FRICTION MEASUREMENT DURING FREE VIBRATION

by

Christoph Andreas Johann Kossack

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Mechanical Engineering

Charlotte

2018

Approved by:

_____
Dr. Tony Schmitz

_____
Dr. John Ziegert

_____
Dr. Harish Cherukuri

ABSTRACT

CHRISTOPH ANDREAS JOHANN KOSSACK. Friction Measurement during free vibration. (Under the direction of DR. TONY SCHMITZ)

Friction is a key consideration in the behavior of many dynamic systems. However, modeling friction behavior accurately remains an engineering challenge. Current friction measurement approaches are limited to application specific tests that attempt to mimic physical situations. Furthermore, these tests are typically accompanied by large uncertainties (approximately parts in 10^2).

This research presents a new approach to determine friction behavior that uses velocity measured during free vibration to quantify the energy dissipation in sliding friction contacts. A flexure based friction measuring machine (FMM) was used to conduct friction tests at multiple initial energy input levels. A single parameter Coulomb friction model was used to determine the average dynamic coefficient of friction for a friction contact pair consisting of a polytetrafluouroethylene (PTFE) pin and a polished steel counterface.

The FMM data was also used to study other friction behaviors that have been experimentally observed in prior work, including the Stribeck effect and non-reversible friction. A three parameter dynamic friction model was used to demonstrate the nonlinear dependence of friction on velocity as a system transitions from the stick to slip regimes (Stribeck effect). The presence of non-reversible friction behavior was tested by sectioning the single dynamic coefficient of friction into two components, one for increasing velocity and one for decreasing velocity. Results showed that the friction coefficient was larger for accelerating motion than for decelerating motion.

ACKNOWLEDGMENTS

I would like to express my thanks and appreciation to my mentor and advisor Dr. Tony Schmitz, who through his guidance and support has helped me immeasurably with this research and in my development into an engineer. I would also like to thank Dr. John Ziegert for his assistance and invaluable insight with this research project. I also want to thank Dr. Harish Cherukuri for being part of my thesis committee and for teaching me the engineering analysis tools that I needed to be successful in my research and as an engineer.

I would also like to thank Dr. Andrew Honeycut for taking the time to teach me how to use the equipment in our lab and for always being willing to assist me when I needed help. I would also like to thank my wife Hilary for her continued support and patience with me during my undergraduate and graduate endeavors.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Although friction is a key consideration in many engineering systems, it remains only partially understood. Friction plays an important role in the performance of many dynamic systems, but measuring and modeling friction behavior accurately remains an engineering challenge. This is due, in part, to the fact that current friction measurement tests are typically accompanied by large uncertainties (approximately parts in 10^2 [1]). Force based tribometers rely on precise angular alignment of the applied normal force. Even small misalignments can result in significant errors when determining friction coefficients. Furthermore, these tests are dependent on the accuracy and repeatability of a continuous input force that needs to be applied to create the normal force and sliding motion. It is common practice to build application specific tests that measure the friction behavior for a contact pair at specific velocities. This requires constant velocity conditions. The friction behavior for instantaneous changes in velocity or directional changes are therefore often not quantified.

This research analyzes the effectiveness of a new friction measurement approach that parameterizes friction models using the friction energy dissipation of a freely oscillating system. A friction measuring machine (FMM) that provides linear motion between a pin and a counterface was designed and built. The FMM uses four leaf springs in a parallelogram leaf-type structure arrangement to provide the relative motion for the contact pair. By commanding an initial displacement from equilibrium to the FMM the leaf springs are loaded with the potential energy that results in motion upon release. Once the FMM is released, a laser vibrometer is used to measure velocity as the FMM oscillates

freely until it comes to rest. There is no additional force applied to the system during oscillation.

The modal parameters for the FMM were determined by analyzing velocity data from free vibration tests with no a friction contact. The energy dissipation rate caused by the friction contact pair is quantified by comparing the free vibration to friction contact results. Furthermore, this measurement technique provides data for all friction regimes: acceleration/deceleration, static/sliding friction, and the transition from static to sliding friction (Stribeck effect).

A single parameter Coulomb friction model was used to demonstrate how the vibrometer velocity data can be used to determine the average coefficient of friction for the contact pair. Based on this friction model a Monte Carlo simulation was performed to determine the sensitivities and uncertainty contributions for the modal parameters (mass, stiffness, and viscous damping). Acceleration dependent and Stribeck effect friction models were also studied using the FMM data.

CHAPTER 2: LITERATURE REVIEW

"Energy loss due to friction and the failure of equipment due to wear represent a considerable percentage of every modern economy" [4] and "it is estimated that one-third to one-half of the world's energy production is wasted through friction" [24]. Considering the enormous impact that friction has on dynamic systems, it is apparent that the ability to model and predict friction behavior is critical. This remains an engineering challenge, however, because friction behavior is not yet completely understood.

The abundance of different friction models can make the selection of the appropriate model for an application a challenging task. It is common practice to build application specific friction tests that attempt to mimic the dynamic system as closely as possible. Tribological tests are typically classified "according to their degree of realism, i.e. how closely they imitate the conditions of a real application." [10]. The need to design, build and test a new friction measuring machine for every new application requires significant time and money and is, therefore, not a preferred solution.

Standard tests exist in the form of tribometers, such as pin-on-disk geometries. These tests suffer from large uncertainties, however. Normal and tangential force components as well as surface velocities must be applied and measured accurately. "Misalignments between the force measurement axis and the directions axes and the directions normal (N) and tangent (T) to the reciprocating or rotating surface constitute one of the most significant contributors to friction coefficient measurement errors." [21]. It is the goal of this research to develop a new friction measurement technique that will reduce these uncertainties and then determine the appropriate friction model to quantify test results.

Modeling friction is a challenging process. There are a multitude of parameters that need to be considered, depending on the selection model. “Experiments indicate a functional dependence upon a large variety of parameters, including sliding speed, acceleration, critical sliding distance, temperature, normal load, humidity, surface preparation, and material combination” [11]. Contamination of the friction contact pair interface adds additional complexity because, even the presence of small particles, additional forces are present that can alter friction coefficient results [12].

One of the most difficult parameters to determine is the break-away coefficient of friction, also known as the static coefficient of friction. Determining a single static coefficient of friction for a contact pair can prove difficult, as demonstrated in experiments performed by Li [20] while attempting to obtain the static friction coefficient for an aluminum and glass contact pair. Li found that there were variations in the measured static coefficient of friction and noted that complications caused by “reactions between surfaces, interlocking surface features” [20] and other sources were the cause of this problem.

This “multi-valued friction at zero relative velocity” [11] problem stems from the fact that the true surface contact area between two samples is never exactly the same. Surfaces are never completely smooth and thus have perturbing features called asperities and it is the contact points of these asperities that define the true surface contact area [4,10,13]. Therefore, when one material surface slides across the surface of the other to a new position, the number of surface asperities in contact between materials changes. By extension, the size and stiffness of the new contact asperities has also changed, altering the overall sticking force between the surfaces. This phenomenon causes challenges when trying to identify a single static coefficient of friction. An additional consideration is dwell

time between surfaces, or the time that a contact pair is at zero relative velocity. Increases in dwell time can cause surfaces to sink further into one another and becoming more locked together. This also causes inconsistencies when determining static friction forces.[11,4,2,14]. This problem was encountered in this research when the initial Coulomb friction model included a static friction coefficient, μ_s, along with the dynamic friction coefficient, μ_d. The fitting results for μ_d were consistent for multiple tests at a specific initial displacement, yet μ_s would be so random that its standard deviation was as much as 100% of its mean. For this reason, a single parameter Coulomb friction model was adopted to determine the average dynamic coefficient of friction.

The challenges of determining static friction coefficients lead to another friction modeling obstacle: pre-sliding displacement. As mentioned before, the surface contact area consists of a large number of asperities in contact with one another. These asperities have an associated stiffness and they are the reason that “friction behaves like a spring if the applied force is less than the break-away force” [2]. This means that once force is applied, there is a region of motion as the spring-like asperities are bent before any sliding motion begins. “There is a displacement (presliding displacement) which is an approximately linear function of the applied force, up to a critical force, at which breakaway occurs” [4]. Friction models such as the Dahl [3] and LuGre [6,15,19] models account for this by adding a spring like effect into the friction model. This is done by assuming that the asperities are a large array of bristles with an average spring stiffness. These models include parameters that need to be experimentally determined using constant velocity tests and were therefore not included in the selection of friction models studied in this research. With the addition

of constant velocity data for the contact pair, however, these models could be applied to the FMM data.

Another friction behavior component is the non-reversibility of friction force. “Hysteretic characteristics of friction can also be observed during regressive (two-way) oscillations with macroscopic sliding. Some experimental and analytical studies confirm the existence of different slopes of friction force for the acceleration and deceleration phases” [14]. This is especially true around zero velocity. The friction coefficient is therefore a function of both relative velocity and acceleration. This friction behavior was examined in this research by determining two dynamic friction coefficients for the oscillating FMM: one for the accelerating phases and one for the decelerating phases. The results confirmed “observations of friction loops in which friction is a function of sliding acceleration as well as sliding velocity” [11].

The Stribeck effect is a friction component that is a large contributor to the friction forces in the FMM. “Stribeck’s work has shown a nonlinear transition from stick to slip” [14]. Each velocity data set obtained from the FMM features numerous passes through zero velocity and therefore provides sliding behavior data for this transition phase. There have been many attempts to model the Stribeck effect and this research applies some of the resulting Stribeck functions to determine friction coefficients. A modified version of a continuously differentiable friction model [1], a three parameter velocity dependent model [11], and new friction functions were tested in an attempt to model the decaying behavior observed in the oscillating FMM data.

CHAPTER 3: FRICTION MEASURING MACHINE

3.1 FMM Components

The FMM was designed so that it provides linear motion between a pin and a counterface for friction contact tests. Four leaf springs that are arranged in a parallelogram leaf-type flexure configuration provide the nominally linear motion, as can be seen in Figure 1. The horizontal oscillation of the leaf springs provides the motion for the system, eliminating the need for any external input forces.

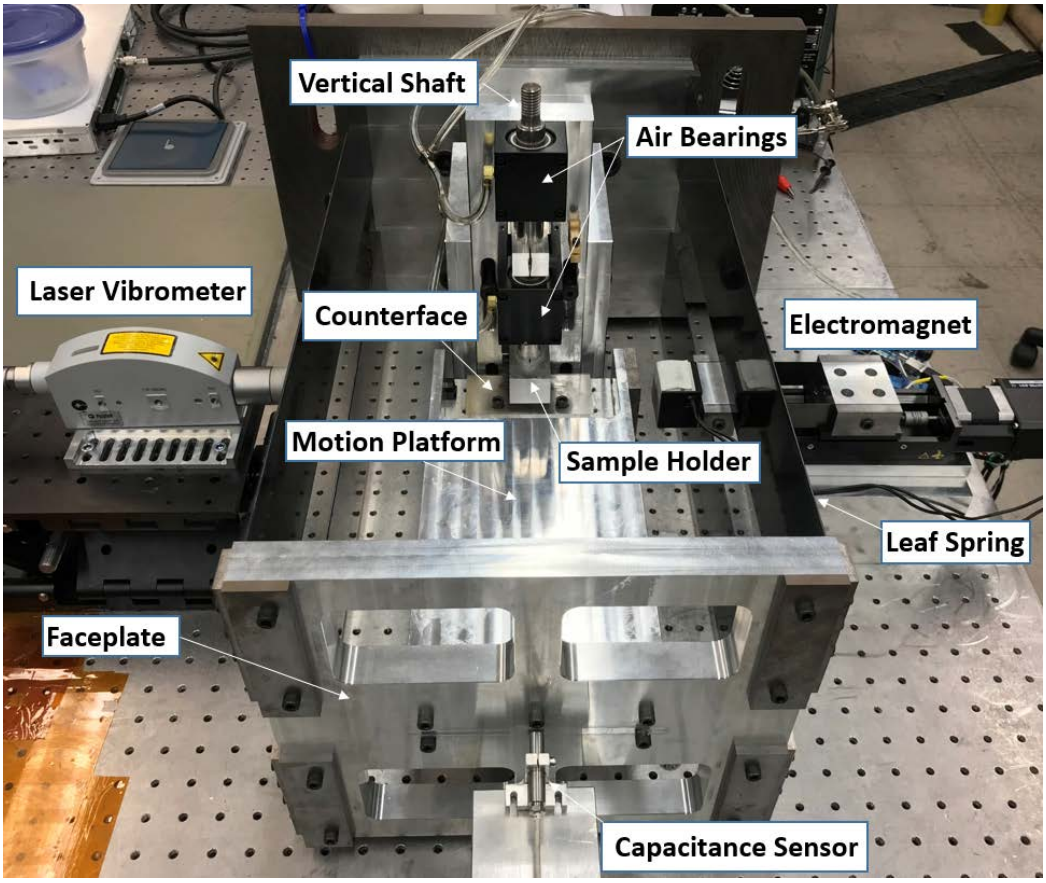

Figure 1 Friction measuring machine and measurement components

One end of the leaf springs is clamped to the base of the assembly which is rigidly mounted to a vibration isolated table. The other end is clamped to an aluminum faceplate to which the motion platform is attached. The motion platform is attached to the center of

the faceplate and extends back towards the base. The counterface for the friction contact pair is mounted on the motion platform at the midpoint between the faceplate and the base.

Two air bearings are attached to the base directly above the center of the counterface when the FMM is in its equilibrium position. The air bearings hold the vertical shaft in place, which is used for the application of the normal load for friction tests. The bottom end of the vertical shaft has a removable sample holder attached to it. A small cylindrical pin of a sample material can be loaded into the holder and the holder can then be attached to the bottom of the vertical shaft. The testing sample is then lowered onto the counterface to complete the friction contact pair. The top end of the vertical shaft is threaded so that different masses can be attached to it. Masses can be added to alter the normal load for friction tests. The forces resulting from the masses of the vertical shaft, sample holder and additional weights are the only contributing factors to the normal load for the tests.

3.2 FMM Operation

An electromagnet is used to move the FMM to the desired initial testing displacement. A MATLAB code was created to move the electromagnet to the motion platform and then pull it back to the input initial displacement. By cutting the power to the electromagnet the FMM is released and allowed to oscillate freely. As soon as the FMM is released the electromagnet is retracted further to ensure that it does not interfere with the oscillating structure.

3.3 FMM Measurement Devices

A laser vibrometer is used to gather velocity data during the tests. The vibrometer is positioned on the opposite side of the electromagnet and takes data at the location of the

friction contact pair interface. SpinScope is used to continuously acquire the laser vibrometer voltage output. In addition to the vibrometer, a capacitance sensor is used to monitor the parasitic motion of the motion platform. The laser vibrometer only measures linear velocity so the capacitance sensor is used to measure the motion normal to the vibrometer measurements. As shown in Chapter 4, the parasitic motion is small enough to be ignored

CHAPTER 4: FMM STRUCTURE CALCULATIONS

4.1 Maximum Allowable Deflection

The range of allowable initial displacements for the FMM were determined to ensure that the structure was not plastically deformed during testing. This was done using Equation 4.1 [23]. The maximum stress in the parallelogram, leaf-type flexure at a specific deflection is calculated with Equation 4.2.

$$\delta_{max} = \frac{\sigma_y L}{3Et} \tag{4.1}$$

$$\sigma_{max} = \frac{3Et\delta_x}{bt^2} \tag{4.2}$$

In Equation 4.1, δ_{max} is the maximum allowable deflection of the structure, t is the thickness of the flexure, L is the length, and σ_y and E are the yield strength and Youngs modulus of the flexure material (steel), respectively. In Equation 4.2, b is the width of the flexure, σ_{max} is the maximum stress, and δ_x is the deflection. Results showed that the maximum allowable deflection for the FMM is 134 mm. To ensure that there is no plastic deformation a factor of safety of six was used to determine the maximum test deflection. From Equation 4.2 it was determined that the maximum allowable stress for this factor of safety is 70.4 MPa, which resulted in a maximum allowable deflection of 22 mm.

4.2 Parasitic Motion

The FMM was designed to eliminate rotations at the friction contact pair interface. This is why the structure utilizes four leaf springs instead of two. By leaving a gap between two leaf springs it is possible to apply the displacement force at the midpoint of the leaf springs, which is crucial for these experiments and displayed in Figure 2.

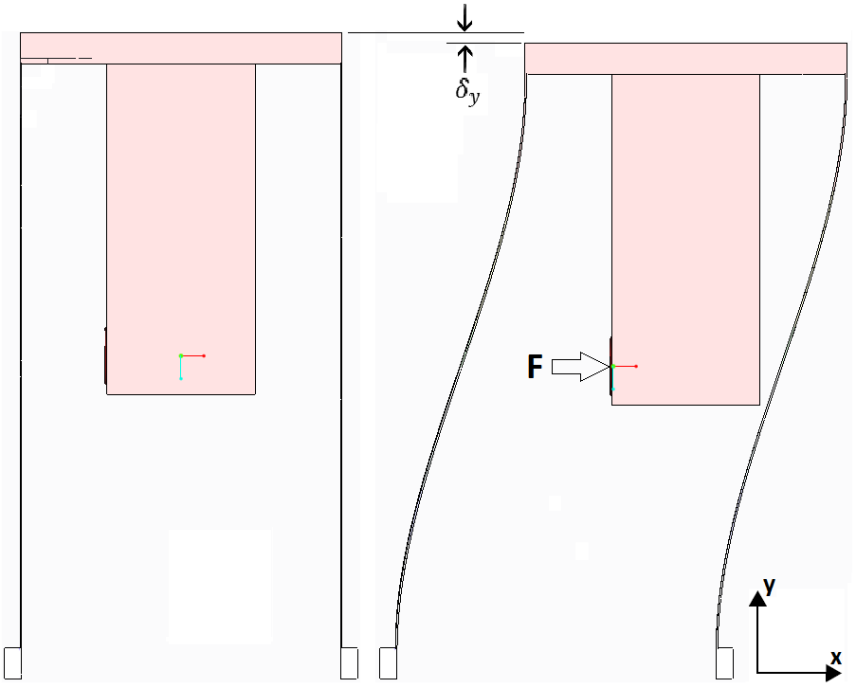

Figure 2 Diagram of the FMM with force input location

When the force is applied as shown in Figure 2, the faceplate, and, by extension, the motion platform, moves parallel to the base. The vibrometer takes velocity measurements in the *x*-direction only. However, there also exists some motion in the y-direction, which is considered the parasitic motion of the system. The parasitic motion can be calculated using Equation 4.3 [23].

$$\delta_y = \frac{-3\delta_x^2}{5L} \tag{4.3}$$

The relationship between parasitic motion and linear deflection is quadratic. The chosen testing range is 4 mm to 22 mm. For a 4 mm test the maximum parasitic motion is 17.4 µm, which is 230 times smaller than the linear displacement. At 22 mm displacement the parasitic motion increases to 500 µm. However, this is still over an order of magnitude lower than the linear displacement, which is 42 times greater. Even at the maximum displacement of 22 mm the parasitic motion has little impact on the testing results, which

will be demonstrated at the end of this section after the parasitic motion calculations have been validated.

The measured parasitic motion was used to validate the results of Equation 4.3. Consider Figure 3, which displays vibrometer and capacitance sensor data for a 6 mm test.

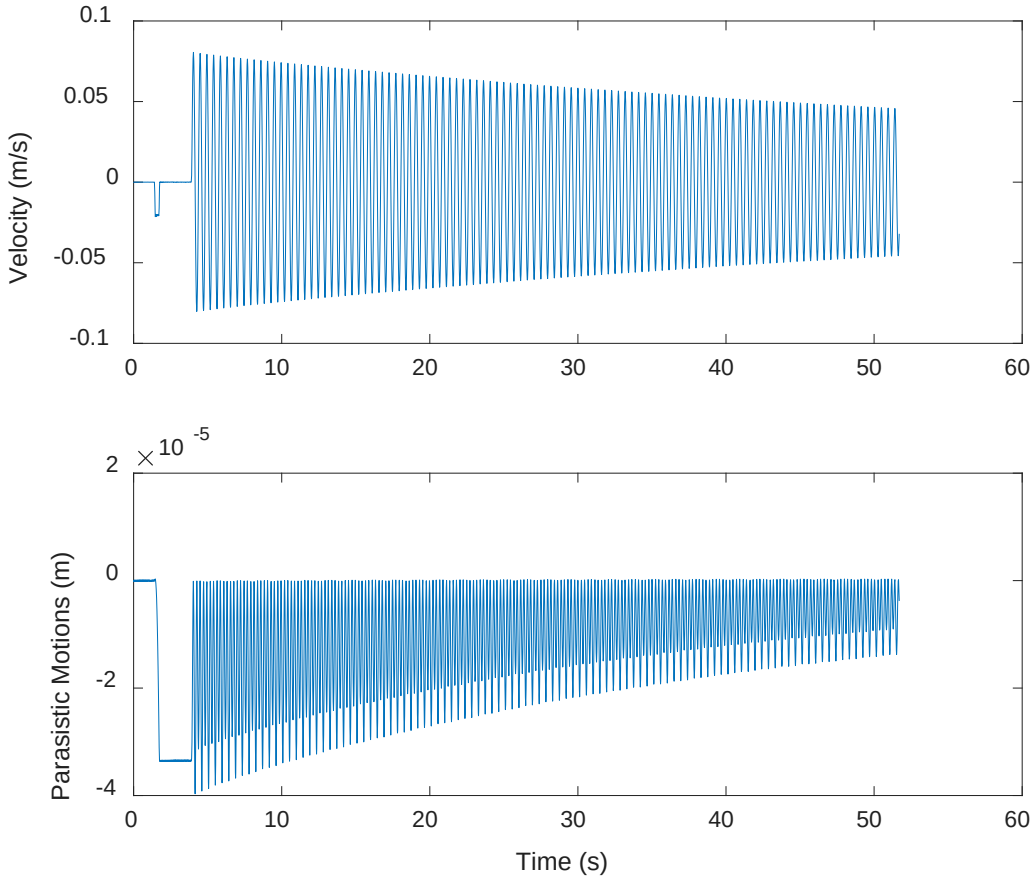

Figure 3 Velocity and capacitance sensor data for a 6 mm initial displacement test

The velocity data was integrated to obtain the position of the FMM during the test. The position vector was then used as the *x*-deflection input, δ_x, for Equation 4.3, to obtain a position vector that represents the analytical parasitic motion. This vector was then plotted and superimposed on the measured parasitic motion as can be seen in Figure 4. The calculated displacement closely matches the displacement measured by the capacitance

sensor. The relationship between parasitic motion, Δy, and linear motion, Δx, is displayed in Figure 5 for one oscillating cycle.

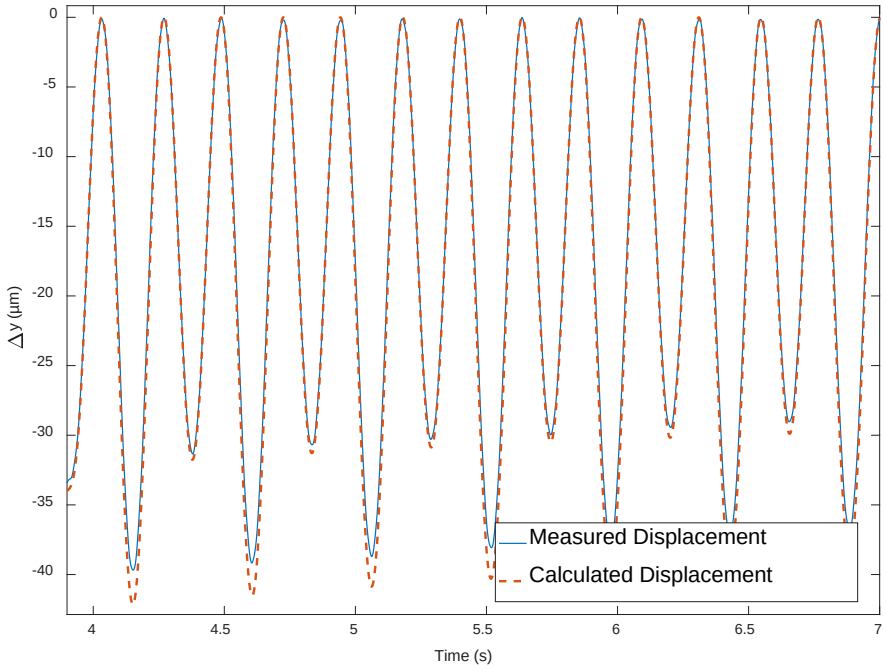

Figure 4 Measured parasitic motion and calculated parasitic motion of the FMM

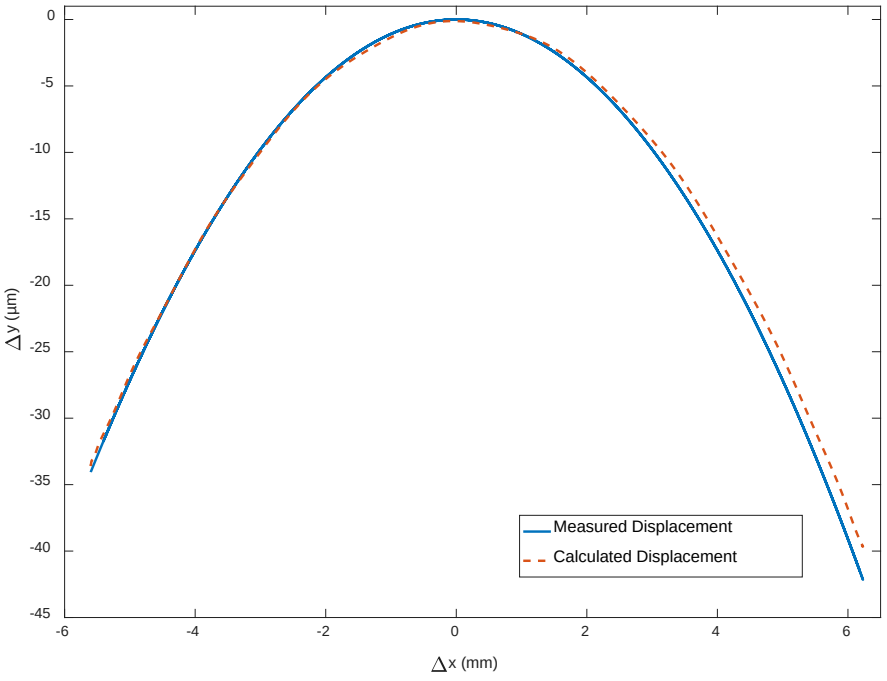

Figure 5 Single cycle comparison between analytical and measured parasitic motion

After the parasitic motion equation was validated, it was used to demonstrate the impact that the parasitic motion magnitude has on the velocity magnitude. The velocity data for a 22 mm test was used for this demonstration. The velocity data was integrated to obtain a position vector and the parasitic displacement was calculated from the position vector. The parasitic displacement along with the time step from the sampling frequency was then used to create a parasitic velocity vector. The linear velocity vector and the parasitic velocity vector were then combined to obtain a resultant velocity vector. Figure 6 displays a comparison between the linear velocity and parasitic velocity (top) as well as the resultant velocity vector superimposed on the linear velocity vector (bottom).

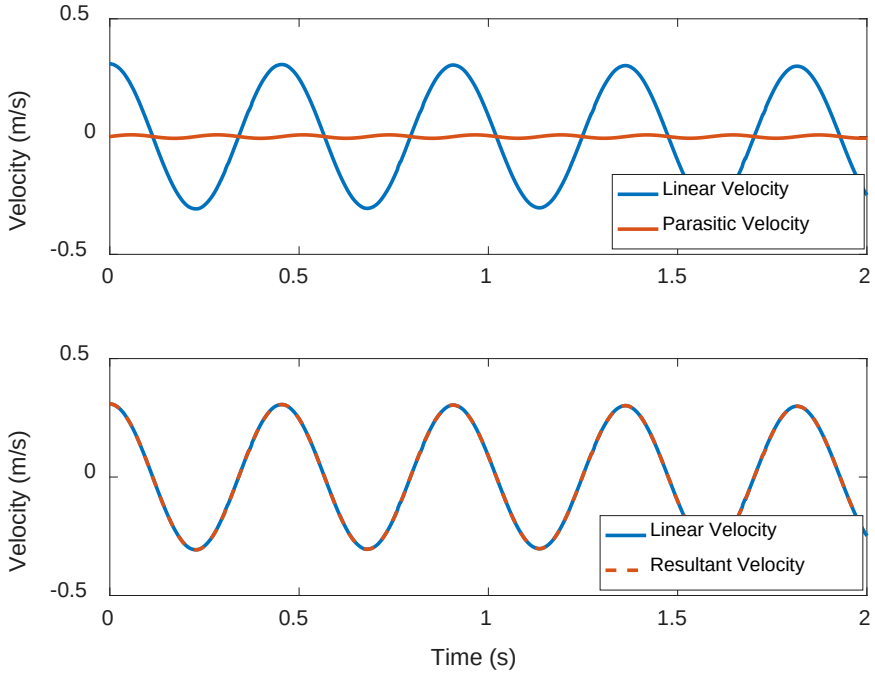

Figure 6 Comparison between linear velocity measured by vibrometer and parasitic motion (top) and linear velocity compared to resultant velocity including both parasitic velocity and linear velocity

As can be seen in Figure 6, even for the largest initial displacement test the parasitic motion has no significant impact on the velocity of the FMM. It is therefore accepted that

the parasitic motion can be ignored when analyzing the velocity data in the friction calculations.

4.3 Vertical Shaft Tilt

In addition to the parasitic motion, attention must be payed to the potential tilt of the vertical shaft during friction contact tests. Even with the two air bearings holding the vertical shaft in place there could still be a tilt in the shaft while the FMM is in motion and the friction force changes direction. This could cause the surfaces of the contact pair to not remain in full contact with each other. To test the shaft tilt, two capacitance sensors were attached to the base and pointed at two locations on the shaft. A friction contact was applied and the FMM was pulled back to an initial displacement of 12 mm and then released. Figure 7 shows the displacement results for the two capacitance sensors performing measurements on the shaft.

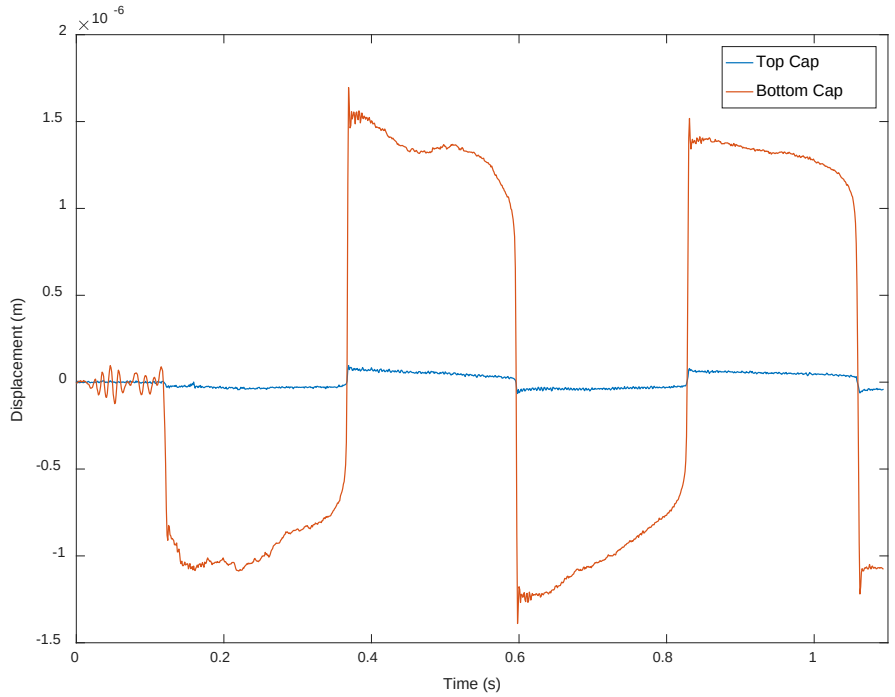

Figure 7 Capacitance sensor displacement data for shaft angular deflection

The maximum horizontal difference between the top and bottom capacitance sensor is 1.6 µm. At a capacitance sensor spacing of 10.8 cm this gives a maximum angular tilt of 8.5×10^{-4} degrees or 3.1 arcseconds, which is small enough to allow for the assumption that the interface contact area remains unchanged during testing.

CHAPTER 5: FMM MODAL PARAMETERS

5.1 Nonlinear Least Squares Fitting

The FMM dynamic characteristics were determined by analyzing free vibration velocity data when no friction contact was present. It was previously shown that the parasitic motion of the FMM is negligible and therefore the machine was analyzed as a single degree of freedom spring-mass-damper system with an equation of motion in the form of Equation 5.1

$$m\ddot{x} + c\dot{x} + kx = 0 \tag{5.1}$$

where *m*, *c* and *k* are the modal mass, damping coefficient and stiffness of the FMM, respectively. The parameters were found by writing a MATLAB code [*Free_Vibration_No_Friction_Fitting.m*] that implemented a nonlinear least-squares fitting algorithm to fit the equation of motion to velocity test data. The free vibration, no friction velocity data used for this fitting process included data from tests with initial displacements, x_0, ranging from 4 mm to 22 mm, in increments of 2 mm (zero initial velocity). Five tests were performed at each increment. For each test, the electromagnet was used to move the motion platform to the desired initial displacement and then the vibrometer was used to collect the velocity data upon release. The motion platform was allowed to oscillate freely for 45 seconds. The visually approximated natural frequency of the structure was about 2 Hz and therefore a sampling frequency of 1000 Hz was selected.

The residual effects of the electromagnet after release were also taken into account. Upon release of the FMM the electromagnet was retracted rapidly as to not interfere with the motion of the FMM. To account for residual magnetism and attractive force, the first five cycles of motion were removed from the data. The fitting algorithm was then applied to the remaining data. The function that was used was the MATLAB nonlinear least squares

fitting function lsqnonlin. This function applies a trust region reflective fitting approach. For each iteration Euler integration was performed to create simulated velocity data and then that data was subtracted from the measured velocity to determine the error. The code altered *m*, *c* and *k* and repeated the process until the error is at or below an acceptable level. Figure 8 shows an example of the velocity data fitting results for a complete data set and Figure 9 shows a close up view of the results for a three cycle window.

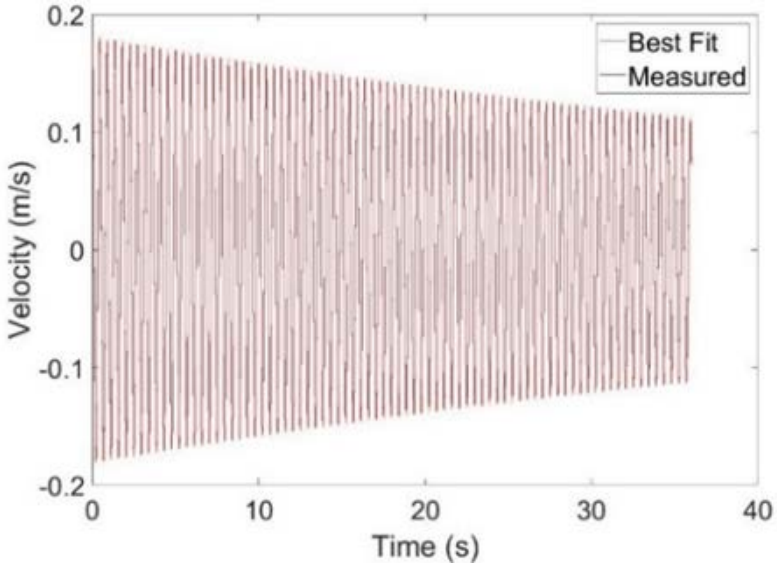

Figure 8 Free vibration, no friction velocity fitting results for a complete test

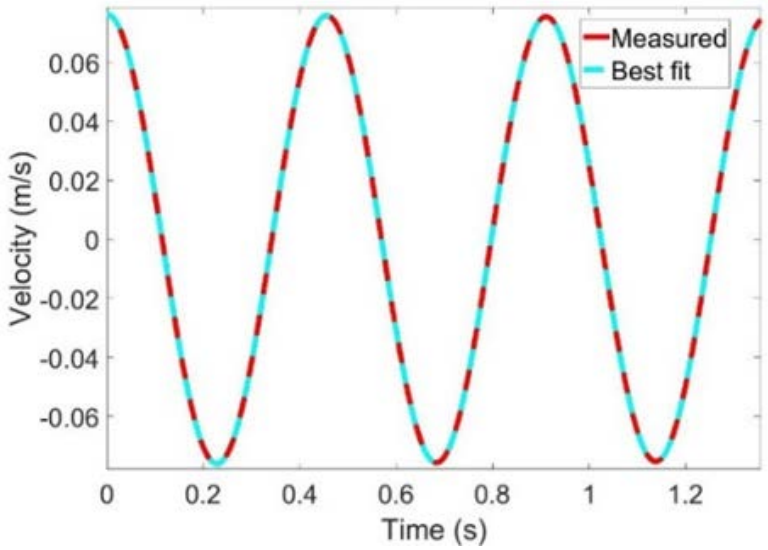

Figure 9 Close up view of fitting results for a three cycle window

Table 1 displays all free vibration fitting results (no friction contact). The velocity in column two is the mean maximum velocity for each of the five tests at a specific initial displacement. Columns three to five represent the mean for each displacement increment. Table 2 displays the accompanying standard deviations.

Table 1 Free vibration modal fitting results (no friction contact)

x_0 (mm)	*Velocity (mm/s)*	*Mass (kg)*	*Stiffness (N/m)*	*Damping (N·s/m)*
4	50	10.394	1981	0.222
6	76	10.395	1981	0.235
8	103	10.393	1981	0.245
10	129	10.397	1982	0.256
14	181	10.392	1982	0.278
16	207	10.389	1983	0.288
18	233	10.389	1984	0.299
20	258	10.385	1984	0.313
22	283	10.382	1984	0.327

Table 2 Mean and standard deviation of modal parameters

Modal Parameters	*m* (kg)	*k* (N/m)	*c* (N·s/m)
Mean	10.391	1982	0.275
Standard Deviation	0.0049	1.3	0.0341

As seen in Table 1, there is an increasing trend in the damping coefficient with increasing initial displacement (and corresponding maximum velocity). Table 2 shows that the damping coefficient's standard deviation is 12 % of its mean. It was discovered that there is a direct linear relationship between velocity and damping; this is presented in section 5.4.

5.2 Log Decrement Damping Analysis

The influence of initial displacement and velocity on the damping coefficient was further investigated. The first step in checking for a correlation between velocity and the damping coefficient was performed by examining the log decrement from cycle to cycle for the data. The FMM was allowed to oscillate for 80 seconds while velocity was

measured. MATLAB code [*Log_Dec_No_Friction.m*] was then created to find peaks and apply the log decrement to determine the damping coefficient for each cycle. The maximum velocity for each cycle was also determined and recorded with the damping coefficient. The results are presented in Figure 10 and clearly demonstrate that an approximately linear relationship exists between velocity and damping coefficient.

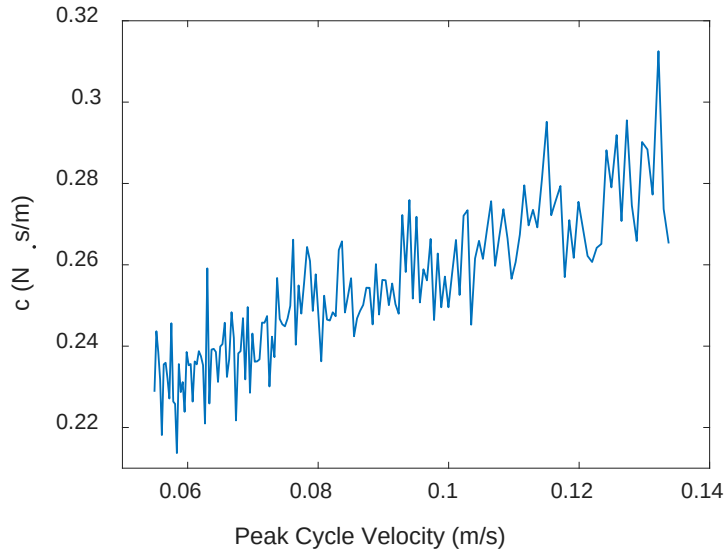

Figure 10 Damping coefficient results from logarithmic decrement method

Initially it was assumed that this change in damping was caused by a drag force from the large leaf springs swinging through the air. A drag term was added to the equation of motion and the fitting routine was repeated. This did improve the fit slightly but offered no change in damping coefficient sensitivity to velocity and was therefore discarded.

5.3 Monte Carlo Simulation

An uncertainty analysis was performed to determine the influence of the modal parameters on the single parameter Coulomb friction model. This was done by creating a Monte Carlo simulation in a MATLAB code [*Monte_Carlo_Single_Parameter_Friction_Model.m*]. For the simulation, a perfect data

set was artificially created and used as the fitting objective for the modified Euler integration. Equation 5.2 is the modified equation of motion for the system when friction is added. The friction force is the product of the normal force, *N*, and the dynamic coefficient of friction, μ_d: $F_f = N \cdot \mu_d$

$$\begin{matrix} m\ddot{x} + c\dot{x} + kx + F_f = 0, \dot{x} > 0 \\ m\ddot{x} + c\dot{x} + kx = 0, \dot{x} = 0 \\ m\ddot{x} + c\dot{x} + kx - F_f = 0, \dot{x} < 0 \end{matrix} \quad (5.2)$$

The Monte Carlo simulation consisted of four separate analyses. A normal distribution was created for each of the three system parameters based on the test data displayed in Table 2. These distributions were randomly sampled 100,000 times and used for the fitting algorithm to determine the resulting dynamic friction coefficient, μ_d. Three of the analyses fixed two modal parameters and randomly sampled the distribution of the third. The fourth analysis sampled the normal distributions of all three modal parameters at the same time. For the artificially created perfect data set a μ_d value of 0.15 was assumed. Table 3 displays the results of the simulations. Column three lists the standard deviation when the selected modal parameter was varied. Column four shows the sensitivity to each modal parameter which was determined using Equation 5.3. Column five represents the products on the right hand side of Equation 5.4.

$$sensitivity_{m,k,c} = \frac{\sigma(\mu_d)}{\sigma(m,k,c)} \quad (5.3)$$

Table 3 Monte Carlo simulation results

Dynamic friction coefficient, μ_d				
System parameter	Mean	Standard deviation	Sensitivity	Product of standard deviation and sensitivity
m	0.150015	6.39×10^{-5}	3.12×10^{-7}	1.99×10^{-11}
k	0.150018	2.09×10^{-5}	2.64×10^{-5}	5.52×10^{-10}
c	0.150001	2.36×10^{-4}	8.04×10^{-6}	1.90×10^{-9}
ALL	0.150021	2.48×10^{-4}	-	-

The law of propagation of uncertainty, represented in Equation 5.4 for this system, was used to determine the combined standard uncertainty for the friction coefficient due to variation in the modal parameters. This equation should yield the same result as the fourth Monte Carlo simulation, when sampling all three modal parameter normal distributions at the same time. Equation 5.4 resulted in u_c = 2.45x10^{-4} and the combined Monte Carlo simulation result was u_c = 2.48x10^{-4}. Figure 11 displays the probability density function for μ_d when all three system parameters were randomly sampled during the same simulation.

$$u_c^2(\mu_d) = \left(\frac{\partial \mu_d}{\partial m}\right)^2 u^2(m) + \left(\frac{\partial \mu_d}{\partial k}\right)^2 u^2(k) + \left(\frac{\partial \mu_d}{\partial c}\right)^2 u^2(c) \tag{5.4}$$

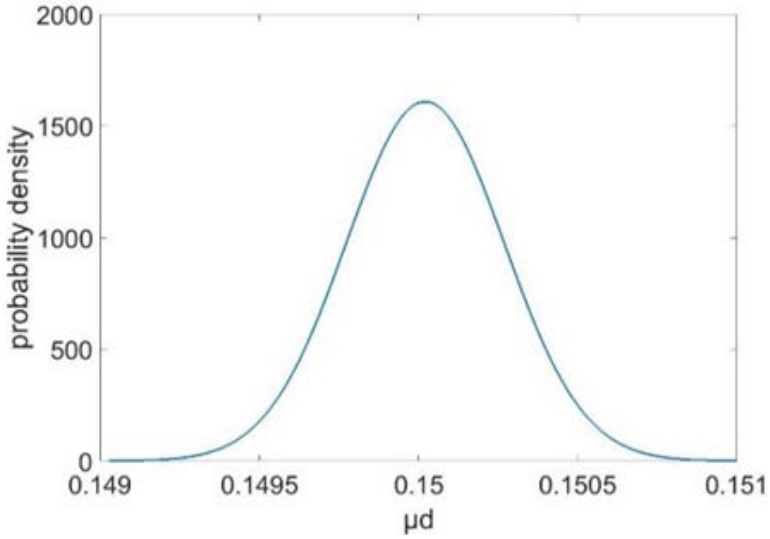

Figure 11 Probability density function for μ_d when all three modal parameters were randomly sampled during the Monte Carlo simulation

Table 3 demonstrates that damping plays a significant role in identifying the friction coefficient. The column five entry is 3.44 times larger than the next largest entry (stiffness). This is not surprising since both viscous damping and Coulomb friction are responsible for energy dissipation in the dynamic system. For this reason, the damping results were further analyzed.

5.4 Linear Viscous Damping Coefficient

The relationship between velocity and viscous damping was quantified by reexamining the original free vibration data from the tests presented in Table 1. This time, however, only five cycles at a time were considered when determining the modal parameters. This ensured that for each sample range, the velocity had not decayed significantly, which made it possible to link the peak cycle velocity to the modal parameters that are identified by the fitting algorithm. It was discovered that there was a linear relationship between peak velocity and the viscous damping coefficient. This relationship is shown in Figure 12 and is expressed in Equation 5.5.

$$c = 0.437\dot{x} + 0.2005 \tag{5.5}$$

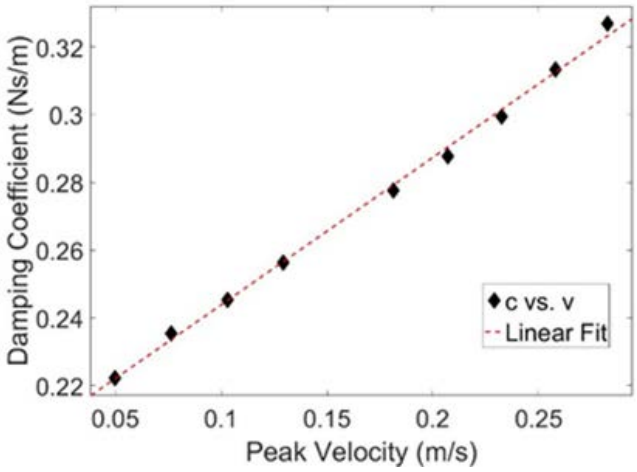

Figure 12 Linear fit of damping coefficient as a function of peak velocity

The linear fit has an R^2 value of 0.997. The most likely source for this phenomenon are the clamps that hold the leaf springs to the base. There could be significant losses in kinetic energy as the leaf springs rub across the clamped surface during oscillation. Several other damping models were tested as well, including Zaitsev's [17] nonlinear damping in double clamped beams model and Elliot and Tehrani's [16] model for nonlinear damping

in single degree of freedom systems, but none provided consistent results for the FMM system.

CHAPTER 6: FRICTION ENERGY DISSIPATION

When a friction contact is added to the FMM, the energy dissipation rate increases dramatically. The difference in FMM energy rate between data sets with and without a friction contact can be used to determine the friction force caused by the contact pair at a specific velocity. A simulation [*Energy_Analysis.m*] was created to illustrate how the difference in energy rate can be used to determine the friction force and friction coefficient. The simulation assumes that the friction force is caused by a constant dynamic coefficient of, μ_d, which yields the friction force when multiplied by the normal load, N. This is the same assumption made when the Monte Carlo simulation was performed in section 5.1 and used the equation of motion provided in Equation 5.2.

For this analysis, the perfect data set for friction contact was created by assuming $\mu_d = 0.25$ and $N = 7$ N. Both the free vibration and friction contact simulations had an initial displacement of 10 mm. Euler integration was used to create two data sets. Figure 13 shows the difference in velocity between the free vibration (no friction) and friction contact (with friction) cases.

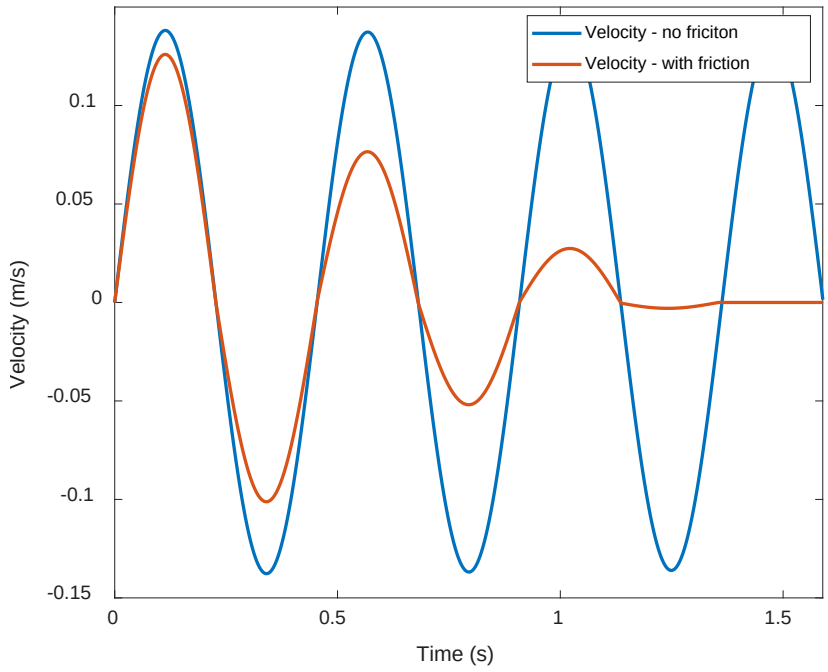

Figure 13 Time dependent velocity for the two cases

The impact that the friction contact has on the energy dissipation rate is easily observable by comparing the velocities in Figure 13. The energy in the system at each time step can be best illustrated as a plot of the kinetic, potential, and total energies. Figure 14 displays the simulation data when converted into energy components. The kinetic and potential energies were calculated using Equation 6.1 and Equation 6.2. The total energy is simply the sum of the two.

$$KE = \frac{1}{2}m\dot{x}^2 \tag{6.1}$$

$$PE = \frac{1}{2}kx^2 \tag{6.2}$$

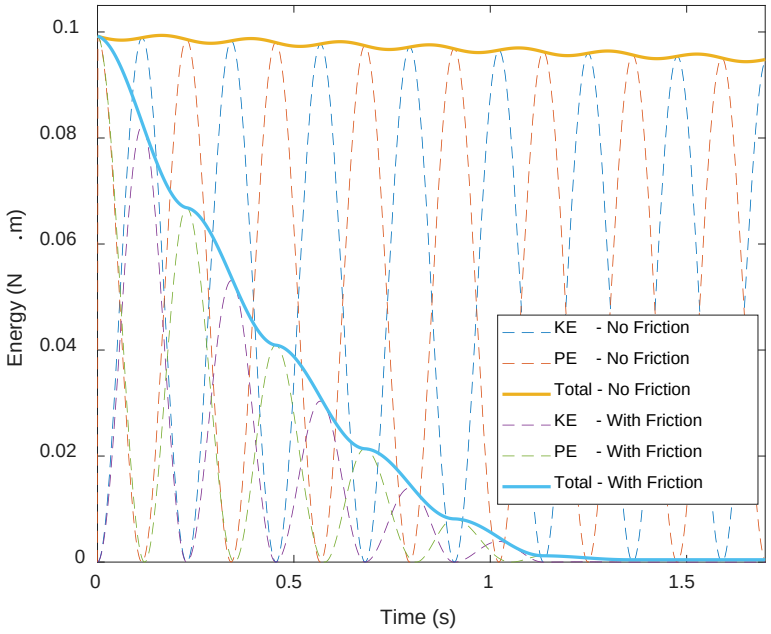

Figure 14 System energy versus time for free vibration with and without friction contact

The energy rate for the system velocities can now be calculated. This was done using Equation 6.3. The energy rate is tabulated at each velocity and plotted; see Figure 15.

$$Energy\ Rate = \dot{x} \cdot [m\ddot{x} + kx] \tag{6.3}$$

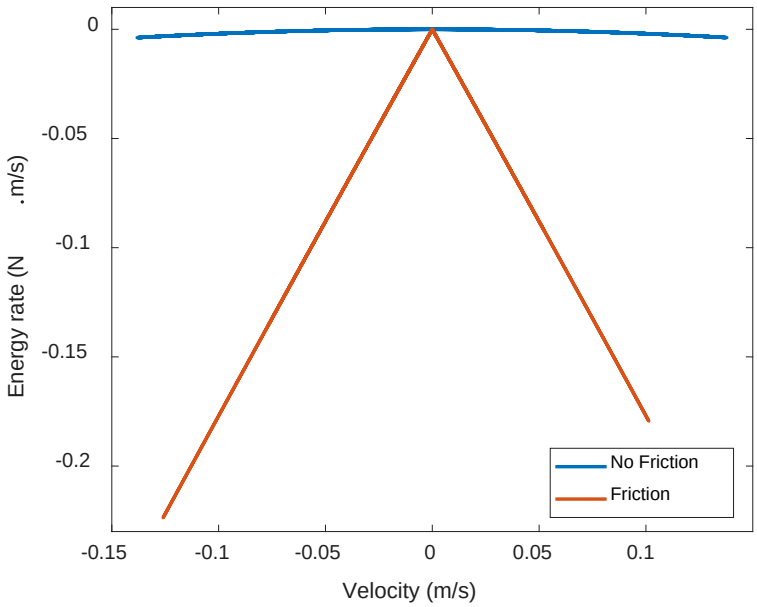

Figure 15 Energy rate for free vibration and friction contacts

To obtain the difference between the two energy rates, two tables were created. The free vibration energy data was input in a two column matrix along with its corresponding velocity data and the matrix was sorted according to velocity, from smallest to largest. The same was done for the friction contact energy rate and velocity data. The absolute value of the magnitudes for the end points of the friction contact data are smaller than those for friction free data and, therefore, the friction contact data lies within the free vibration velocity range. The code creates an energy difference matrix to determine the difference in energy rate at each of the friction contact velocities. The FMM has very low damping and therefore its free vibration energy dissipation is very low, which can be seen by the nearly horizontal blue line in Figure 15. Therefore, the difference in energy rate for the simulation looks like the inverse of the friction contact red line in Figure 15. The energy rate difference for the simulation is displayed in Figure 16.

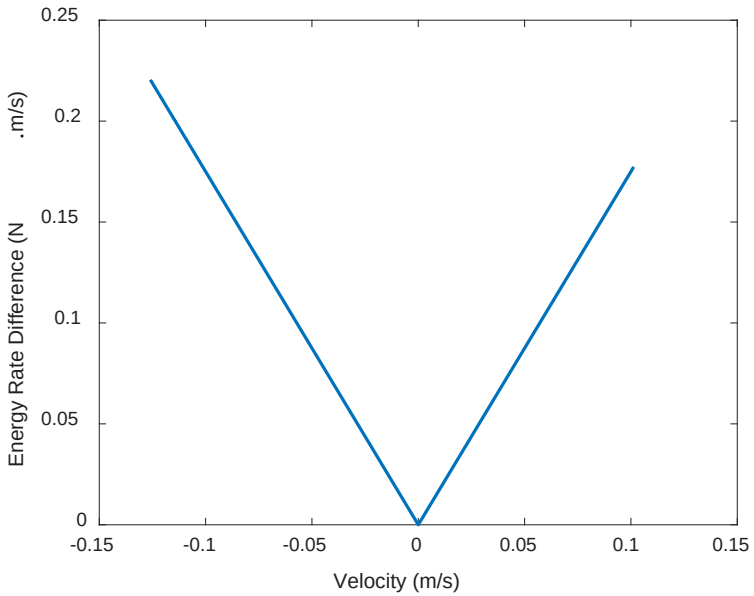

Figure 16 Energy rate difference versus velocity

When dividing the energy rate difference by the velocity it returns a vector with values of -1.75 for all entries when velocity is negative and 1.75 for all values where velocity is positive. This is the force that friction adds to the system at every velocity. Checking against the friction contact simulation: $F_f = \mu_d \cdot N = 0.25 \cdot 7 = 1.75$, it can be seen how this approach works when applied to actual data.

The main challenge in applying this method is the fact that only velocity data (not position) is measured by the vibrometer. Therefore, the displacement and acceleration need to be obtained by integrating and differentiating the velocity, respectively. This causes an increase of noise to be present in the acceleration data. For friction contact data, this is a significant problem because the function to be integrated and differentiated is discontinuous. Adding smoothing functions can reduce this noise, but also makes it more difficult to obtain accurate results when all three components of motion are combined. An example is presented in Figure 17.

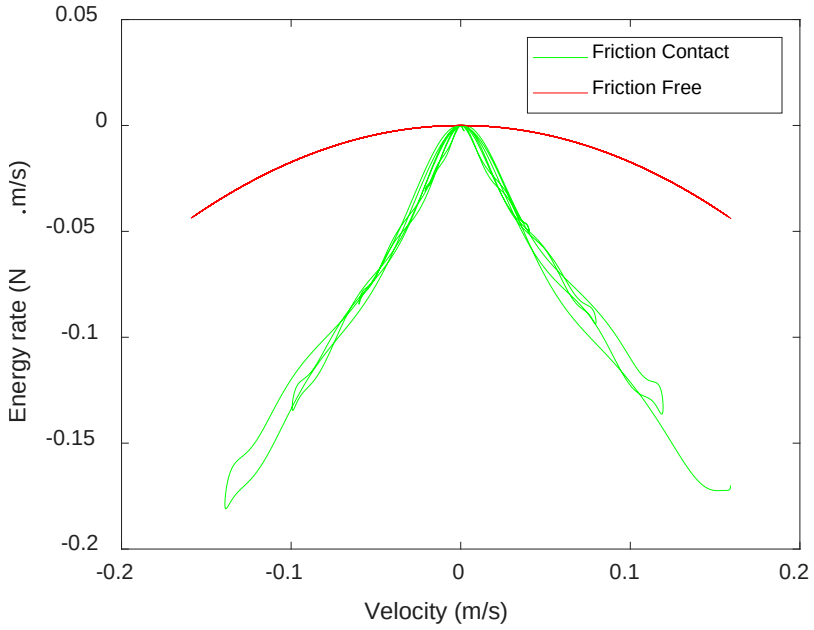

Figure 17 Energy rate calculated from measured data

CHAPTER 7: FRICTION COEFFICIENT FITTING

7.1 Friction Tests

The friction contact pair for the friction tests consisted of a polytetrafluouroethylene (PTFE) pin and a polished steel counterface, which was attached to the motion platform. The PTFE pin was loaded into the sample holder and then attached to the vertical shaft. Four hundred grit sandpaper was placed on the counterface and the pin was lowered on top of it. The sample was then moved back and forth until the surfaces of the pin and counterface appeared to be parallel. The debris was removed with compressed air and then the process was repeated with 1,000 grit sandpaper and then 2,000 grit sandpaper. This was done to ensure that the surfaces were smooth and parallel.

The normal force for these tests was 14 N. The force was determined by measuring the mass of the vertical shaft, sample holder, and added mass and multiplying this mass by the gravitational constant (9.81 m/s^2). Wear is a common side product of friction. Because the friction coefficient results depend on a consistent surface interface, the FMM was checked for repeatability by conducting numerous tests at specific initial displacements. These results were then compared to determine if there was any change in system response as the number of tests increased. Figure 18 displays an example of the velocity data for five tests at the same initial displacement and demonstrates good repeatability. This type of data comparison was performed for all tests prior to the application of the friction fitting code. Friction contact tests were performed at 15 different initial displacements. These tests ranged from 8 mm to 22 mm in 1 mm increments.

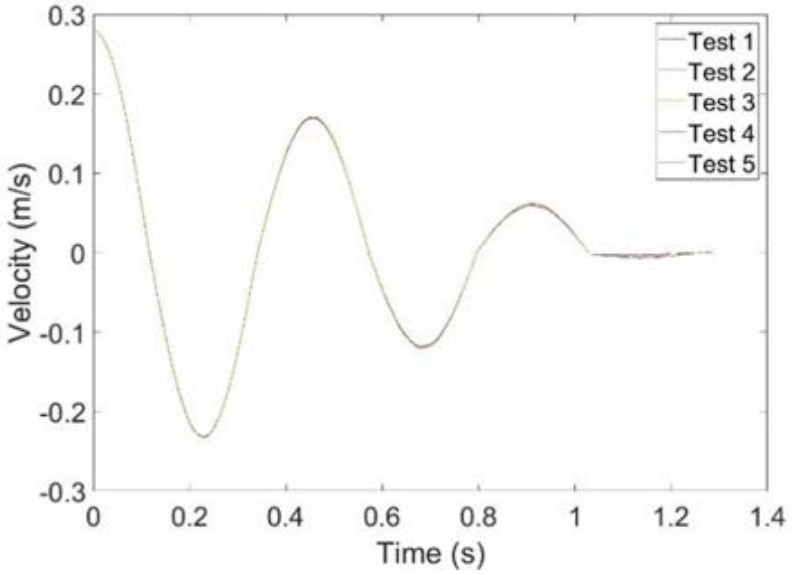

Figure 18 Repeatability test with friction contact

7.2 Single Parameter Coulomb Friction

A single parameter Coulomb friction model was selected as the primary friction response model for this research. This model assumes that friction force is solely a function of normal force and applies a dynamic friction coefficient, μ_d. The friction force is therefore a constant quantity, independent of velocity and acceleration, and always opposes the motion of the system.

For this simplistic model, the friction force, F_f, in Equation 5.2 is replaced by the product of normal force, N, and μ_d: $F_f = N \bullet \mu_d$. The MATLAB code [*Friction_fitting_single_parameter.m*] was used for fitting Equation 5.2 to the velocity data. The velocity data was first filtered using the MATLAB smooth function which used a moving average filter with a default span of 5 samples. The maximum velocity of the data was determined and used as the starting point for the fitting algorithm. This was done for two reasons: to minimize the residual effect of the electromagnet and because at maximum velocity the position of the FMM is at its equilibrium, or zero, displacement. The latter is of particular importance because the initial position is the starting point for the

Euler integration used in the fitting code. For this reason the fitting code has two variables to identify: μ_d and x_{0f}, which are the dynamic friction coefficient and the initial fitting position, respectively. The Euler integration steps are displayed in Equation 7.1

$$\ddot{x}^{(n+1)} = \frac{-\mu_d N - c\dot{x}^{(n)} - kx^{(n)}}{m}$$
$$\dot{x}^{(n+1)} = \dot{x}^{(n)} + \ddot{x}^{(n+1)} \cdot dt \tag{7.1}$$
$$x^{(n+1)} = x^{(n)} + \dot{x}^{(n+1)} \cdot dt$$

where x, $\dot{x}$, and $\ddot{x}$ are position, velocity, and acceleration, respectively. The variable n is an index used to denote the time step number for the variable and dt represents the time step. The constants m, k and c are the modal parameters that were determined during the free vibration analysis with no friction. The velocity data being fit ranges from maximum velocity until the motion comes to a stop. For example, in the case of the data in Figure 18, the fitting range is from 0 to 1.3 seconds.

The fitting results for the initial displacement tests are summarized in Figure 19 and Table 4 (results for μ_d from each initial displacement).

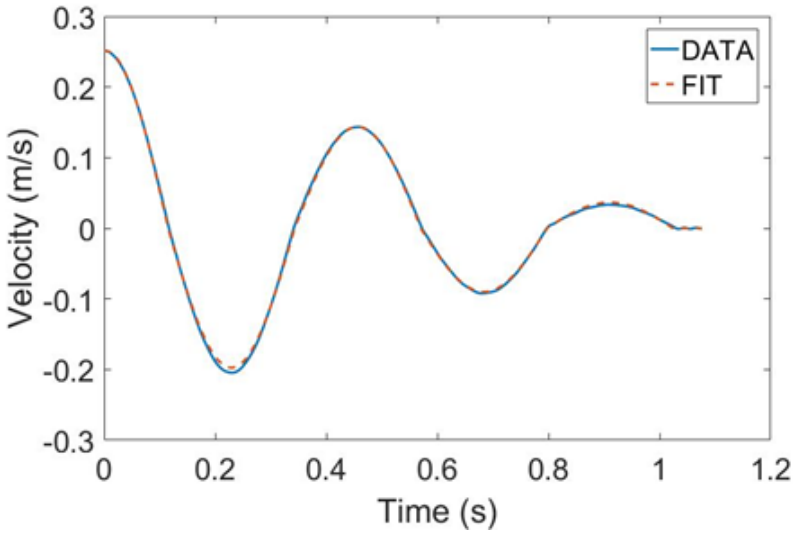

Figure 19 Fitting result example for a single parameter friction model

Table 4 Dynamic coefficient of friction results

	Linear c		Mean c	
x_0(mm)	μ_d	v_{max}(m/s)	μ_d	v_{max}(m/s)
8	0.217	0.086	0.217	0.086
9	0.225	0.100	0.224	0.100
10	0.228	0.114	0.228	0.114
11	0.232	0.128	0.231	0.128
12	0.236	0.141	0.235	0.141
13	0.244	0.155	0.239	0.155
14	0.247	0.168	0.247	0.168
15	0.248	0.182	0.248	0.182
16	0.252	0.196	0.252	0.196
17	0.254	0.209	0.254	0.209
18	0.261	0.223	0.261	0.223
19	0.265	0.237	0.265	0.237
20	0.264	0.251	0.264	0.251
21	0.265	0.265	0.265	0.265
22	0.267	0.280	0.267	0.280

Fitting was performed for both viscous damping models: the constant mean damping coefficient used in the Monte Carlo simulations and the linear damping coefficient from Equation 5.3. The results for the two approaches are virtually identical. The single friction coefficient shows a dependence on initial displacement. The Dahl and LuGre [3,6,11,15,19] model bristle effect discussed in the literature review states that the pre-sliding displacement caused by the spring like effect of the contact asperities can result in different required break away forces, which would impact the friction behavior around zero velocity. The velocity dependence of friction is well documented in most friction models, especially in the transition phase. The velocities of the FMM are low, with a maximum peak velocity of 0.267 m/s for the largest initial displacement test. It is therefore possible that the entire motion regime lies solely inside the Stribeck effect, never reaching the constant Coulomb dynamic friction coefficient. To establish a relationship between velocity and μ_d the maximum velocity for each initial displacement test was recorded and is presented in Table 2. This relationship is represented in Figure 20. The relationship between the dynamic friction coefficient and velocity in this case is not linear and is best

represented as part of a power function. The resulting power law fit is displayed in Equation 7.2.

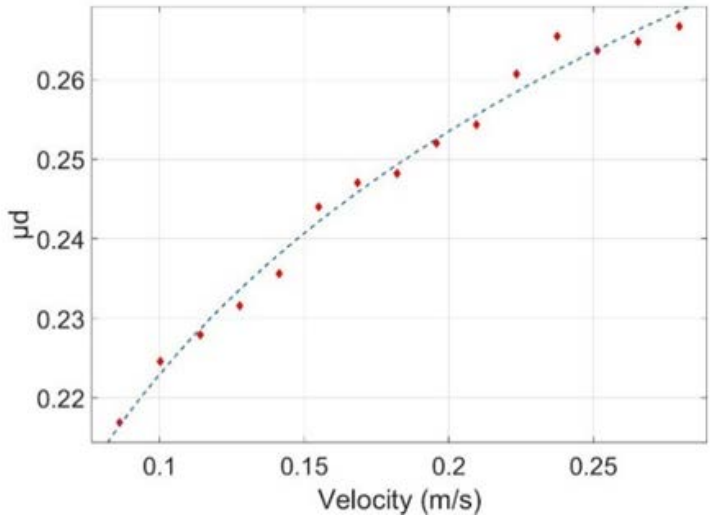

Figure 20 Dynamic friction coefficient variation with peak velocity

$$\mu_d = 0.311\dot{x}^{0.2116} + 0.03148 \qquad (7.2)$$

The results show that the data collected with the FMM can be used to determine the average friction coefficient for a contact pair. In this research a single sliding friction coefficient model was implemented, but other models could be applied as well to study the transition phase phenomena.

7.3 Acceleration Dependent Friction Model

There are an abundance of friction models in existence. An interesting friction model was proposed by Wojewoda and Stefanski [14] and has been experimentally evaluated by many. Their model suggests that the magnitude of the friction force is dependent on acceleration. It states that the friction force is larger when the sliding velocity is increasing than when it is decreasing in magnitude, which leads to a hysteretic effect around zero velocity. The FMM is never at a constant velocity and therefore is continuously accelerating in one direction or another. The acceleration dependent friction model was

tested in a separate simulation by adding a constraint to the fitting algorithm, as seen in Equation 7.3. The modified code determined the acceleration at each time step and applied the corresponding component of Equation 7.3. In Equation 7.3, μ_{d1} represents the friction coefficient for the case when velocity is increasing in magnitude and μ_{d2} is the friction coefficient for the case when velocity is decreasing in magnitude. An example of the fitting results is presented in Figure 21.

$$\begin{aligned} m\ddot{x} + c\dot{x} + kx + \mu_{d1}N = 0, \quad & \dot{x} > 0\,, \ddot{x} > 0 \\ & \dot{x} < 0\,, \ddot{x} < 0 \\ m\ddot{x} + c\dot{x} + kx + \mu_{d2}N = 0, \quad & \dot{x} > 0\,, \ddot{x} < 0 \\ & \dot{x} < 0\,, \ddot{x} > 0 \end{aligned} \tag{7.3}$$

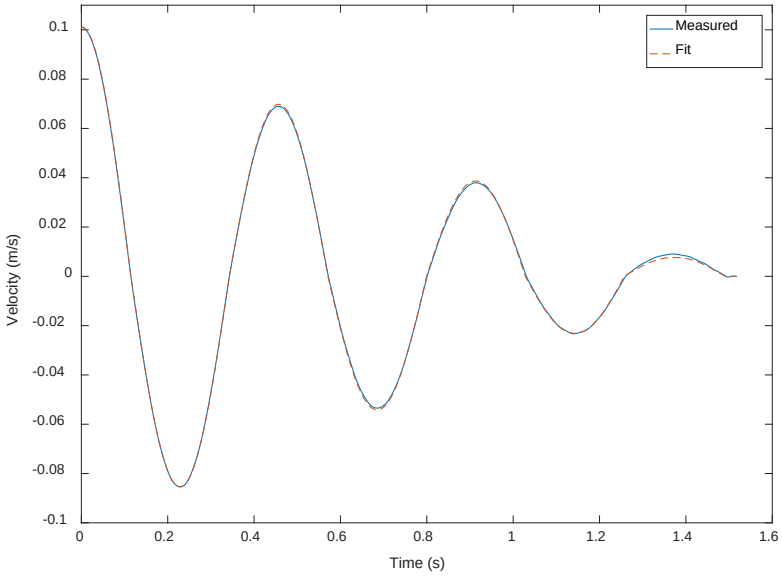

Figure 21 Fitting results for acceleration dependent friction coefficient model

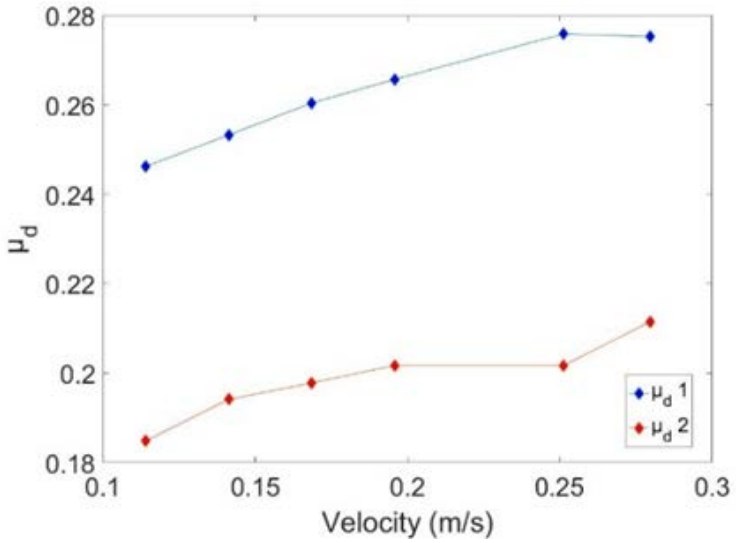

Figure 22 Dynamic friction coefficient for increasing velocity, μ_{d1}, and decreasing velocity, μ_{d2} for different peak velocities

Figure 22 displays the results for multiple initial displacement tests. The results corroborate the theory of acceleration dependent friction. The friction coefficient has a different magnitude depending on whether the sliding velocity was increasing or decreasing. An increasing velocity magnitude always corresponds to a higher friction coefficient. These results also led to an investigation of work done by Den Hartog and Burdekin [14] on an irreversible friction effect model. Instead of constant values for μ_{d1} and μ_{d2} in Equation 7.3, the coefficients were replaced by two exponential functions of velocity. Fitting results were excellent, but the results for most tests were inconsistent and simply confirmed that the coefficient of friction is larger for increasing velocity magnitude than for decreasing velocity magnitude as previously shown.

7.4 Velocity Dependent Three Parameter Friction Model

The transition phase from stick to slip is a component of friction behavior that is yet to be completely successfully modeled. A simple and often used friction model for dynamic systems is presented by Berger in [11]. The model assumes that the friction force

weakens monotonically as the system goes from stick to slip. This is a three parameter model for the coefficient of friction and is presented in Equation 7.4

$$\mu(v_{rel}; \mu_0, \mu_1, \alpha) = \mu_0 + \mu_1 \cdot \exp(-\alpha|v_{rel}|) \tag{7.4}$$

where v_{rel} is the velocity, μ_0 is a coefficient that governs large velocity behavior, μ_1 is a coefficient that controls low velocity behavior, and α controls the rate of change of friction. At zero velocity, the sum of μ_0 and μ_1 represent the static coefficient of friction. As velocity increases the model converges to a constant dynamic coefficient of friction, in this case $\mu_d = \mu_0$. The constant α is responsible for the rate of the transition from stick to slip and resembles the Stribeck effect. MATLAB code [*Friction_fitting_three_parameter.m*] was created to implement this friction model. New tests were performed from 10 mm to 16 mm in 1 mm increments. Five tests were performed at each initial displacement. Figure 23 displays an example fit for a 16 mm initial displacement test and Figure 24 displays the resulting velocity dependent friction coefficient.

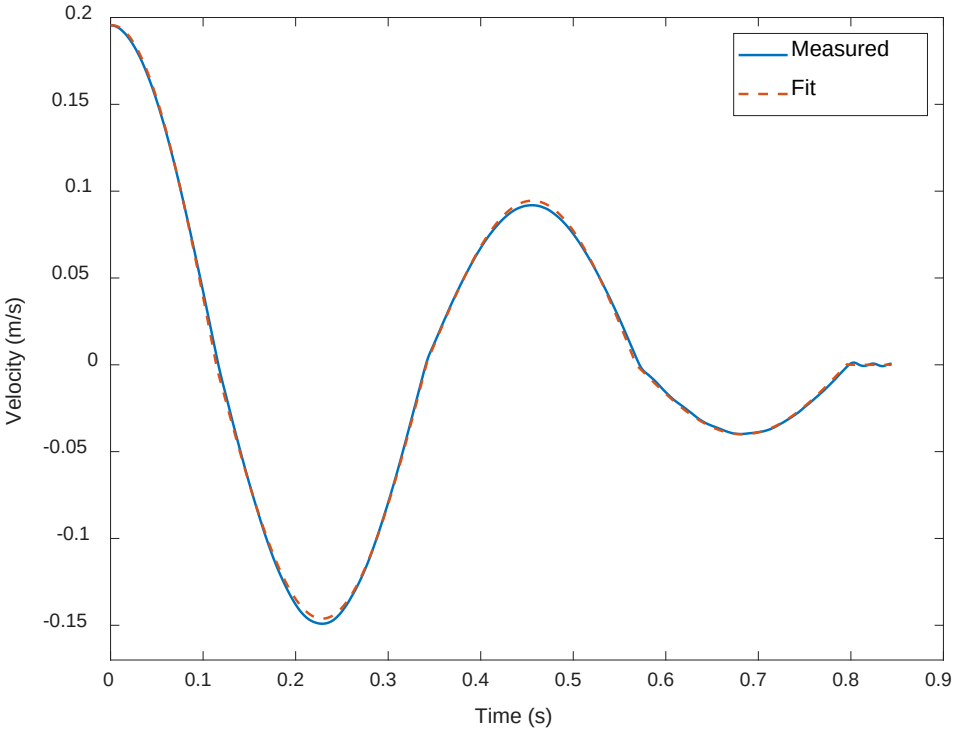

Figure 23 Velocity dependent three parameter friction model

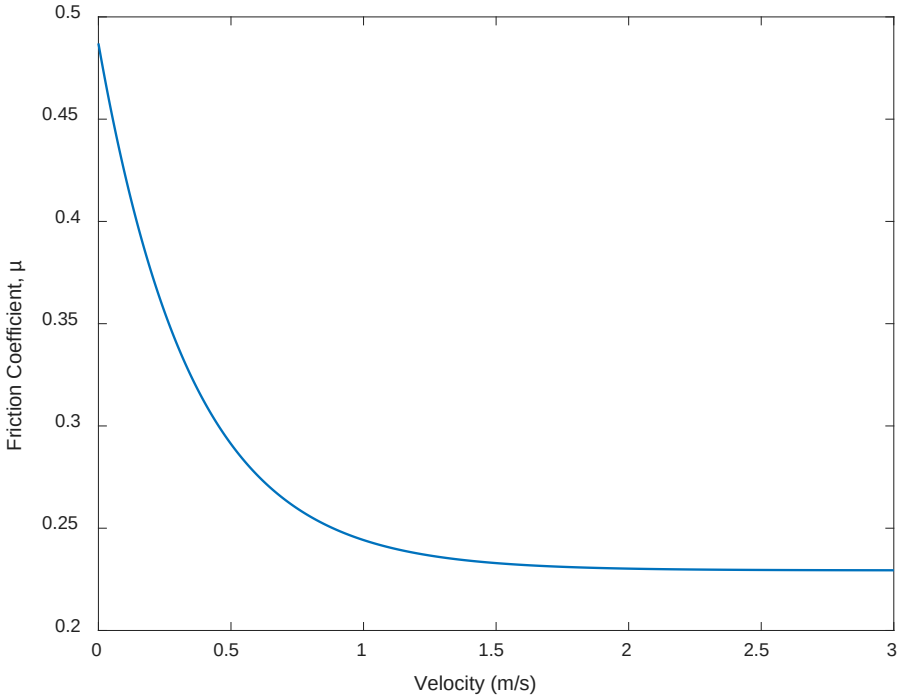

Figure 24 Coefficient of friction versus velocity for three parameter dynamic friction model

The results displayed in Figure 24 predict a static coefficient of friction of around 0.49, a constant dynamic coefficient of friction of 0.23 and the transition phase. Figure 24

also represents the predicted coefficient of friction for velocities larger than the test velocities. The complete results are displayed in Table 5, where the mean of each coefficient at each initial displacement increment is shown.

Table 5 Three parameter friction model results

Initial Displacement (mm)	Peak Velocity (m/s)	μ_0	μ_1	α
10	0.114	0.210	0.238	5.605
11	0.127	0.216	0.244	5.140
12	0.141	0.217	0.259	4.865
14	0.169	0.223	0.250	3.211
15	0.182	0.220	0.260	3.034
16	0.196	0.228	0.261	2.852

Even this model displays an increase in dynamic friction coefficient with increasing peak velocity. The static coefficient of friction also increased with the initial displacement. This is most likely due to the varying break-away force discussed in the literature review, which depends on the rate of the increase of the force when moving from stick to slip. The larger the displacement the more tangential force is being applied by the flexures. The model does manage to predict the friction behavior for the displacement test ranges that were performed, however.

7.5 Continuously Differentiable Friction Model

An interesting model that encapsulates all components of friction was developed by Dixon et al. in the form of a continuously differentiable friction model with a friction coefficient that is a function of velocity [1]. This model captures every aspect of friction behavior: static coefficient, the Stribeck effect, viscous dissipation, and coulombic friction. The model is presented in Equation 7.5.

$$f(\dot{x}) = \gamma_1(\tanh(\gamma_2\dot{x}) - \tanh(\gamma_3\dot{x})) - \gamma_4\tanh(\gamma_5\dot{x}) \tag{7.5}$$

where γ_1 to γ_5 are scaling coefficients. The sum of γ_1 and γ_4 gives the static friction coefficient. Coulomb friction is represented with the $\gamma_4 tanh(\gamma_5 \dot{x})$ term. The Stribeck effect is captured by the $tanh(\gamma_2 \dot{x})$-$tanh(\gamma_3 \dot{x})$ component of Equation 7.5.

The problem with this model is that it passes through zero and it was therefore altered as follows. The fitting algorithm attempts to find γ1 through γ5 using the same nonlinear least squares approach as in previous tests. At each iteration, a velocity vector is created and a friction coefficient vector is calculated using Equation 7.5. The second derivative of Equation 7.5 is then used to determine where the function becomes concave up. The friction coefficient vector is shifted to zero from that point. The friction vector is now combined with the velocity vector to create a table that is used to interpolate friction coefficients at different velocities. Figure 25 displays the modification steps. The circles represent the determined points of inflection and from those two points the plot is shifted toward the vertical axis. The resulting friction coefficient versus velocity function shape can be seen in Figure 26. The data from this graph is used to interpolate the friction coefficient at a specific velocity.

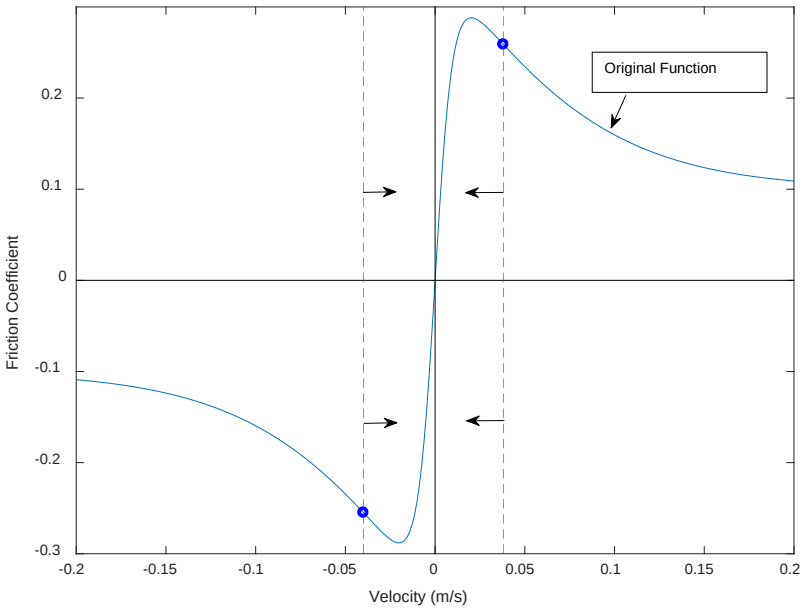

Figure 25 Modification steps of Equation 7.5

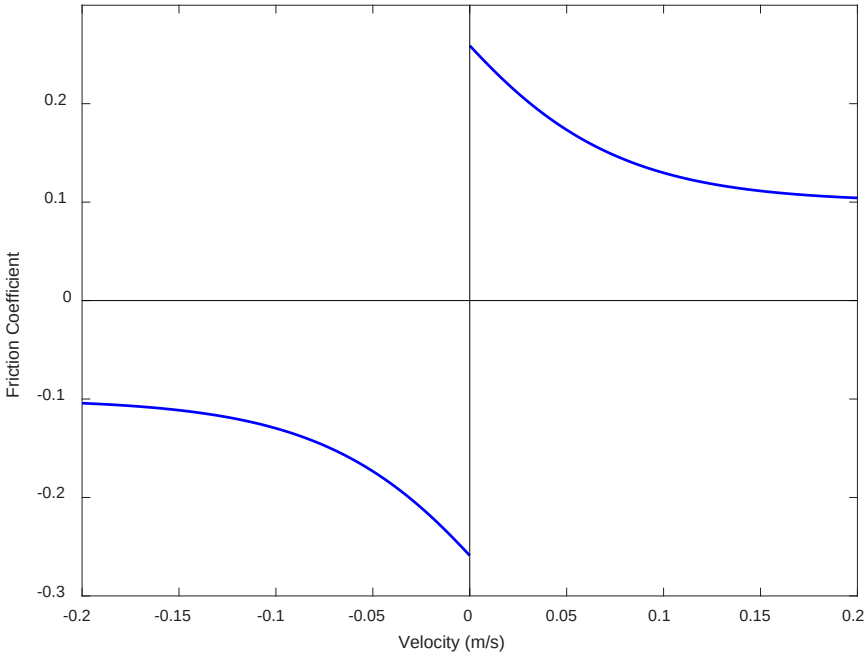

Figure 26 Friction coefficient versus velocity from modified Equation 7.5

It is the shape of the transition phase as well as the magnitudes of the static and sliding coefficients of friction that are of interest in this approach. MATLAB code [*Tanh_Modified_Fitting_Model.m*] was created to implement the modified continuously differentiable friction model in the fitting algorithms. Figure 27 displays an example of the

friction fit results and Figure 28 displays the resulting coefficient of friction as a function of velocity.

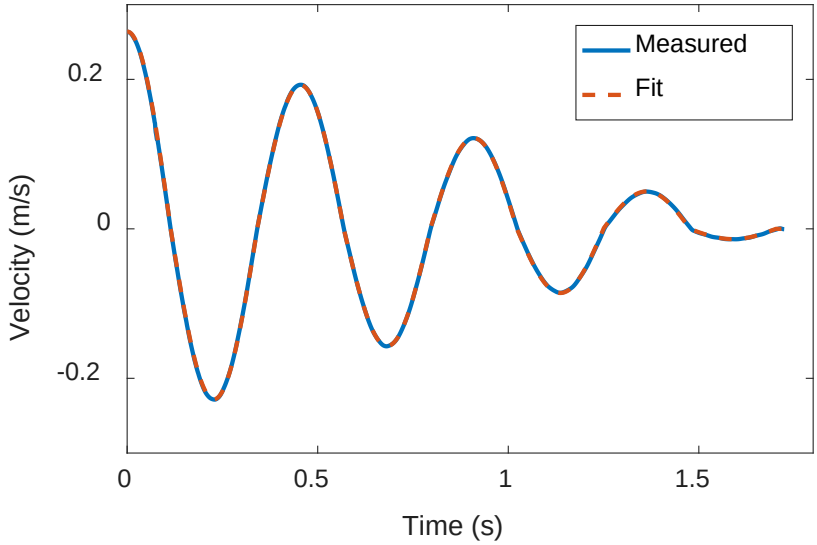

Figure 27 Equation 7.5 friction model fitting results for a 22 mm initial displacement test

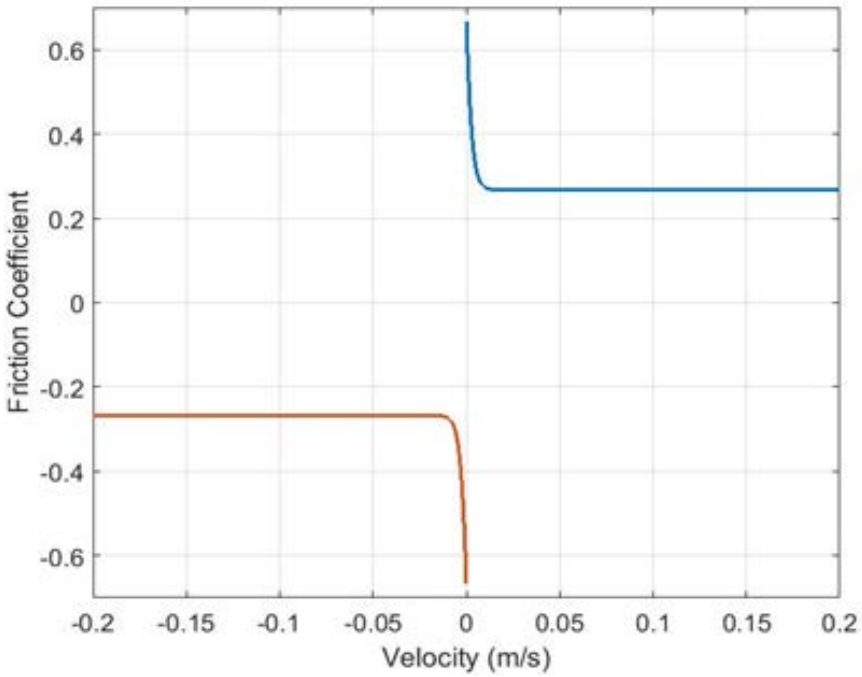

Figure 28 Friction coefficient versus velocity for the fitting results from figure 27

According to this model, the static and dynamic coefficients of friction were 0.63 and 0.24, respectively. Even though this model was able to find a good fit for all initial

displacement tests, the results varied considerably across the tests. However, the model provides an interesting method to approximate friction behavior.

7.6 Discussion

The friction models that were applied in this chapter differ greatly from one another, yet each one manages to provide a good fit to the measured data. The full range of motion data that the FMM provides is beneficial for studying the friction behavior in the stick to slip transition phase. The main goal of this research was to develop a new method for measuring friction for low velocity applications and has been achieved.

CHAPTER 8: FREQUENCY RESPONSE FUNCTION ANALYSIS

8.1 Frequency Response Function

In all previous experiments the input energy to produce FMM motion was supplied by moving the machine from its equilibrium position to a specific initial displacement and then releasing it. All subsequent analyses were performed in the time domain. In this chapter, impact tests are performed, where the energy input is a short duration impact, to obtain the FMM frequency response function (FRF). This enabled the structural dynamics of a system with a sliding contact to be studied in the frequency domain. Velocity and impact force data were transformed into the frequency domain via the discrete Fourier transform. The FMM's mobility and receptance were then calculated as the ratio of the response of the force input. The nonlinearity of a system with a sliding friction contact was observed by performing the impact tests at three different force levels. The previously presented single parameter friction fitting MATLAB code was modified by replacing the initial condition of displacement with that of an impact force. A single sliding (Coulomb) friction coefficient was determined for all impact force levels by fitting the simulated FRF to the experimental FRF.

8.2 Experimental Setup

For the purposes of this study, the input energy was supplied by the hammer impact which was applied to the motion platform at the middle of the leaf spring length to minimize platform rotation. The electromagnet was removed from the previous experimental setup to make room for the impact tests; see Figure 29.

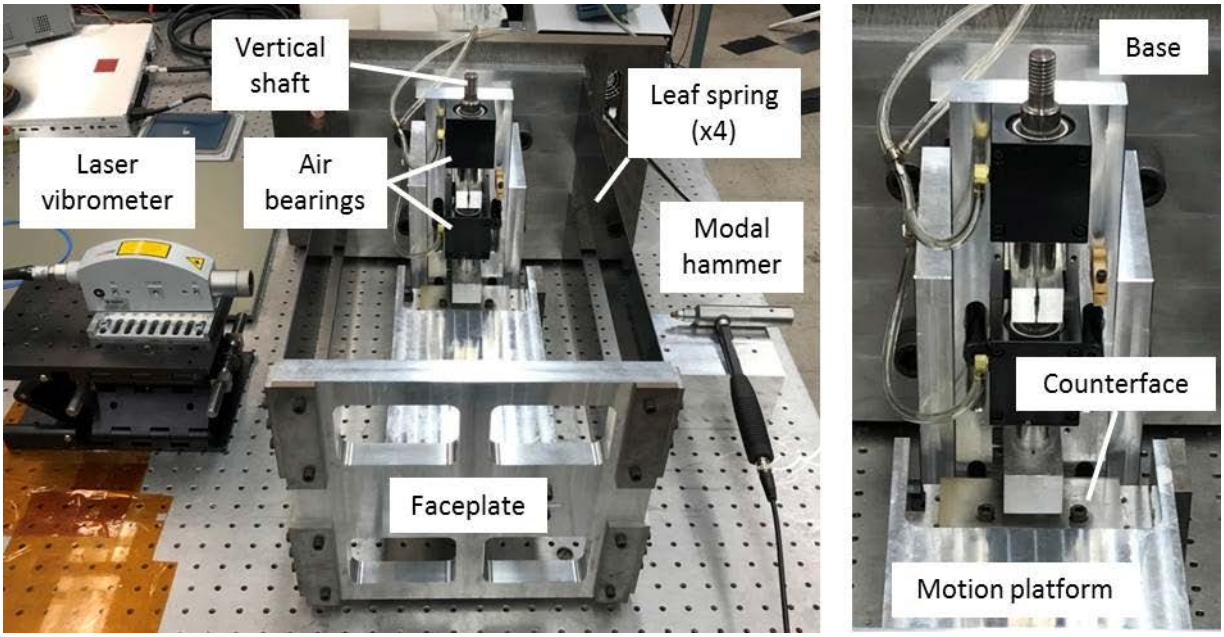

Figure 29 Photograph of FMM. The key components are identified.

The FMM friction contact for FRF testing was produced between the pin and counterface. The normal force between the pin and counterface was provided by a mass attached to the top of the vertical shaft. The mass for the tests completed in this study was 0.680 kg (normal force of 6.67 N).

For the tests performed here, the contact pair consisted of a polytetrafluouroethylene (PTFE) pin on a polished steel counterface. The interface was lubricated using CRC Ultra Lite 3-36 Ultra Thin Non Staining Lubricant. The lubrication was applied to ensure that each impact force level selected for testing would result in a sufficient number of oscillations during the decaying motion.

Three impact force ranges were applied to the FMM using the impact hammer. Since the impact hammer is a manual device and repeating the same force level with each impact was not possible, a tolerance of ±50 N was selected for impact acceptance. The nominal impact force levels were 450 N (low), 1000 N (medium), and 1450 N (high). The impact force was applied using a PCB 086D05 modally tuned hammer with added mass

and a rubber tip. The corresponding velocity was measured using a Polytec OFV-5000 laser vibrometer. The sampling time for each test was selected to be 33 s to ensure adequate frequency resolution for the FRFs. The sampling frequency was 10 kHz. The FMM was impacted by the hammer 10 times for each nominal force level. Between each impact, the CRC Ultra Lite 3-36 was reapplied to the counterface to ensure a consistent lubrication condition.

8.3 Experimental Results

The measured time domain force and velocity signals were imported into MATLAB for analysis. Figure 30 displays the impact force and velocity data for 10 trials at the medium impact level.

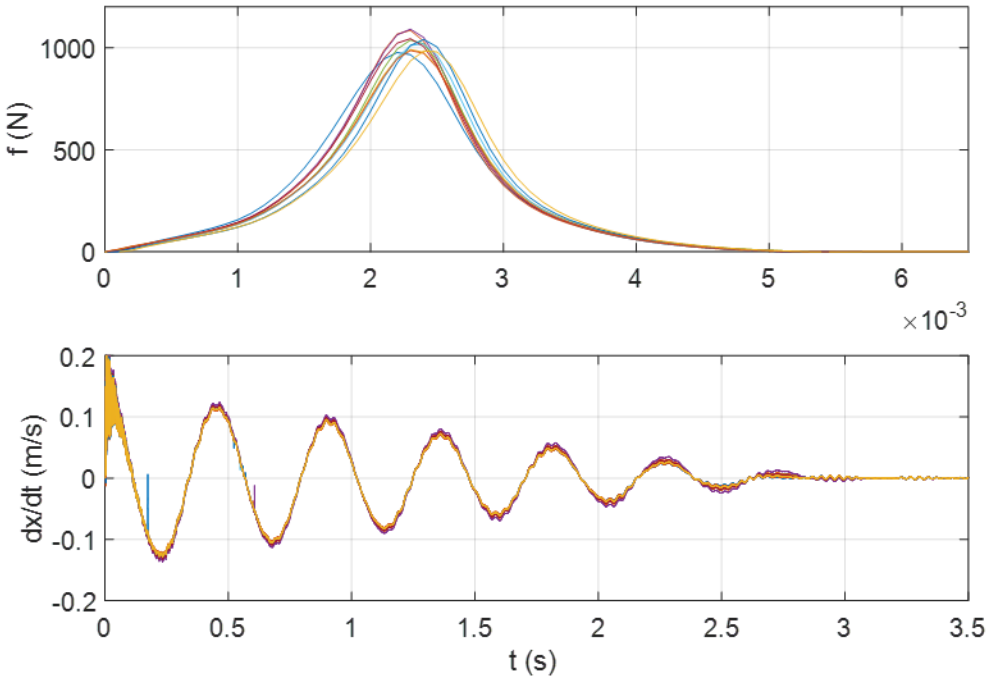

Figure 30 (Top) Time domain force, f, for 1000 N (medium) level. (Bottom) Time domain velocity, $\dot{x}$, due to force input. Note the change in time scale between the top and bottom panels.

For each of the three force levels, the impulse was calculated (i.e., the area under the time domain force profile). It was determined using the trapezoid rule with a step size of 1×10^{-4} s (i.e., the sampling period). Figure 31 displays the results. For the right panel, the mean impulse value at each force level is shown with an error bar that represents ±1 standard deviation over the 10 trials.

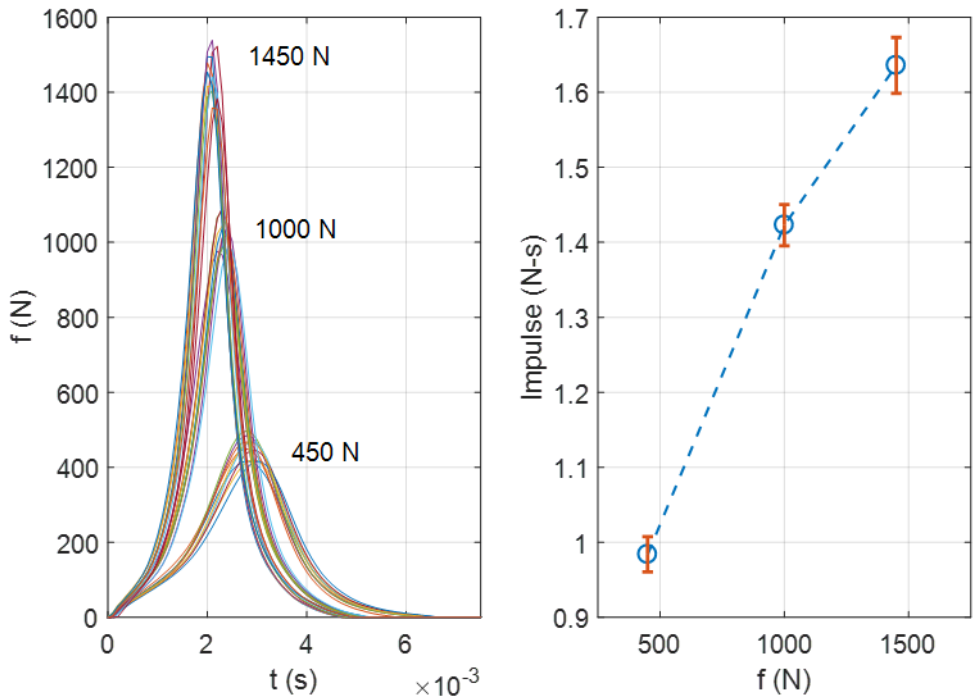

Figure 31 (Left) Three impact force levels. (Right) Impulse values for the three force levels.

The FRFs for the three force levels were determined by converting the time domain force and velocity signals into the frequency domain using the discrete Fourier transform. The mobility for each data set was then calculated by dividing the frequency domain velocity by the frequency domain force. Figure 32 displays the real and imaginary parts of the mobility for 10 trials at the medium force level. To convert to receptance, the mobility FRF was divided by $i\omega$. This follows from an assumption of harmonic motion, where $x = Xe^{i\omega t}$ and $\dot{x} = i\omega Xe^{i\omega t}$. The corresponding receptance is shown in Figure 33.

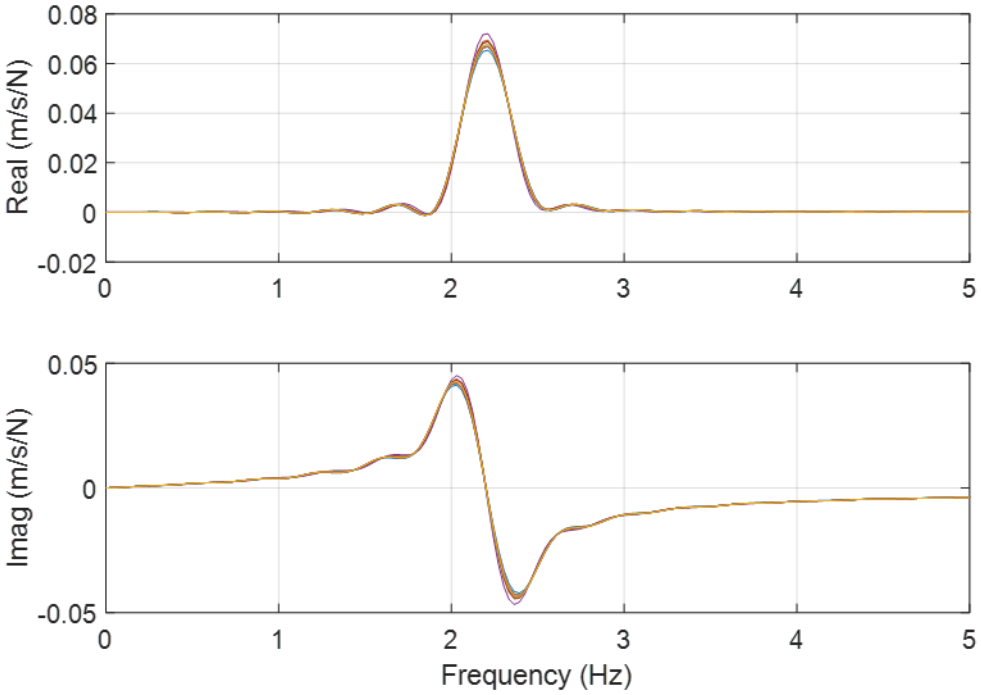

Figure 32 (Top) Real part of mobility FRF for 10 trials at the 1000 N (medium) force level. (Bottom) Imaginary part of mobility FRF for 10 trials

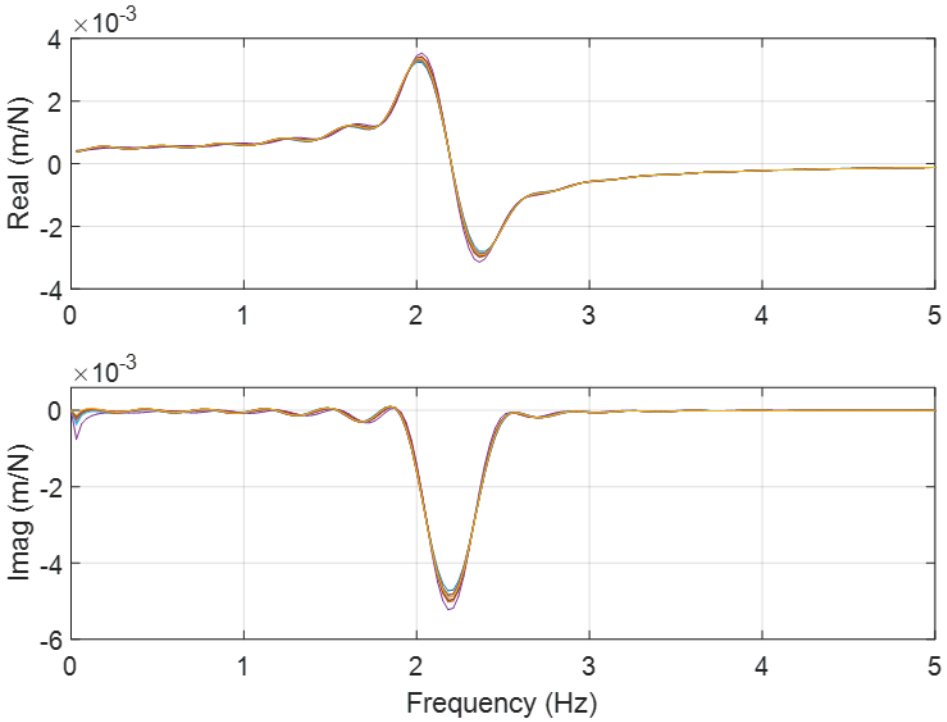

Figure 33 (Top) Real part of receptance FRF for 10 trials at the 1000 N (medium) force level. (Bottom) Imaginary part of receptance FRF for 10 trials.

The receptances are next compared for the three force levels. Figure 34 displays the mean FRF magnitudes for the low, medium, and high levels (10 trials each, averaged in the frequency domain). Figure 35 shows the impulse versus receptance magnitude for the three force levels (mean impulse values of 0.98 N-s, 1.42 N-s and 1.64 N-s), where the + symbol identifies the mean of 10 trials at each level. A linear trend is observed with an R^2 value very close to unity.

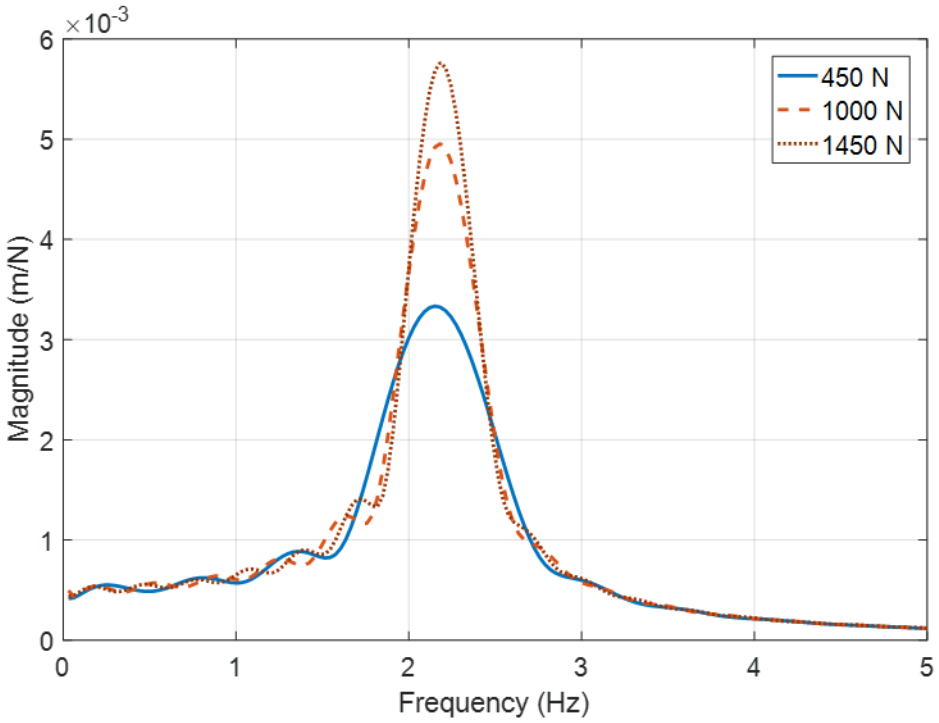

Figure 34 Mean receptance magnitude at three force levels. The magnitude increases with force for the FMM with friction contact.

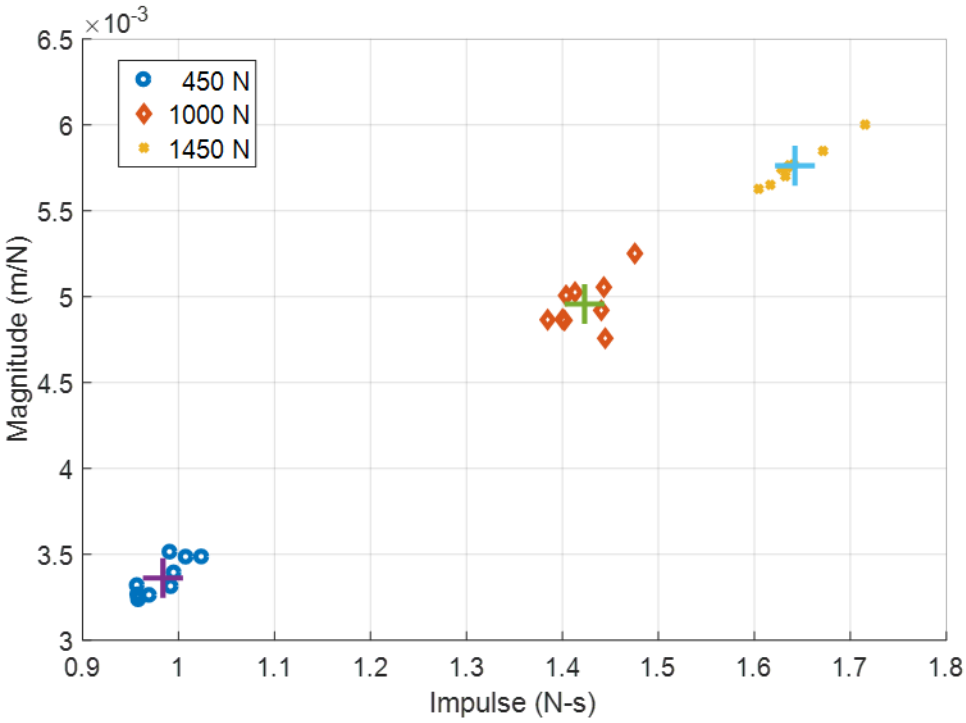

Figure 35 Impulse versus receptance magnitude for the three force levels.

Next, Equation 7.1 was modified to include structural damping for the FMM as shown in Equation 8.1. Equation 8.1 was solved by fixed time step numerical integration and the friction coefficient was identified to provide a best fit to the measured mobility FRFs. The measured force from the impact hammer, f, was used as input to Equation 8.1 to avoid introducing errors due to approximations of the excitation force. Because the sampling frequency for the measured force was 10 kHz, the numerical integration time step was 1×10^{-4} s.

$$\begin{aligned} m\ddot{x} + c\dot{x} + kx + F_f &= f, \dot{x} > 0 \\ m\ddot{x} + c\dot{x} + kx &= f, \dot{x} = 0 \\ m\ddot{x} + c\dot{x} + kx - F_f &= f, \dot{x} < 0 \end{aligned} \tag{8.1}$$

A result for the 450 N (low) force level is shown in Figure 36. The measured force is displayed in the top panel, while the simulated (solid line) and measured (dotted line) velocity are displayed in the bottom panel. The friction coefficient is 0.113. Figure 37

shows the corresponding mobility FRF. The 1000 N (medium) force levels results are provided in Figure 38 and Figure 39. The 1450 N (high) force level results are shown in Figure 40 and Figure 41. The friction coefficient is 0.113 in all cases.

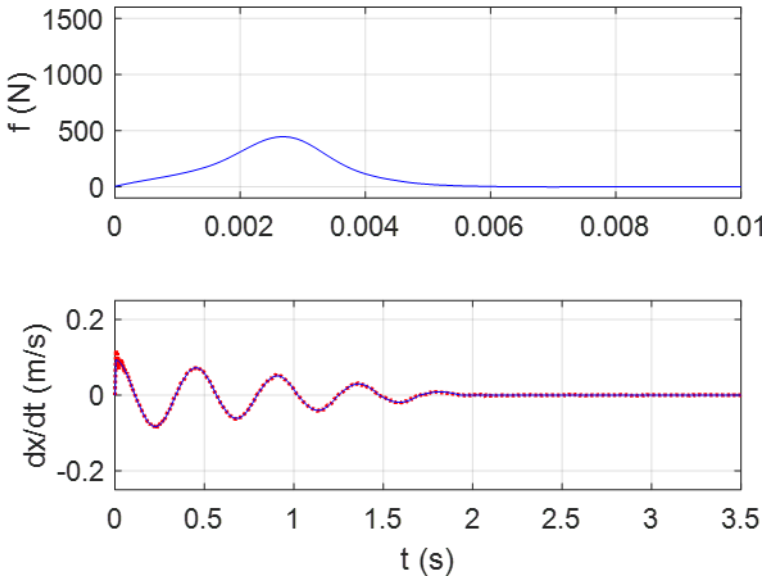

Figure 36 (Top) Low force level (450 N) input. (Bottom) Simulated (solid) and measured (dotted) velocity for 0.113 friction coefficient. Note that the top and bottom panels have different time scales

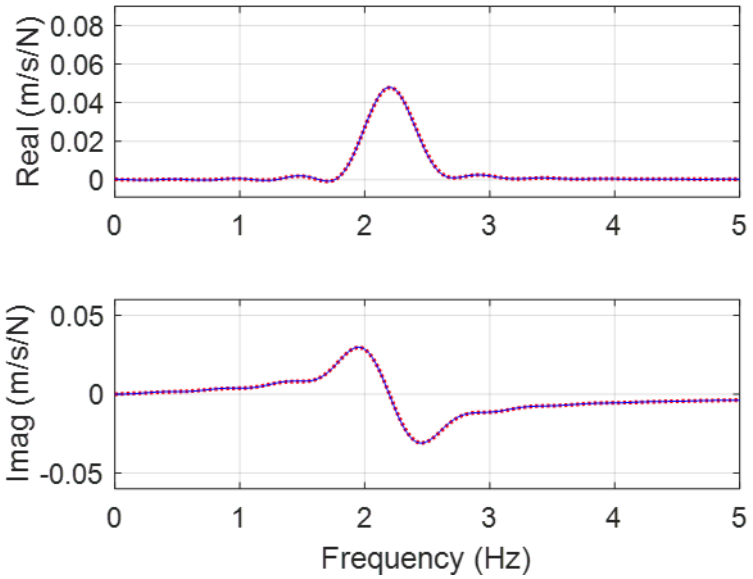

Figure 37 (Top) Real part of simulated (solid) and measured (dotted) mobility FRF for the low (450 N) force level with a friction coefficient of 0.113. (Bottom) Imaginary part of simulated and measured mobility FRFs.

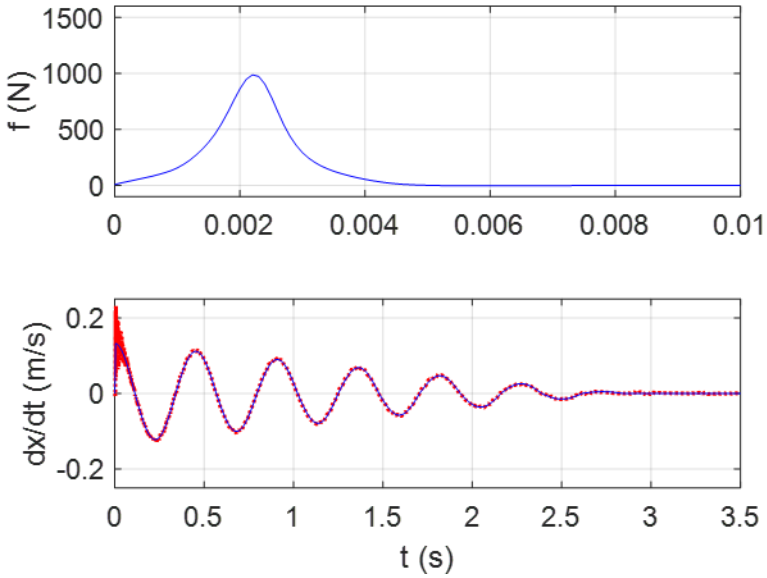

Figure 38 (Top) Medium force level (1000 N) input. (Bottom) Simulated (solid) and measured (dotted) velocity for 0.113 friction coefficient. Note that the top and bottom panels have different time scales

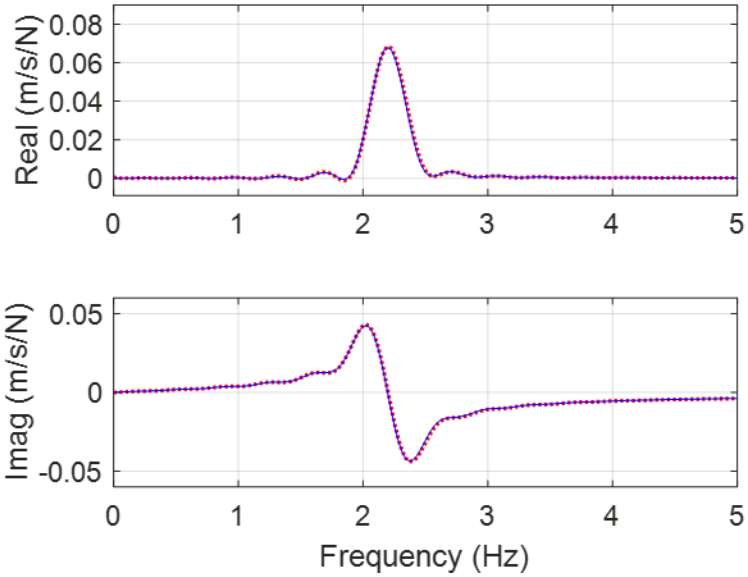

Figure 39 (Top) Real part of simulated (solid) and measured (dotted) mobility FRF for the medium (1000 N) force level with a friction coefficient of 0.113. (Bottom) Imaginary part of simulated and measured mobility FRFs.

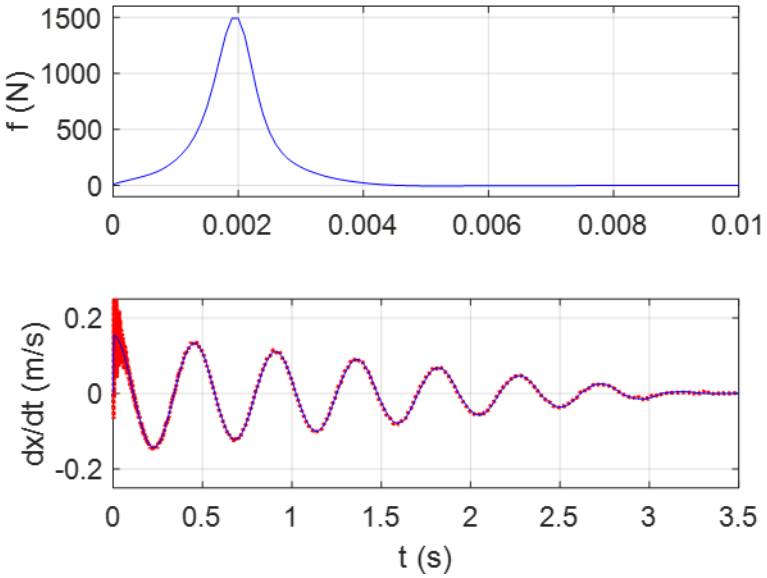

Figure 40 (Top) High force level (1450 N) input. (Bottom) Simulated (solid) and measured (dotted) velocity for 0.113 friction coefficient. Note that the top and bottom panels have different time scales

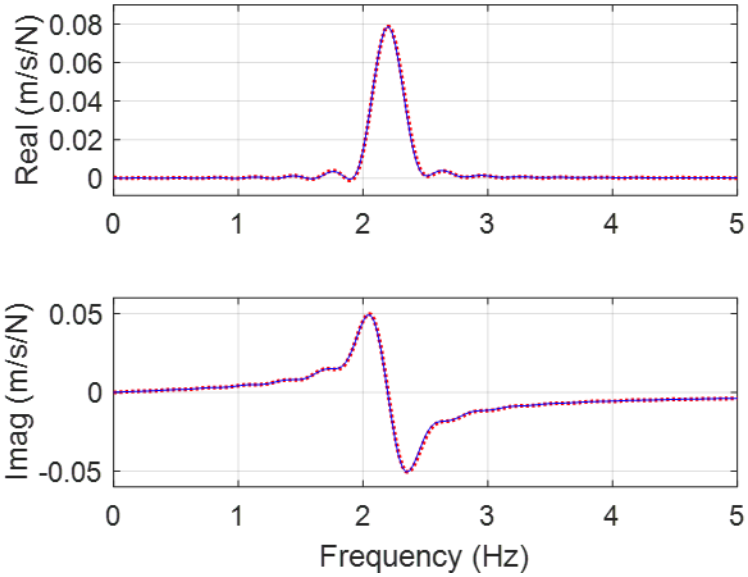

Figure 41 (Top) Real part of simulated (solid) and measured (dotted) mobility FRF for the high (1450 N) force level with a friction coefficient of 0.113. (Bottom) Imaginary part of simulated and measured mobility FRFs

The frequency range for the medium force level result is extended to 20 Hz in Figure 42. Interestingly, it is seen that odd multiples (3, 5, 7, 9, …) of the 2.2 Hz natural

frequency appear in both the simulated and measured mobility FRFs. This result was observed at all three force levels.

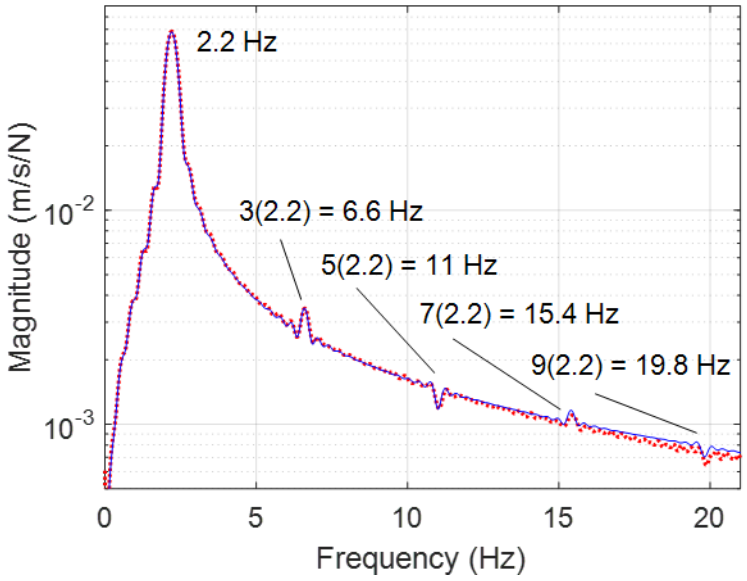

Figure 42 Logarithmic magnitude of simulated (solid) and measured (dotted) mobility FRF for the medium (1000 N) force level with a friction coefficient of 0.113.

CHAPTER 9: CONCLUSION AND FUTURE WORK

9.1 Conclusion

Free vibration (no friction contact) velocity data from the FMM was used to obtain the modal parameters for the system. It was shown that the FMM design produced negligible parasitic motion and enabled the structure to be modeled as a single degree of freedom spring-mass-damper system. The goal of this research was to show that the FMM could be used to determine friction coefficients using oscillating motion. Friction tests using a PTFE-polished steel contact pair were performed for a range of different initial displacements. A single parameter Coulomb friction model was used to demonstrate how it is possible to use the measured velocity data to determine the dynamic friction coefficient for the selected contact pair. These results also showed that friction is indeed a function of velocity, as many current friction models in practice have theorized.

Several additional friction models were applied to the FMM data. By separating the dynamic friction coefficient into two coefficients depending on acceleration it was discovered that there is a correlation between acceleration and friction coefficients, as experimentally observed in prior work. The dynamic friction coefficient was found to be larger when the magnitude of the velocity was increasing than when it was decreasing. The Stribeck effect was also modeled by applying a three parameter dynamic friction model [11] and a hyperbolic tangent function suggested by [1]. Both models were able to provide an excellent fit for the measured velocity and showed the existence of a nonlinear velocity dependence of the friction coefficient for the stick-to-slip transition phase.

Impact tests were performed on the structure with a friction contact and frequency response functions were obtained. This was done for a range of different impulses. The nonlinearity caused by the inclusion of a friction contact was confirmed. By calculating the

magnitude of the receptance of the system and applying an impulse simulation fitting code it was possible to determine the dynamic friction coefficient for the structure.

9.2 Future Work

The development and application of different friction models to study the transition phase of friction behavior is one of the most exciting aspects of future work. The FMM data could offer a great insight into this friction regime. If low constant velocity friction tests can be performed for the contact pair then more advanced models like the LuGre or Maxwell slip models could be used to further analyze friction behavior.

The analysis of different friction contact pairs would also be of great interest. It would be interesting to see if the friction response patterns discovered in this research for the PTFE-polished steel pair are present for different combinations of material as well.

The impact of the clamps on the damping coefficient and on the resulting friction coefficients is another FMM research task. “One of the key roles for friction at a contact interface is to provide passive damping to the structure, and this performance issue has serious implications in, for example, large space structures and other built-up structures whose inherent damping may be low… design of nominally stationary joints – bolted or riveted connections, clamped boundaries, etc. – requires careful consideration of friction modeling”[11]. There is a possibility that the velocity dependent damping in the system is the cause of friction result irregularities.

Work done by Further, Beards and Woohat shows that clamp force influences the dynamic characteristics of a frame by significantly altering its natural frequencies and mode shapes [11]. Tests could be performed on the FMM altering the clamp force holding the leaf springs in place to see if this significantly alters the natural frequency of the

structure. A comparison of damping in two similar structures, one monolithic and one assembled with clamped boundary conditions, would also aid in this analysis. Two scaled down versions of the FMM have been partially constructed; see Figure 43. Tests at different initial displacements could be performed on both structures to determine if increases in velocity also increase the viscous damping coefficient. If this is the case for only the assembled structure then the impact the clamps have on the damping coefficient can be confirmed.

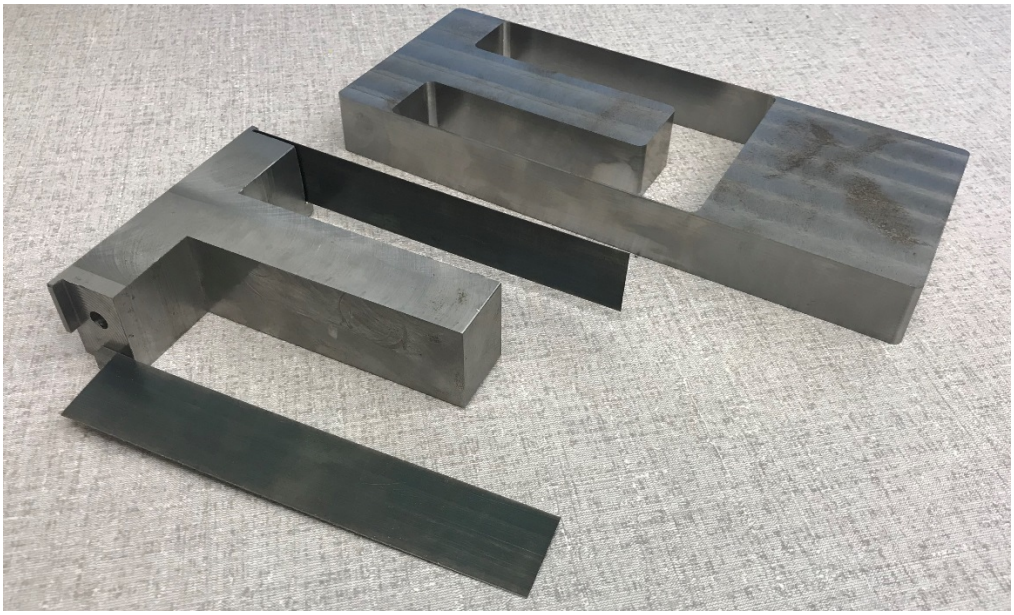

Figure 43 Scaled down version of the FMM: Monolithic (top) and assembled (bottom)

REFERENCES

[1] C. Makkar, C., W.e. Dixon, W.g. Sawyer, and G. Hu. *A New Continuously Differentiable Friction Model for Control Systems Design*. Proceedings, 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics., July 24, 2005. doi:10.1109/aim.2005.1511048.

[2] C. Canudas de Wit, H. Olsson, K.j. Astrom, and P. Lischinsky. *A New Model for Control of Systems with Friction.* IEEE Transactions on Automatic Control (Volume: 40, Issue: 3).March, 1995.

[3] P.R. Dahl. *A Solid Friction Model.* TOR-0158(3107-18)-1, The Aerospace Corporation, El Segundo, CA, 1968.

[4] B. Armstrong-Helouvry, P. Dupont, and C. Canudas De Wit. *A Survey of Models, Analysis Tools and Compensation Methods for the Control of Machines with Friction.* Automatica 30, no. 7 (July 1994): 1083-1138

[5] J. Swevers, F. Al-Bender, C.G. Ganseman, and T. Prajogo. *An Intergrated Friction Model Structure with Improved Presliding Behavior for Accurate Friction Compensation.* IEEE Transactions on Automatic Conrol 45, no. 4 (2000).

[6] Piatkowski, T. *Dahl and LuGre Dynamic Friction Models — The Analysis of Selected Properties*. Mechanism and Machine Theory 73 (2014): 91-100. doi:10.1016/j.mechmachtheory.2013.10.009.

[7] Majdoub, F., J. Perret-Liaudet, M. Belin, and J.m. Martin. *Decaying Law for the Free Oscillating Response with a Pseudo-polynomial Friction Law: Analysis of a Superlow Lubricated Friction Test*. Journal of Sound and Vibration 348 (2015): 263-81. doi:10.1016/j.jsv.2015.03.019.

[8] C. T. Lomascolo, J. Ziegert, and T. L. Schmitz. *Displacement-Based Measurement of Static and Dynamic Coefficients of Friction.* American Society for Precision Engineering Annual Meeting, October 23-28, Portland, OR, 2016.

[9] C. Canudas de Wit, H. Olsson, K.J. Astrom, and P. Lischinsky. *Dynamic Friction Models and Control Design.* IEEE American Control Conference, 1993.

[10] N. Axen, S. Hogmark, S. Jacobson. *Friction and Wear Measurement Techniques.* CRC Press, pp. 493-510. January 2001.

[11] Berger, E. *Friction Modeling for Dynamic System Simulation*. Applied Mechanics Reviews 55, no. 6 (2002): 535. doi:10.1115/1.1501080.

[12] Olsson, H., K.j. Åström, C. Canudas De Wit, M. Gäfvert, and P. Lischinsky. *Friction Models and Friction Compensation*. European Journal of Control 4, no. 3 (1998): 176-95. doi:10.1016/s0947-3580(98)70113-x.

[13] F. Al-Bender. *Fundamentals of Friction Modeling.* ASPE – The American Society of precision Engineering, Proceesings April 2010. 978-1-887706-57-706-53-7

[14] Wojewoda, J., A. Stefanski, M. Wiercigroch, and T. Kapitaniak. *Hysteretic Effects of Dry Friction: Modelling and Experimental Studies*. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 366, no. 1866 (2008): 747-65. doi:10.1098/rsta.2007.2125.

[15] Barabanov, Nikita, and Romeo Ortega. *Necessary and Sufficient Conditions for Passivity of the Lugre Friction Model 1*. IFAC Proceedings Volumes 32, no. 2 (1999): 2524-527. doi:10.1016/s1474-6670(17)56429-0.

[16] Elliott, S. J., M. Ghandchi Tehrani, and R. S. Langley. *Nonlinear Damping and Quasi- linear Modelling.* Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 373, no. 2051 (2015): 20140402. doi:10.1098/rsta.2014.0402.

[17] Zaitsev, S., R. Almog, O. Shtempluck, and E. Buks. *Nonlinear Dynamics in Nanomechanical Oscillators*. 2005 International Conference on MEMS,NANO and Smart Systems. doi:10.1109/icmens.2005.88.

[18] Haessig, Jr. David A., and Bernard Friedland. *On the Modeling and Simulation of Friction. Acquisition, Tracking, and Pointing V*, 1991. doi:10.1117/12.45712..

[19] Astrom, K.j., and C. Canudas-De-Wit. *Revisiting the LuGre Friction Model.* IEEE Control Systems 28, no. 6 (2008): 101-14. doi:10.1109/mcs.2008.929425..

[20] D. Li. *Static Coefficient of Friction Measurement using Tribometer.* Nanovea, 2014.

[21] T. L. Schmitz, J. E. Action, J. C. Ziegert, and W. G. Sawyer. *The Difficulty of Measuring Low Friction: Uncertainty Analysis for Friction Coefficient Measurements.* Journal of Tribology, 127: 673-678, 2005.

[22] T. Schmitz and S. Smith. *Mechanical Vibrations: Modeling and Measurement.* Springer, New York, NY, 2012.

[23] S. Smith. *Flexures: Elements of Elastic Mechanisms.*CRC Press LLC, London, UK, 2000.

[24] Szeri, A.Z. *Tribology: Friction, Lubrication and Wear.* Hemisphere Publishing. 1980.

[25] Kossack, C, Schmitz, T, and Ziegert, J. *Identification of friction energy dissipation using free vibration velocity: measurement and modeling.* American Society for Precision Engineering Annual Meeting, October 29-November 3, Charlotte, NC. 2017.

APPENDIX

A1 Free_Vibration_no_friction_fitting.m

```
clc;close all;clear all;

plotNo = 1;
testend = 5e3;                  % length of time to be fit

% ******************** SETUP ***************************************

% ************************************************************ Test 1
load('NF20mm1');
time1 = NF20mm1(:, 1);
velocity1 = NF20mm1(:, 2);

index = find(velocity1 == max(velocity1))
index(1)
time1 = time1(index(1) - 2010 : length(time1));
velocity1 = velocity1(index(1) - 2010 : length(velocity1));
time1 = time1 - time1(1);

velocity1 = smooth(velocity1);
index = find(velocity1 == max(velocity1));
index = index(1) - 2000;

offset = mean(velocity1(index:index+1000));
velocity1 = velocity1 - offset;
plot(time1,velocity1)
index = find(velocity1 == max(velocity1));

index = index(1) + 12000;
time1 = time1(index:length(time1));
velocity1 = velocity1(index:length(velocity1));

index = find(velocity1 == max(velocity1));
time1 = time1(index:length(time1));
time1 = time1 - time1(1);
velocity1 = velocity1(index:length(velocity1));

time1 = time1(1:testend);
velocity1 = velocity1(1:testend);

clear NF20mm1;

% ************************************************************ Test 2
load('NF20mm2');
time2 = NF20mm2(:, 1);
velocity2 = NF20mm2(:, 2);

index = find(velocity2 == max(velocity2));
time2 = time2(index - 2500 : length(time2));
```

```
velocity2 = velocity2(index - 2500 : length(velocity2));
time2 = time2 - time2(1);

velocity2 =smooth(velocity2);
index = find(velocity2 == max(velocity2));
index = index(1) - 2000;

offset = mean(velocity2(index:index+1000));
velocity2 = velocity2 - offset;

index = find(velocity2 == max(velocity2));

index = index(1) + 12000;
time2 = time2(index:length(time2));
velocity2 = velocity2(index:length(velocity2));

index = find(velocity2 == max(velocity2));
time2 = time2(index:length(time2));
time2 = time2 - time2(1);
velocity2 = velocity2(index:length(velocity2));

time2 = time2(1:testend);
velocity2 = velocity2(1:testend);

clear NF20mm2
% ************************************************************* Test 3
load('NF20mm3');
time3 = NF20mm3(:, 1);
velocity3 = NF20mm3(:, 2);

index = find(velocity3 == max(velocity3));
time3 = time3(index - 2500 : length(time3));
velocity3 = velocity3(index - 2500 : length(velocity3));
time3 = time3 - time3(1);

velocity3 =smooth(velocity3);
index = find(velocity3 == max(velocity3));
index = index(1) - 2000;

offset = mean(velocity3(index:index+1000));
velocity3 = velocity3 - offset;

index = find(velocity3 == max(velocity3));

index = index(1) + 12000;
time3 = time3(index:length(time3));
velocity3 = velocity3(index:length(velocity3));

index = find(velocity3 == max(velocity3));
time3 = time3(index:length(time3));
time3 = time3 - time3(1);
velocity3 = velocity3(index:length(velocity3));

time3 = time3(1:testend);
```

```
velocity3 = velocity3(1:testend);

clear NF20mm3
% ************************************************************* Test 4
load('NF20mm4');
time4 = NF20mm4(:, 1);
velocity4 = NF20mm4(:, 2);

index = find(velocity4 == max(velocity4));
time4 = time4(index - 2500 : length(time4));
velocity4 = velocity4(index - 2500 : length(velocity4));
time4 = time4 - time4(1);
velocity4 =smooth(velocity4);

index = find(velocity4 == max(velocity4));
index = index(1) - 2000;
offset = mean(velocity4(index:index+1000));
velocity4 = velocity4 - offset;

index = find(velocity4 == max(velocity4));

index = index(1) + 12000;
time4 = time4(index:length(time4));
velocity4 = velocity4(index:length(velocity4));

index = find(velocity4 == max(velocity4));
time4 = time4(index:length(time4));
time4 = time4 - time4(1);
velocity4 = velocity4(index:length(velocity4));

time4 = time4(1:testend);
velocity4 = velocity4(1:testend);

clear NF20mm4
% ************************************************************* Test 5
load('NF20mm5');
time5 = NF20mm5(:, 1);
velocity5 = NF20mm5(:, 2);

index = find(velocity5 == max(velocity5));
time5 = time5(index - 2500 : length(time5));
velocity5 = velocity5(index - 2500 : length(velocity5));
time5 = time5 - time5(1);

velocity5 = smooth(velocity5);

index = find(velocity5 == max(velocity5));
index = index(1) - 2000;

offset = mean(velocity5(index:index+1000));
velocity5 = velocity5 - offset;

index = find(velocity5 == max(velocity5));
```

```
index = index(1) + 12000;
time5 = time5(index:length(time5));
velocity5 = velocity5(index:length(velocity5));

index = find(velocity5 == max(velocity5));
time5 = time5(index:length(time5));
time5 = time5 - time5(1);
velocity5 = velocity5(index:length(velocity5));

time5 = time5(1:testend);
velocity5 = velocity5(1:testend);

clear NF20mm5
% ************************************************************* Test 6
load('NF20mm6');
time6 = NF20mm6(:, 1);
velocity6 = NF20mm6(:, 2);

index = find(velocity6 == max(velocity6));
time6 = time6(index - 2500 : length(time6));
velocity6 = velocity6(index - 2500 : length(velocity6));
time6 = time6 - time6(1);

velocity6 = smooth(velocity6);

index = find(velocity6 == max(velocity6));
index = index(1) - 2000;
offset = mean(velocity6(index:index+1000));
velocity6 = velocity6 - offset;

index = find(velocity6 == max(velocity6));

index = index(1) + 12000;
time6 = time6(index:length(time6));
velocity6 = velocity6(index:length(velocity6));

index = find(velocity6 == max(velocity6));
time6 = time6(index:length(time6));
time6 = time6 - time6(1);
velocity6 = velocity6(index:length(velocity6));

time6 = time6(1:testend);
velocity6 = velocity6(1:testend);

clear NF20mm6
% ************************************************************* Test 7
load('NF20mm7');
time7 = NF20mm7(:, 1);
velocity7 = NF20mm7(:, 2);

index = find(velocity7 == max(velocity7));
time7 = time7(index - 2500 : length(time7));
velocity7 = velocity7(index - 2500 : length(velocity7));
time7 = time7 - time7(1);
```

```
velocity7 = smooth(velocity7);

index = find(velocity7 == max(velocity7));
index = index(1) - 2000;
offset = mean(velocity7(index:index+1000));
velocity7 = velocity7 - offset;

index = find(velocity7 == max(velocity7));

index = index(1) + 12000;
time7 = time7(index:length(time7));
velocity7 = velocity7(index:length(velocity7));

index = find(velocity7 == max(velocity7));
time7 = time7(index:length(time7));
time7 = time7 - time7(1);
velocity7 = velocity7(index:length(velocity7));

time7 = time7(1:testend);
velocity7 = velocity7(1:testend);

clear NF20mm7
% ************************************************************* Test 8
load('NF20mm8');
time8 = NF20mm8(:, 1);
velocity8 = NF20mm8(:, 2);

index = find(velocity8 == max(velocity8));
time8 = time8(index - 2500 : length(time8));
velocity8 = velocity8(index - 2500 : length(velocity8));
time8 = time8 - time8(1);

velocity8 = smooth(velocity8);

index = find(velocity8 == max(velocity8));
index = index(1) - 2000;
offset = mean(velocity8(index:index+1000));
velocity8 = velocity8 - offset;

index = find(velocity8 == max(velocity8));

index = index(1) + 12000;
time8 = time8(index:length(time8));
velocity8 = velocity8(index:length(velocity8));

index = find(velocity8 == max(velocity8));
time8 = time8(index:length(time8));
time8 = time8 - time8(1);
velocity8 = velocity8(index:length(velocity8));

time8 = time8(1:testend);
velocity8 = velocity8(1:testend);

clear NF20mm8
```

```
% ************************************************************* Test 9
load('NF20mm9');
time9 = NF20mm9(:, 1);
velocity9 = NF20mm9(:, 2);

index = find(velocity9 == max(velocity9));
time9 = time9(index - 2500 : length(time9));
velocity9 = velocity9(index - 2500 : length(velocity9));
time9 = time9 - time9(1);

velocity9 = smooth(velocity9);

index = find(velocity9 == max(velocity9));

index = index(1) - 2000;

offset = mean(velocity9(index:index+1000));
velocity9 = velocity9 - offset;
index = find(velocity9 == max(velocity9));

index = index(1) + 12000;
time9 = time9(index:length(time9));
velocity9 = velocity9(index:length(velocity9));

index = find(velocity9 == max(velocity9));
time9 = time9(index:length(time9));
time9 = time9 - time9(1);
velocity9 = velocity9(index:length(velocity9));

time9 = time9(1:testend);
velocity9 = velocity9(1:testend);

clear NF20mm9
% ************************************************************* Test
10
load('NF20mm10');
time10 = NF20mm10(:, 1);
velocity10 = NF20mm10(:, 2);

index = find(velocity10 == max(velocity10));
time10 = time10(index - 2500 : length(time10));
velocity10 = velocity10(index - 2500 : length(velocity10));
time10 = time10 - time10(1);

velocity10 = smooth(velocity10);

index = find(velocity10 == max(velocity10));
index = index(1) - 2000;
offset = mean(velocity10(index:index+1000));
velocity10 = velocity10 - offset;

index = find(velocity10 == max(velocity10));

index = index(1) + 12000;
```

```
time10 = time10(index:length(time10));
velocity10 = velocity10(index:length(velocity10));

index = find(velocity10 == max(velocity10));
time10 = time10(index:length(time10));
time10 = time10 - time10(1);
velocity10 = velocity10(index:length(velocity10));

time10 = time10(1:testend);
velocity10 = velocity10(1:testend);

clear NF20mm10
% ***************** END SETUP
*****************************************


figure(plotNo)
plot(time1,velocity1,time2,velocity2,time3,velocity3,time4,velocity4,..
.
    time5,velocity5,time6,velocity6,time7,velocity7,time8,velocity8,...
   time9,velocity9,time10,velocity10)
plotNo = plotNo + 1;

Results_20mm = zeros(10,5);

for test = 1:10

if test == 1
    dx=velocity1(1);
    velocity = velocity1;
    time = time1;
end

if test == 2
    dx=velocity2(1);
    velocity = velocity2;
    time = time2;
end

if test == 3
    dx=velocity3(1);
    velocity = velocity3;
    time = time3;
end

if test == 4
    dx=velocity4(1);
    velocity = velocity4;
    time = time4;
end

if test == 5
    dx=velocity5(1);
    velocity = velocity5;
    time = time5;
end
```

```
if test == 6
    dx=velocity6(1);
    velocity = velocity6;
    time = time6;
end

if test == 7
    dx=velocity7(1);
    velocity = velocity7;
    time = time7;
end

if test == 8
    dx=velocity8(1);
    velocity = velocity8;
    time = time8;
end

if test == 9
    dx=velocity9(1);
    velocity = velocity9;
    time = time9;
end

if test == 10
    dx=velocity10(1);
    velocity = velocity10;
    time = time10;
end

if test ==1
    m = 14.4;       % guess for mass in kg
    k = 1980;       % guess for stiffness in N/m
    c1 = 0.27;      % guess for damping N·s/m
    dista = 1e-3;   % guess for position at t0
else
    m = x(1);
    k = x(2);
    c1 = x(3);
    dista = x(4);
end

x0 = [m k c1 dista];

LB = [1      0    0   -12e-3 ];
UB = [100  3000  100   12e-3 ];

options = optimset('Display', 'iter', 'MaxIter', 2000, 'MaxFunEvals',
5000);
[x, resnorm, residual, exitflag, output] = lsqnonlin(@fit_func_Vmax,
x0, LB, UB, options,time,velocity,plotNo);

 Results_20mm(test,1) = x(1);Results_20mm(test,2) = x(2);
Results_20mm(test,3) = x(3);
 Results_20mm(test,4) = x(4);
```

```
 plotNo = plotNo + 1;

 end

function [PI] = fit_func_Vmax(x,time,velocity,plotNo)

dx = velocity(1);

m = x(1);
k = x(2);
c1 = x(3);
dista = x(4);

dt = time(2) - time(1);
steps = length(time) - 1;
vel_f = zeros(length(time),1);
vel_f(1,1)=dx;
distance(1) = dista;

for cnt = 1:steps

        dx_previous = dx;
        if dx_previous > 0                      % velocity is positive
            ddx = ( - c1*dx - k*dista)/m;         % m/s^2
            dx = dx + ddx*dt;                     % m/s
            dista = dista + dx*dt;                % m
            vel_f(cnt+1,1) = dx;
            distance(cnt+1)= dista;
        else                                    % velocity is negative
            ddx = ( - c1*dx - k*dista)/m;         % m/s^2
            dx = dx + ddx*dt;                     % m/s
            dista = dista + dx*dt;                % m
            vel_f(cnt+1,1) = dx;
            distance(cnt+1)= dista;
        end

end

Actual(1,:) = velocity(1:length(velocity));
Simulation(1,:) = vel_f(1:length(vel_f));

vel_diff = Actual - Simulation;

% Performance index, PI, is the objective function to minimize
PI = abs(vel_diff)*1e3;
```

A2 Log_Dec_No_Friction.m

```
clc;
clear all;
close all;

% Load data
load('Vib10mm80sec');

m = 10.415;     % kg
k = 1982.17;    % N/m

% ************** 10 ************
time = Vib10mm80sec(:, 1);
velocity = Vib10mm80sec(:, 2);

index = find(velocity == max(velocity));
time = time(index:length(time));
time = time-time(1);
velocity = velocity(index:length(velocity));
velocity = movmean(velocity,15);

plot(time,velocity);

[pks_find ,locs_find]=findpeaks(velocity);

cnt3=1;

for i = 3 : length(pks_find)

    if pks_find(i) > 0
        pks_2(cnt3) = pks_find(i);
        zero_locs_2(cnt3) = locs_find(i);
        cnt3=cnt3+1;
    else
    end

end

cnt3 = 1;

for i = 2: length(zero_locs_2)

    difference(i-1) = zero_locs_2(i) - zero_locs_2(i-1);

    if difference(i-1) > 400

        pks(cnt3) = pks_2(i);
        zero_locs(cnt3) = zero_locs_2(i);
        cnt3 = cnt3+1;

    end
end
```

```
for i = 2 : length(zero_locs)

    cycles_test(i-1) = i-1;
    difference2(i-1) = zero_locs(i) - zero_locs(i-1);

end

for i = 2 : length(pks)

    gamma = log(pks(i-1)/pks(i));

    zeta = (gamma)/(sqrt((4*pi^2+gamma^2)));

    c_log_dec (i-1) = zeta * 2*sqrt(k*m);
    v0_cycle(i-1) = pks(i-1);
    cycle(i-1)=i-1;
end

figure(2)
plot(v0_cycle,c_log_dec,'LineWidth',1)
xlabel('Peak Cycle Velocity (m/s)');ylabel('c (N\cdots/m)');
set(gca,'FontSize',16);
axis([0.05 0.14 0.21 0.32])
```

A3 Monte_Carlo_single_parameter_friction_model.m

```
clc;close all;clear all;

load('Results_6mm'); load('Results_8mm'); load('Results_10mm');
% Results_Xmm Columns: 1.) m , 2.) k, 3.) c , 4.) A , 5.) d0

%%      Combining all displacement fit results for m,k,c,A,d0 vectors

m_combined(1:10) = Results_6mm(:,1);
m_combined(11:20) = Results_8mm(:,1);
m_combined(21:30) = Results_10mm(:,1);

k_combined(1:10) = Results_6mm(:,2);
k_combined(11:20) = Results_8mm(:,2);
k_combined(21:30) = Results_10mm(:,2);

c_combined(1:10) = Results_6mm(:,3);
c_combined(11:20) = Results_8mm(:,3);
c_combined(21:30) = Results_10mm(:,3);

A_combined(1:10) = Results_6mm(:,4);
A_combined(11:20) = Results_8mm(:,4);
A_combined(21:30) = Results_10mm(:,4);

d0_combined(1:10) = Results_6mm(:,5);
d0_combined(11:20) = Results_8mm(:,5);
d0_combined(21:30) = Results_10mm(:,5);

%%      Mean and Standard Deviation for system components

m_mean = mean(m_combined);
m_stdev = std(m_combined);

k_mean = mean(k_combined);
k_stdev = std(k_combined);

c_mean = mean(c_combined);
c_stdev = std(c_combined);

A_mean = mean(A_combined);
A_stdev = std(A_combined);

d0_mean = mean(d0_combined);
d0_stdev = std(d0_combined);

%%     Create normal distribution for variable being tested
%      and pick 100,000 random samples

n=100000;
x1_dist=makedist('Normal','mu',m_mean,'sigma',m_stdev);
x1=random(x1_dist,n,1);
x1_stats= [mean(x1) std(x1) var(x1)];
```

```
figure(1)
histogram(x1,100,'Normalization','probability');
title('m random sample histogram')

n=100000;
x2_dist=makedist('Normal','mu',k_mean,'sigma',k_stdev);
x2=random(x2_dist,n,1);
x2_stats= [mean(x2) std(x2) var(x2)];

figure(2)
histogram(x1,100,'Normalization','probability');
title('k random sample histogram')

n=100000;
x3_dist=makedist('Normal','mu',c_mean,'sigma',c_stdev);
x3=random(x3_dist,n,1);
x3_stats= [mean(x3) std(x3) var(x3)];

figure(3)
histogram(x1,100,'Normalization','probability');
title('c random sample histogram')

n=100000;
x4_dist=makedist('Normal','mu',A_mean,'sigma',A_stdev);
x4=random(x4_dist,n,1);
x4_stats= [mean(x4) std(x4) var(x4)];

figure(4)
histogram(x1,100,'Normalization','probability');
title('A random sample histogram')


%%                                                              Set
Constants
plotNo = 5;
mud = 0.15;
v0 = 0.101;
dista = 0;
t = 0:0.001:1.5;
N = 7.2;

%%   Call function to create "data" at selected μ

velocity_sim_data = create_data(m_mean,k_mean,c_mean,N,A_mean,dista,...
    v0,t,mud);

%%  Loop that takes each random sample and runs nonlinear least squares
fit to "data"
Results = zeros(n,2);

    for test = 1:length(x1)

        % model parameters
        m = x1(test);     % kg
```

```
        k = x2(test);     % N/m                   CHANGE DESIRED
VARIABLE HERE
        c = x3(test);     % N-s/m
        A = x4(test);

        dista = 0.0001;
        x0 = [mud dista];

        LB = [0.1 -0.003 ];
        UB = [1    0.003 ];

        options = optimset('Display', 'iter', 'MaxIter', 2000, ...
            'MaxFunEvals', 5000);
        [x, resnorm, residual, exitflag, output] = ...
            lsqnonlin(@fit_func_friction_monte, x0, LB, UB,
options,t,...
            velocity_sim_data,m,k,c,A,N,plotNo);

        Results(test,1) = x(1);
        Results(test,2) = x(2);

    end

%%   Mean and Standard Deviation for μ from fit results
mud_mean = mean(Results(:,1));
mud_stdev = std(Results(:,1));

mud_mean_ALL_variable = mean(Results(:,1));
mud_stdev_ALL_variable = std(Results(:,1));

%%   Normal Probability Density Function Plot
H = mud_mean-4*mud_stdev:1e-6:mud_mean+4*mud_stdev;
G = normpdf(H,mud_mean,mud_stdev);

figure(plotNo)
plot(H,G)
xlabel('mud')
ylabel('probability density')
title('μ-NPDF w/ ALL PARAMETERS VARIABLE') % CHANGE TITLE depending on
test
plotNo = plotNo+1;

function [ velocity_sim_data ] = create_data( m,k,c,N,A,dista,v0,t,mud
)

% This function creates "data" for a specific selected mud value using
% the mean values for the system dynamics found via free vibration fit

dx_previous = v0;
mus = mud;
dx = v0;
dt = t(2)-t(1);
vel_f(1) = dx;
```

```
for cnt = 1:length(t)-1
    % friction
    if dx*dx_previous > 0   % velocity is not zero
        dx_previous = dx;
        if dx_previous > 0  % velocity is positive
            ddx = (-mud*N - c*dx -A*dx^2- k*dista)/m;
            dx = dx + ddx*dt;
            dista = dista + dx*dt;
            pos_f(cnt) = dista;
            vel_f(cnt+1) = dx;
            acc_f(cnt) = ddx;
            Friction(cnt) = -mud*N;
        else   % velocity is negative
            ddx = (mud*N - c*dx +A*dx^2 - k*dista)/m;
            dx = dx + ddx*dt;
            dista = dista + dx*dt;
            pos_f(cnt) = dista;
            vel_f(cnt+1) = dx;
            acc_f(cnt) = ddx;
            Friction(cnt) = mud*N;
        end
    else % velocity is zero (passed through zero)

        dx_previous = dx;
        if dista < mus*N/k && dista > -mus*N/k
            ddx = 0;
            dx = 0;
            pos_f(cnt) = dista;
            vel_f(cnt+1) = dx;
            acc_f(cnt) = ddx;
            Friction(cnt) = k*dista;

        else    % motion continues
            if dista > 0
                ddx = (mus*N - c*dx + A*dx^2- k*dista)/m;
                dx = dx + ddx*dt;
                dista = dista + dx*dt;
                pos_f(cnt) = dista;
                vel_f(cnt+1) = dx;
                acc_f(cnt) = ddx;
                Friction(cnt) = mus*N;
            else
                ddx = (-mus*N - c*dx - A*dx^2- k*dista)/m;
                dx = dx + ddx*dt;
                dista = dista + dx*dt;
                pos_f(cnt) = dista;
                vel_f(cnt+1) = dx;
                acc_f(cnt) = ddx;
                Friction(cnt) = -mus*N;
            end
        end
    end
end
velocity_sim_data = vel_f;

end
```

A4 Friction_Fitting_Single_Parameter.m

```
clc;close all;clear all;

load('PS18mm1'); load('PS18mm2'); load('PS18mm3');
load('PS18mm4'); load('PS18mm5'); load('PS18mm6');
load('PS18mm7'); load('PS18mm8'); load('PS18mm9');
load('PS18mm10');

plotNo = 1;
testend = 1750;

% ******************** SETUP ****************************************

% ********************************************************** Test 1

time1 = PS18mm1(:, 1);
velocity1 = PS18mm1(:, 2);

velocity1 =smooth(velocity1);

index = find(velocity1 == max(velocity1));

index = index(1) - 1900;

offset = mean(velocity1(index:index+824));
velocity1 = velocity1 - offset;

index = find(velocity1 == max(velocity1));
time1 = time1(index:length(time1));
time1 = time1 - time1(1);
velocity1 = velocity1(index:length(velocity1));

time1 = time1(1:testend);
velocity1 = velocity1(1:testend);

plot(time1,velocity1);

% ********************************************************** Test 2

time2 = PS18mm2(:, 1);
velocity2 = PS18mm2(:, 2);

velocity2 =smooth(velocity2);

index = find(velocity2 == max(velocity2));

index = index(1) - 1900;

offset = mean(velocity2(index:index+824));
velocity2 = velocity2 - offset;
```

```
index = find(velocity2 == max(velocity2));
time2 = time2(index:length(time2));
time2 = time2 - time2(1);
velocity2 = velocity2(index:length(velocity2));

time2 = time2(1:testend);
velocity2 = velocity2(1:testend);

plot(time2,velocity2);
% ************************************************************ Test 3

time3 = PS18mm3(:, 1);
velocity3 = PS18mm3(:, 2);

velocity3 =smooth(velocity3);

index = find(velocity3 == max(velocity3));

index = index(1) - 1900;

offset = mean(velocity3(index:index+824));
velocity3 = velocity3 - offset;

index = find(velocity3 == max(velocity3));
time3 = time3(index:length(time3));
time3 = time3 - time3(1);
velocity3 = velocity3(index:length(velocity3));

time3 = time3(1:testend);
velocity3 = velocity3(1:testend);

plot(time3,velocity3);
% ************************************************************ Test 4

time4 = PS18mm4(:, 1);
velocity4 = PS18mm4(:, 2);

velocity4 =smooth(velocity4);

index = find(velocity4 == max(velocity4));

index = index(1) - 1900;

offset = mean(velocity4(index:index+824));
velocity4 = velocity4 - offset;

index = find(velocity4 == max(velocity4));
time4 = time4(index:length(time4));
time4 = time4 - time4(1);
velocity4 = velocity4(index:length(velocity4));

time4 = time4(1:testend);
velocity4 = velocity4(1:testend);
```

```
plot(time4,velocity4);
% ************************************************************ Test 5

time5 = PS18mm5(:, 1);
velocity5 = PS18mm5(:, 2);

velocity5 =smooth(velocity5);

index = find(velocity5 == max(velocity5));

index = index(1) - 1900;

offset = mean(velocity5(index:index+824));
velocity5 = velocity5 - offset;

index = find(velocity5 == max(velocity5));
time5 = time5(index:length(time5));
time5 = time5 - time5(1);
velocity5 = velocity5(index:length(velocity5));

time5 = time5(1:testend);
velocity5 = velocity5(1:testend);

plot(time5,velocity5);

% ************************************************************ Test 6

time10 = PS18mm10(:, 1);
velocity10 = PS18mm10(:, 2);

velocity10 =smooth(velocity10);

index = find(velocity10 == max(velocity10));

index = index(1) - 1900;

offset = mean(velocity10(index:index+824));
velocity10 = velocity10 - offset;

index = find(velocity10 == max(velocity10));
time10 = time10(index:length(time10));
time10 = time10 - time10(1);
velocity10 = velocity10(index:length(velocity10));

time10 = time10(1:testend);
velocity10 = velocity10(1:testend);

plot(time10,velocity10);

%************************************************************ Test 7

time6 = PS18mm6(:, 1);
velocity6 = PS18mm6(:, 2);
```

```
velocity6 =smooth(velocity6);

index = find(velocity6 == max(velocity6));

index = index(1) - 1900;

offset = mean(velocity6(index:index+824));
velocity6 = velocity6 - offset;

index = find(velocity6 == max(velocity6));
time6 = time6(index:length(time6));
time6 = time6 - time6(1);
velocity6 = velocity6(index:length(velocity6));

time6 = time6(1:testend);
velocity6 = velocity6(1:testend);

plot(time6,velocity6);
% ************************************************************* Test 8

time7 = PS18mm7(:, 1);
velocity7 = PS18mm7(:, 2);

velocity7 =smooth(velocity7);

index = find(velocity7 == max(velocity7));

index = index(1) - 1900;

offset = mean(velocity7(index:index+824));
velocity7 = velocity7 - offset;

index = find(velocity7 == max(velocity7));
time7 = time7(index:length(time7));
time7 = time7 - time7(1);
velocity7 = velocity7(index:length(velocity7));

time7 = time7(1:testend);
velocity7 = velocity7(1:testend);

plot(time7,velocity7);
% ************************************************************* Test 9

time8 = PS18mm8(:, 1);
velocity8 = PS18mm8(:, 2);

velocity8 =smooth(velocity8);

index = find(velocity8 == max(velocity8));

index = index(1) - 1900;

offset = mean(velocity8(index:index+824));
```

```
velocity8 = velocity8 - offset;

index = find(velocity8 == max(velocity8));
time8 = time8(index:length(time8));
time8 = time8 - time8(1);
velocity8 = velocity8(index:length(velocity8));

time8 = time8(1:testend);
velocity8 = velocity8(1:testend);

plot(time8,velocity8);
% ************************************************************ Test
10

time9 = PS18mm9(:, 1);
velocity9 = PS18mm9(:, 2);

velocity9 =smooth(velocity9);

index = find(velocity9 == max(velocity9));

index = index(1) - 1900;

offset = mean(velocity9(index:index+824));
velocity9 = velocity9 - offset;

index = find(velocity9 == max(velocity9));
time9 = time9(index:length(time9));
time9 = time9 - time9(1);
velocity9 = velocity9(index:length(velocity9));

time9 = time9(1:testend);
velocity9 = velocity9(1:testend);

plot(time9,velocity9);

% ***************** END SETUP
*****************************************


figure(plotNo)
plot(time1,velocity1,time2,velocity2,time3,velocity3,time4,velocity4,..
.
    time5,velocity5,time6,velocity6,time7,velocity7,time8,velocity8,...
    time9,velocity9,time10,velocity10)
plotNo = plotNo + 1;

Results_PS_18mm = zeros(10,2);

load('SystemParameters')

for test = 1:10

% model parameters
```

```
m = SystemParameters(1);      % kg
k = SystemParameters(2);      % N/m
c = SystemParameters(3);      % N-s/m
N = 0.8771*9.81;              % N
A = SystemParameters(4);
dista = 1e-4;
mud = 0.1;

if test == 1
    dx=velocity1(1);
    velocity = velocity1;
    time = time1;
end

if test == 2
    dx=velocity2(1);
    velocity = velocity2;
    time = time2;
end

if test == 3
    dx=velocity3(1);
    velocity = velocity3;
    time = time3;
end

if test == 4
    dx=velocity(1);
    velocity = velocity4;
    time = time4;
end

if test == 5
    dx=velocity(1);
    velocity = velocity5;
    time = time5;
end

if test == 6
    dx=velocity(1);
    velocity = velocity6;
    time = time6;
end

if test == 7
    dx=velocity(1);
    velocity = velocity7;
    time = time7;
end

if test == 8
    dx=velocity(1);
    velocity = velocity8;
    time = time8;
end
```

```
if test == 9
    dx=velocity(1);
    velocity = velocity9;
    time = time9;
end

if test == 10
    dx=velocity(1);
    velocity = velocity10;
    time = time10;
end

x0 = [mud dista];

LB = [0  -2e-3 ];
UB = [1   2e-3 ];

 options = optimset('Display', 'iter', 'MaxIter', 2000, 'MaxFunEvals',
5000);
 [x, resnorm, residual, exitflag, output] =
lsqnonlin(@fit_func_friction_1mud,...
     x0, LB, UB, options,time,velocity,m,k,c,A,N,plotNo);

 FOO(test,1) = output.firstorderopt;

 plotNo = plotNo + 2;

 Results_PS_18mm(test,1) = x(1);
 Results_PS_18mm(test,2) = x(2);

end

function [PI] = fit_func_friction_1mud(x,t,velocity,m,k,c,A,N,plotNo);

dx=velocity(1);

mud = x(1);
dista = x(2);

dt=1e-3;
steps=length(t)-1;

vel_f = 0*t;

vel_f(1)=dx;
time_sim(1) = 0;

dx_previous = 1; % set to negative velocity for positive initial
displacement to start simulation
for cnt = 1:steps
    % friction
    if dx*dx_previous > 0   % velocity is not zero
        dx_previous = dx;
        if dx_previous > 0  % velocity is positive
```

```
            ddx = (-mud*N - c*dx - A*dx^2- k*dista)/m;
            dx = dx + ddx*dt;
            dista = dista + dx*dt;
            vel_f(cnt+1) = dx;

        else   % velocity is negative

            ddx = (mud*N - c*dx +A*dx^2 - k*dista)/m;
            dx = dx + ddx*dt;
            dista = dista + dx*dt;
            vel_f(cnt+1) = dx;

        end
    else % velocity is zero (passed through zero)

        dx_previous = dx;
        if dista < (mud)*N/k && dista > -(mud)*N/k
            ddx = 0;
            dx = 0;
            vel_f(cnt+1) = dx;

        else    % motion continues
            if dista > 0
                ddx = (mud*N - c*dx + A*dx^2- k*dista)/m;
                dx = dx + ddx*dt;
                dista = dista + dx*dt;
                vel_f(cnt+1) = dx;
            else

                ddx = (-mud*N - c*dx - A*dx^2- k*dista)/m;
                dx = dx + ddx*dt;
                dista = dista + dx*dt;
                vel_f(cnt+1) = dx;
            end
        end
    end

    time_sim(cnt+1) = time_sim(cnt)+dt;

end

figure(plotNo)
plot(t,velocity,time_sim,vel_f)
drawnow

vel_diff = velocity - vel_f;

% Performance index, PI, is the objective function to minimize
PI = abs(vel_diff)*1e3;
```

A5 Friction_fitting_three_parameter.m

```
clc;close all;clear all;

load('PS22mm1'); load('PS22mm2'); load('PS22mm3');
load('PS22mm4'); load('PS22mm5');
load('PS22mm6'); load('PS22mm7'); load('PS22mm8');
load('PS22mm9'); load('PS22mm10');

plotNo = 1;
testend = 1750;

% ******************** SETUP ****************************************

% ********************************************************** Test 1

time1 = PS22mm1(:, 1);
velocity1 = PS22mm1(:, 2);

velocity1 =smooth(velocity1);

index = find(velocity1 == max(velocity1));

index = index(1) - 1900;

offset = mean(velocity1(index:index+824));
velocity1 = velocity1 - offset;

index = find(velocity1 == max(velocity1));
time1 = time1(index:length(time1));
time1 = time1 - time1(1);
velocity1 = velocity1(index:length(velocity1));

time1 = time1(1:testend);
velocity1 = velocity1(1:testend);

plot(time1,velocity1);

% ********************************************************** Test 2

time2 = PS22mm2(:, 1);
velocity2 = PS22mm2(:, 2);

velocity2 =smooth(velocity2);

index = find(velocity2 == max(velocity2));

index = index(1) - 1900;

offset = mean(velocity2(index:index+824));
velocity2 = velocity2 - offset;
```

```
index = find(velocity2 == max(velocity2));
time2 = time2(index:length(time2));
time2 = time2 - time2(1);
velocity2 = velocity2(index:length(velocity2));

time2 = time2(1:testend);
velocity2 = velocity2(1:testend);

plot(time2,velocity2);
% ********************************************************** Test 3

time3 = PS22mm3(:, 1);
velocity3 = PS22mm3(:, 2);

velocity3 =smooth(velocity3);

index = find(velocity3 == max(velocity3));

index = index(1) - 1900;

offset = mean(velocity3(index:index+824));
velocity3 = velocity3 - offset;

index = find(velocity3 == max(velocity3));
time3 = time3(index:length(time3));
time3 = time3 - time3(1);
velocity3 = velocity3(index:length(velocity3));

time3 = time3(1:testend);
velocity3 = velocity3(1:testend);

plot(time3,velocity3);
% ********************************************************** Test 4

time4 = PS22mm4(:, 1);
velocity4 = PS22mm4(:, 2);

velocity4 =smooth(velocity4);

index = find(velocity4 == max(velocity4));

index = index(1) - 1900;

offset = mean(velocity4(index:index+824));
velocity4 = velocity4 - offset;

index = find(velocity4 == max(velocity4));
time4 = time4(index:length(time4));
time4 = time4 - time4(1);
velocity4 = velocity4(index:length(velocity4));

time4 = time4(1:testend);
velocity4 = velocity4(1:testend);
```

```
plot(time4,velocity4);
% ************************************************************ Test 5

time5 = PS22mm5(:, 1);
velocity5 = PS22mm5(:, 2);

velocity5 =smooth(velocity5);

index = find(velocity5 == max(velocity5));

index = index(1) - 1900;

offset = mean(velocity5(index:index+824));
velocity5 = velocity5 - offset;

index = find(velocity5 == max(velocity5));
time5 = time5(index:length(time5));
time5 = time5 - time5(1);
velocity5 = velocity5(index:length(velocity5));

time5 = time5(1:testend);
velocity5 = velocity5(1:testend);

plot(time5,velocity5);

% ************************************************************ Test 6

time6 = PS22mm6(:, 1);
velocity6 = PS22mm6(:, 2);

velocity6 =smooth(velocity6);

index = find(velocity6 == max(velocity6));

index = index(1) - 1900;

offset = mean(velocity6(index:index+824));
velocity6 = velocity6 - offset;

index = find(velocity6 == max(velocity6));
time6 = time6(index:length(time6));
time6 = time6 - time6(1);
velocity6 = velocity6(index:length(velocity6));

time6 = time6(1:testend);
velocity6 = velocity6(1:testend);

plot(time6,velocity6);

% ************************************************************ Test 7

time7 = PS22mm7(:, 1);
velocity7 = PS22mm7(:, 2);
```

```
velocity7 =smooth(velocity7);

index = find(velocity7 == max(velocity7));

index = index(1) - 1900;

offset = mean(velocity7(index:index+824));
velocity7 = velocity7 - offset;

index = find(velocity7 == max(velocity7));
time7 = time7(index:length(time7));
time7 = time7 - time7(1);
velocity7 = velocity7(index:length(velocity7));

time7 = time7(1:testend);
velocity7 = velocity7(1:testend);

plot(time7,velocity7);
% ************************************************************ Test 8

time8 = PS22mm8(:, 1);
velocity8 = PS22mm8(:, 2);

velocity8 =smooth(velocity8);

index = find(velocity8 == max(velocity8));

index = index(1) - 1900;

offset = mean(velocity8(index:index+824));
velocity8 = velocity8 - offset;

index = find(velocity8 == max(velocity8));
time8 = time8(index:length(time8));
time8 = time8 - time8(1);
velocity8 = velocity8(index:length(velocity8));

time8 = time8(1:testend);
velocity8 = velocity8(1:testend);

plot(time8,velocity8);
% ************************************************************ Test 9
time9 = PS22mm9(:, 1);
velocity9 = PS22mm9(:, 2);

velocity9 =smooth(velocity9);

index = find(velocity9 == max(velocity9));

index = index(1) - 1900;

offset = mean(velocity9(index:index+824));
velocity9 = velocity9 - offset;
```

```
index = find(velocity9 == max(velocity9));
time9 = time9(index:length(time9));
time9 = time9 - time9(1);
velocity9 = velocity9(index:length(velocity9));

time9 = time9(1:testend);
velocity9 = velocity9(1:testend);

plot(time9,velocity9);
% ********************************************************** Test
10

time10 = PS22mm10(:, 1);
velocity10 = PS22mm10(:, 2);

velocity10 =smooth(velocity10);

index = find(velocity10 == max(velocity10));

index = index(1) - 1900;

offset = mean(velocity10(index:index+824));
velocity10 = velocity10 - offset;

index = find(velocity10 == max(velocity10));
time10 = time10(index:length(time10));
time10 = time10 - time10(1);
velocity10 = velocity10(index:length(velocity10));

time10 = time10(1:testend);
velocity10 = velocity10(1:testend);

plot(time10,velocity10);

% ***************** END SETUP
****************************************

figure(plotNo)
plot(time1,velocity1,time2,velocity2,time3,velocity3,time4,velocity4,..
.

time5,velocity5,time6,velocity6,time7,velocity7,time8,velocity8,time9,v
elocity9,...
    time10,velocity10)
plotNo = plotNo + 1;

Results_PS_22mm = zeros(10,4);

load('SystemParameters')

for test = 1:10

% model parameters
m = SystemParameters(1);      % kg
```

```
k = SystemParameters(2);      % N/m
c = SystemParameters(3);      % N-s/m
N = 0.8771*9.81;              % N
dista = 1e-4;
mud0 = 0.2;
mud1 = 0.2;
alpha = 5;

if test == 1
    dx=velocity1(1);
    velocity = velocity1;
    time = time1;
end

if test == 2
    dx=velocity2(1);
    velocity = velocity2;
    time = time2;
end

if test == 3
    dx=velocity3(1);
    velocity = velocity3;
    time = time3;
end

if test == 4
    dx=velocity4(1);
    velocity = velocity4;
    time = time4;
end

if test == 5
    dx=velocity5(1);
    velocity = velocity5;
    time = time5;
end
if test == 6
    dx=velocity6(1);
    velocity = velocity6;
    time = time6;
end

if test == 7
    dx=velocity7(1);
    velocity = velocity7;
    time = time7;
end

if test == 8
    dx=velocity8(1);
    velocity = velocity8;
    time = time8;
end
if test == 9
```

```
    dx=velocity9(1);
    velocity = velocity9;
    time = time9;
end

if test == 10
    dx=velocity10(1);
    velocity = velocity10;
    time = time10;
end

x0 = [mud0 mud1 alpha dista];

LB = [0    0     0     -2e-3 ];
UB = [1    1    1e6    2e-3 ];

 options = optimset('Display', 'iter', 'MaxIter', 2000, 'MaxFunEvals',
5000);
 [x, resnorm, residual, exitflag, output] =
lsqnonlin(@fit_func_friction_exp,...
     x0, LB, UB, options,time,velocity,m,k,N,plotNo);

 plotNo = plotNo + 2;

 Results_PS_22mm(test,1) = x(1);
 Results_PS_22mm(test,2) = x(2);
 Results_PS_22mm(test,3) = x(3);
 Results_PS_22mm(test,4) = x(4);

end

function [PI] = fit_func_friction_exp(x,t,velocity,m,k,N,plotNo);

dx=velocity(1);

mud0 = x(1);
mud1 = x(2);
alpha = x(3);
dista = x(4);

dt=1e-3;
steps=length(t)-1;

vel_f = 0*t;

vel_f(1)=dx;
time_sim(1) = 0;

FN = N;

dx_previous = 1; % set to negative velocity for positive initial
displacement to start simulation
for cnt = 1:steps
    % friction
```

```
    if dx*dx_previous > 0   % velocity is not zero
        dx_previous = dx;
        if dx_previous > 0  % velocity is positive
            f = mud0 + mud1*exp(-alpha*abs(dx));
            ddx = ( - (0.437*abs(dx) + 0.2005)*dx - FN*f - k*dista)/m;
            dx = dx + ddx*dt;
            dista = dista + dx*dt;
            vel_f(cnt+1) = dx;

        else   % velocity is negative
            f = mud0 + mud1*exp(-alpha*abs(dx));
            ddx = ( - (0.437*abs(dx) + 0.2005)*dx + FN*f - k*dista)/m;
            dx = dx + ddx*dt;
            dista = dista + dx*dt;
            vel_f(cnt+1) = dx;

        end
    else % velocity is zero (passed through zero)

        dx_previous = dx;
        if dista < (mud0 + mud1)*N/k && dista > -(mud0+mud1)*N/k
            ddx = 0;
            dx = 0;
            vel_f(cnt+1) = dx;

        else    % motion continues
            if dista > 0
                f = mud0 + mud1*exp(-alpha*abs(dx));
                ddx = ( - (0.437*abs(dx) + 0.2005)*dx + FN*f -
k*dista)/m;
                dx = dx + ddx*dt;
                dista = dista + dx*dt;
                vel_f(cnt+1) = dx;
            else
                f = mud0 + mud1*exp(-alpha*abs(dx));
                ddx = ( - (0.437*abs(dx) + 0.2005)*dx - FN*f -
k*dista)/m;
                dx = dx + ddx*dt;
                dista = dista + dx*dt;
                vel_f(cnt+1) = dx;
            end
        end
    end

    time_sim(cnt+1) = time_sim(cnt)+dt;

end

figure(plotNo)
plot(t,velocity,time_sim,vel_f)
drawnow

dx_friction_plot = 0:0.001:10;
for i = 1:length(dx_friction_plot)
    f_plot(i) = mud0 + mud1*exp(-alpha*abs(dx_friction_plot(i)));
```

```
end
figure(plotNo + 1)
plot(dx_friction_plot,f_plot)

vel_diff = velocity - vel_f;

% Performance index, PI, is the objective function to minimize
PI = abs(vel_diff)*1e3;
```

A6 Tanh_Modified_fitting_model.m

```
clc;close all;clear all;

load('PS22mm1'); load('PS22mm2'); load('PS22mm3');
load('PS22mm4'); load('PS22mm5'); load('PS22mm6');
load('PS22mm7'); load('PS22mm8'); load('PS22mm9');
load('PS22mm10');

testend = 1840;
plotNo = 1;
% ******************** SETUP ****************************************

% ************************************************************ Test 1

time1 = PS22mm1(:, 1);
velocity1 = PS22mm1(:, 2);

velocity1 =smooth(velocity1);

index = find(velocity1 == max(velocity1));

index = index(1) - 1900;

offset = mean(velocity1(index:index+824));
velocity1 = velocity1 - offset;

index = find(velocity1 == max(velocity1));
time1 = time1(index:length(time1));
time1 = time1 - time1(1);
velocity1 = velocity1(index:length(velocity1));

plot(time1,velocity1)

time1 = time1(1:testend);
velocity1 = velocity1(1:testend);

% ************************************************************ Test 2

time2 = PS22mm2(:, 1);
velocity2 = PS22mm2(:, 2);

velocity2 =smooth(velocity2);

index = find(velocity2 == max(velocity2));

index = index(1) - 1900;

offset = mean(velocity2(index:index+824));
velocity2 = velocity2 - offset;

index = find(velocity2 == max(velocity2));
time2 = time2(index:length(time2));
```

```
time2 = time2 - time2(1);
velocity2 = velocity2(index:length(velocity2));

time2 = time2(1:testend);
velocity2 = velocity2(1:testend);

plot(time2,velocity2);
% ************************************************************* Test 3

time3 = PS22mm3(:, 1);
velocity3 = PS22mm3(:, 2);

velocity3 =smooth(velocity3);

index = find(velocity3 == max(velocity3));

index = index(1) - 1900;

offset = mean(velocity3(index:index+824));
velocity3 = velocity3 - offset;

index = find(velocity3 == max(velocity3));
time3 = time3(index:length(time3));
time3 = time3 - time3(1);
velocity3 = velocity3(index:length(velocity3));

time3 = time3(1:testend);
velocity3 = velocity3(1:testend);

plot(time3,velocity3);
% ************************************************************* Test 4

time4 = PS22mm4(:, 1);
velocity4 = PS22mm4(:, 2);

velocity4 =smooth(velocity4);

index = find(velocity4 == max(velocity4));

index = index(1) - 1900;

offset = mean(velocity4(index:index+824));
velocity4 = velocity4 - offset;

index = find(velocity4 == max(velocity4));
time4 = time4(index:length(time4));
time4 = time4 - time4(1);
velocity4 = velocity4(index:length(velocity4));

time4 = time4(1:testend);
velocity4 = velocity4(1:testend);

plot(time4,velocity4);
```

```
% ************************************************************ Test 5

time5 = PS22mm5(:, 1);
velocity5 = PS22mm5(:, 2);

velocity5 =smooth(velocity5);

index = find(velocity5 == max(velocity5));

index = index(1) - 1900;

offset = mean(velocity5(index:index+824));
velocity5 = velocity5 - offset;

index = find(velocity5 == max(velocity5));
time5 = time5(index:length(time5));
time5 = time5 - time5(1);
velocity5 = velocity5(index:length(velocity5));

time5 = time5(1:testend);
velocity5 = velocity5(1:testend);

plot(time5,velocity5);

% ************************************************************ Test 6

time10 = PS22mm10(:, 1);
velocity10 = PS22mm10(:, 2);

velocity10 =smooth(velocity10);

index = find(velocity10 == max(velocity10));

index = index(1) - 1900;

offset = mean(velocity10(index:index+824));
velocity10 = velocity10 - offset;

index = find(velocity10 == max(velocity10));
time10 = time10(index:length(time10));
time10 = time10 - time10(1);
velocity10 = velocity10(index:length(velocity10));

time10 = time10(1:testend);
velocity10 = velocity10(1:testend);

plot(time10,velocity10);

%************************************************************ Test 7

time6 = PS22mm6(:, 1);
velocity6 = PS22mm6(:, 2);
```

```
velocity6 =smooth(velocity6);

index = find(velocity6 == max(velocity6));

index = index(1) - 1900;

offset = mean(velocity6(index:index+824));
velocity6 = velocity6 - offset;

index = find(velocity6 == max(velocity6));
time6 = time6(index:length(time6));
time6 = time6 - time6(1);
velocity6 = velocity6(index:length(velocity6));

time6 = time6(1:testend);
velocity6 = velocity6(1:testend);

plot(time6,velocity6);
% ************************************************************* Test 8

time7 = PS22mm7(:, 1);
velocity7 = PS22mm7(:, 2);

velocity7 =smooth(velocity7);

index = find(velocity7 == max(velocity7));

index = index(1) - 1900;

offset = mean(velocity7(index:index+824));
velocity7 = velocity7 - offset;

index = find(velocity7 == max(velocity7));
time7 = time7(index:length(time7));
time7 = time7 - time7(1);
velocity7 = velocity7(index:length(velocity7));

time7 = time7(1:testend);
velocity7 = velocity7(1:testend);

plot(time7,velocity7);
% ************************************************************* Test 9

time8 = PS22mm8(:, 1);
velocity8 = PS22mm8(:, 2);

velocity8 =smooth(velocity8);

index = find(velocity8 == max(velocity8));

index = index(1) - 1900;

offset = mean(velocity8(index:index+824));
```

```
velocity8 = velocity8 - offset;

index = find(velocity8 == max(velocity8));
time8 = time8(index:length(time8));
time8 = time8 - time8(1);
velocity8 = velocity8(index:length(velocity8));

time8 = time8(1:testend);
velocity8 = velocity8(1:testend);

plot(time8,velocity8);
% ************************************************************ Test
10

time9 = PS22mm9(:, 1);
velocity9 = PS22mm9(:, 2);

velocity9 =smooth(velocity9);

index = find(velocity9 == max(velocity9));

index = index(1) - 1900;

offset = mean(velocity9(index:index+824));
velocity9 = velocity9 - offset;

index = find(velocity9 == max(velocity9));
time9 = time9(index:length(time9));
time9 = time9 - time9(1);
velocity9 = velocity9(index:length(velocity9));

time9 = time9(1:testend);
velocity9 = velocity9(1:testend);

plot(time9,velocity9);

% ***************** END SETUP
*****************************************


figure(plotNo)
plot(time1,velocity1,time2,velocity2,time3,velocity3,time4,velocity4,..
.
    time5,velocity5,time6,velocity6,time7,velocity7,time8,velocity8,...
    time9,velocity9,time10,velocity10)
plotNo = plotNo + 1;


Results_PS_22mm = zeros(10,2);

load('SystemParameters')

for test = 1:10
```

```
% model parameters
m = SystemParameters(1);      % kg
k = SystemParameters(2);     % N/m
c = SystemParameters(3);       % N-s/m
N = 0.8771*9.81;                          % MEASURED MASS into Force
A = SystemParameters(4);
dista = 1e-4;
gamma1 = 0.25;
gamma2 = 100;
gamma3 = 10;
gamma4 = 0.1;
gamma5 = 100;

if test == 1
    dx=velocity1(1);
    velocity = velocity1;
    time = time1;
end

if test == 2
    dx=velocity2(1);
    velocity = velocity2;
    time = time2;
end

if test == 3
    dx=velocity3(1);
    velocity = velocity3;
    time = time3;
end

if test == 4
    dx=velocity(1);
    velocity = velocity4;
    time = time4;
end

if test == 5
    dx=velocity(1);
    velocity = velocity5;
    time = time5;
end

if test == 6
    dx=velocity(1);
    velocity = velocity6;
    time = time6;
end

if test == 7
    dx=velocity(1);
    velocity = velocity7;
    time = time7;
end
```

```
if test == 8
    dx=velocity(1);
    velocity = velocity8;
    time = time8;
end

if test == 9
    dx=velocity(1);
    velocity = velocity9;
    time = time9;
end

if test == 10
    dx=velocity(1);
    velocity = velocity10;
    time = time10;
end

x0 = [gamma1 gamma2 gamma3 gamma4 gamma5 dista];

LB = [-1e15 -1e15 -1e15 -1e15 -1e15 -2e-3];
UB = [1e15 1e15 1e15 1e15 1e15 2e-3];

 options = optimset('Display', 'iter', 'MaxIter', 2000, 'MaxFunEvals',
5000);
 [x, resnorm, residual, exitflag, output] = lsqnonlin(@fit_func_tanh,
x0, LB, UB, options,time,velocity,m,k,c,A,N,plotNo);

 plotNo = plotNo + 2;

 Results_PS_22mm(test,1) = x(1);
 Results_PS_22mm(test,2) = x(2);
 Results_PS_22mm(test,3) = x(3);
 Results_PS_22mm(test,4) = x(4);
 Results_PS_22mm(test,5) = x(5);
 Results_PS_22mm(test,6) = x(6);

end

function [PI] = fit_func_tanh(x,time,velocity,m,k,c,A,N,plotNo)

dx = velocity(1);

dx_max = max(velocity)*10;      % set up bounds for interpolation table

dx_fric = 0:1e-3:dx_max;

% Initial conditions
y = 0;
dy = velocity(1);

gamma1 = x(1);
gamma2 = x(2);
gamma3 = x(3);
```

```
gamma4 = x(4);
gamma5 = x(5);
dista = x(6);

Fric_Coeff=gamma1*(tanh(gamma2*dx_fric)-
tanh(gamma3*dx_fric))+gamma4*tanh(gamma5*dx_fric);

cnt=1;
for i = 0:1e-3:dx_fric(length(dx_fric));

        ddF(cnt)=gamma1*(2*gamma2^2*tanh(i*gamma2)*(tanh(i*gamma2)^2 -
1) ...
            - 2*gamma3^2*tanh(i*gamma3)*(tanh(i*gamma3)^2 - 1)) ...
            + 2*gamma4*gamma5^2*tanh(i*gamma5)*(tanh(i*gamma5)^2 - 1);

    cnt=cnt+1;
end

cnt=1;
cnt2=2;

if max(Fric_Coeff) == Fric_Coeff(length(Fric_Coeff))
else
while cnt == 1

    next = ddF(cnt2+1);
    previous = ddF(cnt2);

    if next*previous > 0
        cnt2 = cnt2+1;
    else
        cnt = cnt + 1;
    end

end

index = find(ddF == next);

Fric_Coeff = Fric_Coeff(index:length(Fric_Coeff));
dx_fric = dx_fric(index:length(dx_fric));
dx_fric = dx_fric - dx_fric(1);
end

figure(plotNo)
plot(dx_fric,Fric_Coeff)
drawnow
plotNo = plotNo + 1;

dt=1e-3;
steps=length(time) - 1;
t=0:dt:length(time)*dt-dt;

vel_f = 0*t;
```

```
vel_f(1)=dx;

dx_previous = 1;
for cnt = 1:steps
    % friction
    if dx*dx_previous > 0   % velocity is not zero
        dx_previous = dx;
        if dx_previous > 0  % velocity is positive
            dx_abs = abs(dx);
            F = gamma1*(tanh(gamma2*dx_abs)-
tanh(gamma3*dx_abs))+gamma4*tanh(gamma5*dx_abs);
            ddx = (-F*N -A*dx^2- c*dx - k*dista)/m;
            dx = dx + ddx*dt;
            dista = dista + dx*dt;
            vel_f(cnt+1) = dx;
        else   % velocity is negative
            dx_abs = abs(dx);
            F = gamma1*(tanh(gamma2*dx_abs)-
tanh(gamma3*dx_abs))+gamma4*tanh(gamma5*dx_abs);
            ddx = (F*N +A*dx^2- c*dx - k*dista)/m;
            dx = dx + ddx*dt;
            dista = dista + dx*dt;
            vel_f(cnt+1) = dx;
        end

    else % velocity is zero (passed through zero)

        dx_previous = dx;

                dx_abs = abs(dx);
                F = gamma1*(tanh(gamma2*dx_abs)-
tanh(gamma3*dx_abs))+gamma4*tanh(gamma5*dx_abs);
        if abs(dista)*k < F*N
            ddx = 0;
            dx = 0;
            vel_f(cnt+1) = dx;
        else    % motion continues
            if dista > 0
                dx_abs = abs(dx);
                F = gamma1*(tanh(gamma2*dx_abs)-
tanh(gamma3*dx_abs))+gamma4*tanh(gamma5*dx_abs);
                ddx = (F*N +A*dx^2 - c*dx - k*dista)/m;
                dx = dx + ddx*dt;
                dista = dista + dx*dt;
                vel_f(cnt+1) = dx;
            else
                dx_abs = abs(dx);
                F = gamma1*(tanh(gamma2*dx_abs)-
tanh(gamma3*dx_abs))+gamma4*tanh(gamma5*dx_abs);
                ddx = (-F*N -A*dx^2 - c*dx - k*dista)/m;
                dx = dx + ddx*dt;
                dista = dista + dx*dt;
                vel_f(cnt+1) = dx;
            end
        end
```

```
    end


end
figure(plotNo)
plot(time,velocity, time,vel_f)

vel_f=vel_f(:);

% Calculate difference between model and data
vel_diff = velocity - vel_f;

% Performance index, PI, is the objective function to minimize
PI = abs(vel_diff)*1e3;
```