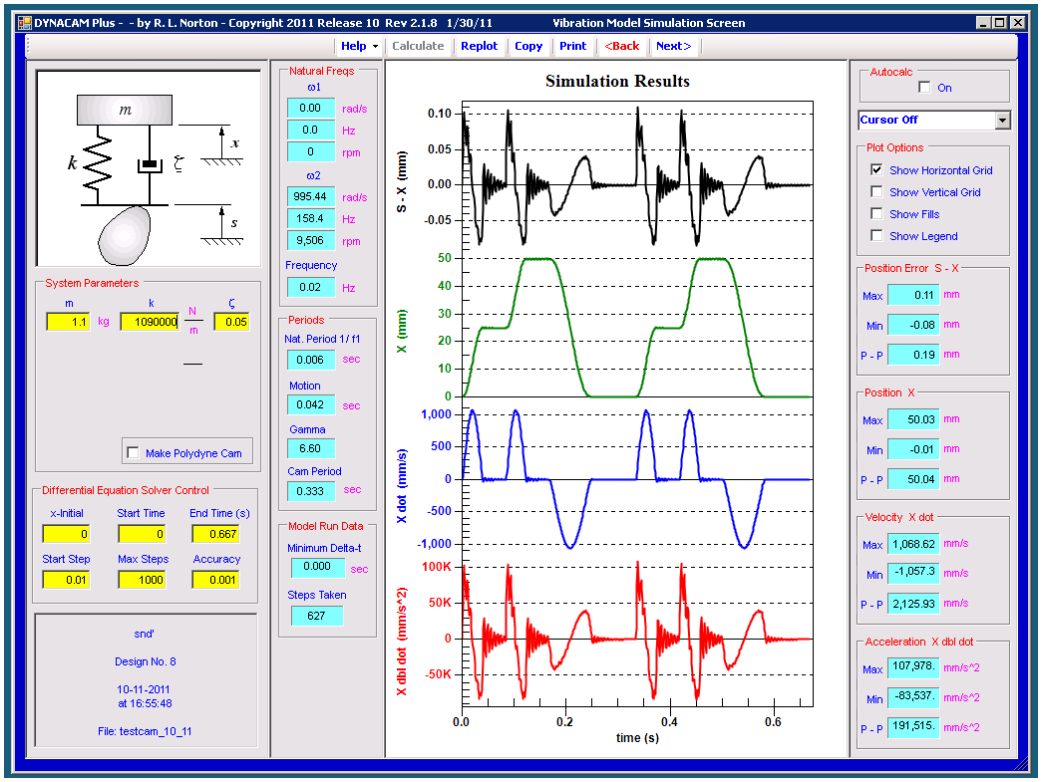

Figure 14: Vibrations tab for the test cam.

4.2.1 Creating Different Cam Profiles

To determine if different cam profiles would reduce the vibrations, the team created several different cam profiles in Dynacam and ran the MATLAB script to determine the reactions in the links. Each of these different profiles generates their own reactions as their displacement, velocity, acceleration, and jerk act differently and thus the vibrations change. The cam profiles examined were modified sinusoid, modified trapezoid, full cycloid, 3-4-5 polynomial, and 4-5-6-7 polynomial. All the cams have a speed of 180 RPM, two rise periods which have a displacement of 112.5 mm each and last for 45°, while each dwell lasts for 45°. The fall is 225 mm and lasts for 90° while the final dwell also lasts 90°.

4.2.2 Exporting Cam Profile Data

The cam profiles are the input into the MATLAB script. In order for different cam profiles to be analyzed in the computer model, they must be exported from Dynacam. The Dynacam solution was exported into a text file. It was necessary to remove the unneeded data and keep the displacement, velocity, acceleration, and jerk. Additionally, for the one DOF model it was necessary to keep the displacement, velocity, and acceleration of the follower. These are not needed by the multiple DOF models. Finally, only the first 180 degrees of the cam profile data were kept due to the creation of the cam profiles. The second 180 degrees was the return to zero, which was only needed for the Runge-Kutta solution to the cam profile. The text file is read by the MATLAB script as the input for the conveyor system.

4.3 Sprocket

The system has a sprocket on the drive end which holds six links. Originally the team decided that because the sprocket holds six links, it would act as one effective link of six times the mass of one link. This would have made the final DOF of the system 79 instead of 85. After consideration, the team decided that instead of lumping the first six links, they would be treated individually. This was done because it was determined that the links are already rigid in their configuration in the chain system. By lumping them together in the sprocket, there would not be any effect on the vibrations produced by the system.

4.4 Solving Differential Equations

To solve the differential equation in Equation 10, the team replaced the independent variables with the variables shown in Equation 11.

$$\ddot{x} + \frac{c_1 + c_2}{m}\dot{x} + \frac{k_1}{m}x = \frac{k_1}{m}s + \frac{c_1}{m}\dot{s}$$

Equation 10: Equation of Motion for an SDOF Cam System

$$x_2 = \dot{x}$$

$$x_1 = x$$

Equation 11: Replacement Equation

When Equation 11 is substituted into Equation 10, Equation 12 is formed.

$$\dot{x_2} + \frac{c_1 + c_2}{m}x_2 + \frac{k_1}{m}x_1 = \frac{k_1}{m}s + \frac{c_1}{m}\dot{s}$$

Equation 12: SDOF Equation used in MATLAB

Equation 12 was then coupled with Equation 13.

$$x_2 = \dot{x}_1$$

Equation 13: Coupled Equation

MATLAB solved Equation 12 and Equation 13 simultaneously to provide values of *x* based on the cam profile *s* and velocity $\dot{s}$. For larger DOF models, the masses, spring constants, and damping factors are arranged in a matrix so that the equation can be solved for multiple DOFs using the state-space method.

4.4.1 Converting the Second Order System into an Equivalent First Order System

For MATLAB to be able to solve the second order system, Equation 14 , it must be converted to an equivalent first order system, Equation 15.

$$[M]\{\ddot{x}\} + [C]\{\dot{x}\} + [K]\{x\} = \{F(t)\}$$

Equation 14: Second Order Differential Equation

$$\frac{d}{dt}\begin{Bmatrix} x_1 \\ \vdots \\ x_n \\ \dot{x}_1 \\ \vdots \\ \dot{x}_n \end{Bmatrix} = \begin{bmatrix} 0_{(nxn)} & I_{(nxn)} \\ -M^{-1}K & -M^{-1}C \end{bmatrix} \begin{Bmatrix} x_1 \\ \vdots \\ x_n \\ \dot{x}_1 \\ \vdots \\ \dot{x}_n \end{Bmatrix} + \begin{Bmatrix} 0 \\ \vdots \\ 0 \\ F_1 \\ \vdots \\ F_n \end{Bmatrix}$$

Equation 15: Equivalent First Order System

This is represented in the MATLAB script by Equation 16.

$$\frac{d}{dt}\begin{Bmatrix} x_1 \\ \vdots \\ x_n \\ x_{n+1} \\ \vdots \\ x_{2n} \end{Bmatrix} = \begin{bmatrix} 0_{(nxn)} & I_{(nxn)} \\ -M^{-1}K & -M^{-1}C \end{bmatrix} \begin{Bmatrix} x_1 \\ \vdots \\ x_n \\ x_{n+1} \\ \vdots \\ x_{2n} \end{Bmatrix} + \begin{Bmatrix} 0 \\ \vdots \\ 0 \\ F_1 \\ \vdots \\ F_n \end{Bmatrix}$$

Equation 16: Equivalent First Order System as Represented in the MATLAB Script

This process uncouples the differential equations, allowing MATLAB to solve for the displacement of each link for the desired cam input using ode45.

4.5 Creating MATLAB Models

To create an nDOF MATLAB script, it was first necessary to validate a 1 DOF model against the program Dynacam's calculations. Next this 1 DOF model was expanded to a 2 DOF model to introduce the concept of matrices into the system. The results of that model were compared to the 1 DOF results to ensure accuracy. The 2 DOF model was then altered to create an nDOF model which allows the user to input the number of degrees of freedom that they would like to analyze. As the focus of this project is on the nDOF model, the description of that model is more detailed than the 1 DOF and 2 DOF models.

4.5.1 MATLAB Script for 1 DOF Model

Using the spring-mass-damper system as a foundation for the MATLAB functions, a 1 DOF script is written in MATLAB. The MATLAB script runs a Runge-Kutta fourth-order differential equation solver. The script is written to calculate the displacement, velocity, and acceleration of the indexing cam for 1 DOF. The cam profile that was generated by Dynacam was imported into the MATLAB script for reference.

The first step is setting the system's parameters including the mass, spring constant, and damping coefficients. These must be the same as the parameters used in Dynacam, for the purpose of comparison. A forcing function is written using Equation 12 to excite the system. This function was referenced in the MATLAB script and used to calculate the displacement, velocity, and acceleration of the cam. Then the ODE needs to be solved for one revolution of the cam. The script uses MATLAB's "ode45" function and a *for* loop to solve the ODE at each step. To ensure that the script is correct, the Dynacam data is plotted along with the calculated MATLAB data as a comparison of the MATLAB's accuracy.

4.5.2 MATLAB Script for 2 DOF Model

The MATLAB script that was created to model the 1 DOF system was altered to calculate a 2 DOF system. To do this, the 1 DOF script was modified to include matrices instead of constants (see Equation 17). Three matrices were introduced, one for the damping, mass, and spring constant. These are square matrices that have two by two dimensions to accommodate for each link. The mass matrix is structured so that the mass of each link is on the diagonal, while the spring constant and damping matrices are structured differently. Due to the nature of damping, the matrix needs to be constructed in such a way that the damping of one link is able to affect the damping of another (see Equation 18). The same matrix configuration is used to calculate the spring constant of each link.

$$\dot{x_2} + [C][M]^{-1}x_2 + [K][M]^{-1}x_1 = [K][M]^{-1}s + [C][M]^{-1}\dot{s}$$

Equation 17: nDOF Equation used in MATLAB

$$\begin{bmatrix} c_1 + c_2 & -c_1 \\ -c_1 & c_1 + c_2 \end{bmatrix}$$

Equation 18: Damping 2 DOF Matrix

The script reads the cam data from the Dynacam file for both the SVAJ of the cam and the cams dynamic response. The parameters are then put in which include the mass, spring constant and damping. The mass, damping and spring constant matrices are then defined. The script then determines the excitation to determine the force of the spring. The ODE defined in the function is then solved using a *for* loop. The plots are finally output to show the responses of the system.

4.5.3 Creating Multiple DOF Models in MATLAB

To simulate the vibrations produced in each link in the model, an nDOF model was created in MATLAB. This model allows the user to enter in the number of links that they would like to be tested. The script then uses loops and ordinary differential equations to calculate the vibrations in the multiple links. Since the script calculates the displacement, velocity, and acceleration of the links in each cam profile simultaneously, each cam profile was assigned a number that was added onto each variable for distinction in the calculations.

To begin the MATLAB script, the user must define the number of degrees of freedom, or nDOF, that are being looked at. The data from each cam profile is then imported and the displacement, velocity, and acceleration are read individually. Once the data is imported the mass of each link, spring constant, damping coefficient, and friction coefficient of each link are entered. Since the links are identical, there is only need to have one value for each property. To calculate the damping that is imparted on the machine, the damping coefficient is multiplied by two, the natural frequency, and the mass of the link.

To determine the time parameter of the model, the speed of the cam was used. The speed used is 180 RPM which is also the speed used in the Dynacam model of each cam profile. The amount of time that it would take to go through one degree of motion is calculated using Equation 19. The time matrix was then generated by multiplying each degree of motion by *dt*.

$$dt = \frac{1}{\frac{WCam/60}{360}}$$

Equation 19: Time Difference Calculation

To calculate the displacement, velocity, and acceleration of the multiple links, matrices are used as seen in Equation 17. The matrices were generated in a similar fashion using nDOF by nDOF identity matrices for each mass, damping, and spring matrix. Since the mass of each link does not affect the link next to it, the mass identity matrix was multiplied by the mass value which was input at the beginning of the script. Generating the damping and spring matrices is slightly more complicated.

Because each link is in series, the damping and spring of one link has an effect on the link next to it. It is also important to remember that the links are being modeled as a conveyor belt and that the first link is connected to the last link. The identity matrix for the spring constant was multiplied by the constant that was input earlier. The matrix is then adjusted for each link to account for the forces that are acting on it due to the other links. The final matrix for the spring constant is shown in Equation 20. This matrix can be expanded to nDOF by repeating the same pattern of addition.

$$\begin{pmatrix} k_1 + k_2 + k_5 & -k_1 & 0 & 0 & -k_5 \\ -k_1 & k_2 + k_3 & -k_2 & 0 & 0 \\ 0 & -k_2 & k_3 + k_4 & -k_3 & 0 \\ 0 & 0 & -k_3 & k_4 + k_5 & -k_4 \\ -k_5 & 0 & 0 & -k_4 & k_5 + k_1 \end{pmatrix}$$

Equation 20: Spring nDOF Matrix

Each link has a different damping acting on it because the natural frequency of each link is different. To calculate the damping imparted on each link, a diagonal matrix of eigenvalues for the mass and spring matrices was produced. The square root of that diagonal was taken to obtain the natural frequency of each link. This value was multiplied by two, the damping coefficient, and the mass of a link, creating the proportional damping vector. This vector was then used to generate the damping matrix seen in Equation 21. Like the spring constant matrix, this matrix can be increased to nDOF by repeating the pattern of addition.

$$\begin{pmatrix} c_1 + c_2 + c_5 & -c_1 & 0 & 0 & -c_5 \\ -c_1 & c_2 + c_3 & -c_2 & 0 & 0 \\ 0 & -c_2 & c_3 + c_4 & -c_3 & 0 \\ 0 & 0 & -c_3 & c_4 + c_5 & -c_4 \\ -c_5 & 0 & 0 & -c_4 & c_5 + c_1 \end{pmatrix}$$

Equation 21: Damping nDOF Matrix

The inverse mass matrix is then calculated and given its own label to save time in further calculations done in the MATLAB script. These matrices are generic and are used in each of the cam profile calculations.

The excitation of the first link for each cam profile is then calculated. This is calculated using Equation 22 below where *c* is the damping of the first link, *z1dot* is the velocity input from Dynacam, *k* is the spring constant, and *z* is the displacement input from Dynacam. The initial conditions for each cam profile are then set.

$$f = c * z1dot + k * z$$

Equation 22: Excitation Force

The ordinary differential equation for each cam profile is then written. These equations use *ode 45* and *for* loops to calculate the displacement, velocity, and acceleration for each link in the system. Each *for* loop is set to run for the designated number of time steps minus one so that each link's values are calculated at each time step. Within each *for* loop is a function equation that is customized for the various cam profiles. The loop is used to interpolate the points between each link and to calculate the force matrix. The loop then uses the function imbedded in it to calculate the displacement of each link using the force matrix. The velocity of the links are then calculated using the previously generated matrices and an additional frictional value that is generated using Equation 23 where *m* is the mass, *g* is the gravitational force, *b* is the friction coefficient, and *x* is the displacement that was just calculated. The velocity of the links is calculated, and then the frictional force is subtracted from each link's from it to obtain the final velocity. The acceleration is calculated in the MATLAB script using the same matrices as used in the displacement calculation.

$$Fr = m * g * b * sign(x)$$

Equation 23: Friction Calculation

The RMS values of the link a quarter of the way in on the tight side and the Dynacam data are then calculated. These RMS values are exported to a text file so they can be compared. The maximum acceleration of each link is then calculated for each cam profile. These values are also exported into a text file for analysis. To compare the data more accurately, the difference between the calculated vales of a link one quarter of the way down the tight side and the Dynacam data is calculated.

Upon completion of the calculations, eight figures are produced. One which plots the displacement, velocity, and acceleration of three links, the first, middle, and last, of all the cam profiles. Another figure plots only the results from the middle link. One figure was created for each cam profile to plot the comparison of the results from the middle link of each cam to its respective Dynacam profile. While the last figure plots the difference between the calculated values of the middle link and the Dynacam profile for each cam profile. The script is also written to determine the RMS values for the middle link on the tight side of each cam profile and the maximum acceleration of each cam profile.

To test the scripts to ensure that they were working, the team ran them at varying DOFs. First the team tested 1 DOF and obtained accurate results. The team then tested two, five, twenty, forty, and eighty-five DOFs and obtained the same results. The complete script can be seen in Appendix C: n Degree of Freedom MATLAB Script.

5 Results

The following section details the results of the computer simulations.

5.1 SolidWorks Stiffness Testing

The spring constant of each link was obtained using SolidWorks' FEA program. The links were loaded with a force and the displacement was measured. Bearing load applied to the aluminum and titanium links was a 100 Newton force. The displacements were 6.950 * 10^4 mm for the aluminum link and 1.053* 10^3 mm for the titanium link. The resulting spring constant for the aluminum link is 9.270*10^7 N/m shown in Equation 24. While the resulting spring constant for the titanium link is 1.439* 10^8 N/m as shown in Equation 25

$$k = \frac{Force}{Displacement} = \frac{100N}{6.950 * 10^4 mm} * \frac{1000mm}{1m} = 9.270 * 10^7 \frac{N}{m}$$

Equation 24 Spring Constant Evaluation for Aluminum Link

$$k = \frac{Force}{Displacement} = \frac{100N}{1.053 * 10^3 mm} * \frac{1000mm}{1m} = 1.439 * 10^8 \frac{N}{m}$$

Equation 25 Spring Constant Evaluation for Titanium Link

5.2 Model Testing

To check that the MATLAB code was accurately simulating the behavior of the system the code needed to be tested. This was done with using the data imputed from the program Dynacam and by adjusting numbers and checking if the output behaved the way it should. The testing will be explained for each different DOF computer model.

5.2.1 1 DOF Testing

The 1 DOF model was tested by running it using the same input values that were used in the creation of the cam profile in the program Dynacam. The results from the MATLAB script were then plotted with those from Dynacam and can be seen below in Figure 15. As can be seen by the figure, the script is correct since the Dynacam results and MATLAB results are nearly identical. For the complete 1 DOF script, see Appendix A: 1 Degree of Freedom MATLAB Script.

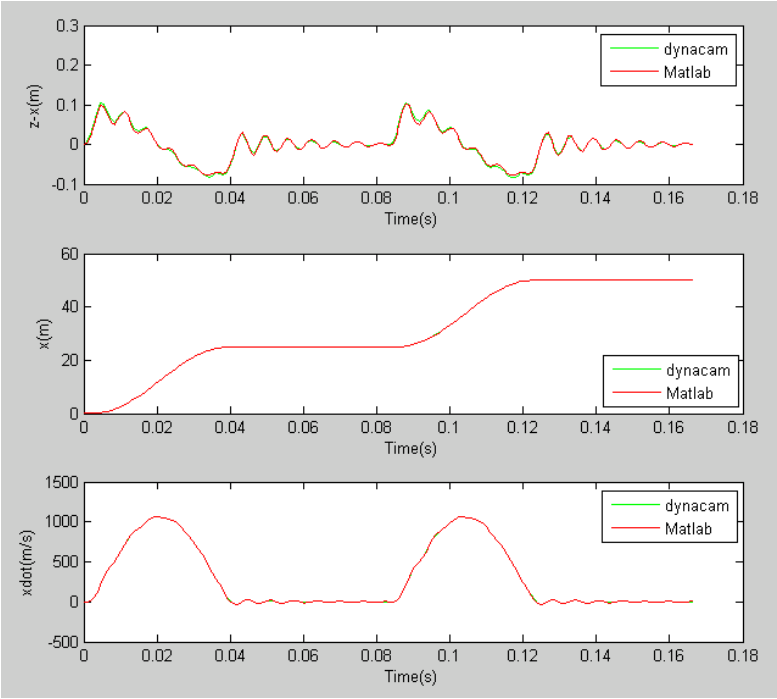

Figure 15: 1 DOF MATLAB Output

5.2.2 2 DOF Testing

The 2 DOF model was tested in a similar fashion to that of the 1 DOF model. The output of the script is shown in Figure 16 below. This output graphs the displacement and velocities of each conveyor link. It can be seen that there are more vibrations in the second link than in the first link. The full script is shown in Appendix B: 2 Degree of Freedom MATLAB Script.

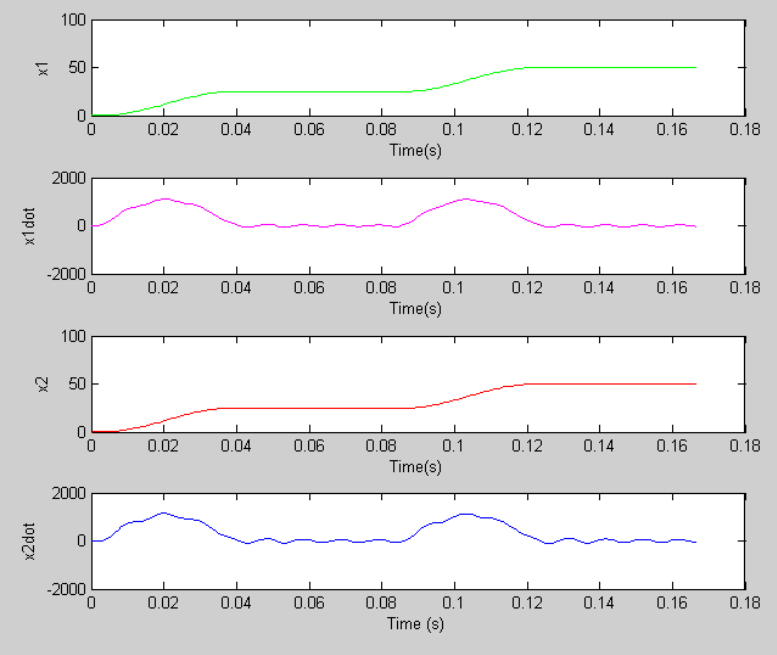

Figure 16: 2 DOF MATLAB Output

5.2.3 nDOF Testing

To test the performance of the nDOF model, it was run four different times using varying degrees of freedom and friction coefficients. The damping coefficient was kept at 0.13 because that was the damping coefficient provided by the sponsor.

The initial test of the MATLAB script used 20 DOF and a friction coefficient of 0.25. The results are shown in Figure 17. This figure shows the displacement, velocity, and acceleration for three links, Link 1, Link 10, and Link 20. Additional figures from this test can be seen in Appendix D: Additional Figures for 20 DOF Testing.

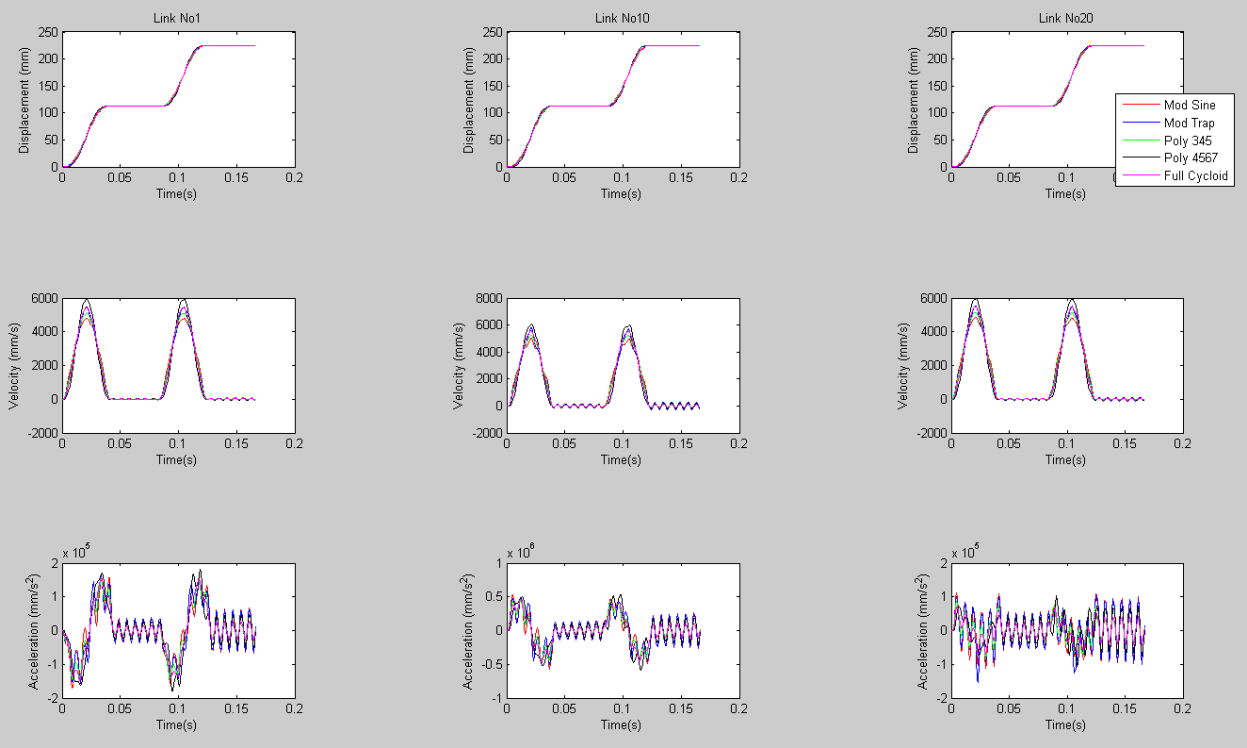

Figure 17: 20 DOF MATLAB Test

The second test of the MATLAB script used 40 DOF and a friction coefficient of 0.25. The results are shown in Figure 18. This figure shows the displacement, velocity, and acceleration of three links, Link 1, Link 20, and Link 40. Additional figures for this test can be seen in Appendix E: Additional Figures for 40 DOF Testing.

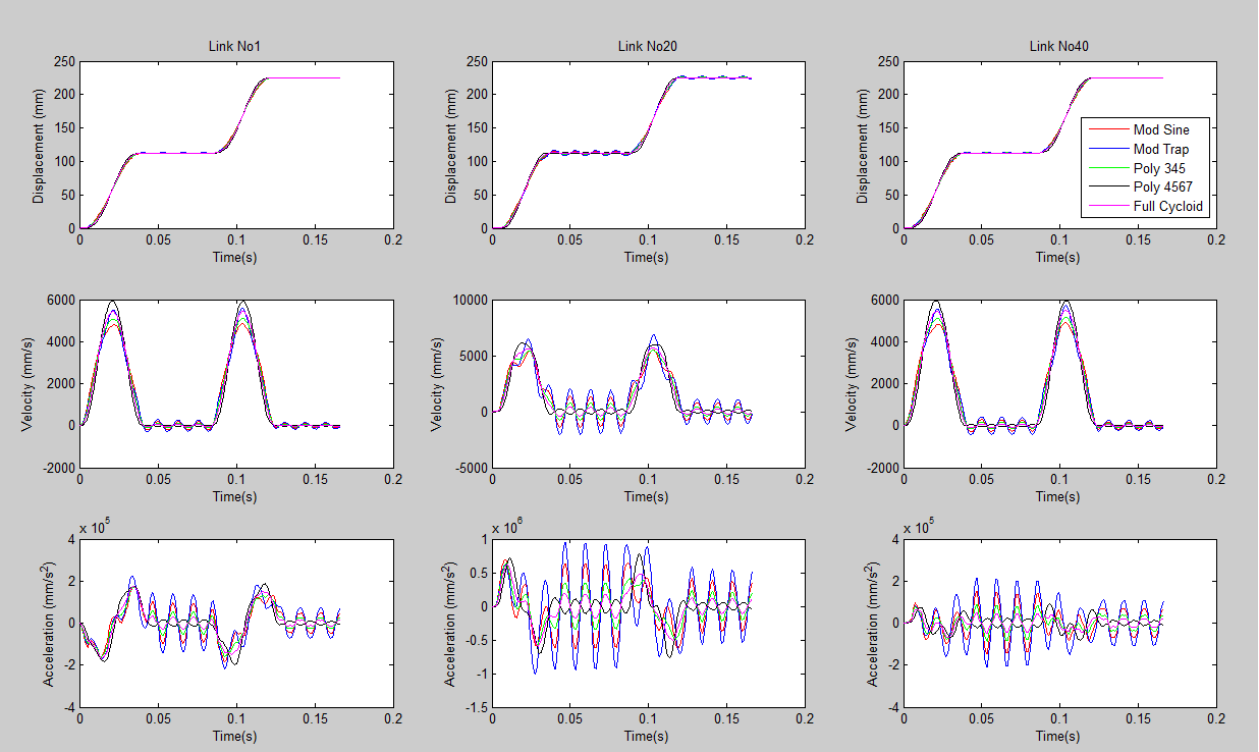

Figure 18: 40 DOF MATLAB Test

The script was then run three times at 85 DOF using varying coefficients of friction with values of 0.15, 0.25, and 0.35. The figures can be seen in Figure 19, Figure 20, and Figure 21 respectively. Additional graphs can be seen in Appendix F: Additional Figures for 85 DOF and Frictional Coefficient of 0.15, Appendix G: Additional Figures for 85 DOF and Frictional Coefficient of 0.25, and Appendix H: Additional Figures for 85 DOF and Frictional Coefficient of 0.35.

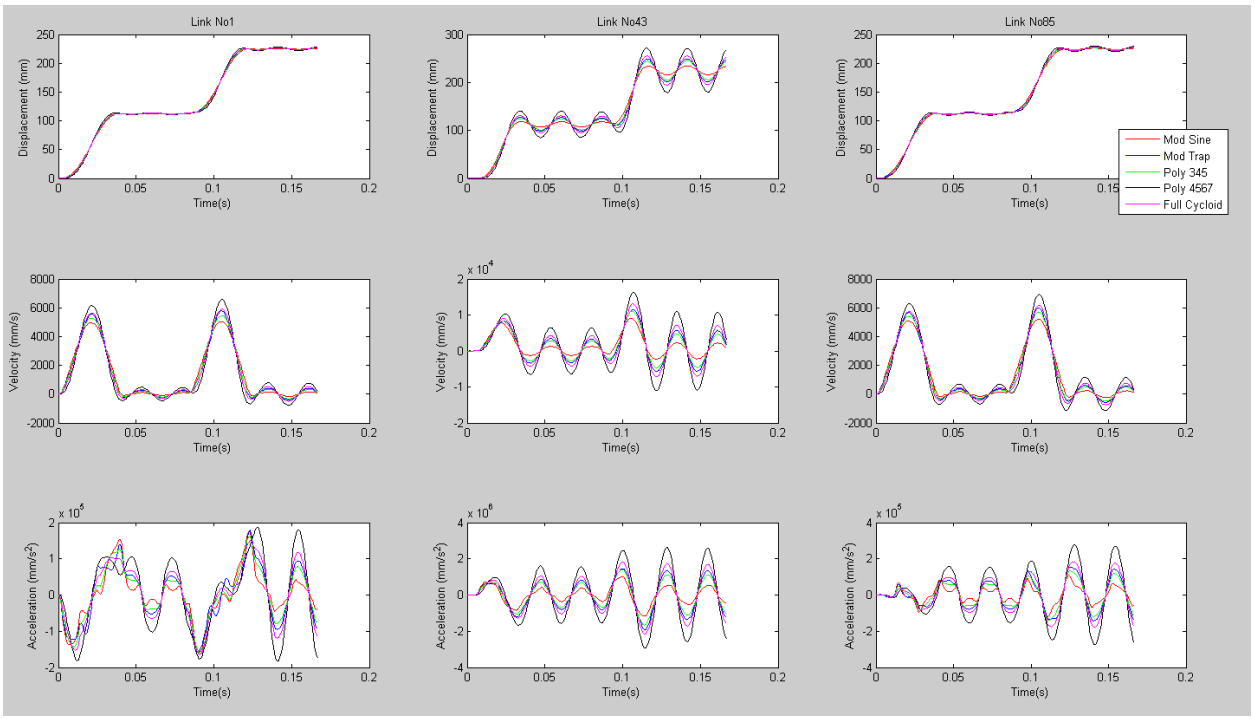

Figure 19: 85 DOF 0.15 Friction Coefficient MATLAB Test

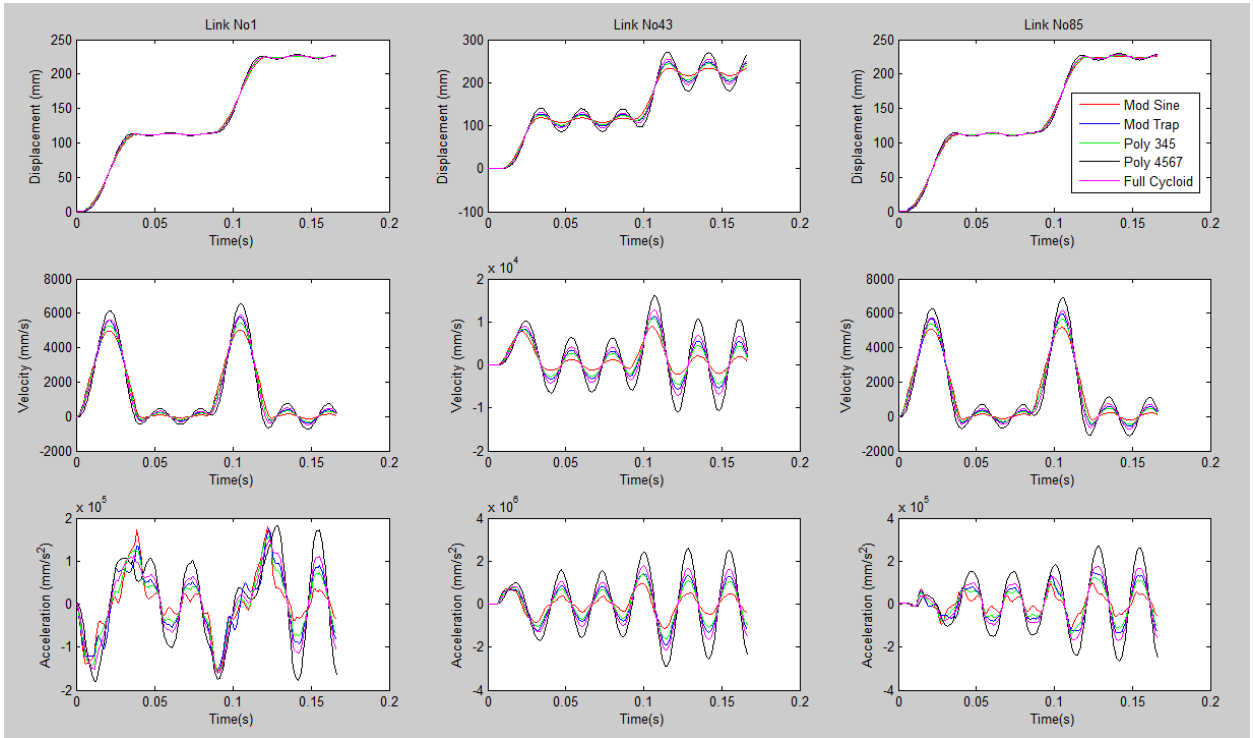

Figure 20: 85 DOF 0.25 Friction Coefficient MATLAB Test

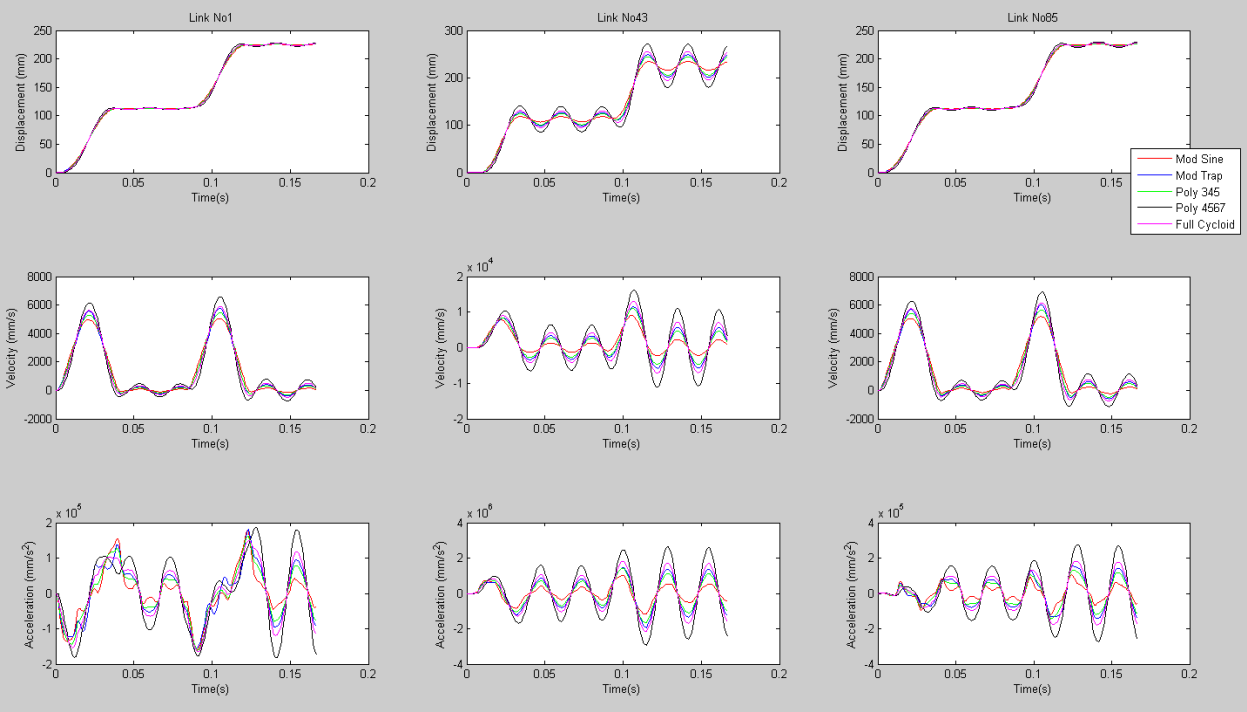

Figure 21: 85 DOF 0.35 Friction Coefficient MATLAB Test

For link 63 of the 85 DOF model, the friction coefficients were changed and the results for modified sinusoid were plotted against the Dynacam profile. The results of this test can be seen in Figure 22.

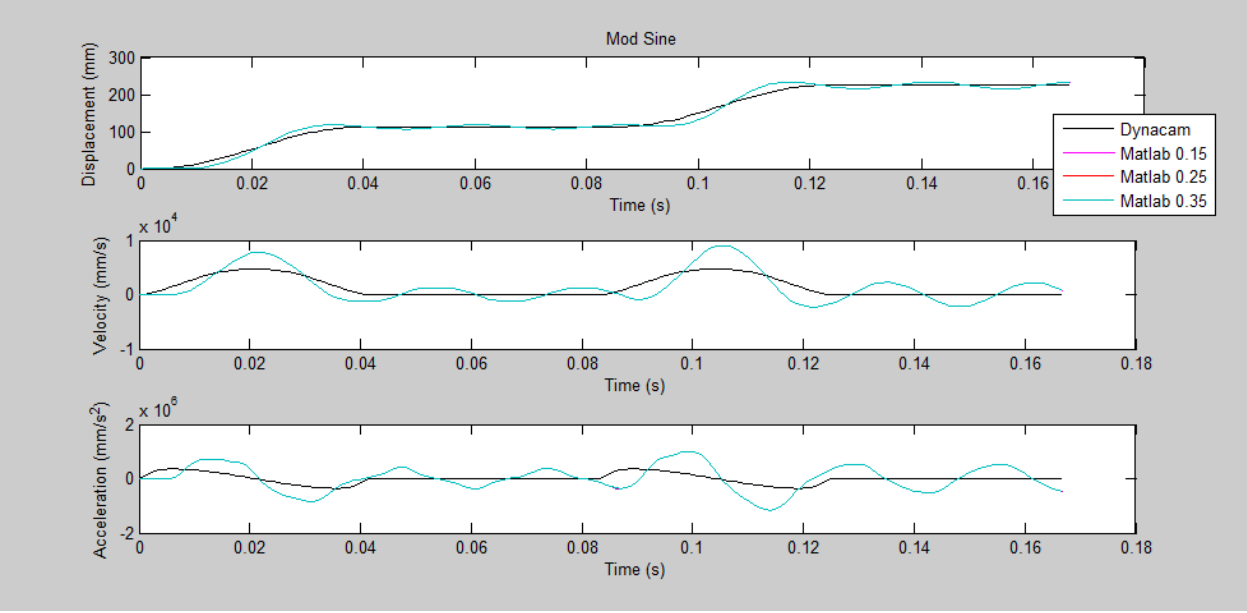

Figure 22: 85 DOF Link 63 Modified Sinusoid Comparison of 0.15, 0.25, 0.35 Friction Coefficients

5.3 Effects of Cam Profile Change

To determine the effects that the cam profile has on the system, the difference between the theoretical Dynacam data and the calculated MATLAB data was graphed. Figure 23 shows the difference between link 63's motion and the original Dynacam input data when running the MATLAB script as 85 DOF and a frictional coefficient of 0.35. The differences for the other nDOF script tests can be seen in Appendix D: Additional Figures for 20 DOF Testing, Appendix E: Additional Figures for 40 DOF Testing, Appendix F: Additional Figures for 85 DOF and Frictional Coefficient of 0.15, Appendix G: Additional Figures for 85 DOF and Frictional Coefficient of 0.25, and Appendix H: Additional Figures for 85 DOF and Frictional Coefficient of 0.35.

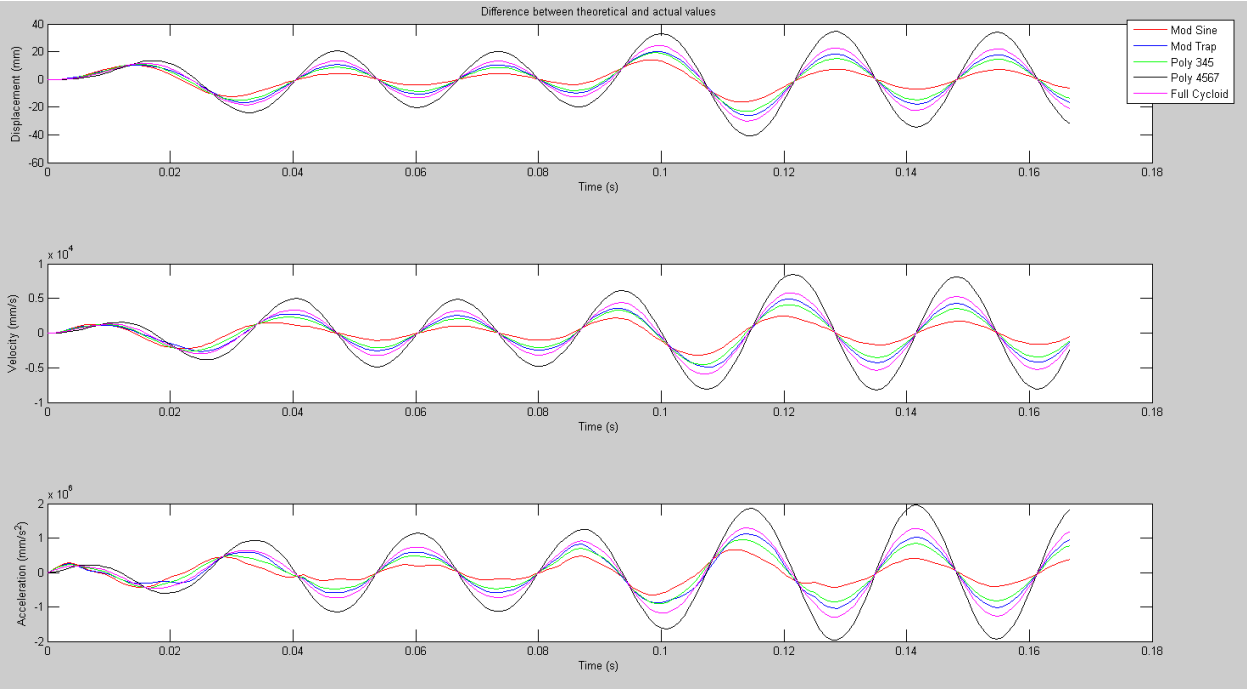

Figure 23: 85 DOF 0.35 Friction Coefficient Difference between Link 63 Calculated Values and Dynacam Values

5.4 Material Variation

The MATLAB script was run at 85 DOF with a frictional coefficient of 0.35 and a damping coefficient of 0.13. The material used in this test was titanium. The results are shown in Figure 24. Additional figures for this test can be seen in Appendix I: Additional Figures for 85 DOF and Frictional Coefficient of 0.35 for a Titanium Link.

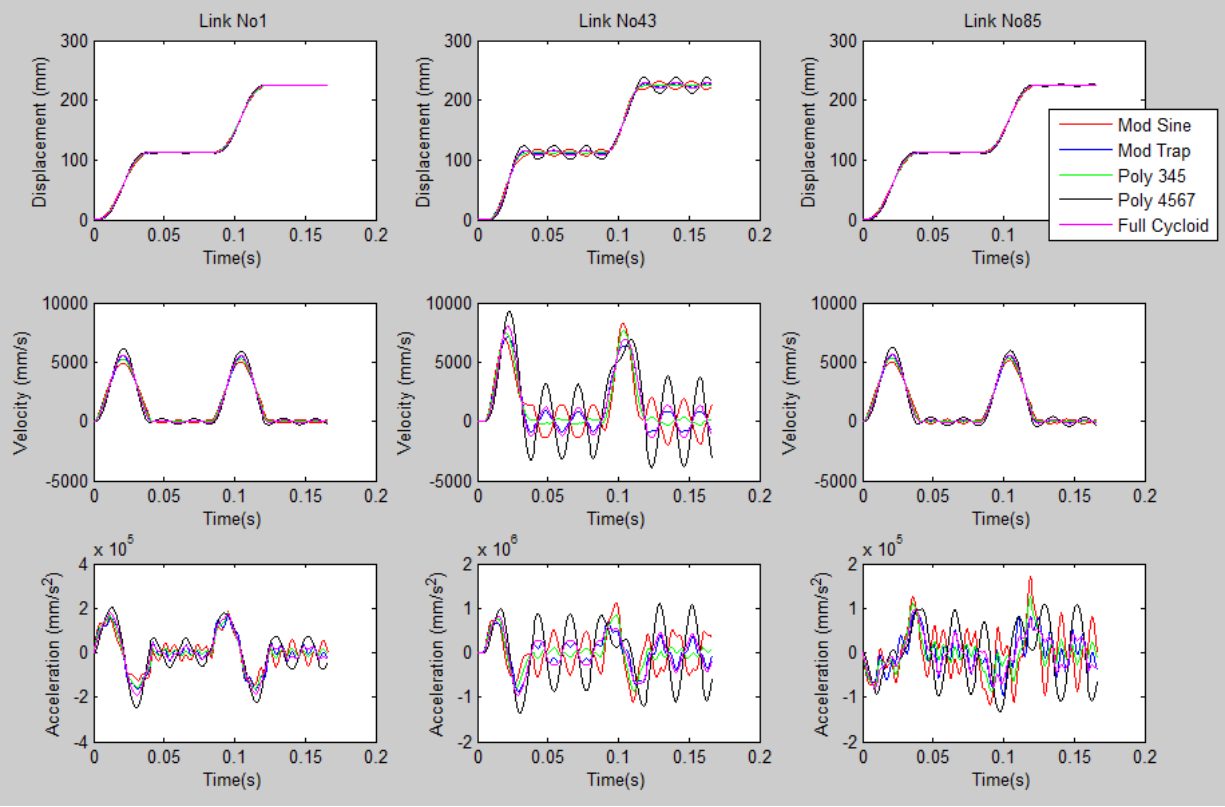

Figure 24: 85 DOF 0.35 Friction Three Link Comparison (Titanium)

5.5 RMS and Maximum Acceleration Values

The MATLAB script exported RMS and maximum acceleration values for each test that was run. The RMS and maximum acceleration values for 85 DOF and a friction coefficient of 0.35 are shown below in Table 1 and Table 2, respectively.

Table 1 shows the RMS values for the displacement, velocity, and acceleration of each cam profile that were calculated using the program Dynacam's output versus the calculated MATLAB script displacement of link 63 for the aluminum and titanium link.

Cam Profile	Maximum Acceleration (mm/s2)	
	Aluminum Link	Titanium Link
Modified Sinusoid	944,040	918,580
Modified Trapezoid	1,478,200	738,320
3-4-5 Polynomial	1,332,500	728,790
4-5-6-7 Polynomial	2,384,900	1,150,500
Full Cycloid	1,741,900	817,740

Table 2Table 2 shows the maximum acceleration that is experienced by link 63 for the aluminum and titanium link in the calculated MATLAB script for each cam profile.

	Cam Profile	Program Dynacam	Calculated MATLAB	
			Aluminum Link	Titanium Link
Displacement (mm)	Modified Sinusoid	156.84	157.63	157.25
	Modified Trapezoidal	156.95	158.00	157.58
	3-4-5 Polynomial	156.94	157.91	157.48
	4-5-6-7 Polynomial	157.21	159.11	158.23
	Full Cycloid	157.04	158.29	157.74
Velocity (mm/s)	Modified Sinusoid	2223.3	2881.9	2617.2
	Modified Trapezoidal	2283.2	3709.7	2626.2
	3-4-5 Polynomial	2275.7	3464.2	2602.8
	4-5-6-7 Polynomial	2432.1	5466.3	3255.4
	Full Cycloid	2331.9	4190.8	2751.5
Acceleration (mm/s^2)	Modified Sinusoid	1.79E+05	3.84E+05	3.53E+05
	Modified Trapezoidal	2.28E+03	6.46E+05	3.01E+05
	3-4-5 Polynomial	1.89E+05	5.70E+05	2.93E+05
	4-5-6-7 Polynomial	2.31E+05	1.11E+06	5.53E+05
	Full Cycloid	2.03E+05	7.77E+05	3.42E+05

Table 1: RMS Values for 85 DOF

Cam Profile	Maximum Acceleration (mm/s^2)	
	Aluminum Link	Titanium Link
Modified Sinusoid	944,040	918,580
Modified Trapezoid	1,478,200	738,320
3-4-5 Polynomial	1,332,500	728,790
4-5-6-7 Polynomial	2,384,900	1,150,500
Full Cycloid	1,741,900	817,740

Table 2: Maximum Acceleration Values of 85 DOF Link 63

6 Discussion

In this section, the results of the computer simulation are discussed.

6.1 Accuracy of the Model

Based on the testing, previously described in Model Testing, the model that the team has created is an accurate model of the system. The testing showed that the model outputs were close to the theoretical outputs from the program Dynacam for the 1 and 2 DOF models.

The accuracy of the script was also tested based on the RMS values. The script outputs the RMS values of the displacement, velocity, and acceleration. These values were compared to the RMS values of the Dynacam output. These values are close and therefore the team can say that the model is an accurate portrayal of the system, shown in Table 3.

	Cam Profile	Program Dynacam	Calculated MATLAB for Aluminum	Percent Difference
Displacement (mm)	Modified Sinusoid	156.84	157.63	0.50%
	Modified Trapezoidal	156.95	158.00	0.67%
	3-4-5 Polynomial	156.94	157.91	0.62%
	4-5-6-7 Polynomial	157.21	159.11	1.21%
	Full Cycloid	157.04	158.29	0.80%
Velocity (mm/s)	Modified Sinusoid	2223.3	2881.9	29.62%
	Modified Trapezoidal	2283.2	3709.7	62.48%
	3-4-5 Polynomial	2275.7	3464.2	52.23%
	4-5-6-7 Polynomial	2432.1	5466.3	124.76%
	Full Cycloid	2331.9	4190.8	79.72%
Acceleration (mm/s^2)	Modified Sinusoid	1.79E+05	3.84E+05	114.53%
	Modified Trapezoidal	2.28E+03	6.46E+05	28193.62%
	3-4-5 Polynomial	1.89E+05	5.70E+05	201.59%
	4-5-6-7 Polynomial	2.31E+05	1.11E+06	380.52%
	Full Cycloid	2.03E+05	7.77E+05	282.76%

Table 3: RMS Values for 85 DOF with Percent Difference

6.2 Comparison of the Cam Profiles

In addition to changing the main parameters of the system, the mass, stiffness, and damping, changing the cam function will affect the vibrations of the system. The various cam functions will create different variances in the system. By changing the cam function it is possible to compare the resulting displacement, velocity, and accelerations for each individual link. Modified Sine is the most commonly used cam profile in industry, but due to the complexity of this system, looking at other cam profiles is necessary.

The ranking of the profiles is based on the difference in the link motion from the original Dynacam input. There is an irregularity in the final script that the "ranking" of the cam profiles changes as the degrees of freedom of the system changes. This irregularity makes the cam profiles ranking invert, making modified sinusoid the worst choice at a low degree of freedom.

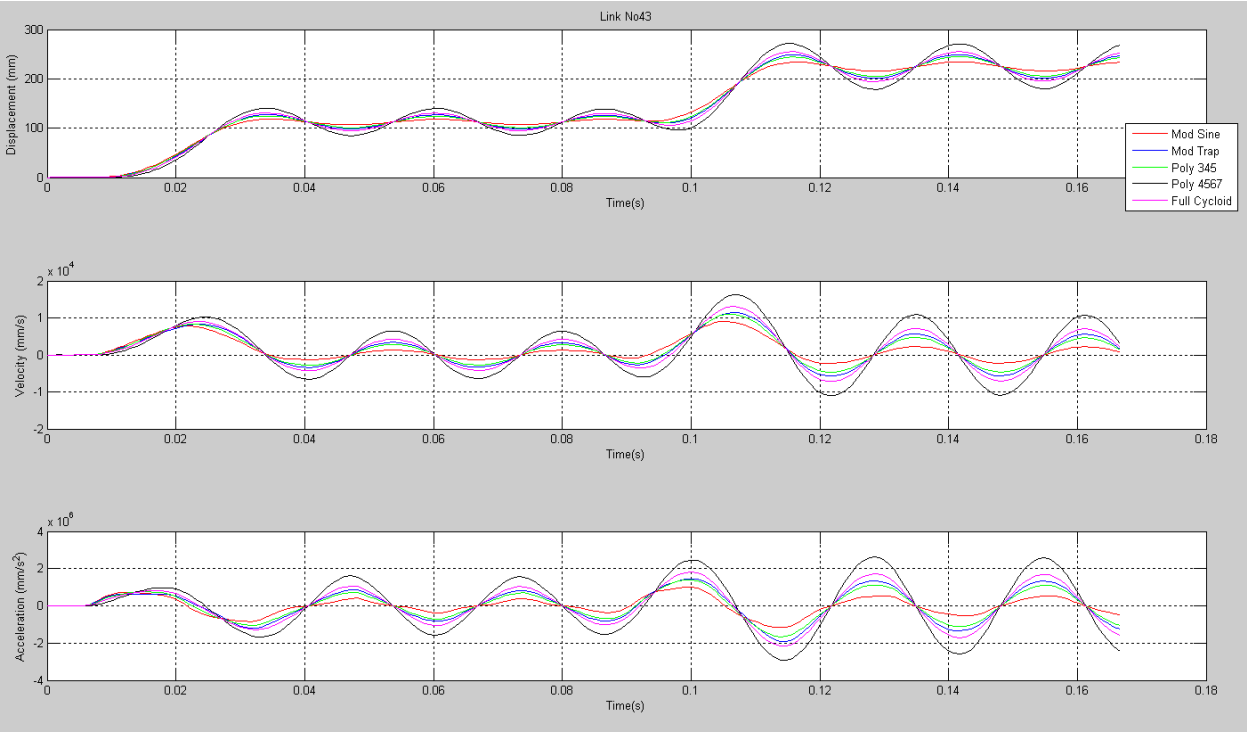

Figure 25: 85 DOF 0.15 Friction Link 43 Comparison for all Cam Profiles

With the final degree of freedom set to 85, mimicking the actual conveyor, Figure 25 shows a comparison of the all the cam profiles, while the full results are shown in Appendix F: Additional Figures for 85 DOF and Frictional Coefficient of 0.15, Appendix G: Additional Figures for 85 DOF and Frictional Coefficient of 0.25, and Appendix H: Additional Figures for 85 DOF and Frictional Coefficient of 0.35. In these results, the friction coefficient is varied, yet the "ranking" of the cam profiles remains the same. In all instances, modified sinusoid appears to have the lowest discrepancy in displacement, velocity, and acceleration compared to the cam input, yet the acceleration graph is the least smooth of the profiles. The displacement is, at maximum, 5mm off from the theoretical. The velocity is at maximum around 2500mm/s off from theoretical. The acceleration is at maximum 5 x 10^5 mm/s^2 off from the theoretical. The next three profiles, in order of expanding variance, are the 3-4-5 polynomial, modified trapezoid, and then the full cycloid. Finally, the 4-5-6-7 polynomial showed the most variance, with a maximum displacement of 20mm off from the theoretical.

6.3 Material Effects

Table 4 shows the percent difference between the Dynacam RMS values and the MATLAB RMS values for both Aluminum and Titanium links. The RMS values are shown for the displacement, velocity, and acceleration of each cam profile. As can be seen, the Titanium link has an average percent

difference of 24.41% while the Aluminum link has an average percent difference of 38.15%. The Titanium link produces the least amount of vibrations making it the ideal material for the link conveyor system.

	Cam Profile	Program Dynacam	Calculated MATLAB		Percent Difference between MATLAB and Dynacam	
			Aluminum Link	Titanium Link	Aluminum Link	Titanium Link
Displacement (mm)	Modified Sinusoid	156.84	157.63	157.25	0.50%	0.26%
	Modified Trapezoidal	156.95	158.00	157.58	0.66%	0.40%
	3-4-5 Polynomial	156.94	157.91	157.48	0.61%	0.34%
	4-5-6-7 Polynomial	157.21	159.11	158.23	1.19%	0.64%
	Full Cycloid	157.04	158.29	157.74	0.79%	0.44%
Velocity (mm/s)	Modified Sinusoid	2223.3	2881.9	2617.2	22.85%	15.05%
	Modified Trapezoidal	2283.2	3709.7	2626.2	38.45%	13.06%
	3-4-5 Polynomial	2275.7	3464.2	2602.8	34.31%	12.57%
	4-5-6-7 Polynomial	2432.1	5466.3	3255.4	55.51%	25.29%
	Full Cycloid	2331.9	4190.8	2751.5	44.36%	15.25%
Acceleration (mm/s^2)	Modified Sinusoid	1.79E+05	3.84E+05	3.53E+05	53.39%	49.29%
	Modified Trapezoidal	2.28E+03	6.46E+05	3.01E+05	99.65%	99.24%
	3-4-5 Polynomial	1.89E+05	5.70E+05	2.93E+05	66.84%	35.49%
	4-5-6-7 Polynomial	2.31E+05	1.11E+06	5.53E+05	79.19%	58.23%
	Full Cycloid	2.03E+05	7.77E+05	3.42E+05	73.87%	40.64%
			Average Percent Difference		38.15%	24.41%

Table 4: RMS Percent Differences between Dynacam, Aluminum, and Titanium Links

6.4 Frictional Effects

In order to better test the model, the frictional coefficient was varied and the results were plotted, as shown in Figure 26. The results show that the frictional coefficient is not having any effect on the script.

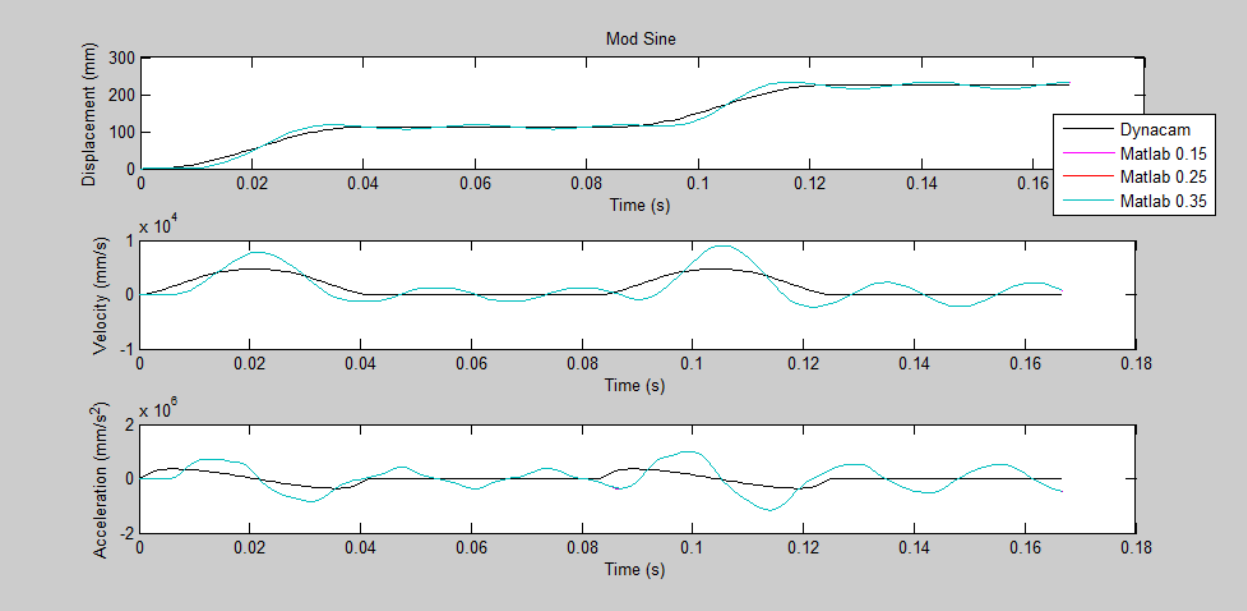

Figure 26: 85 DOF Link 63 Modified Sinusoid Comparison of 0.15, 0.25, 0.35 Friction Coefficients

7 Conclusions

Based on the results, the best cam for the system was a modified sinusoid profile while the best material is Titanium. Looking at the figures from the program, the modified sinusoid matches the theoretical output best and had the least vibrations. From looking at Figure 27, the conclusions about the best cam were made. The figure is the 85 DOF with a 0.13 damping factor and 0.35 friction coefficient. The 0.35 friction coefficient was used as a base for determining the best cam because it is the aspect ratio that most accurately mimics the tested system. The vibrations are the worst on the middle link but, of all of the cam profiles, the best vibrations are shown with the red color line which is the modified sinusoid cam profile. The modified sinusoid cam had the smallest acceleration curves for 85 DOF. The 4-5-6-7 polynomial cam had the smoothest acceleration curve but also has the highest peak acceleration. The high accelerations that occur using the 4-5-6-7 polynomial cam could be too much for the system; therefore, the team has determined that the modified sinusoid profile is the best cam profile for this system.

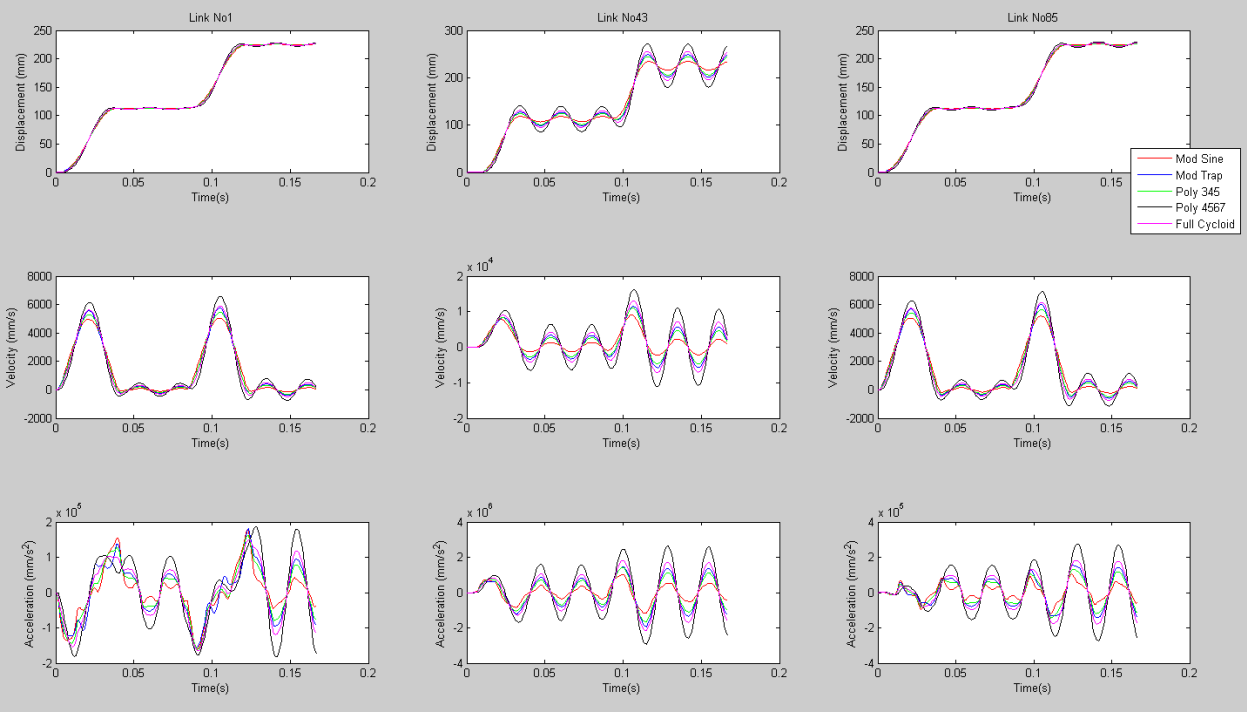

Figure 27: 85 DOF 0.35 Friction Coefficient Cam Profile Comparisons

Based on the displacement curves, all five cam profiles are comparable. They line up almost on top of one another. Figure 28 shows the modified sinusoid link 63 profile versus the modified sinusoid theoretical output from Dynacam. The velocity and acceleration for the modified sinusoid were not the smoothest, but they were the smallest, and therefore, the better option for the system.

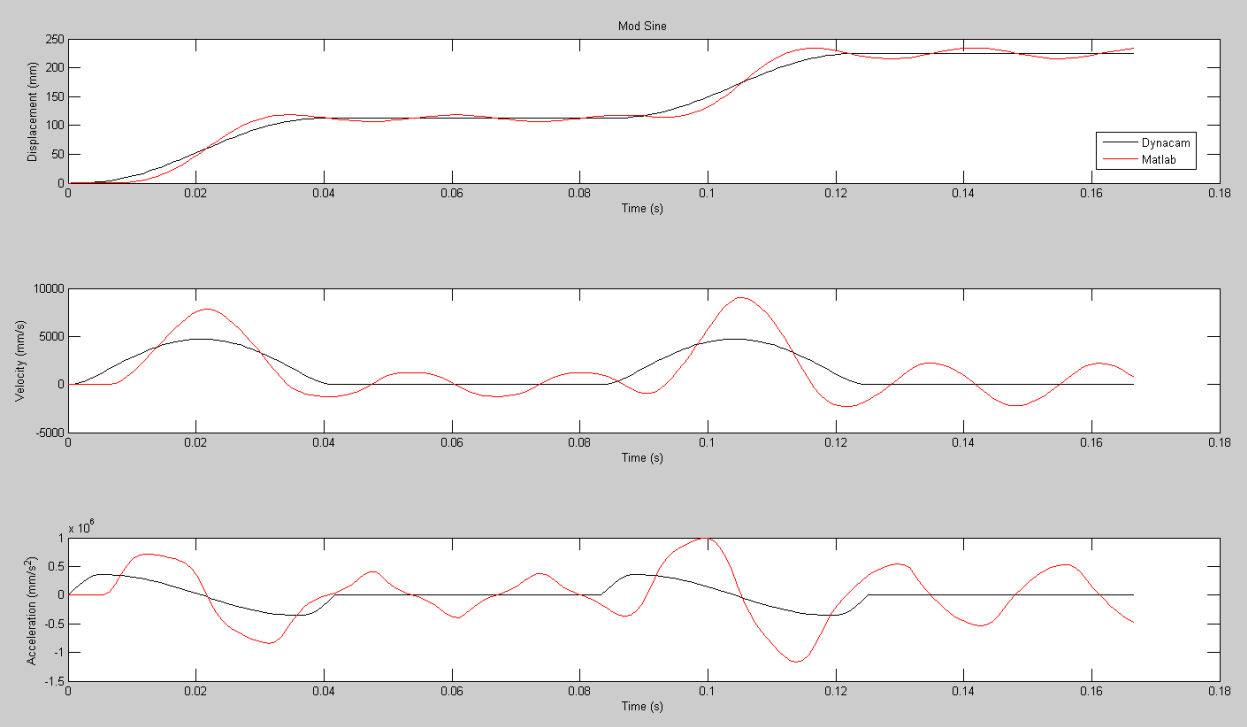

Figure 28: 85 DOF 0.35 Friction Coefficient Link 63 Modified Sinusoid Comparison

When running the script the DOF was run at 20, 40 and 85. When the script ran at 20 DOF and 40 DOF the better cam profile was the 4-5-6-7 polynomial. The velocity and acceleration were the smallest on the polynomial with those DOF. But once the system was run with 85 DOF the better profile turned out to be the modified sinusoid cam.

8 Recommendations

From the results from our script, the project team recommends a few additions and changes to the project. These are both to improve the performance of the script and other parameters that could be investigated.

Future studies should include rotational effects of the conveyor system. The script treats the system as a linear system, and does not consider the rotation around the idler and drive pulleys. Additionally, an investigation of the temperature variation which may impact the system could be performed. A change in temperature would affect the damping of the system. Also, further studies into frictional effects are necessary because no frictional effects were seen when running the MATLAB script. Friction should have some effect on the results which were not seen in the graphs.

The spring movement of the idler pulley could also be taken into consideration. The force due to the preload is placed on the links, but movement of the spring is not taken into consideration.

A modified sinusoid cam profile seems to be the best choice for this system. It has the lowest variance in the three categories. It might have some issues with jerk due to the acceleration being not as smooth as the other profiles.

Bibliography

1. **Camco.** Indexer Frequently Asked Questions. *Camco Ferguson Solutions in Motion.* [Online] June 24, 2010. [Cited: September 25, 2011.] http://www.camcoindex.com/indfaq.htm#HowCam.

2. **Norton, Robert L.** *Cam Design and Manufacturing Handbook.* New York : Industrial Press, Inc., 2009.

3. **Motion Index Drives, Inc.** Customizing an Indexer Part I. *Motion Index Drives, Inc.* [Online] [Cited: September 28, 2011.] motionindexdrives.com.

4. **Merriam-Webster Incorporated.** Sprocket. [Online] 2011. [Cited: November 14, 2011.] http://www.merriam-webster.com/dictionary/sprocket.

5. **American Chain Association.** *Standard Handbook of Chains: chains for power transmission and material handling.* Boca Raton, FL : CRC Press, 2006.

6. **wiseGEEK.** What is a Sprocket? *wiseGEEK clear answers for common questions.* [Online] Conjecture Corporation, 2011. [Cited: November 14, 2011.] http://www.wisegeek.com/what-is-a-sprocket.htm.

7. **Barnes, Nicholas J., Koniers, James P. and Wilk, Peter J.** *Analysis of an Index-Dwell Conveyor System.* Worcester : Worcester Polytechnic Institute, 2002. Major Qualifying Project.

8. **Norton Associates.** *Report on Testing of the FCAM Chassis.* 2011.

9. **The MathWorks Inc.** Product Documentation. *MathWorks.* [Online] 2011. [Cited: Sept. 8, 2011.] http://www.mathworks.com/help/toolbox/simulink/ug/f7-5734.html.

Acknowledgements

The team would formally like to thank Dr. Adrianna Hera for all of her help with the MATLAB code, Prof. Stephen Nestinger for his help with the modeling of the system, and Prof. Zhikun Hou for his help with the vibrational calculation. Gillette Co. for providing the project, especially Loren Gjata and Charlie Gillis for their assistance. Last but not least, Prof. Robert Norton for advising the project and providing the support and guidance needed to complete this project.

Appendix A: 1 Degree of Freedom MATLAB Script

1 DOF Function

```
function ydot=camfunction_10_11(t,y);
global m1 k1 c1 fi f1 f2 t1 t2


fi=f1+((f2-f1)*(t-t1))/(t2-t1); % linear interpolation
ydot(1,1)=y(2);
ydot(2,1)=1/m1*(-(c1)*y(2)-k1*y(1)+fi);
```

1 DOF Script

```
%comment
clear;close all;clc;
global m1 k1 c1 c2 fi f1 f2 t1 t2

%% import cam data
thData=dlmread('testcamsvaj_10_11.txt', '', 6, 0);
angle=thData(:,1);
z=thData(:,2);
zdot=thData(:,3);
zdotdot=thData(:,4);

%% dynamic response
dynacamData=dlmread('testcamxdot_10_11.txt', '', 6, 0);
xDynacam=dynacamData(:,3);
xdotDynacam=dynacamData(:,4);
z_xDynacam=dynacamData(:,2);

%% system parameters
m1=1.1; k1=1090000; zeta1=0.05;
w1=sqrt(k1/m1);
c1=2*zeta1*w1*m1;


%% Excitation
% M*x..+C*x.+K*x=f(t);
f=c1*zdot+k1*z;

%% Initial Conditions
x(1)=0.000108224; xdot(1)=0.180964;

wCam=180; %rpm
dt=(1/(wCam/60))/360; % sec
t=[0:1:180]*dt;     % time for a rotation

%% Solve ODE
nip=3;
for i=1:length(t)-1
    y0=[x(i); xdot(i)]; %
    t1=t(i); t2=t(i+1);
```

```
    tspan = (t1:(t2-t1)/nip:t2);
    %fi=f(i);
    f1=f(i); f2=f(i+1); % for better accuracy

    [t_y,y] = ode45('camfunction_10_11',tspan,y0);
    x(i+1,1)=y(nip+1,1); xdot(i+1,1)=y(nip+1,2);
end

%% Plot Results
figure
subplot(3,1,1);
plot(t, z_xDynacam, 'g'); hold on; plot(t, z-x, 'r')
ylabel('z-x(m)'); xlabel('Time(s)')
legend('dynacam', 'MATLAB')

subplot(3,1,2);
plot(t, xDynacam, 'g'); hold on; plot(t, x, 'r');
ylabel('x(m)'); xlabel('Time(s)')
legend('dynacam', 'MATLAB')

subplot(3,1,3)
plot(t, xdotDynacam, 'g'); hold on; plot(t, xdot, 'r');
ylabel('xdot(m/s)'); xlabel('Time(s)')
legend('dynacam', 'MATLAB')
```

Appendix B: 2 Degree of Freedom MATLAB Script

2 DOF Function

```
% FUNCTION ysystem_ndof.m
function ydot = camfunction_2dof_10_12(t,y)
global K C inv_M  ndof A B fi f1 f2 t1 t2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%xdot=[zeros(ndof), eye(ndof); -K*inv_M, -C*inv_M]*y;

fi=f1+((f2-f1)*(t-t1))/(t2-t1); % linear interpolation
force=[fi; zeros(ndof-1, 1)];
A=[zeros(ndof), eye(ndof); -inv_M*K, -inv_M*C];


B=[zeros(ndof,1);inv_M*force];
ydot= A*y+B;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

2 DOF Script

```
%comment
clear;close all;clc;
global  K C inv_M  ndof A B fi f1 f2 t1 t2
ndof=2;

%% import cam data
thData=dlmread('testcamsvaj_10_11.txt', '', 6, 0);
angle=thData(:,1);
z=thData(:,2);
zdot=thData(:,3);
zdotdot=thData(:,4);

%% dynamic response
dynacamData=dlmread('testcamxdot_10_11.txt', '', 6, 0);
xDynacam=dynacamData(:,3);
xdotDynacam=dynacamData(:,4);
z_xDynacam=dynacamData(:,2);

%% system parameters
m1=1.1; k1=1090000; zeta1=0.05;
m2=1.1; k2=1090000; zeta2=0.05;
w1=sqrt(k1/m1);
c1=2*zeta1*w1*m1;
c2=2*zeta2*w1*m1;

M=[m1 0; 0 m2];
inv_M=inv(M);

C=[c1+c2 -c2; -c2 c2];

K=[k1+k2 -k2; -k2 k2];
```

```
%% Excitation
% M*x..+C*x.+K*x=f(t);
f=c1*zdot+k1*z;

x10=0.000108224; x20=0; x1dot0=0.180964; x2dot0=0;

y0=[x10 x20 x1dot0 x2dot0];

% y0=zeros(2*ndof,1);
%
% y0(1,1)=x10;
% y0(1,ndof+1)=x1dot0;

y(1,:)=y0;

wCam=180; %rpm
dt=(1/(wCam/60))/360; % sec
t=[0:1:180]*dt;     % time for a rotation

%% Solve ODE
nip=3;

for i=1:length(t)-1
    y0=y(i,:); %
    t1=t(i); t2=t(i+1);
    tspan = (t1:(t2-t1)/nip:t2);
    %fi=f(i);
    f1=f(i); f2=f(i+1); % for better accuracy

    [t_y,yS] = ode45('camfunction_2dof_10_12',tspan,y0);
    y(i+1,:)=yS(end,:);
    %x(i+1,1)=y(nip+1,1); xdot(i+1,1)=y(nip+1,2);
end

%% Plot Graphs
figure
plot(t,y(:,1))

subplot(4,1,1);
plot(t, y(:,1), 'g')
ylabel('x1'); xlabel('Time(s)')

subplot(4,1,3);
plot(t, y(:,2), 'r')
ylabel('x2'); xlabel('Time(s)')

subplot(4,1,2)
plot(t, y(:,3), 'm')
ylabel('x1dot'); xlabel('Time(s)')

subplot(4,1,4)
plot(t, y(:,4),'b')
ylabel('x2dot'); xlabel('Time (s)')
```

Appendix C: n Degree of Freedom MATLAB Script

nDOF Modified Sinusoid Function

```
% FUNCTION ysystem_ndof.m
function y1dot = ndofcamfunction_mod_sine_2_10(t,y)

global K C inv_M  ndof A Bs fsi fs1 fs2  t1 t2 b M forces
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%xdot=[zeros(ndof), eye(ndof); -K*inv_M, -C*inv_M]*y;

fsi=fs1+((fs2-fs1)*(t-t1))/(t2-t1); % linear interpolation
if round(ndof/2)-(ndof/2)==0
    forces=[fsi; zeros((ndof-8)/2, 1);2224 ; zeros(5,1); -2224; zeros((ndof-
8)/2,1)];
else forces=[fsi; zeros((ndof-7)/2, 1); 2224; zeros(4,1); -2224; zeros((ndof-
7)/2,1)];
end

A=[zeros(ndof), eye(ndof); -inv_M*K, -inv_M*C];

Bs=[zeros(ndof,1);inv_M*forces];
y1dot= A*y+Bs;

y1dot(ndof+1:2*ndof,1)=y1dot(ndof+1:2*ndof,1)-
diag(M).*b.*sign(y(ndof+1:2*ndof,1));
```

nDOF Modified Trapezoidal Function

```
% FUNCTION ysystem_ndof.m
function y2dot = ndofcamfunction_mod_trap_2_10(t,y)
global K C inv_M  ndof A Bt fti ft1 ft2 t1 t2 b M forcet
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%xdot=[zeros(ndof), eye(ndof); -K*inv_M, -C*inv_M]*y;

fti=ft1+((ft2-ft1)*(t-t1))/(t2-t1); % linear interpolation
if round(ndof/2)-(ndof/2)==0
    forcet=[fti; zeros((ndof-8)/2, 1);2224 ; zeros(5,1); -2224; zeros((ndof-
8)/2,1)];
else forcet=[fti; zeros((ndof-7)/2, 1); 2224; zeros(4,1); -2224; zeros((ndof-
7)/2,1)];
end
A=[zeros(ndof), eye(ndof); -inv_M*K, -inv_M*C];


Bt=[zeros(ndof,1);inv_M*forcet];
y2dot= A*y+Bt;

y2dot(ndof+1:2*ndof,1)=y2dot(ndof+1:2*ndof,1)-
diag(M).*b.*sign(y(ndof+1:2*ndof,1));
```

nDOF 3-4-5 Polynomial Function

```
% FUNCTION ysystem_ndof.m
function y3dot = ndofcamfunction_poly_345_2_10(t,y)
global K C inv_M  ndof A B3 f3i f31 f32 t1 t2 M b force3
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%xdot=[zeros(ndof), eye(ndof); -K*inv_M, -C*inv_M]*y;

f3i=f31+((f32-f31)*(t-t1))/(t2-t1); % linear interpolation
if round(ndof/2)-(ndof/2)==0
    force3=[f3i; zeros((ndof-8)/2, 1);2224 ; zeros(5,1); -2224; zeros((ndof-
8)/2,1)];
else force3=[f3i; zeros((ndof-7)/2, 1); 2224; zeros(4,1); -2224; zeros((ndof-
7)/2,1)];
end
A=[zeros(ndof), eye(ndof); -inv_M*K, -inv_M*C];


B3=[zeros(ndof,1);inv_M*force3];
y3dot= A*y+B3;

y3dot(ndof+1:2*ndof,1)=y3dot(ndof+1:2*ndof,1)-
diag(M).*b.*sign(y(ndof+1:2*ndof,1));
```

nDOF 4-5-6-7 Polynomial Function

```
% FUNCTION ysystem_ndof.m
function y4dot = ndofcamfunction_poly_4567_2_10(t,y)
global K C inv_M  ndof A B4 f4i f41 f42 t1 t2 M b force4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%xdot=[zeros(ndof), eye(ndof); -K*inv_M, -C*inv_M]*y;

%1000 pounds = 4448 N
f4i=f41+((f42-f41)*(t-t1))/(t2-t1); % linear interpolation
if round(ndof/2)-(ndof/2)==0
    force4=[f4i; zeros((ndof-8)/2, 1);2224 ; zeros(5,1); -2224; zeros((ndof-
8)/2,1)];
else force4=[f4i; zeros((ndof-7)/2, 1); 2224; zeros(4,1); -2224; zeros((ndof-
7)/2,1)];
end
A=[zeros(ndof), eye(ndof); -inv_M*K, -inv_M*C];


B4=[zeros(ndof,1);inv_M*force4];
y4dot= A*y+B4;

y4dot(ndof+1:2*ndof,1)=y4dot(ndof+1:2*ndof,1)-
diag(M).*b.*sign(y(ndof+1:2*ndof,1));
```

nDOF Full Cycloid Function

```
% FUNCTION ysystem_ndof.m
function y5dot = ndofcamfunction_full_cycloid_2_10(t,y)
global K C inv_M  ndof A Bc fci fc1 fc2 t1 t2 M b forcec
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%xdot=[zeros(ndof), eye(ndof); -K*inv_M, -C*inv_M]*y;

fci=fc1+((fc2-fc1)*(t-t1))/(t2-t1); % linear interpolation
if round(ndof/2)-(ndof/2)==0
    forcec=[fci; zeros((ndof-8)/2, 1);2224 ; zeros(5,1); -2224; zeros((ndof-
8)/2,1)];
```

```
else forcec=[fci; zeros((ndof-7)/2, 1); 2224; zeros(4,1); -2224; zeros((ndof-
7)/2,1)];
end

A=[zeros(ndof), eye(ndof); -inv_M*K, -inv_M*C];


Bc=[zeros(ndof,1);inv_M*forcec];
y5dot= A*y+Bc;

y5dot(ndof+1:2*ndof,1)=y5dot(ndof+1:2*ndof,1)-
diag(M).*b.*sign(y(ndof+1:2*ndof,1));
```

nDOF Script

```
%comment
clear all;close all;clc;
global  K C inv_M ndof A Bs Bt B3 B4 Bc fs1 ft1 f31 f41 fc1 fs2 ft2 f32 f42
fc2 t1 t2 b M forces forcet forcec force3 force4
ndof=85;
tic

%% import cam data mod sine
thData1=dlmread('Barrel_Cam_Mod_Sine_10_31.txt', '', 6, 0);
angle=thData1(:,1);
z1=thData1(:,2);
z1dot=thData1(:,3);
z1dotdot=thData1(:,4);

%% import cam data mod trap
thData2=dlmread('Mod_Trap_Cam_SVAJ.txt', '', 6, 0); %reads cam data
angle=thData2(:,1); %reads angle column
z2=thData2(:,2); %reads displacement column
z2dot=thData2(:,3); %reads velocity column
z2dotdot=thData2(:,4); %reads acceleration column

%% import cam data poly 345
thData3=dlmread('Poly_345_Cam_SVAJ.txt', '', 6, 0);
angle=thData3(:,1);
z3=thData3(:,2);
z3dot=thData3(:,3);
z3dotdot=thData3(:,4);

%% import cam data poly 4567
thData4=dlmread('Poly_4567_Cam_SVAJ.txt', '', 6, 0);
angle=thData4(:,1);
z4=thData4(:,2);
z4dot=thData4(:,3);
z4dotdot=thData4(:,4);

%% import cam data full cycloid
thData5=dlmread('Full_Cycloid_Cam_SVAJ.txt', '', 6, 0);
angle=thData5(:,1);
z5=thData5(:,2);
z5dot=thData5(:,3);
```

```
z5dotdot=thData5(:,4);

%% system parameters
%Stiffness for Aluminum is 92696969.7 mass is 2.03 kg
%Stiffness for Titanium is 143884892.1 mass is 2.28 kg
%Damping is 0.13 tested
%1000 pounds = 4448 N
m1=2.03; k1=92696969.7; zeta1=0.13; b=0.25*9.8; %zeta1=damping back to link,
b=friction coefficient


%% Initial speed and time conditions
wCam=180; %rpm
dt=(1/(wCam/60))/360; % sec
t=[0:1:180]*dt;     % time for a rotation

%% Mass Matrix
M=m1*eye(ndof);
inv_M=inv(M);

% %% Friction Matrix
% Fr=eye(ndof);
% Fr=Fr*-2*b;


%% Spring Constant Matrix
K=zeros(ndof);

k=k1*ones(1,ndof);

for i=1:ndof-1
    K(i,i)=k(i)+k(i+1);
    K(i,i+1)=-k(i+1);
    K(i+1,i)=-k(i+1);
end
%Should K(1,1) be a different value because there are 6 links that are
%connected directly?

K(ndof,ndof)=2*k(1);
K(1,1)=3*k(1);
K(ndof,1)=-k(1);
K(1,ndof)=-k(1);
%K(end,end)=k(end);

%% Natural Frequency
[V,D]=eig(K,M);
w=sqrt(diag(D));
c=2*zeta1*w*m1;

%% Damping to Link Coefficient Matrix
C=zeros(ndof);

for i=1:ndof-1
    C(i,i)=c(i)+c(i+1);
```

```
    C(i,i+1)=-c(i+1);
    C(i+1,i)=-c(i+1);
end

C(ndof,ndof)=c(1)+c(ndof);
C(1,1)=3*c(1);
C(ndof,1)=-c(1);
C(1,ndof)=-c(ndof);
%C(end,end)=c(end);


%% Excitition mod sine
% M*x..+C*x.+K*x=f(t);
f11=c(1)*z1dot+k1*z1;

%% Excitition mod trap
% M*x..+C*x.+K*x=f(t);
f22=c(1)*z2dot+k1*z2;

%% Excitition poly 345
% M*x..+C*x.+K*x=f(t);
f33=c(1)*z3dot+k1*z3;

%% Excitition poly 4567
% M*x..+C*x.+K*x=f(t);
f44=c(1)*z4dot+k1*z4;

%% Excitition full cycloid
% M*x..+C*x.+K*x=f(t);
f55=c(1)*z5dot+k1*z5;
%% Initial Conditions mod sine

x10=zeros(1,ndof); x10dot=zeros(1,ndof);
y10=[x10 x10dot];

y1(1,:)=y10;

%% Initial Conditions mod trap

x20=zeros(1,ndof); x20dot=zeros(1,ndof);
y20=[x20 x20dot];

y2(1,:)=y20;

%% Initial Conditions poly 345

x30=zeros(1,ndof); x30dot=zeros(1,ndof);
y30=[x30 x30dot];

y3(1,:)=y30;

%% Initial Conditions poly 4567
```

```
x40=zeros(1,ndof); x40dot=zeros(1,ndof);
y40=[x40 x40dot];

y4(1,:)=y40;

%% Initial Conditions full cycloid

x50=zeros(1,ndof); x50dot=zeros(1,ndof);
y50=[x50 x50dot];

y5(1,:)=y50;

%% Solve ODE Mod Sine
nip=3;

s1(1,:)=[z1(1) zeros(1, ndof-1)];
s1dot(1,:)=[z1dot(1) zeros(1,ndof-1)];

for i=1:length(t)-1
    y10=y1(i,:); %
    t1=t(i); t2=t(i+1);
    tspan = (t1:(t2-t1)/nip:t2);
    %fi=f(i);
    fs1=f11(i); fs2=f11(i+1); % for better accuracy

    [t_y,yS] = ode45('ndofcamfunction_mod_sine_2_10',tspan,y10);
    y1(i+1,:)=yS(end,:);
    %x(i+1,1)=y(nip+1,1); xdot(i+1,1)=y(nip+1,2);


    x1(i+1,:)=y1(i+1,1:ndof);
    x1dot(i+1,:)=y1(i+1,ndof+1:end);

%     s(i+1,:)=[z(i+1) zeros(1, ndof-1)];
%     sdot(i+1,:)=[zdot(i+1) zeros(1,ndof-1)];
    x1dotdot(i+1,:)=inv_M*(-C*x1dot(i+1,:)'-K*x1(i+1,:)'+forces);

end


%% Solve ODE Mod Trap
nip=3;

s2(1,:)=[z2(1) zeros(1, ndof-1)];
s2dot(1,:)=[z2dot(1) zeros(1,ndof-1)];

for i=1:length(t)-1
    y20=y2(i,:); %
    t1=t(i); t2=t(i+1);
    tspan = (t1:(t2-t1)/nip:t2);
    %fi=f(i);
    ft1=f22(i); ft2=f22(i+1); % for better accuracy
```

```
    [t_y,yS] = ode45('ndofcamfunction_mod_trap_2_10',tspan,y20);
    y2(i+1,:)=yS(end,:);
    %x(i+1,1)=y(nip+1,1); xdot(i+1,1)=y(nip+1,2);


    x2(i+1,:)=y2(i+1,1:ndof);
    x2dot(i+1,:)=y2(i+1,ndof+1:end);

%     s(i+1,:)=[z(i+1) zeros(1, ndof-1)];
%     sdot(i+1,:)=[zdot(i+1) zeros(1,ndof-1)];
    x2dotdot(i+1,:)=inv_M*(-C*x2dot(i+1,:)'-K*x2(i+1,:)'+forcet);

end

%% Solve ODE poly 345
nip=3;

s3(1,:)=[z3(1) zeros(1, ndof-1)];
s3dot(1,:)=[z3dot(1) zeros(1,ndof-1)];

for i=1:length(t)-1
    y30=y3(i,:); %
    t1=t(i); t2=t(i+1);
    tspan = (t1:(t2-t1)/nip:t2);
    %fi=f(i);
    f31=f33(i); f32=f33(i+1); % for better accuracy

    [t_y,yS] = ode45('ndofcamfunction_poly_345_2_10',tspan,y30);
    y3(i+1,:)=yS(end,:);
    %x(i+1,1)=y(nip+1,1); xdot(i+1,1)=y(nip+1,2);


    x3(i+1,:)=y3(i+1,1:ndof);
    x3dot(i+1,:)=y3(i+1,ndof+1:end);

%     s(i+1,:)=[z(i+1) zeros(1, ndof-1)];
%     sdot(i+1,:)=[zdot(i+1) zeros(1,ndof-1)];
    x3dotdot(i+1,:)=inv_M*(-C*x3dot(i+1,:)'-K*x3(i+1,:)'+force3);

end

%% Solve ODE poly 4567
nip=3;

s4(1,:)=[z4(1) zeros(1, ndof-1)];
s4dot(1,:)=[z4dot(1) zeros(1,ndof-1)];

for i=1:length(t)-1
    y40=y4(i,:); %
    t1=t(i); t2=t(i+1);
    tspan = (t1:(t2-t1)/nip:t2);
    %fi=f(i);
    f41=f44(i); f42=f44(i+1); % for better accuracy
```

```
    [t_y,yS] = ode45('ndofcamfunction_poly_4567_2_10',tspan,y40);
    y4(i+1,:)=yS(end,:);
    %x(i+1,1)=y(nip+1,1); xdot(i+1,1)=y(nip+1,2);


    x4(i+1,:)=y4(i+1,1:ndof);
    x4dot(i+1,:)=y4(i+1,ndof+1:end);

%     s(i+1,:)=[z(i+1) zeros(1, ndof-1)];
%     sdot(i+1,:)=[zdot(i+1) zeros(1,ndof-1)];
    x4dotdot(i+1,:)=inv_M*(-C*x4dot(i+1,:)'-K*x4(i+1,:)'+force4);

end

%% Solve ODE full cycloid
nip=3;

s5(1,:)=[z5(1) zeros(1, ndof-1)];
s5dot(1,:)=[z5dot(1) zeros(1,ndof-1)];

for i=1:length(t)-1
    y50=y5(i,:); %
    t1=t(i); t2=t(i+1);
    tspan = (t1:(t2-t1)/nip:t2);
    %fi=f(i);
    fc1=f55(i); fc2=f55(i+1); % for better accuracy

    [t_y,yS] = ode45('ndofcamfunction_full_cycloid_2_10',tspan,y50);
    y5(i+1,:)=yS(end,:);
    %x(i+1,1)=y(nip+1,1); xdot(i+1,1)=y(nip+1,2);


    x5(i+1,:)=y5(i+1,1:ndof);
    x5dot(i+1,:)=y5(i+1,ndof+1:end);

%     s(i+1,:)=[z(i+1) zeros(1, ndof-1)];
%     sdot(i+1,:)=[zdot(i+1) zeros(1,ndof-1)];
    x5dotdot(i+1,:)=inv_M*(-C*x5dot(i+1,:)'-K*x5(i+1,:)'+forcec);

end

%% Plot three links (first, middle, last)
Links2plot=[1, round(ndof/2), ndof];

figure('Name','Comparison of 3 Links')

for i=1:length(Links2plot)
subplot(3,length(Links2plot),i)
linkNo=Links2plot(i);

subplot(3,length(Links2plot),i)
plot(t,y1(:,linkNo), 'r')
hold on
plot(t,y2(:,linkNo), 'b')
```

```
hold on
plot(t,y3(:,linkNo), 'g')
hold on
plot(t,y4(:,linkNo), 'k')
hold on
plot(t,y5(:,linkNo), 'm')
title(['Link No' num2str(linkNo)])
ylabel('Displacement (mm)'); xlabel('Time(s)');

subplot(3,length(Links2plot),length(Links2plot)+i)
plot(t,y1(:,ndof+linkNo), 'r')
hold on
plot(t,y2(:,ndof+linkNo), 'b')
hold on
plot(t,y3(:,ndof+linkNo), 'g')
hold on
plot(t,y4(:,ndof+linkNo), 'k')
hold on
plot(t,y5(:,ndof+linkNo), 'm')
ylabel('Velocity (mm/s)'); xlabel('Time(s)');

subplot(3,length(Links2plot),2*length(Links2plot)+i)
plot(t,x1dotdot(:,linkNo), 'r')
hold on
plot(t,x2dotdot(:,linkNo), 'b')
hold on
plot(t,x3dotdot(:,linkNo), 'g')
hold on
plot(t,x4dotdot(:,linkNo), 'k')
hold on
plot(t,x5dotdot(:,linkNo), 'm')
ylabel('Acceleration (mm/s^2)'); xlabel('Time(s)');
end

subplot(3,length(Links2plot),i)
legend('Mod Sine', 'Mod Trap', 'Poly 345', 'Poly 4567', 'Full Cycloid')

%% Plot Only Middle Link
figure('Name','Middle Link Comparison')
linkNo1=Links2plot(:,2);

subplot(3,1,1)
plot(t,y1(:,linkNo1), 'r')
hold on
plot(t,y2(:,linkNo1), 'b')
hold on
plot(t,y3(:,linkNo1), 'g')
hold on
plot(t,y4(:,linkNo1), 'k')
hold on
plot(t,y5(:,linkNo1), 'm')
title(['Link No' num2str(linkNo1)])
ylabel('Displacement (mm)'); xlabel('Time(s)');
grid on

subplot(3,1,2)
```

```
plot(t,y1(:,ndof+linkNo1), 'r')
hold on
plot(t,y2(:,ndof+linkNo1), 'b')
hold on
plot(t,y3(:,ndof+linkNo1), 'g')
hold on
plot(t,y4(:,ndof+linkNo1), 'k')
hold on
plot(t,y5(:,ndof+linkNo1), 'm')
ylabel('Velocity (mm/s)'); xlabel('Time(s)');
grid on

subplot(3,1,3)
plot(t,x1dotdot(:,linkNo1), 'r')
hold on
plot(t,x2dotdot(:,linkNo1), 'b')
hold on
plot(t,x3dotdot(:,linkNo1), 'g')
hold on
plot(t,x4dotdot(:,linkNo1), 'k')
hold on
plot(t,x5dotdot(:,linkNo1), 'm')
ylabel('Acceleration (mm/s^2)'); xlabel('Time(s)');


grid on
subplot(3,1,1)
legend('Mod Sine', 'Mod Trap', 'Poly 345', 'Poly 4567', 'Full Cycloid')

%% Mod Sine Comparison
linkNo2=Links2plot(:,2);

figure('Name', 'Mod Sine')

subplot(3,1,1)
plot(t,z1,'color','k')
hold on
plot(t,y1(:,linkNo2),'color','r');
ylabel('Displacement (mm)')
xlabel('Time (s)')
title('Mod Sine')

subplot(3,1,2)
plot(t,z1dot,'color','k')
hold on
plot(t,y1(:,ndof+linkNo2),'color','r');
ylabel('Velocity (mm/s)')
xlabel('Time (s)')

subplot(3,1,3)
plot(t,z1dotdot,'color','k')
hold on
plot(t,x1dotdot(:,linkNo2),'color','r');
ylabel('Acceleration (mm/s^2)')
xlabel('Time (s)')
```

```
subplot(3,1,1)
legend('Dynacam','MATLAB')

%% Mod Trap Comparison
figure('Name','Mod Trap')

subplot(3,1,1)
plot(t,z2,'color','k')
hold on
plot(t,y2(:,linkNo2),'color','b');
ylabel('Displacement (mm)')
xlabel('Time (s)')
title('Mod Trap')

subplot(3,1,2)
plot(t,z2dot,'color','k')
hold on
plot(t,y2(:,ndof+linkNo2),'color','b');
ylabel('Velocity (mm/s)')
xlabel('Time (s)')

subplot(3,1,3)
plot(t,z2dotdot,'color','k')
hold on
plot(t,x2dotdot(:,linkNo2),'color','b');
ylabel('Acceleration (mm/s^2)')
xlabel('Time (s)')

subplot(3,1,1)
legend('Dynacam','MATLAB')

%% Poly 345 Comparison
figure('Name','Poly 345')

subplot(3,1,1)
plot(t,z3,'color','k')
hold on
plot(t,y3(:,linkNo2),'color','g');
ylabel('Displacement (mm)')
xlabel('Time (s)')
title('Poly 345')

subplot(3,1,2)
plot(t,z3dot,'color','k')
hold on
plot(t,y3(:,ndof+linkNo2),'color','g');
ylabel('Velocity (mm/s)')
xlabel('Time (s)')

subplot(3,1,3)
plot(t,z3dotdot,'color','k')
hold on
plot(t,x3dotdot(:,linkNo2),'color','g');
ylabel('Acceleration (mm/s^2)')
```

```
xlabel('Time (s)')

subplot(3,1,1)
legend('Dynacam','MATLAB')

%% Poly 4567 Comparison
figure('Name','Poly 4567')

subplot(3,1,1)
plot(t,z4,'color','k')
hold on
plot(t,y4(:,linkNo2),'color','c');
ylabel('Displacement (mm)')
xlabel('Time (s)')
title('Poly 4567')

subplot(3,1,2)
plot(t,z4dot,'color','k')
hold on
plot(t,y4(:,ndof+linkNo2),'color','c');
ylabel('Velocity (mm/s)')
xlabel('Time (s)')

subplot(3,1,3)
plot(t,z4dotdot,'color','k')
hold on
plot(t,x4dotdot(:,linkNo2),'color','c');
ylabel('Acceleration (mm/s^2)')
xlabel('Time (s)')

subplot(3,1,1)
legend('Dynacam','MATLAB')

%% Full Cycloid Comparison
figure('Name','Full Cycloid')

subplot(3,1,1)
plot(t,z5,'color','k')
hold on
plot(t,y5(:,linkNo2),'color','m');
ylabel('Displacement (mm)')
xlabel('Time (s)')
title('Full Cycloid')

subplot(3,1,2)
plot(t,z5dot,'color','k')
hold on
plot(t,y5(:,ndof+linkNo2),'color','m');
ylabel('Velocity (mm/s)')
xlabel('Time (s)')

subplot(3,1,3)
plot(t,z5dotdot,'color','k')
hold on
plot(t,x5dotdot(:,linkNo2),'color','m');
```

```
ylabel('Acceleration (mm/s^2)')
xlabel('Time (s)')

subplot(3,1,1)
legend('Dynacam','MATLAB')

%% RMS Displacement Values of link 1/4 of the way down the chain

hrms2d=dsp.RMS;

hrms2d.Dimension = 'Column';
p1=[z1 y1(:,round(3*ndof/4))];
q1=step(hrms2d,p1);

p2=[z2 y2(:,round(3*ndof/4))];
q2=step(hrms2d,p2);

p3=[z3 y3(:,round(3*ndof/4))];
q3=step(hrms2d,p3);

p4=[z4 y4(:,round(3*ndof/4))];
q4=step(hrms2d,p4);

p5=[z5 y5(:,round(3*ndof/4))];
q5=step(hrms2d,p5);

RMSd=[1 q1; 2 q2; 3 q3; 4 q4; 5 q5];

%% RMS Velocity Values of link 1/4 of the way down the chain
p1dot=[z1dot y1(:,ndof+round(3*ndof/4))];
q1dot=step(hrms2d,p1dot);

p2dot=[z2dot y2(:,ndof+round(3*ndof/4))];
q2dot=step(hrms2d,p2dot);

p3dot=[z3dot y3(:,ndof+round(3*ndof/4))];
q3dot=step(hrms2d,p3dot);

p4dot=[z4dot y4(:,ndof+round(3*ndof/4))];
q4dot=step(hrms2d,p4dot);

p5dot=[z5dot y5(:,ndof+round(3*ndof/4))];
q5dot=step(hrms2d,p5dot);

RMSv=[1 q1dot; 2 q2dot; 3 q3dot; 4 q4dot; 5 q5dot];
%% RMS Acceleration Values of link 1/4 of the way down the chain
p1dotdot=[z1dotdot x1dotdot(:,round(3*ndof/4))];
q1dotdot=step(hrms2d,p1dotdot);

p2dotdot=[z2dot x2dotdot(:,round(3*ndof/4))];
q2dotdot=step(hrms2d,p2dotdot);

p3dotdot=[z3dotdot x3dotdot(:,round(3*ndof/4))];
```

```
q3dotdot=step(hrms2d,p3dotdot);

p4dotdot=[z4dotdot x4dotdot(:,round(3*ndof/4))];
q4dotdot=step(hrms2d,p4dotdot);

p5dotdot=[z5dotdot x5dotdot(:,round(3*ndof/4))];
q5dotdot=step(hrms2d,p5dotdot);

RMSa=[1 q1dotdot; 2 q2dotdot; 3 q3dotdot; 4 q4dotdot; 5 q5dotdot];

%% Maximum Acceleration
Ma1=max(abs(x1dotdot));
Ma2=max(abs(x2dotdot));
Ma3=max(abs(x3dotdot));
Ma4=max(abs(x4dotdot));
Ma5=max(abs(x5dotdot));

%% Compile All Data
%RMS Values
RMS=[RMSd; RMSv; RMSa];

%Maximum Acceleration
Ma=[1 Ma1; 2 Ma2; 3 Ma3; 4 Ma4; 5 Ma5];

%% Difference in displacement 1/4 of the way down the chain
D1=z1-y1(:,round(3*ndof/4));
D2=z2-y2(:,round(3*ndof/4));
D3=z3-y3(:,round(3*ndof/4));
D4=z4-y4(:,round(3*ndof/4));
D5=z5-y5(:,round(3*ndof/4));

%% Difference in velocity 1/4 of the way down the chain
Dv1=z1dot-y1(:,round(3*ndof/4)+ndof);
Dv2=z2dot-y2(:,round(3*ndof/4)+ndof);
Dv3=z3dot-y3(:,round(3*ndof/4)+ndof);
Dv4=z4dot-y4(:,round(3*ndof/4)+ndof);
Dv5=z5dot-y5(:,round(3*ndof/4)+ndof);

%% Difference in acceleration 1/4 of the way down the chain
Da1=z1dotdot-x1dotdot(:,round(3*ndof/4));
Da2=z2dotdot-x2dotdot(:,round(3*ndof/4));
Da3=z3dotdot-x3dotdot(:,round(3*ndof/4));
Da4=z4dotdot-x4dotdot(:,round(3*ndof/4));
Da5=z5dotdot-x5dotdot(:,round(3*ndof/4));

%% Plot Differences
figure ('Name','Plot Differences')
subplot(3,1,1)
plot(t,D1, 'r')
hold on
plot(t,D2,'b')
hold on
plot(t,D3,'g')
hold on
plot(t,D4,'k')
```

```
hold on
plot(t,D5,'m')
ylabel('Displacement (mm)')
xlabel('Time (s)')
legend('Mod Sine','Mod Trap','Poly 345','Poly 4567','Full Cycloid')
title('Difference between theoretical and actual values')

subplot(3,1,2)
plot(t,Dv1, 'r')
hold on
plot(t,Dv2,'b')
hold on
plot(t,Dv3,'g')
hold on
plot(t,Dv4,'k')
hold on
plot(t,Dv5,'m')
ylabel('Velocity (mm/s)')
xlabel('Time (s)')

subplot(3,1,3)
plot(t,Da1, 'r')
hold on
plot(t,Da2,'b')
hold on
plot(t,Da3,'g')
hold on
plot(t,Da4,'k')
hold on
plot(t,Da5,'m')
ylabel('Acceleration (mm/s^2)')
xlabel('Time (s)')

%% Export Data

if k1==92696969.7
dlmwrite('RMS_Values_Aluminum.txt',RMS,' ');
dlmwrite('Maximum_Acceleration_Aluminum.txt',Ma,' ');
else
    dlmwrite('RMS_Values_Titanium.txt',RMS,' ');
    dlmwrite('Maximum_Acceleration_Titanium.txt',Ma,' ');
end

tComp=toc
```

Appendix D: Additional Figures for 20 DOF Testing

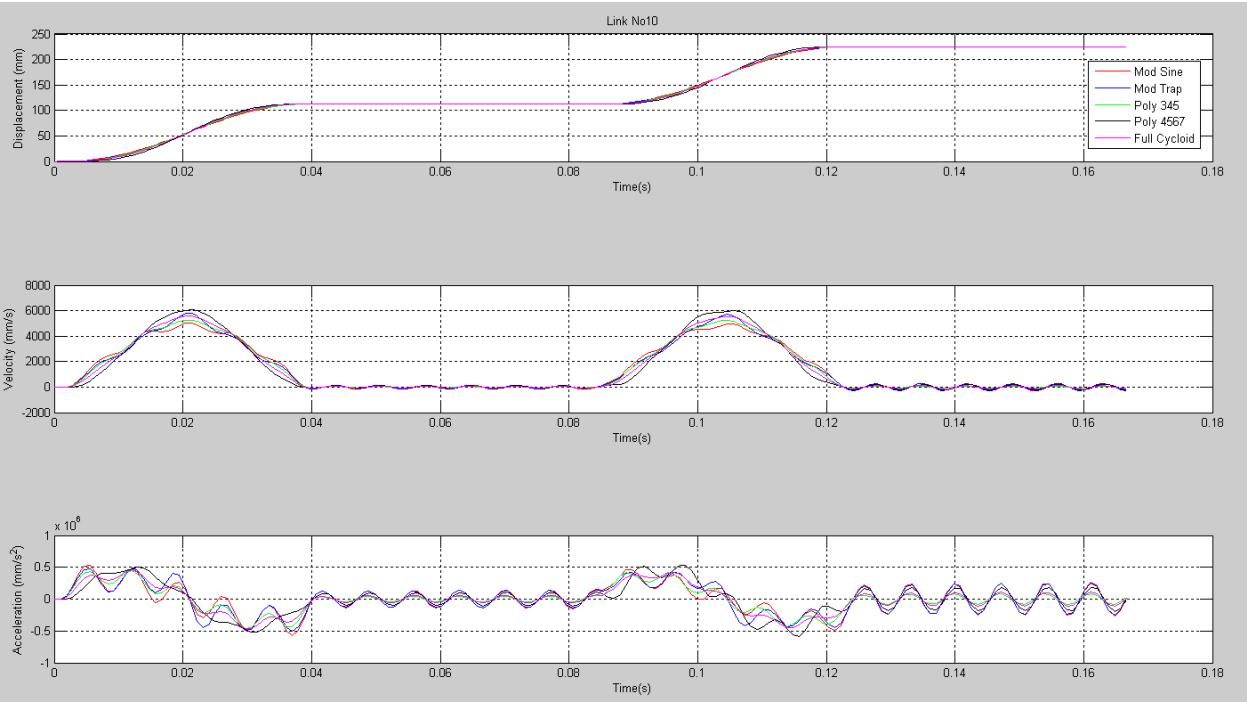

Figure 29: 20 DOF Link 10 Comparison for all Cam Profiles

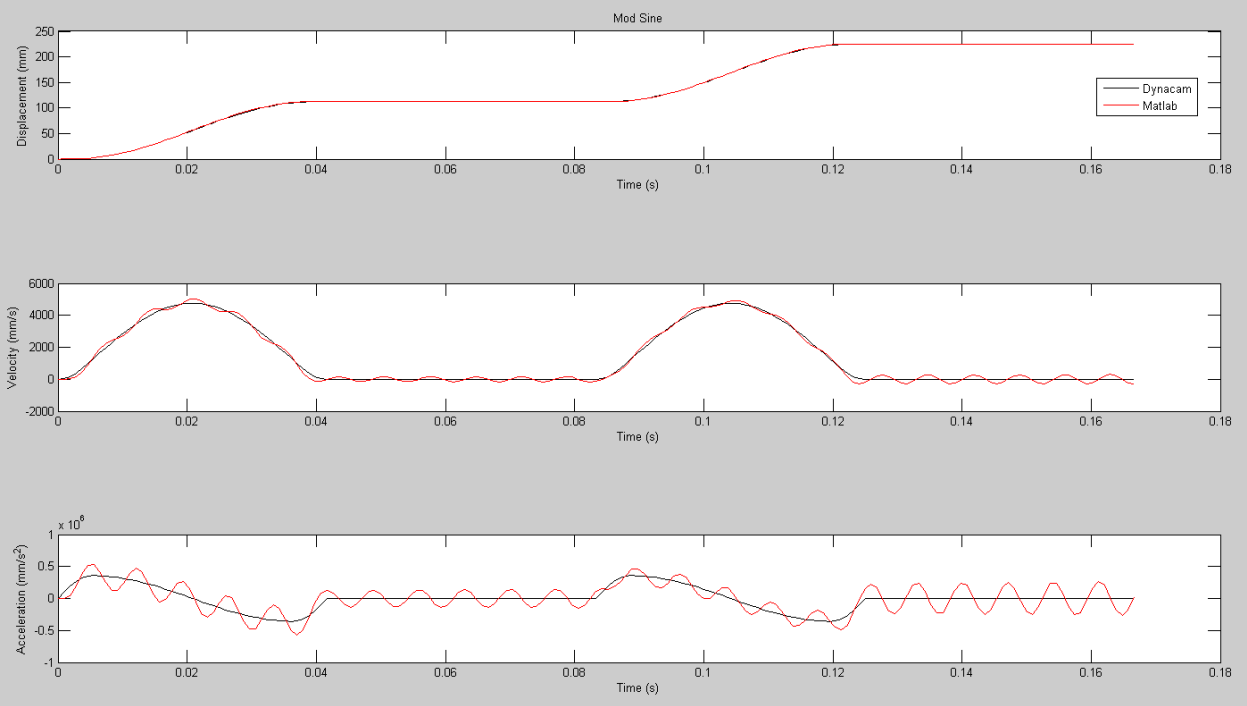

Figure 30: 20 DOF Link 15 Modified Sinusoid

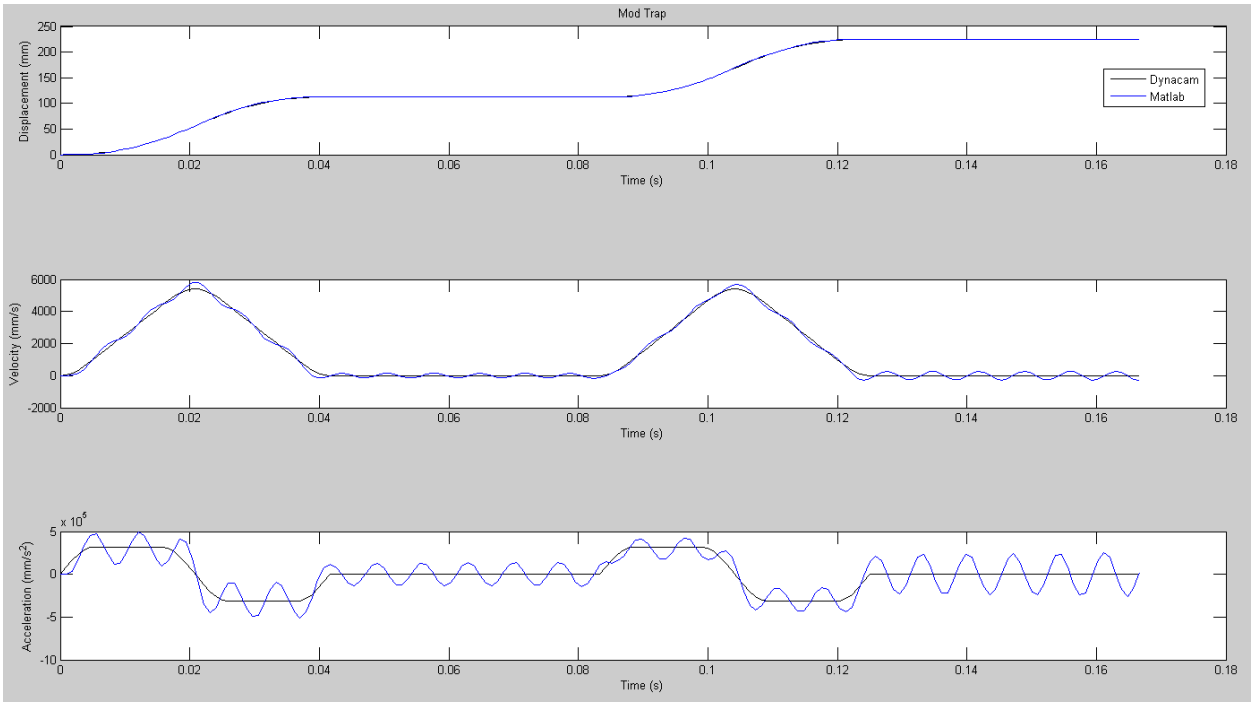

Figure 31: 20 DOF Link 15 Modified Trapezoidal

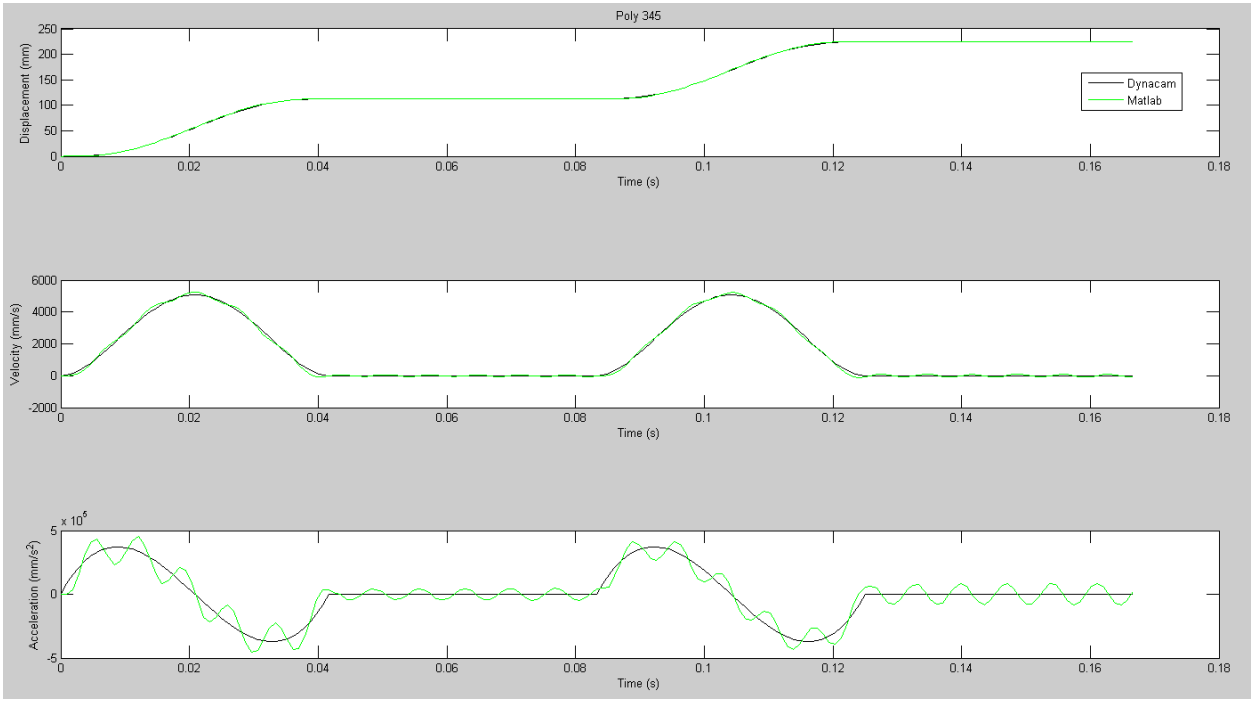

Figure 32: 20 DOF Link 15 3-4-5 Polynomial

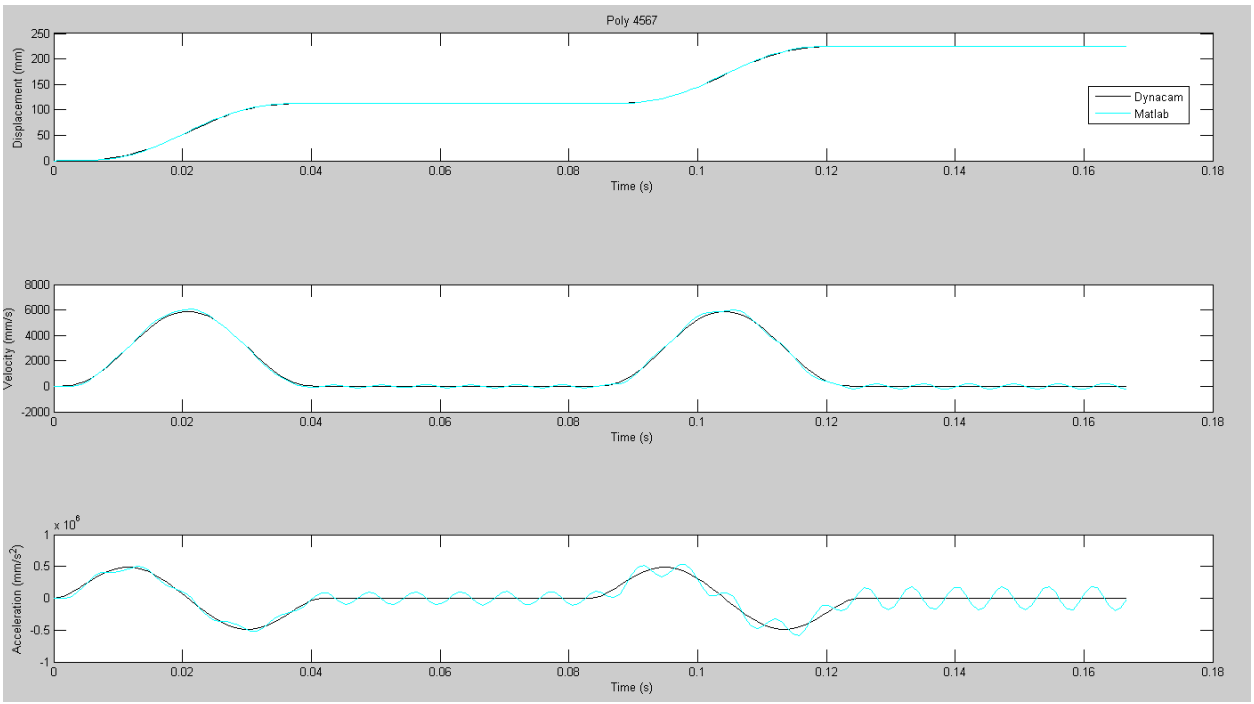

Figure 33: 20 DOF Link 15 4-5-6-7 Polynomial

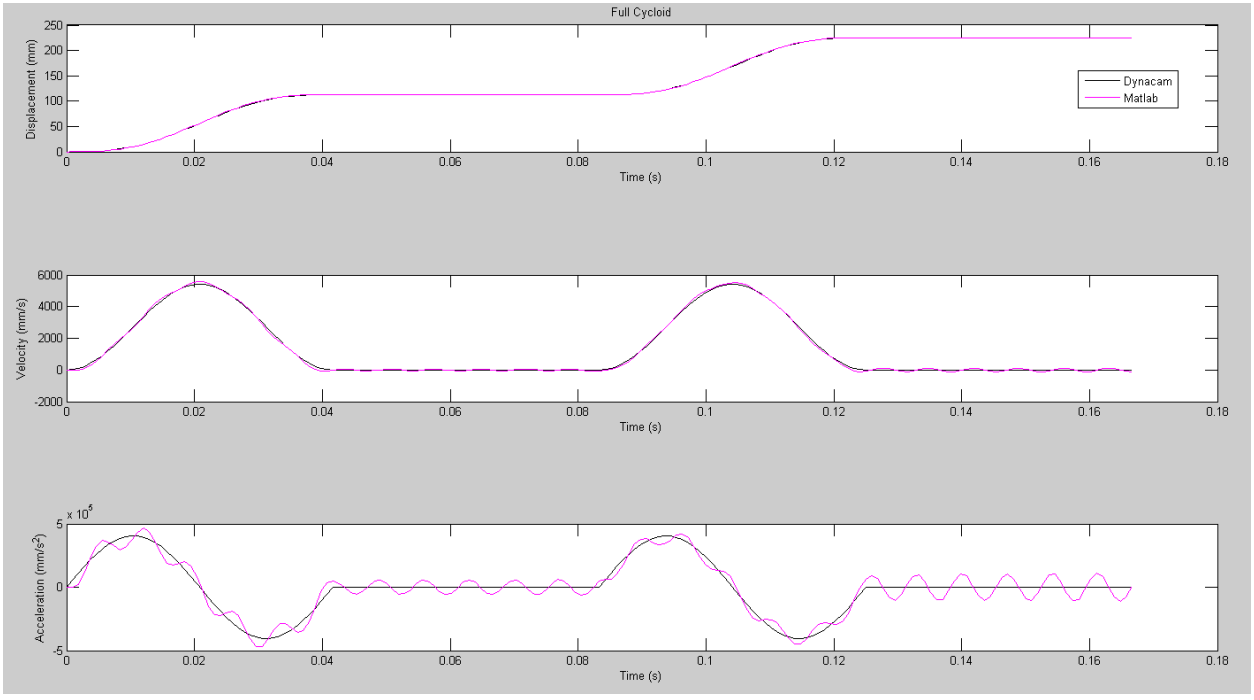

Figure 34: 20 DOF Link 15 Full Cycloid

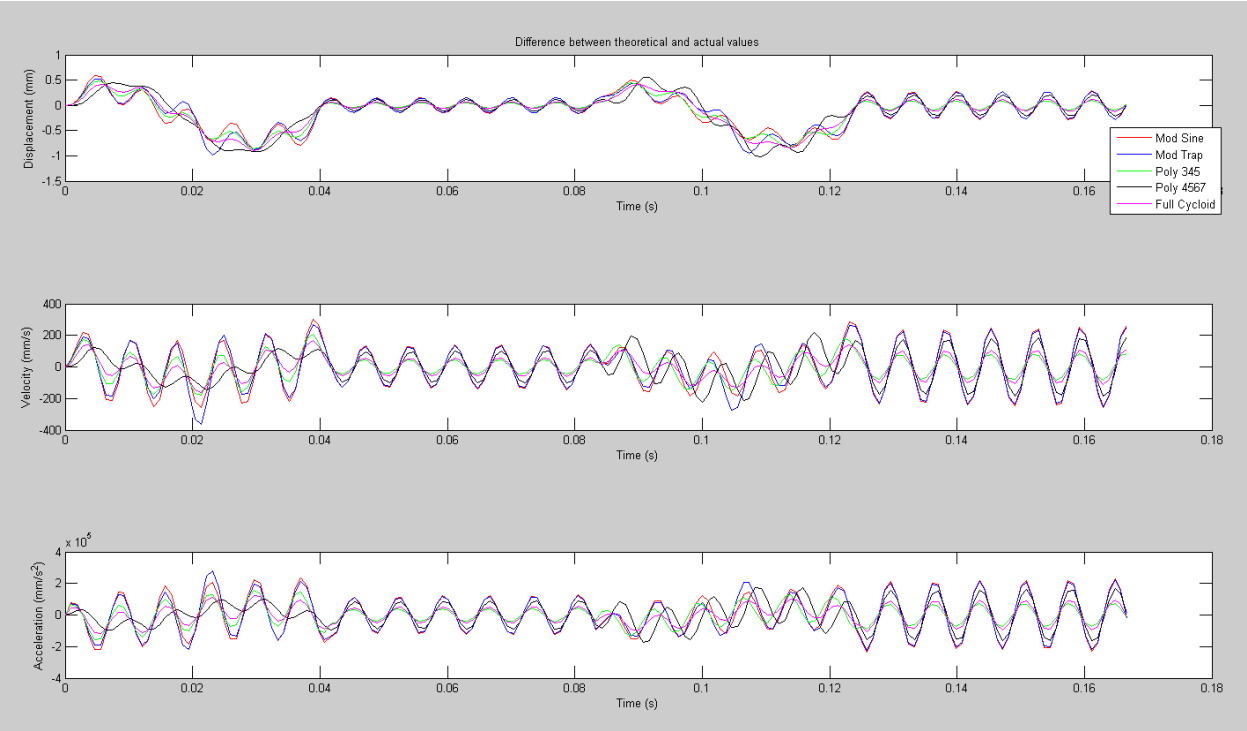

Figure 35: 20 DOF Difference between Link 15 Calculated Values and Dynacam Values

Appendix E: Additional Figures for 40 DOF Testing

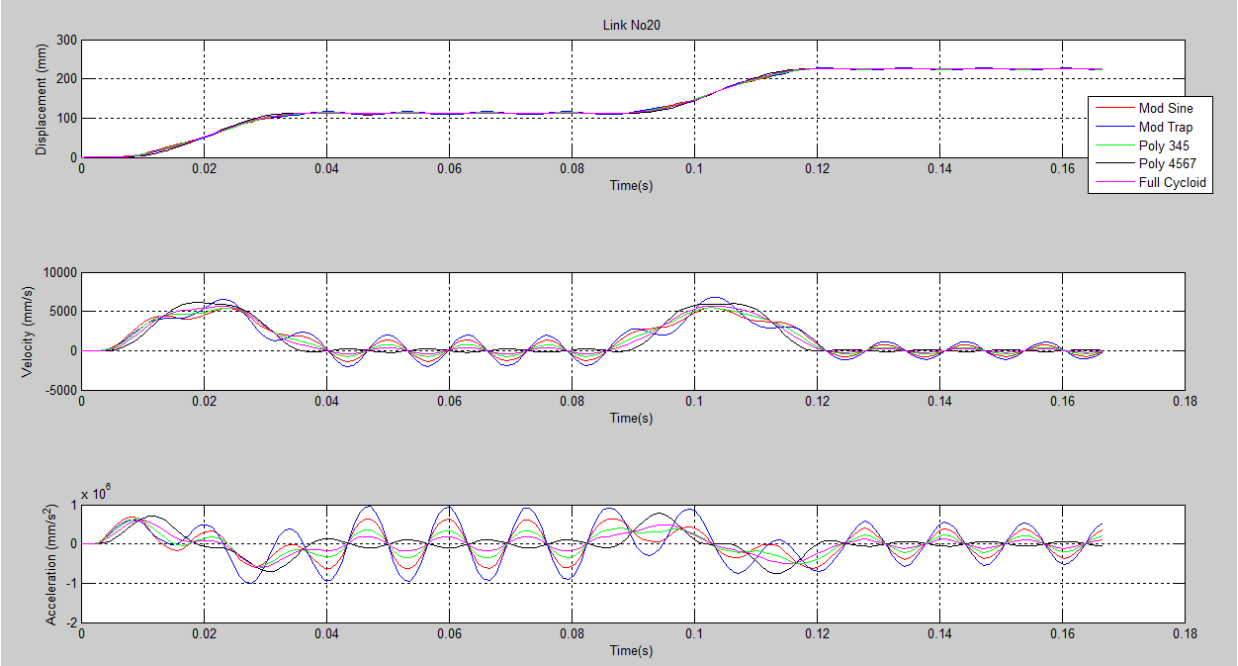

Figure 36: 40 DOF Link 20 Comparison for all Cam Profiles

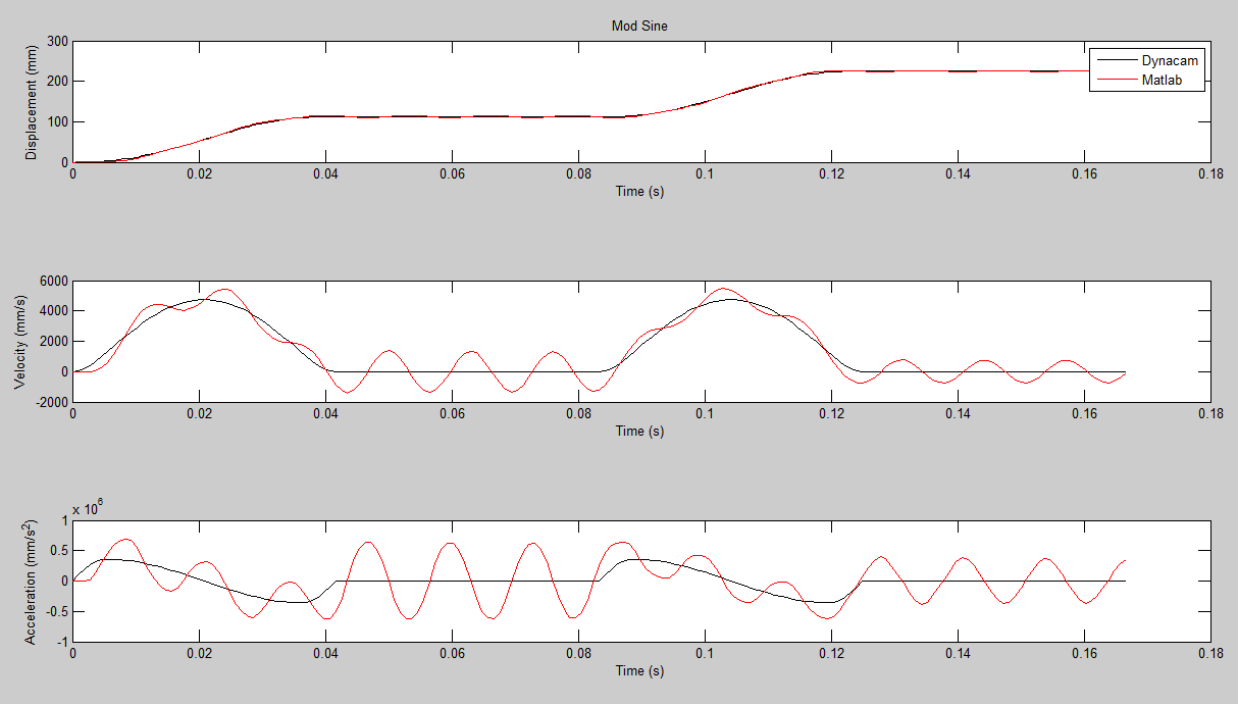

Figure 37: 40 DOF Link 30 Modified Sinusoid

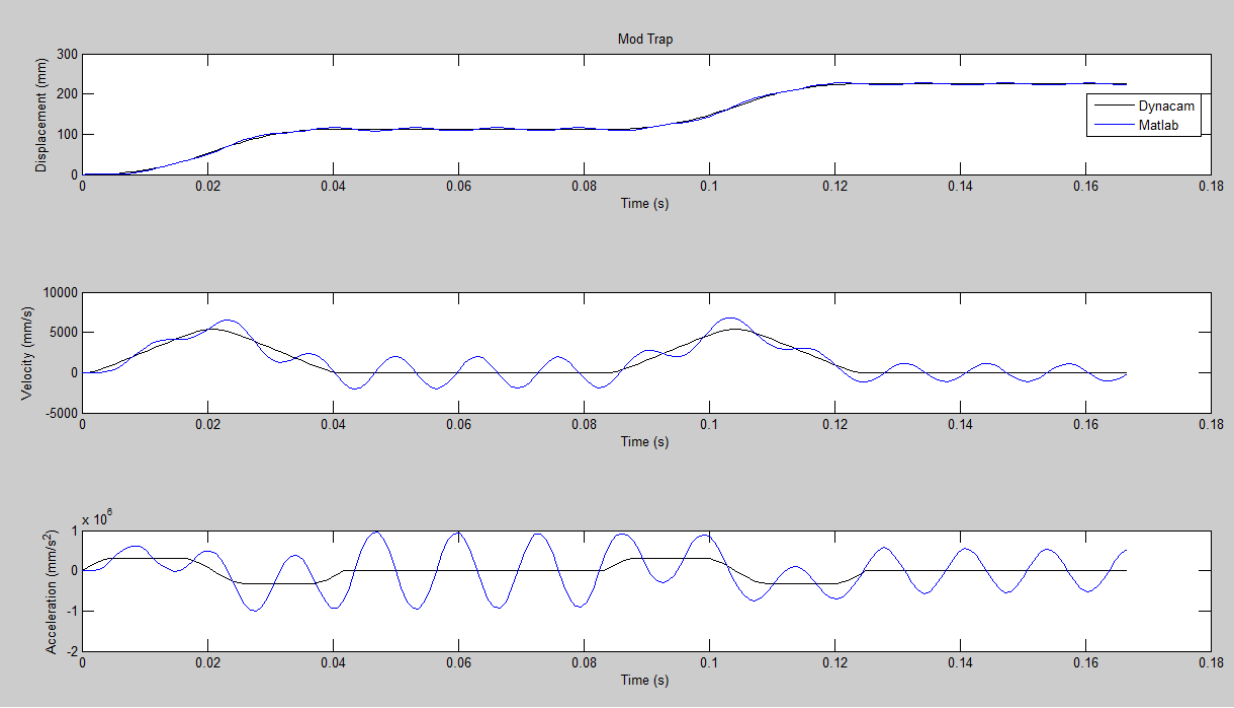

Figure 38: 40 DOF Link 30 Modified Trapezoidal

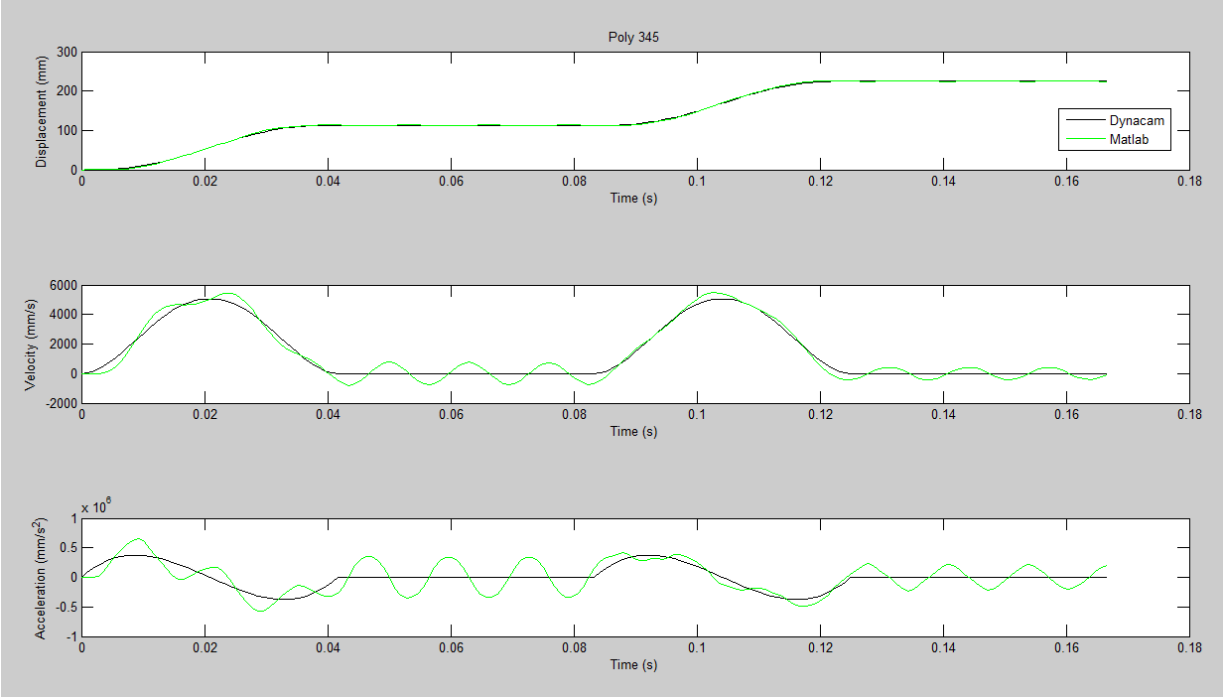

Figure 39: 40 DOF Link 30 3-4-5 Polynomial

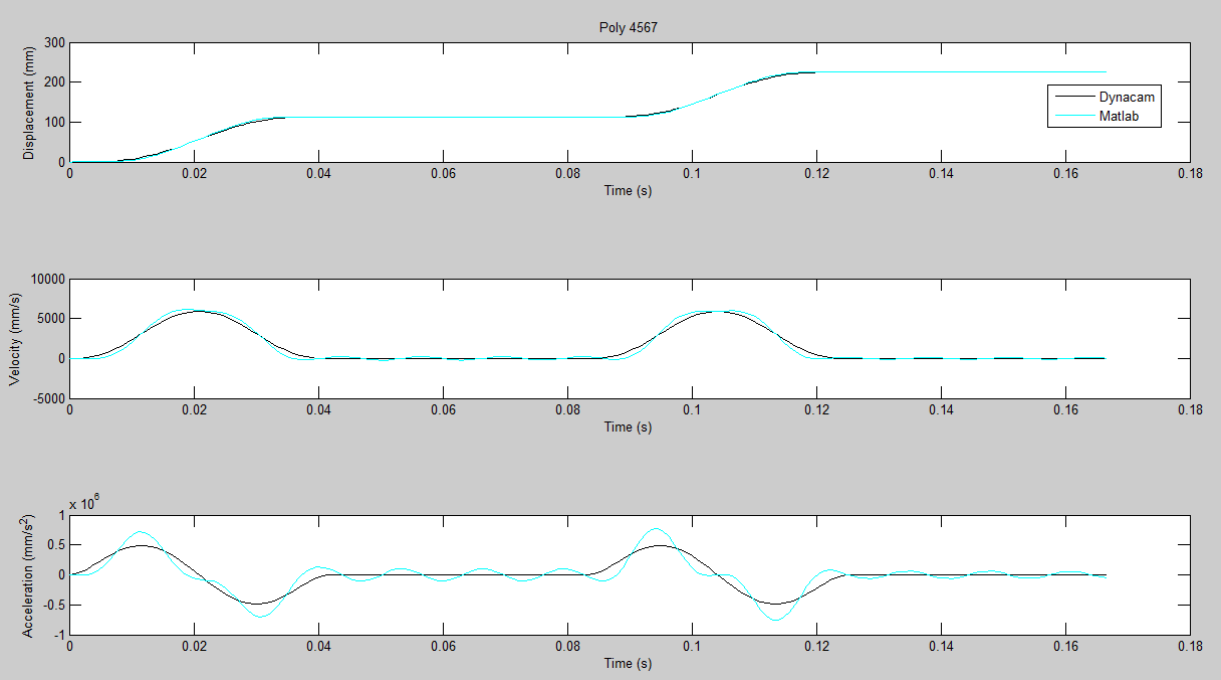

Figure 40: 40 DOF Link 30 4-5-6-7 Polynomial

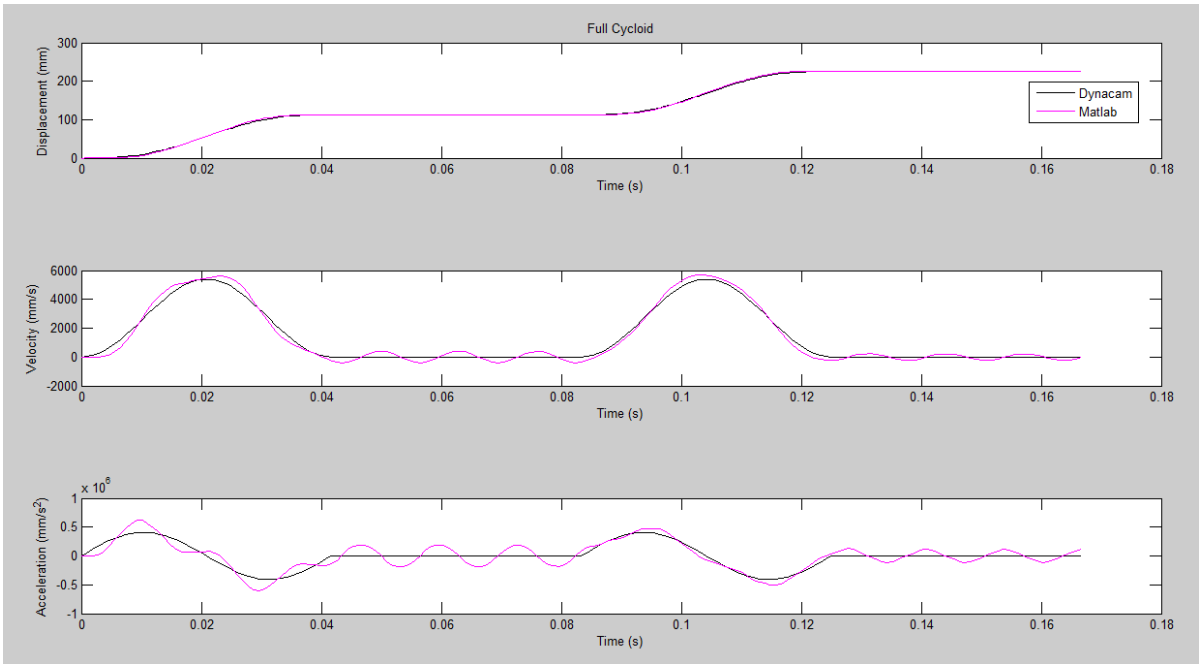

Figure 41: 40 DOF Link 30 Full Cycloid

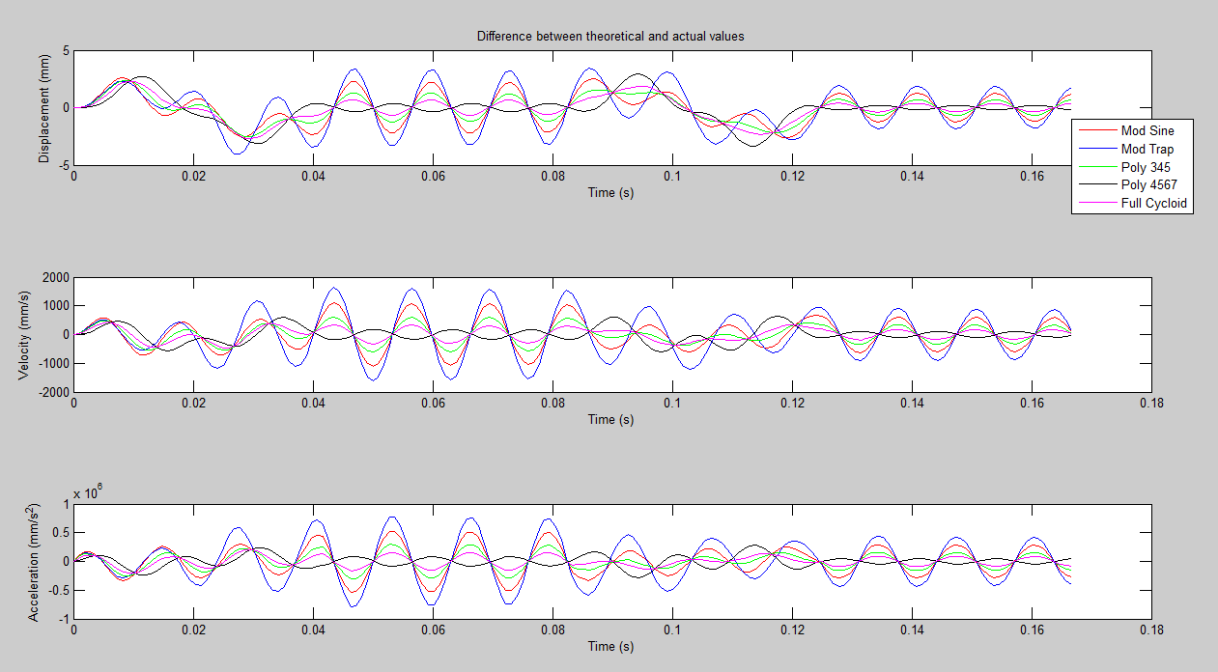

Figure 42: 40 DOF Difference between Link 30 Calculated Values and Dynacam Values

Appendix F: Additional Figures for 85 DOF and Frictional Coefficient of 0.15

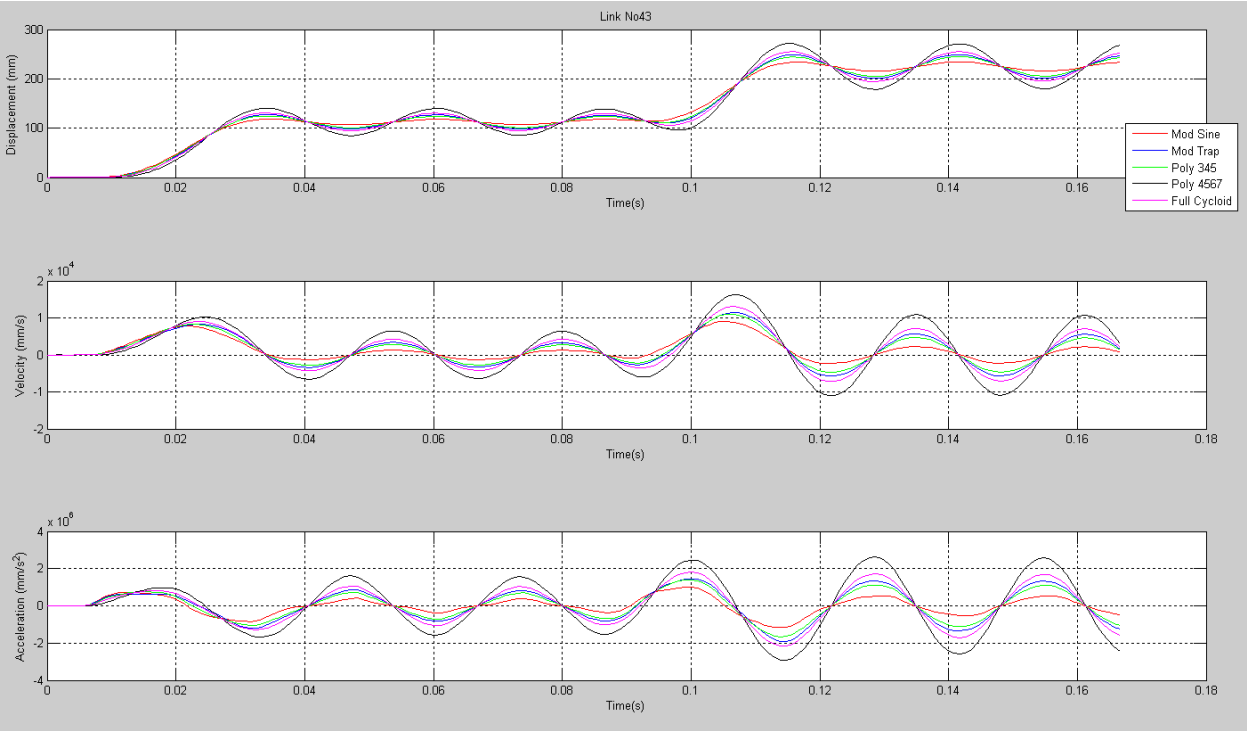

Figure 43: 85 DOF 0.15 Friction Link 43 Comparison for all Cam Profiles

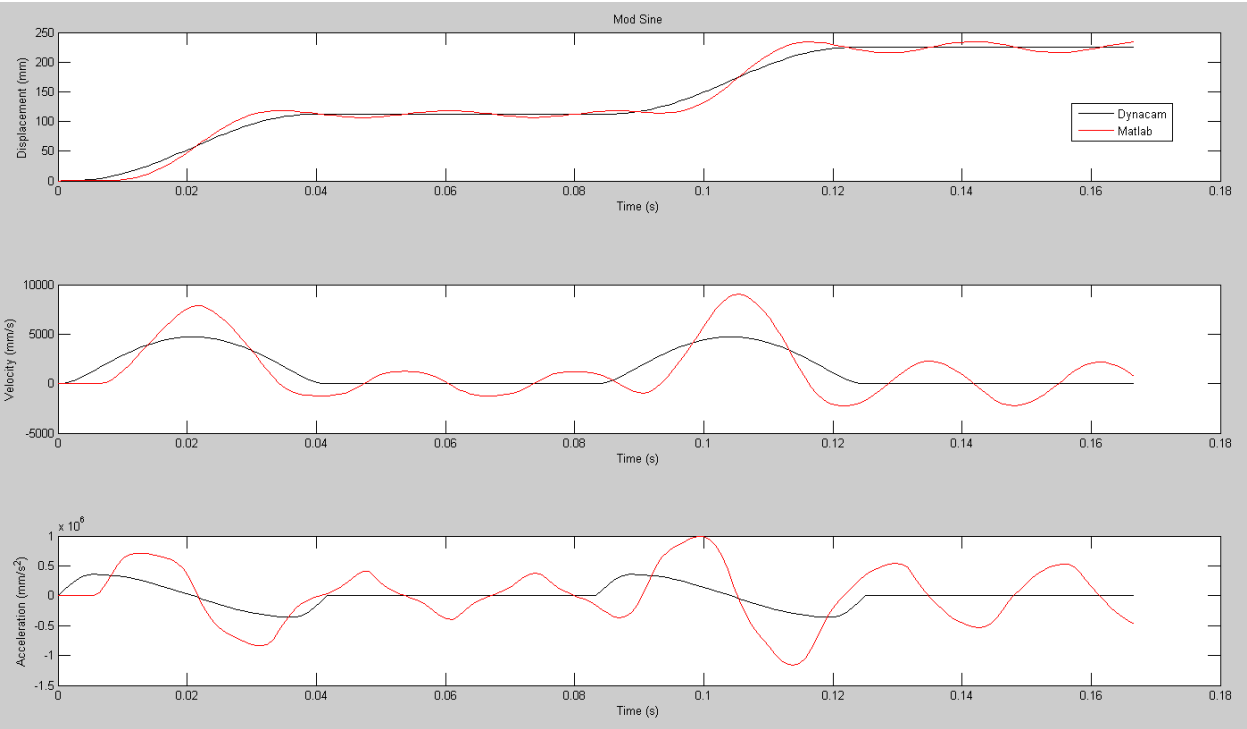

Figure 44: 85 DOF 0.15 Friction Link 63 Modified Sinusoid

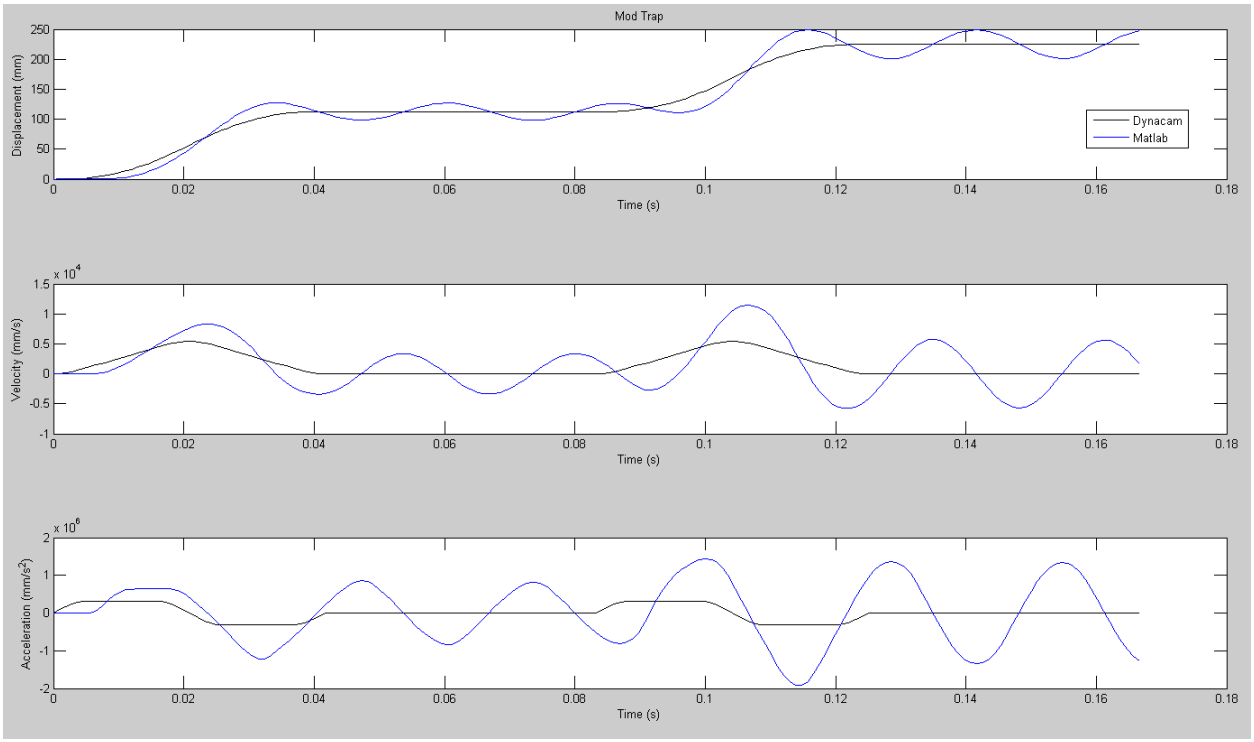

Figure 45: 85 DOF 0.15 Friction Link 63 Modified Trapezoidal

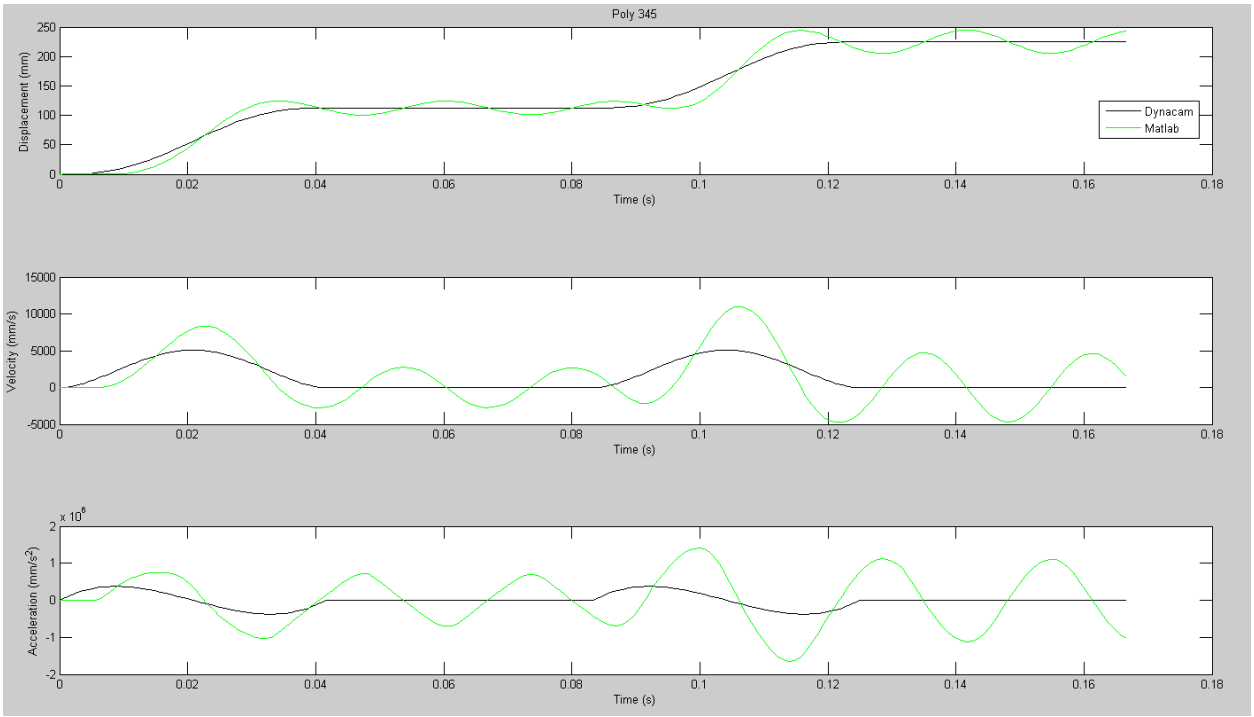

Figure 46: 85 DOF 0.15 Friction Link 63 3-4-5 Polynomial

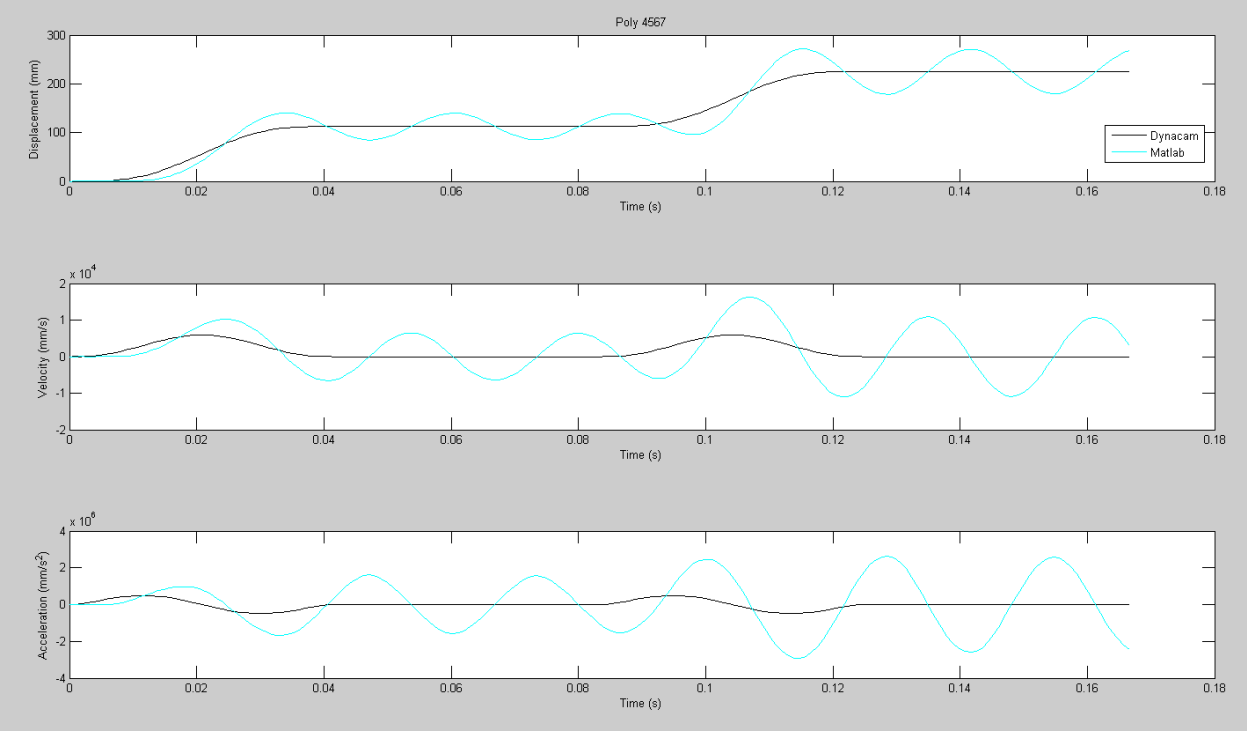

Figure 47: 85 DOF 0.15 Friction Link 63 4-5-6-7 Polynomial

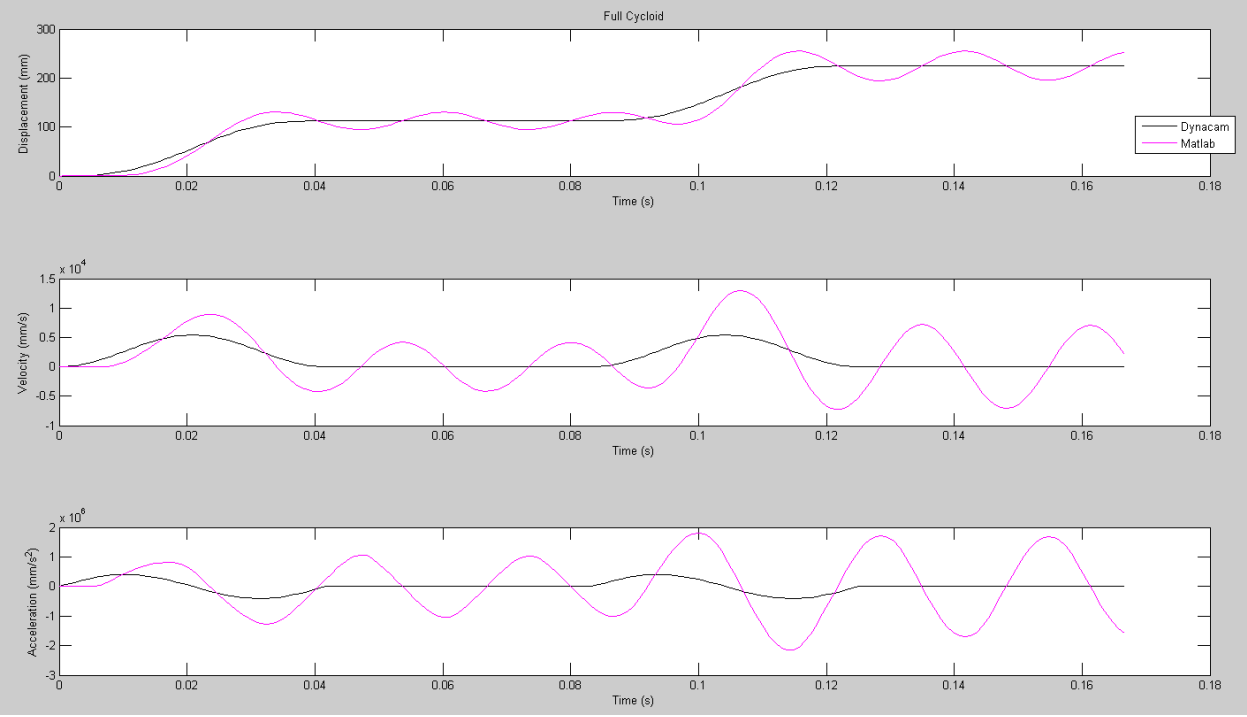

Figure 48: 85 DOF 0.15 Friction Link 63 Full Cycloid

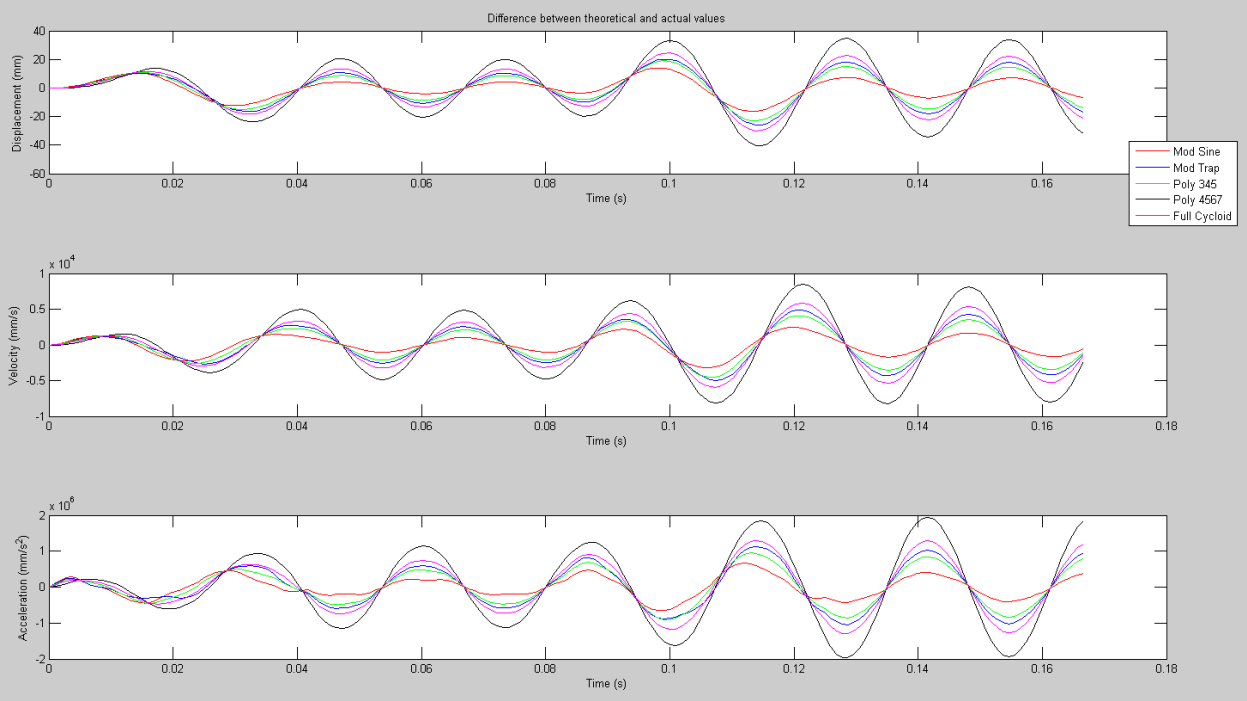

Figure 49: 85 DOF 0.15 Friction Coefficient Difference between Link 63 Calculated Values and Dynacam Values

Appendix G: Additional Figures for 85 DOF and Frictional Coefficient of 0.25

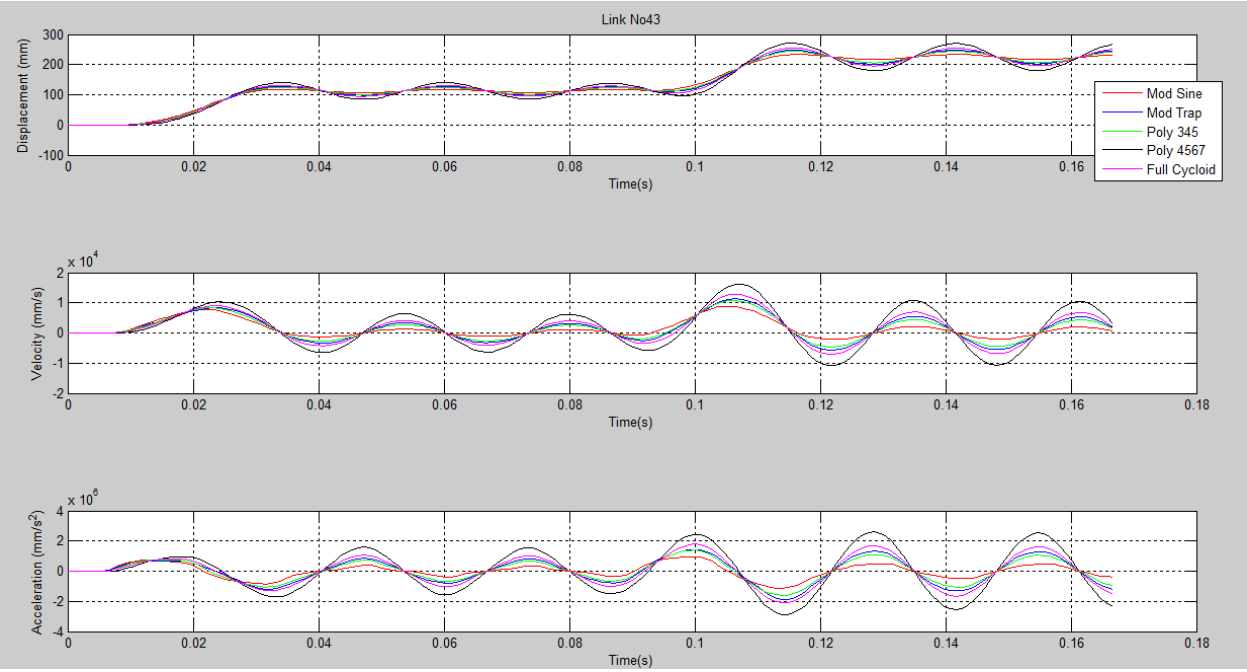

Figure 50: 85 DOF 0.25 Friction Link 43 Comparison for all Cam Profiles

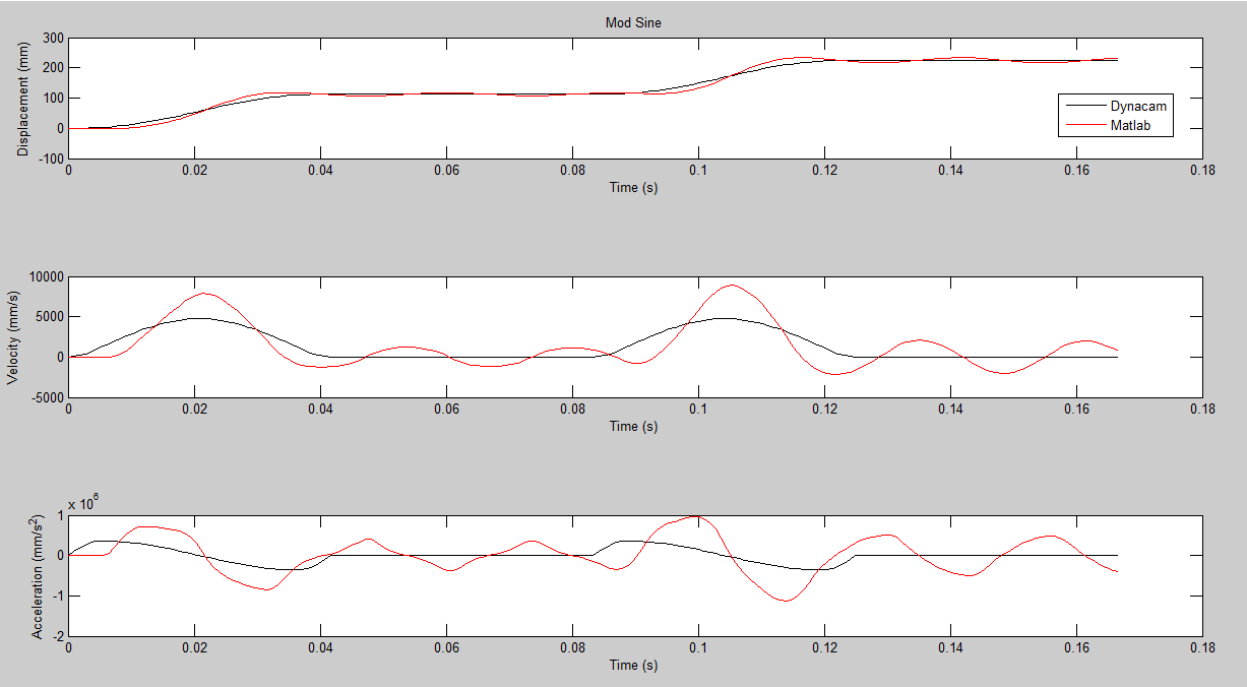

Figure 51: 85 DOF 0.25 Friction Link 63 Modified Sinusoid

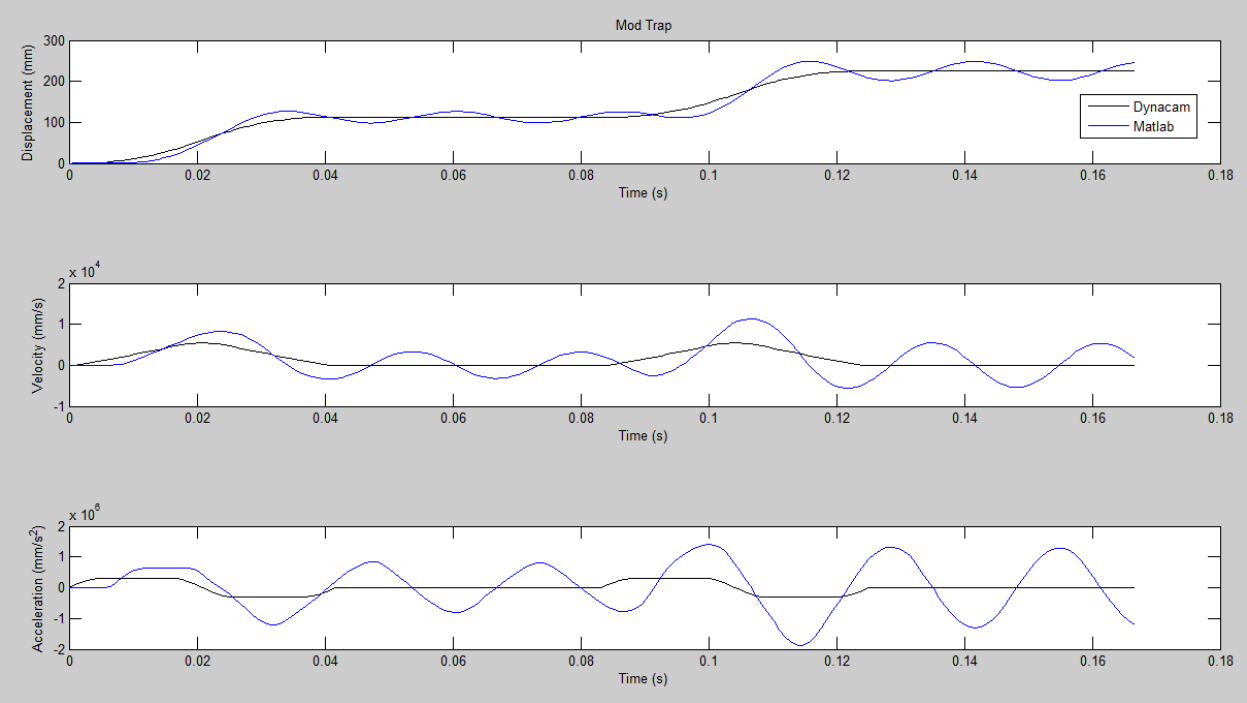

Figure 52: 85 DOF 0.25 Friction Link 63 Modified Trapezoidal

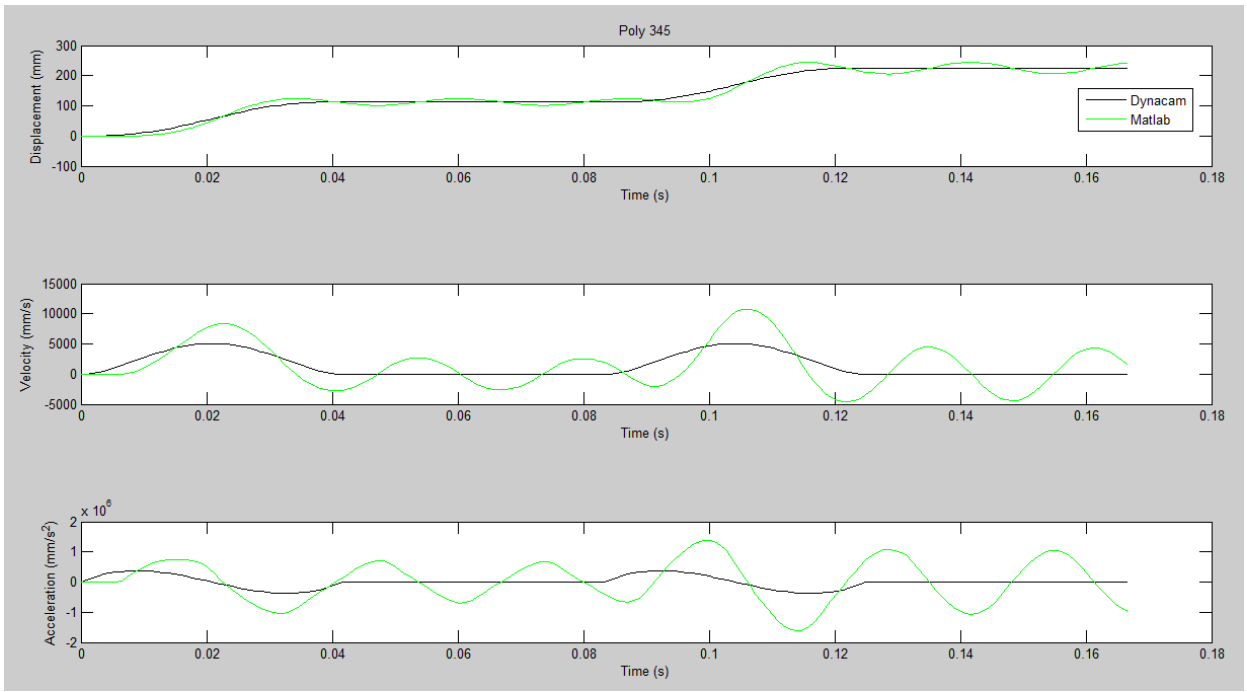

Figure 53: 85 DOF 0.25 Friction Link 63 3-4-5 Polynomial

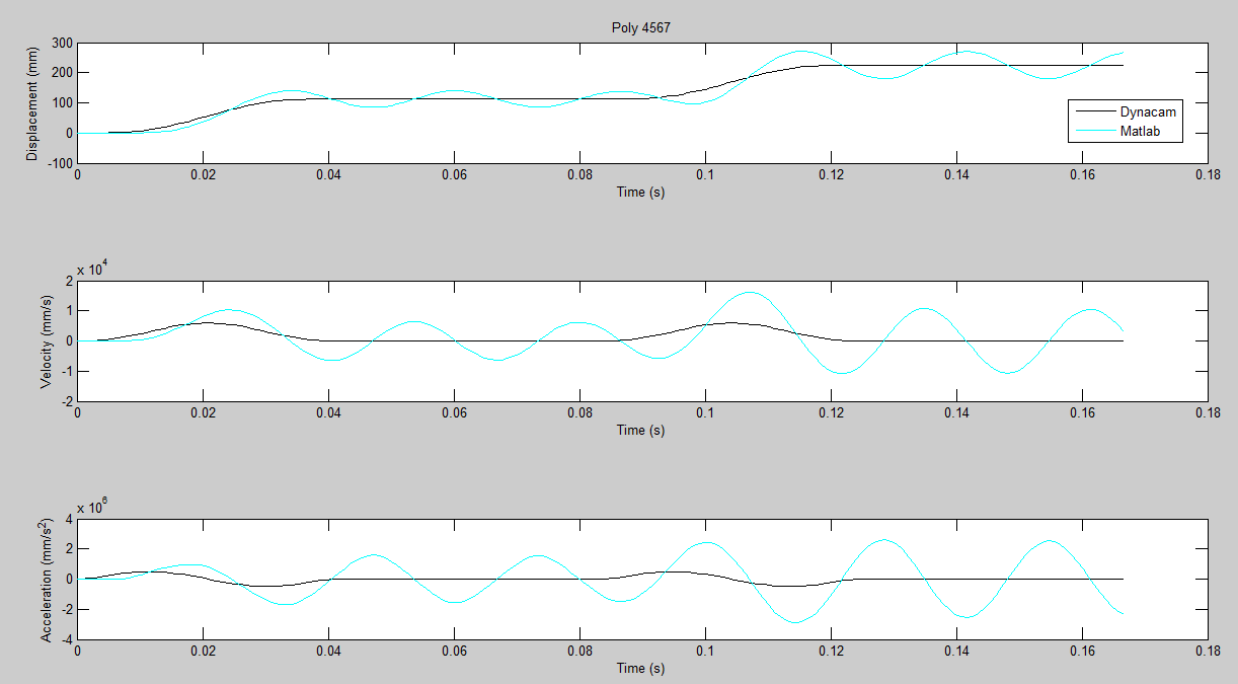

Figure 54: 85 DOF 0.25 Friction Link 63 4-5-6-7 Polynomial

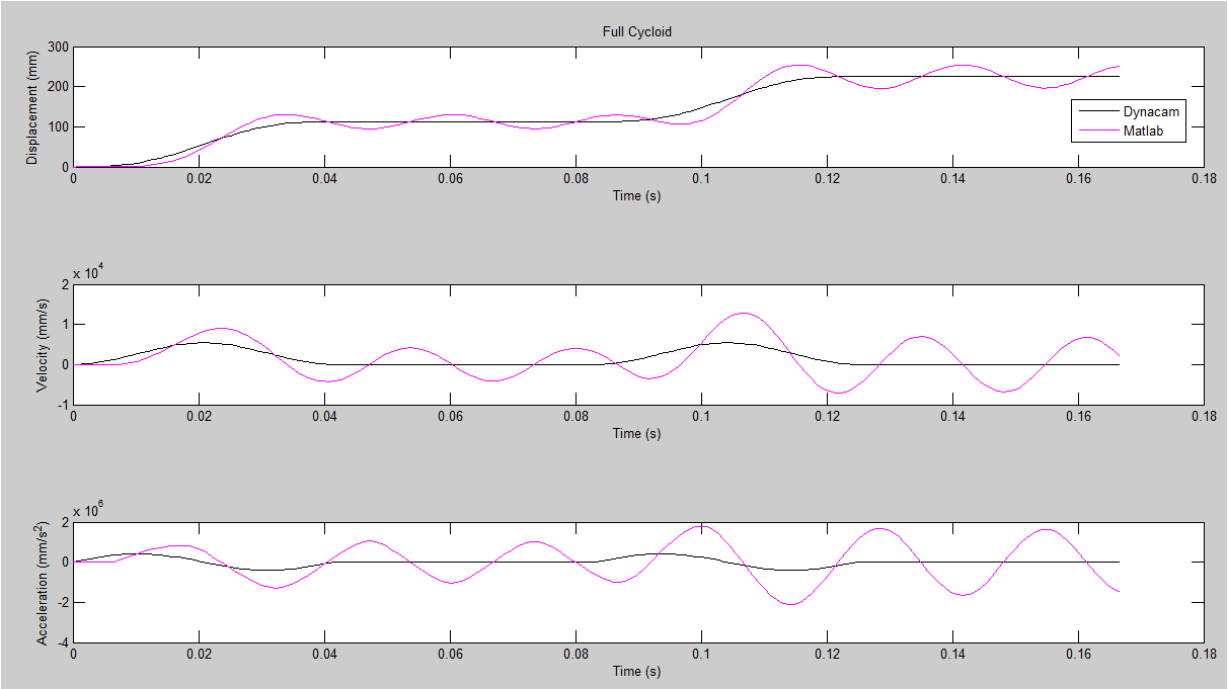

Figure 55: 85 DOF 0.25 Friction Link 63 Full Cycloid

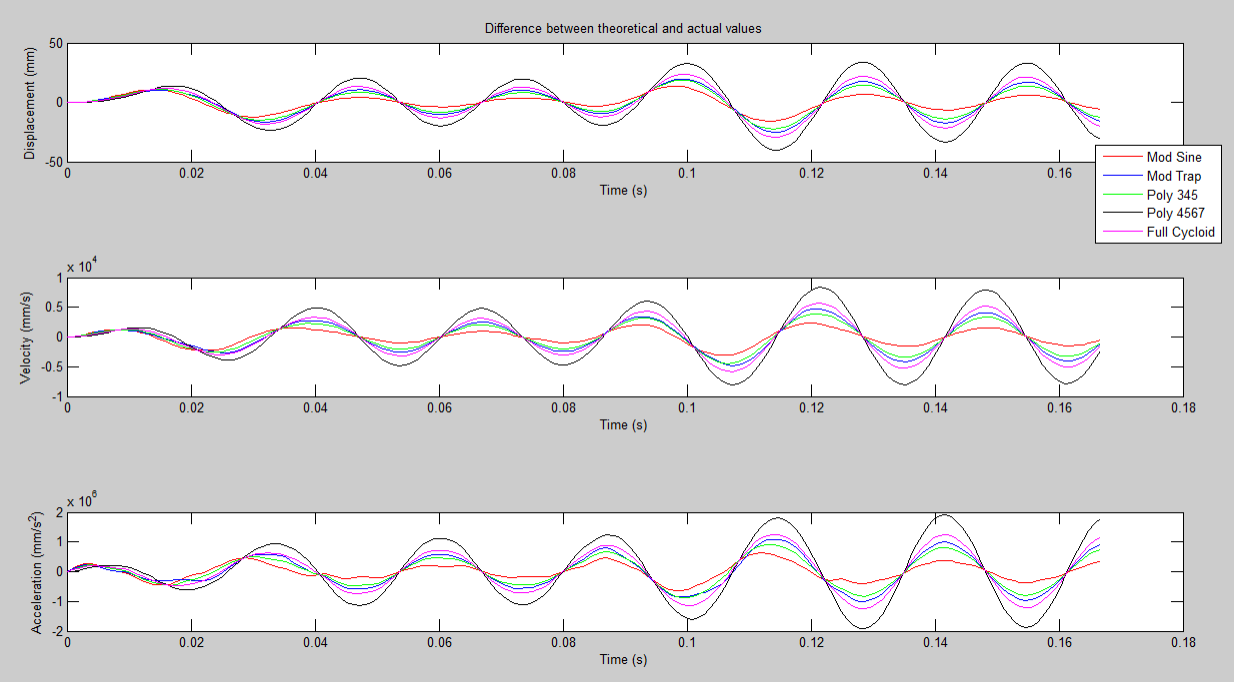

Figure 56: 85 DOF 0.25 Friction Coefficient Difference between Link 63 Calculated Values and Dynacam Values

Appendix H: Additional Figures for 85 DOF and Frictional Coefficient of 0.35

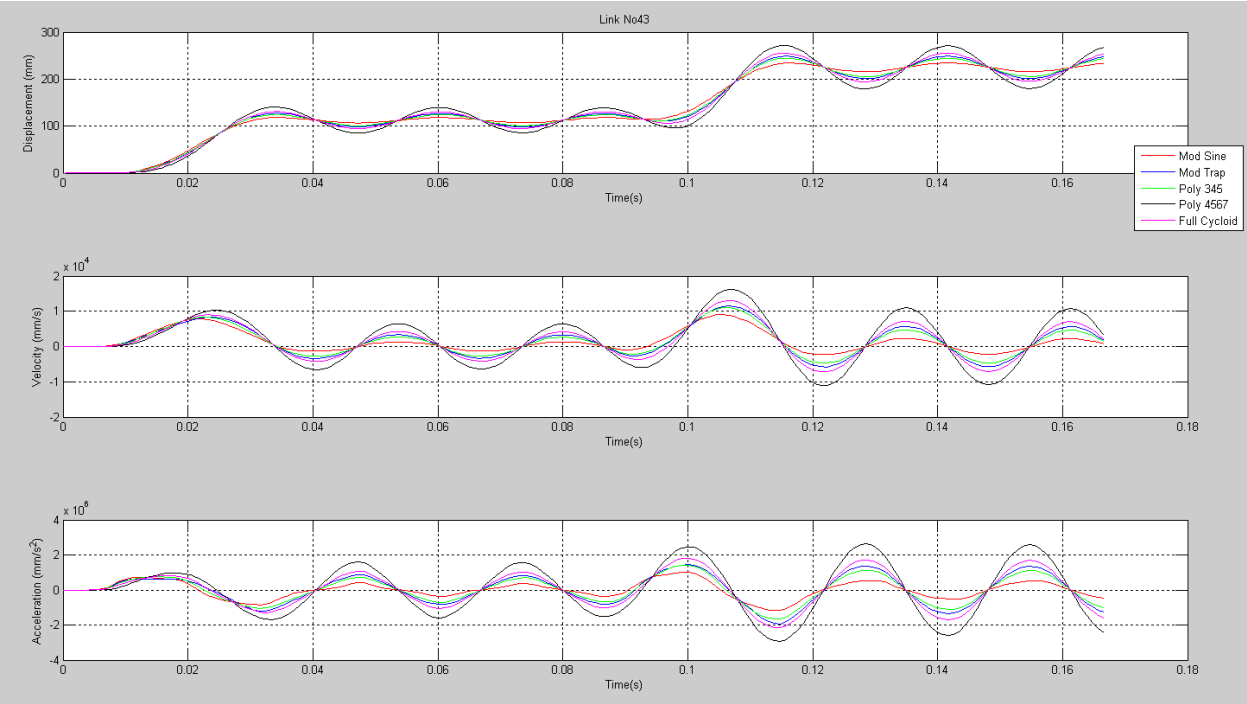

Figure 57: 85 DOF 0.35 Friction Link 43 Comparison for all Cam Profiles

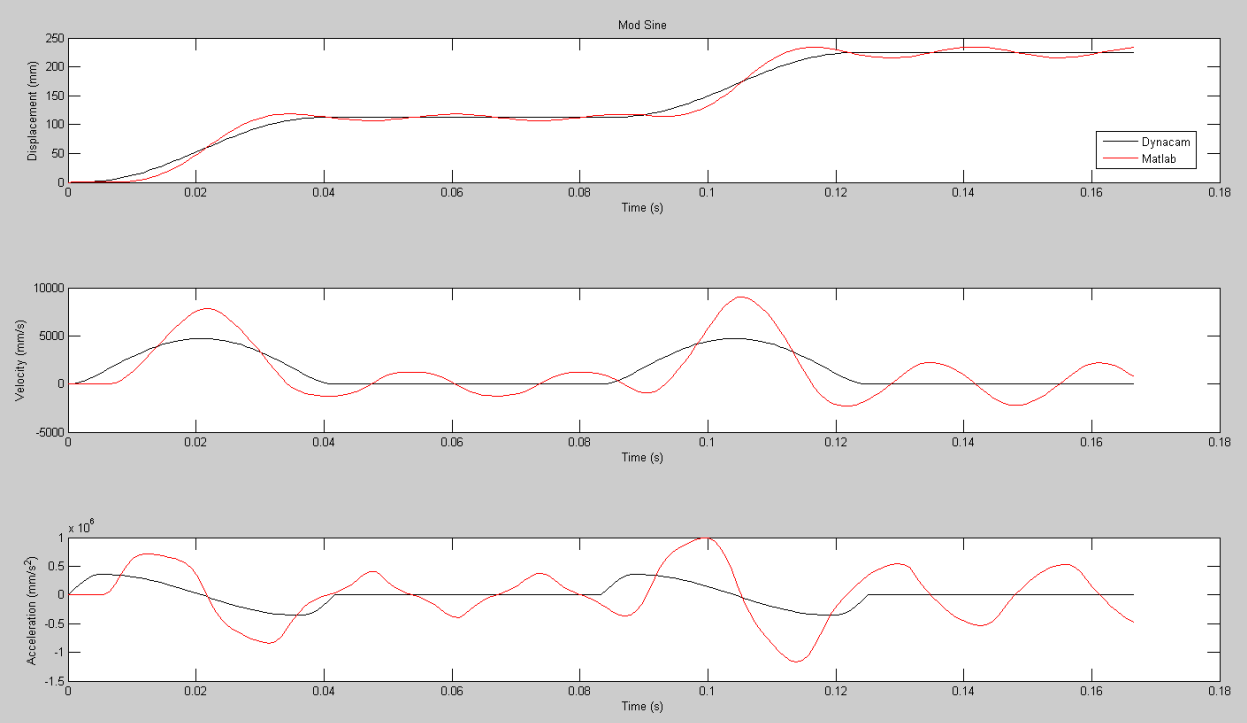

Figure 58: 85 DOF 0.35 Friction Link 63 Modified Sinusoid

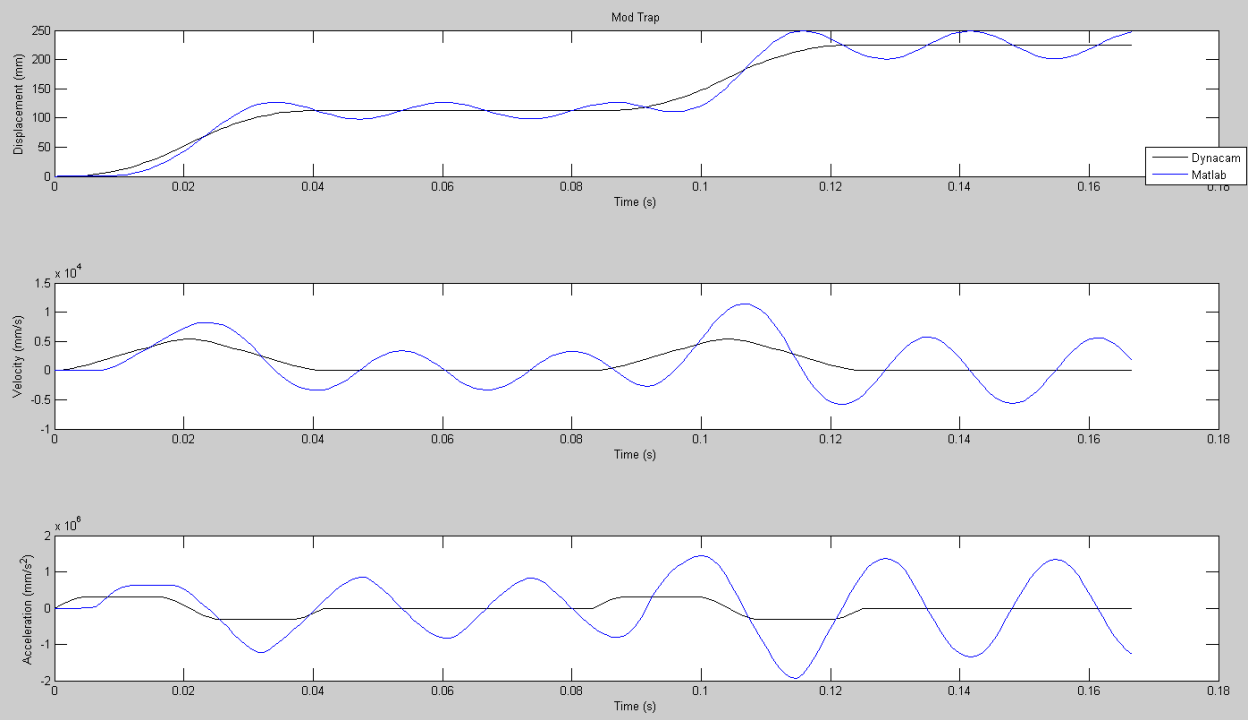

Figure 59: 85 DOF 0.35 Friction Link 63 Modified Trapezoidal

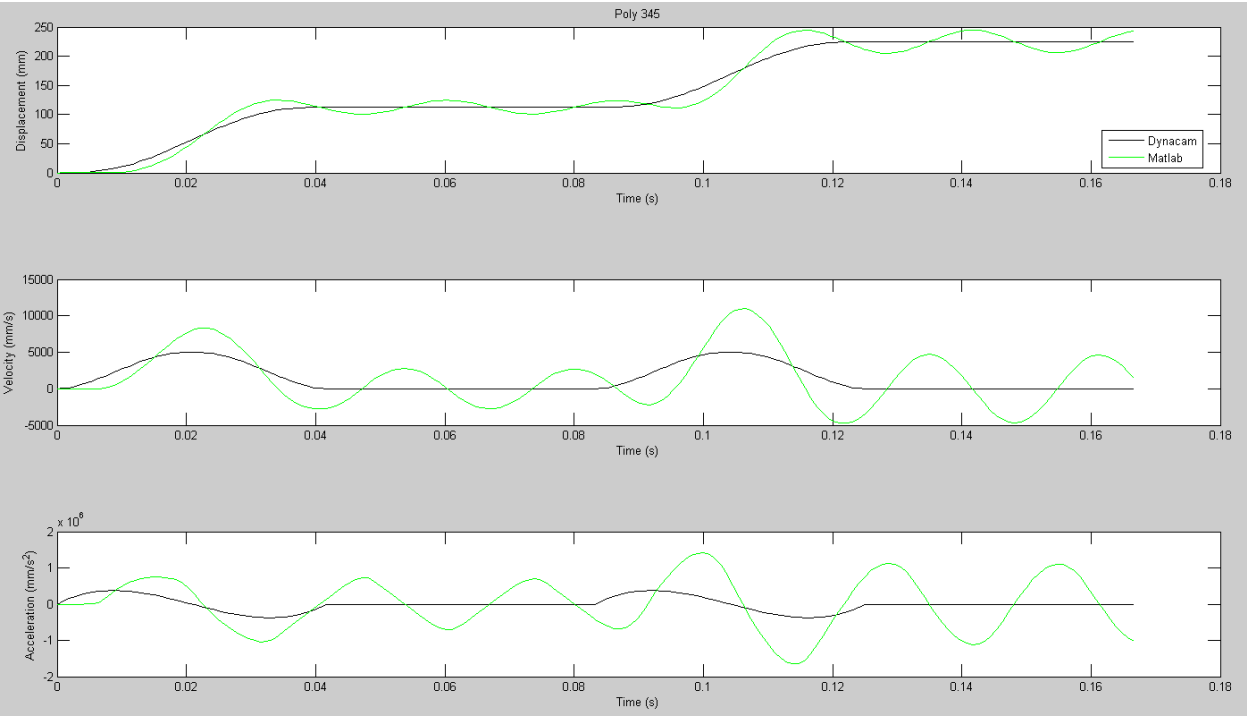

Figure 60: 85 DOF 0.35 Friction Link 63 3-4-5 Polynomial

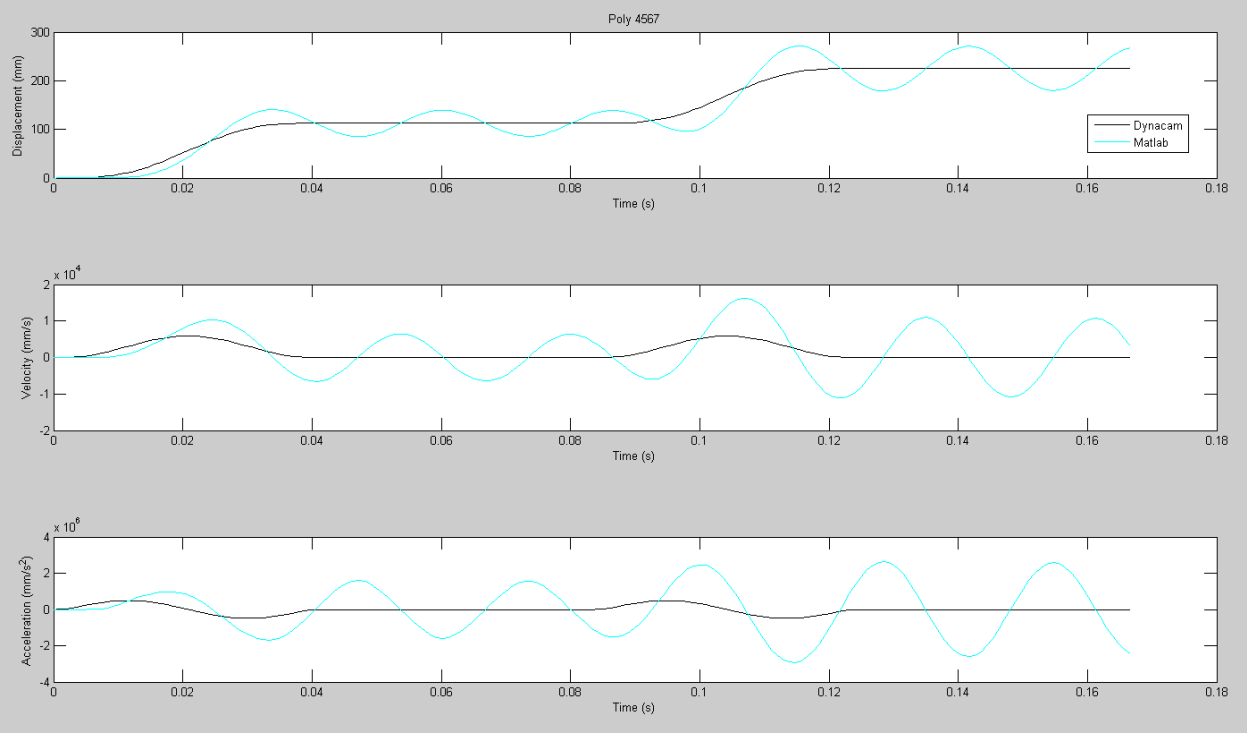

Figure 61: 85 DOF 0.35 Friction Link 63 4-5-6-7 Polynomial

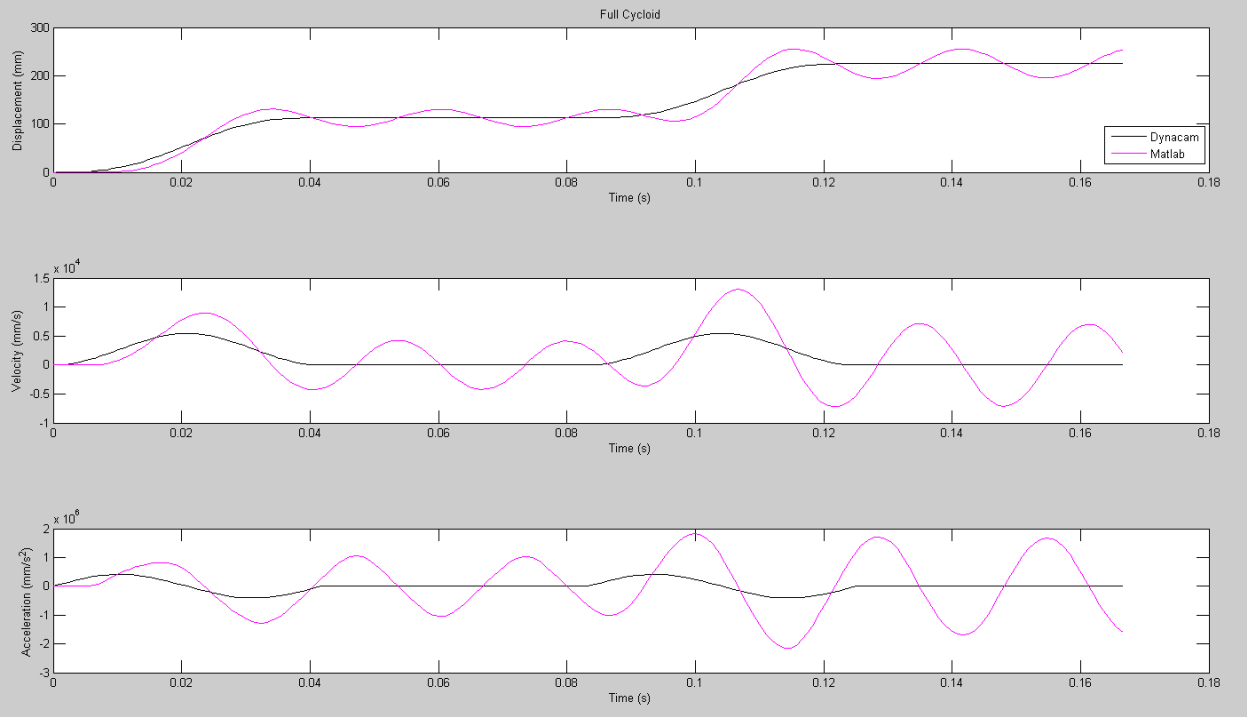

Figure 62: 85 DOF 0.35 Friction Link 63 Full Cycloid

Appendix I: Additional Figures for 85 DOF and Frictional Coefficient of 0.35 for a Titanium Link

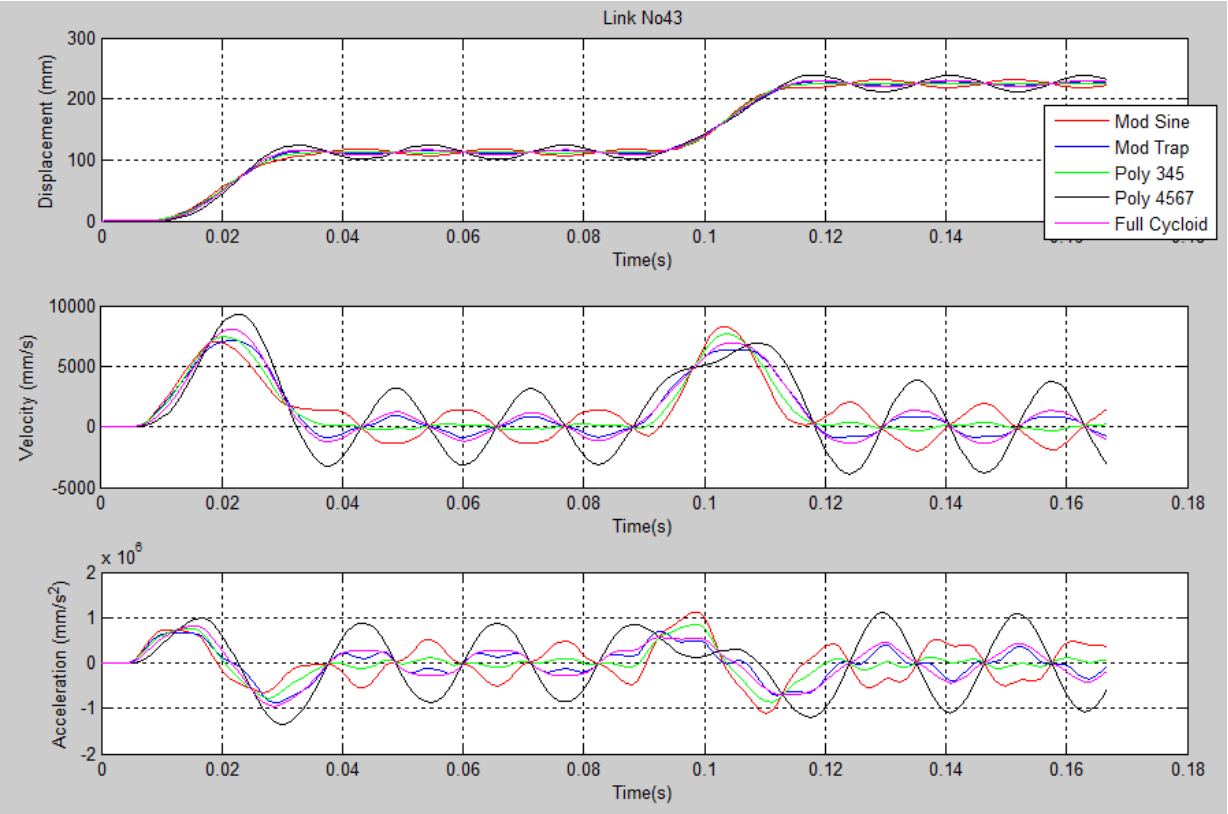

Figure 63: 85 DOF 0.35 Friction Link 43 (Titanium)

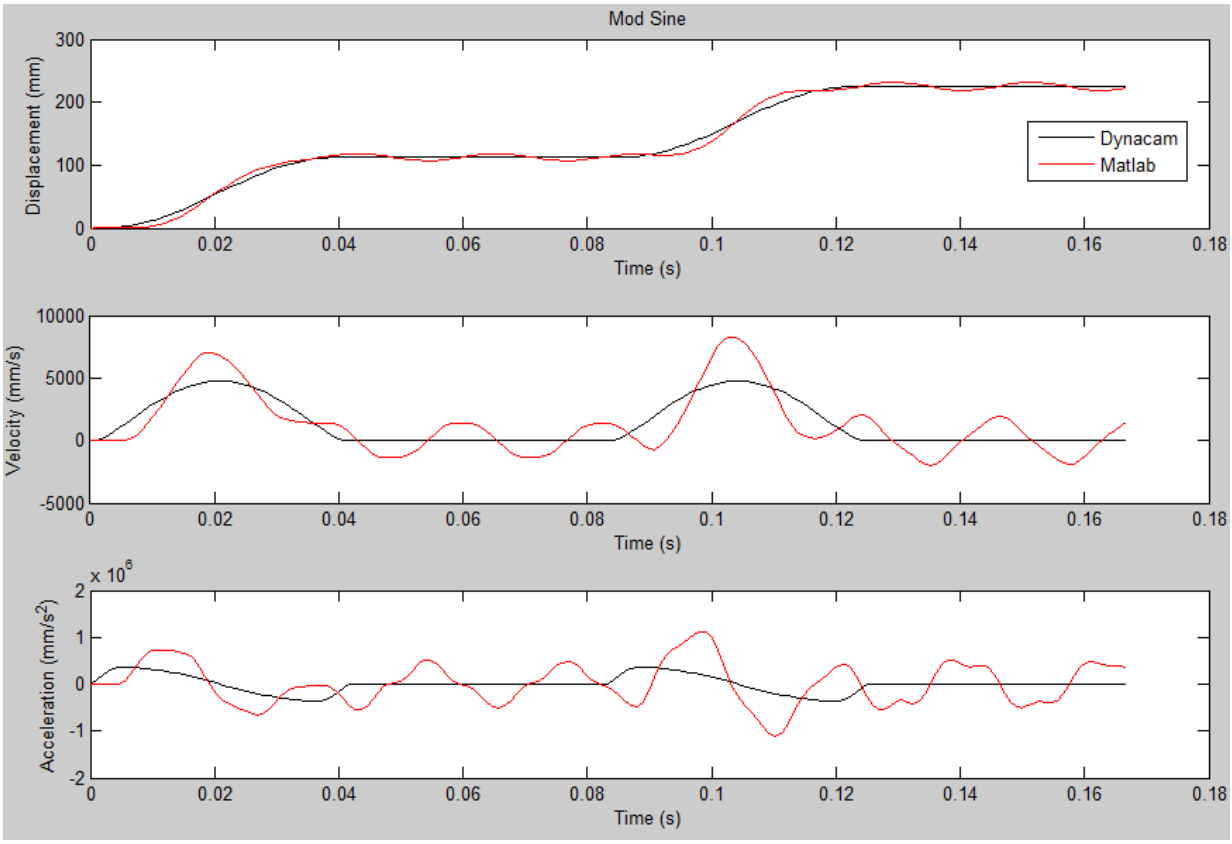

Figure 64:85 DOF 0.35 Friction Link 63 Modified Sinusoid (Titanium)

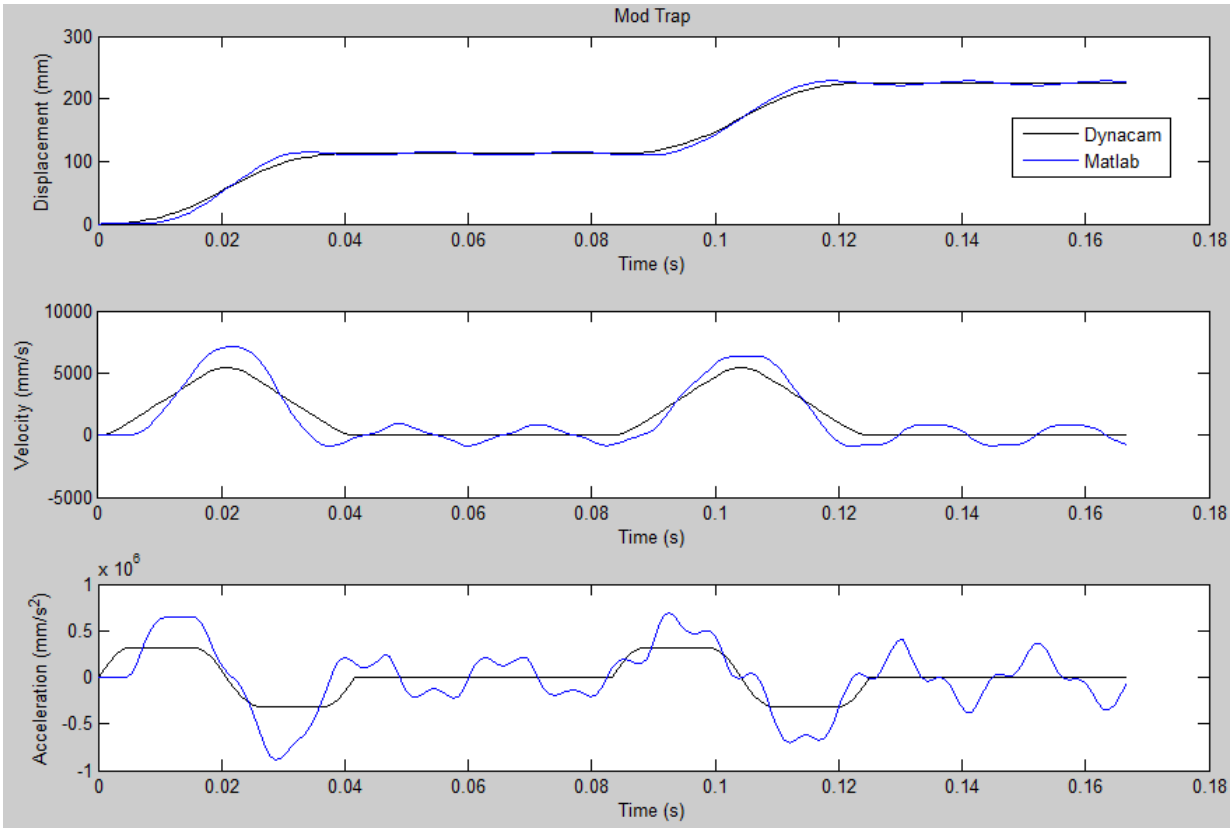

Figure 65:85 DOF 0.35 Friction Link 63 Modified Trapezoidal (Titanium)

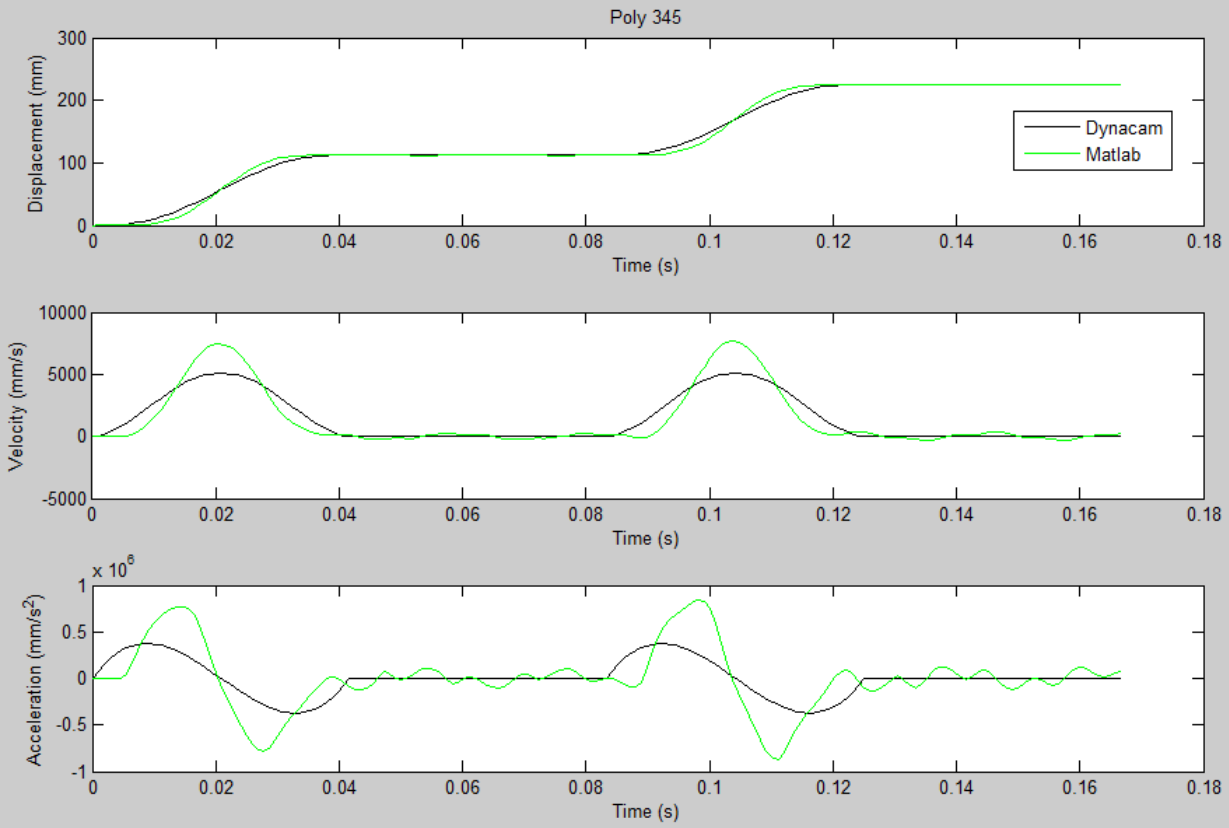

Figure 66:85 DOF 0.35 Friction Link 63 3-4-5 Polynomial (Titanium)

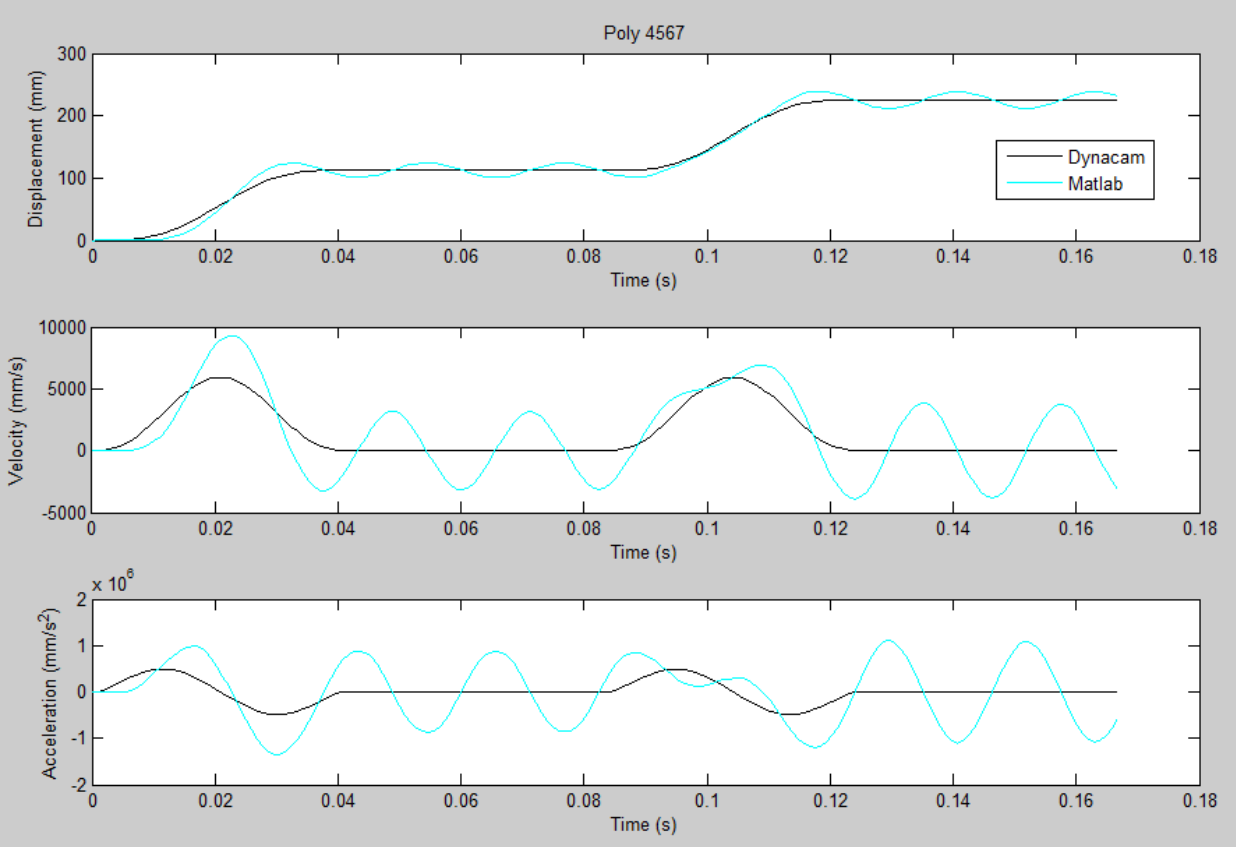

Figure 67:85 DOF 0.35 Friction Link 63 4-5-6-7 Polynomial (Titanium)

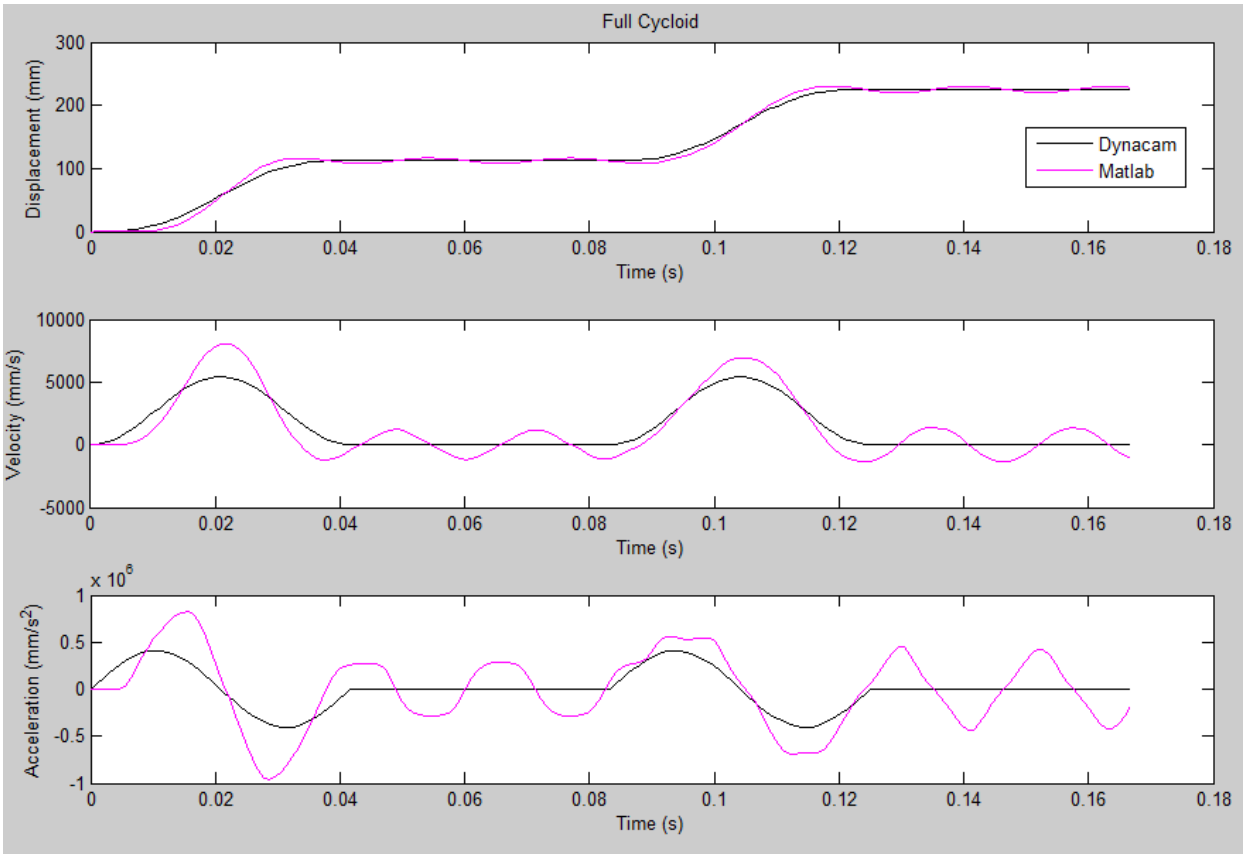

Figure 68: 85 DOF 0.35 Friction Link 63 Full Cycloid (Titanium)

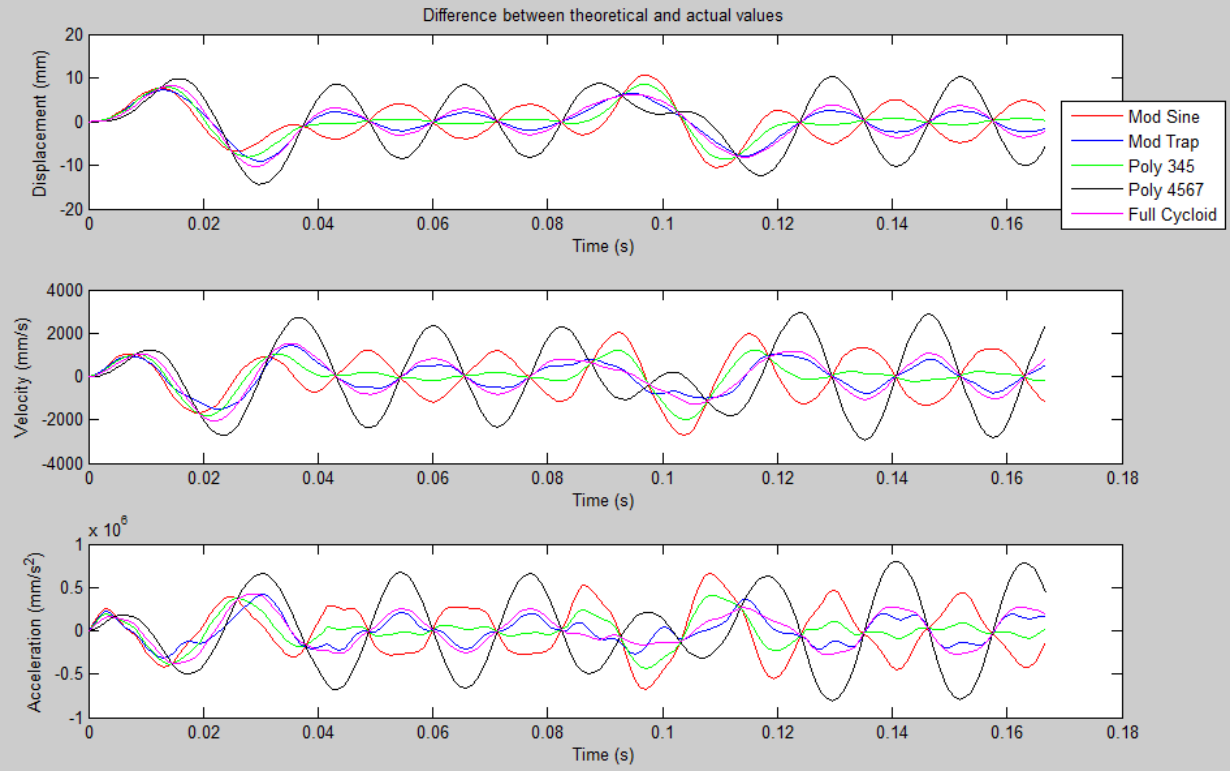

Figure 69: 85 DOF 0.35 Friction Coefficient Difference between Link 63 Calculated Values and Dynacam Values (Titanium)