Fall 1-31-2012

Dynamic friction: measurement and results

Yanal Jij
New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

DYNAMIC FRICTION: MEASUREMENT AND RESULTS

By
Yanal Jij

Friction has been known as a phenomenon since Leonardo da Vinci era. From that time and through the last four centuries, scientists tried to understand this phenomenon in an effort to overcome its undesired side effects which include energy loss, parts wearing, and errors in control systems. Much has been achieved in studying friction and many experiments have been conducted in this field. These experiments resulted in a mounting evidence that friction is a dynamic phenomenon.

This thesis presents an apparatus that measures dynamic friction in linear motion and the results obtained using this apparatus. The apparatus consists of a reciprocating platform and a Test Mass on top of it. The platform moves in a sinusoidal pattern with user changeable frequencies and amplitudes. The ultimate objective of the apparatus is to measure the acceleration and the relative velocity of the Test Mass to produce a friction coefficient vs. velocity curve that is used in studying friction. The force needed to move the base is provided by two solenoids which are controlled through a C++ program that allows real-time control. A graphical user interface (GUI) has been developed to allow the user to change the frequency and amplitude of the motion. The results, presented in this thesis, are from experiments with different frequencies, amplitudes, materials and loads.

DYNAMIC FRICTION: MEASUREMENT AND RESULTS

by
Yanal Jij

A thesis
Submitted to the Faculty of
New Jersey Institute of Technology
In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

Department of Electrical Engineering and Computer Engineering

January 2012

APPROVAL PAGE

DYNAMIC FRICTION: MEASUREMENT AND RESULTS

Yanal Jij

Dr. Bernard Friedland, Thesis Advisor — Date
Distinguished Professor of Electrical and Computer Engineering

---

Dr. Avraham Harnoy — Date
Professor of Mechanical and Industrial Engineering

Dr. David A. Haessig — Date
Adjunct Professor of Electrical and Computer Engineering

BIOGRAPHICAL SKETCH

Author Yanal Jij

Degree Master of Science

Date January 2012

Undergraduate and Graduate Education:

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2011

- Professional Master's degree in Safety and Environmental Management of Industrial Systems,
  Damascus University, Damascus, Syrian Arab Republic, 2008

- Bachelor of Science in Electrical Engineering,
  Damascus University, Damascus, Syrian Arab Republic, 2004

Major: Electrical Engineering

DEDICATION

To the one that taught me that seeking knowledge and education is an important goal in a man's life, my hero, my moral compass and my role model. To my father, ***Mohamed Ali****, thank you for everything that you did to me, Rest in peace Dad.*

To the one that always has me in her heart and mind, and will be the most happy to see me finish this master, your passion and love will always guide and protect me. To my mother, ***Waffa****. No words will be enough to thank you Mom.*

To the most annoying three persons in my life though I love them to death. To ***Lina, Shamel and Hussam****. Thank you for being there for me.*

To the man that supported me through this journey and never gave me NO as an answer. Thank you ***Eyad.***

To the people that helped me achieve this master and were an unending source of help, support, guidance and advice. My cousins ***Ghiath, Ahmad and Aiman****. Thank you for your help and support.*

To my friends, especially ***Mexhit****, thank you for your help and support.*

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my advisor, Dr. Bernard Friedland, for his guidance, advice, suggestions, and most importantly his patience in all aspects that related to my research work including the writing of this thesis.

Special thanks to Professor Avraham Harnoy for giving me insights and for serving as a member of my thesis committee. Thanks also to Professor David Haessig for his insight and for serving as a member in my thesis committee.

I would also like to express my appreciation to Dr. Marino Xanthos and Mrs. Clarisa González-Lenahan for their help in reviewing and editing this thesis.

Finally, I would like to thank all my friends and family for their moral support and understanding during the past several years, especially the times when many were neglected while I concentrated on my thesis.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

CHAPTER 1

INTRODUCTION

Friction is the force that exists when a moving body contacts any kind of surfaces. Depending on the situation, this force may be useful or harmful. It is the reason that we can walk, and it is also the reason why we need to replace our brake pads every now and then. In control systems specifically, friction has undesired effects that produce error in tracking and sometimes destabilize the whole system.

Leonardo da Vinci (1452-1519), in his Notebook, describes friction as the reason of the heaven music. He mentioned that friction is related to velocity [1], and notes that friction is not related to the area of contact, which has been later proven slightly wrong by F.P. Bowden; he also noted that doubling the load will result in doubling the friction force; and that friction changes with changing the material of contacts. His work remained unpublished and stayed in his journals [1].

Guillaume Amontons rediscovered what Leonardo already found about friction and he also formulated a new set of theories in which he attributed friction to the work needed to lift one surface over the roughness of the other [2].

The earliest mathematical description of friction was developed by Charles-Augustin de Coulomb as a development of Amontons work [2].

Leonhard Euler, also studied friction and he assumed that friction results from gravitational forces and he found the relation between the angle of inclination and the friction coefficient [2].

In the 1950's , Philip Bowden and David Tabor studied friction intensively and gave a physical explanation for the laws of friction, and they introduced the term "true

area of contact" which is produced by the asperities in the two surfaces in contact [1].

Nowadays, friction occupies a vast area of the science of tribology and many studies have been carried out to understand and produce accurate models of friction.

In a paper published in the Control System Magazine [6], Friedland et al. mentioned an idea for an apparatus that consists of a moving base and a Test Mass free to slide above it. He suggested that the moving base will overcome the problems that results from keeping the base fixed in dynamic friction measurement near zero velocities.

Starting with Alexander Rokhvarg's apparatus, which he built as a single-axis agnostic system in his thesis 1996 [8], an apparatus has been built to measure dynamic friction.

The apparatus consist of a reciprocating platform and a Test Mass a top of it, the platform moves in a sinusoidal pattern with user changeable frequencies and amplitudes. The ultimate objective of the apparatus is to measure the acceleration and the relative velocity of the Test Mass and produce the friction coefficient vs. velocity curve that is used in studying friction.

The first chapter is a background survey of some of the experiments and apparatuses built to measure friction and the results of some of these experiments.

The second chapter is a physical description of the hardware and software components of the apparatus, and how they operate to produce the final result of an experiment. The third chapter has some of the results obtained using the apparatus with comments about how they relate to previous work of other scientists.

The fourth chapter is the conclusion of this thesis with suggestion on improvements that can enhance the apparatus to fit a wider range of applications.

Appendences include the source code, of the programs used in this thesis, and the

schematic of the apparatus circuitry. Appendix D is a short guide on how to operate the apparatus.

1.1 Background

Leonardo da Vinci has the credit of being the first who made quantitative studies of the problem of friction [2]. Leonardo's experimental setups for friction measurements were rather simple. For example, one of his experiments involved the measuring of the angle, of an inclined plane, that will cause the body placed on the plane to start moving, Figure 1.1(a). Another setup, Figure 1.1 (b), was to measure the weight needed to make a block, placed horizontally on a table, moves [2]. With his methods, he was only able to measure static friction and there is no report that he knew the difference between static and dynamic friction [2]. Leonardo found the following two laws of friction, in which we essentially recover the first and second laws of friction:

- The friction caused by the same weight will be of equal resistance at the beginning of its movement although the contact may be of different breadths and lengths.
- Friction produces double the amount of effort if the weight be doubled.

The first statement means that Da Vinci believed that the area of the surfaces in contact will not affect friction. Leonardo defined a friction coefficient as the ratio of the friction divided by the mass of the slider [2]. Some sources reports that Da Vinci believed in a universal friction coefficient with a value of 0.25.

Two centuries after Leonardo's discoveries, the French physicist Guillaume Amontons (1663-1705) considered the problem of friction again [2]. In his experiments, Figure 1.2, he used springs to measure lateral forces and was able to measure both static and dynamic friction [2], but it is not clear if Amontons was aware of the difference

between the two friction phenomena.

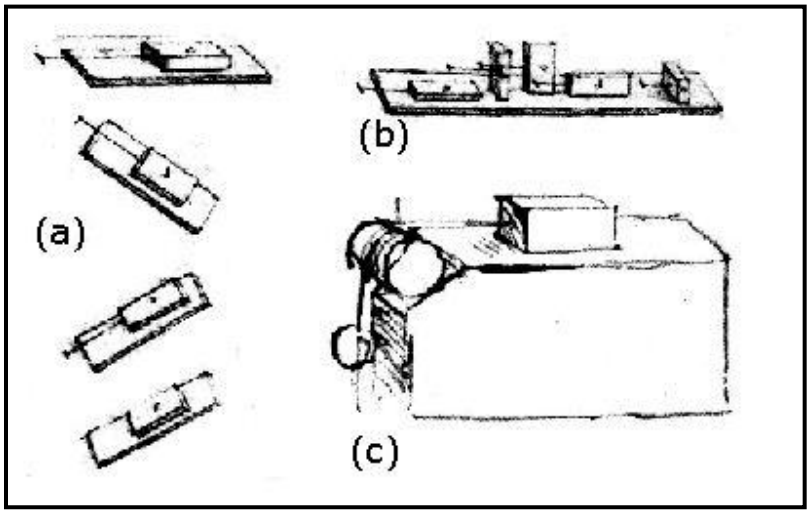


Figure 1.1 Leonardo da Vinci experimental setups which he used in measuring the angle of inclination or what later became the friction coefficient.

Source: [2]

Amontons postulated the following friction laws from his experiments [2]:

- The resistance caused by rubbing, only increases or diminishes in proportion to greater or lesser pressure (load) and not according to the greater or lesser area of the surfaces.
- The resistance caused by rubbing is more or less the same for iron, lead, copper and wood in any combination if the surfaces are coated with pork fat
- The resistance is approximately equal to one-third of the load.

Like Da Vinci, Amontons also believed in a universal friction coefficient and, experimentally, he found a universal friction coefficient of 0.33 independent of the material[2]. Later scientists proved this idea to be wrong.

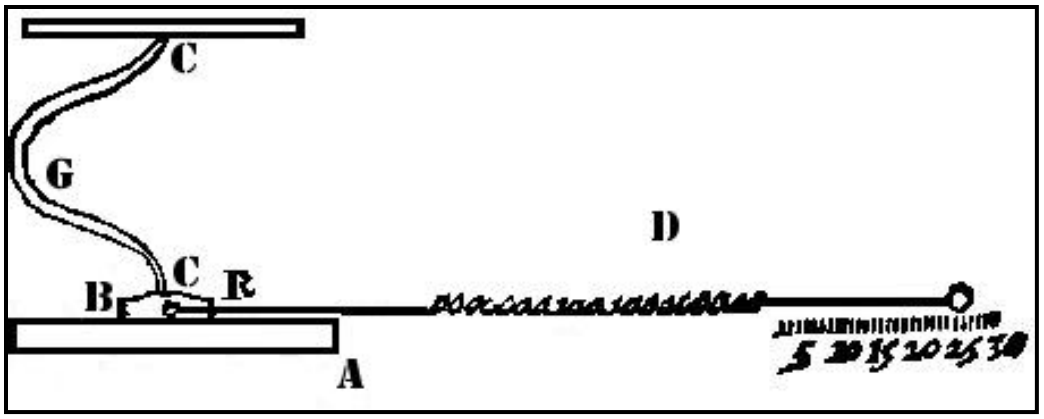


Figure 1.2 Amontons experiment setup in which he used springs to measure lateral forces and was able to measure both static and dynamic friction.

Source [2]

Leonhard Euler (1707-1783), following Amontons, considered the theory of the sliding motion of a block on an inclined plane. He adopted the concept of rigid interlocking asperities as the cause of frictional resistance. He assumed that friction resulted from gravitational forces trying to minimize the potential energy of the block. According to [2], he found the relation between the inclination angle α and friction coefficient μ that indicates:

$$\mu = \tan \alpha \tag{1.1}$$

Assuming a velocity-independent friction coefficient, he found that, for a critical angle, the acceleration of the block should be exceedingly small, since gravity is nearly compensated by dynamic friction [2]. This result was against the experimental facts, where sliding started relatively fast. He concluded that one has to distinguish between static and dynamic friction and that static friction is always larger than dynamic friction. With these assumptions he was able to describe the motion of a block on an inclined plane. Euler was the first who distinguished between static and dynamic friction [2].

Charles August Coulomb (1736-1806) learned about Amontons' work and became so interested in this quotidian phenomenon that he started making measurements himself. He was not only interested in friction coefficients, but also in the time dependence of the static friction force on the time of rest. He found an increase of the friction force with the time of rest and tried to find a mathematical description [2].

He published his major results in a paper "Essai sur la theorie du frottement" namely the friction theory [2], and often referred to as Coulomb's laws of friction:

- For wood sliding on wood under dry conditions, the friction rises initially but soon reaches a maximum. Thereafter, the force of friction is essentially proportional to load.
- For wood sliding on wood the force of friction is essentially proportional to load at any speed, but dynamic friction is much lower than the static friction to long periods of repose.
- For metals sliding on metals without lubricant the force of friction is essentially proportional to load and there is no difference between static and dynamic friction.
- For metals on wood under dry conditions the static friction rises very slowly with time of repose and might take four, five or even more days to reach its limit. With metal-on-metal the limit is reached almost immediately and with wood-on-wood it takes only one or two minutes. For wood-on-wood or metal-on-metal under dry conditions speed has very little effect on dynamic friction, but in the case of wood-on-metal the dynamic friction increases with speed.

The second part of the fourth law, which describes the velocity independence of dynamic friction, is nowadays well known as Coulomb's Law and describes a type of friction named *Coulomb friction*. In Coulomb friction the friction force is independent of the velocity and has a constant value equals to μg where μ is the friction coefficient and g is the garavitational constant.

Following Coulomb's work, Frank P. Bowden and David Tabor (1939) conducted further experiments, and presented a physical explanation for the laws of friction. They

determined that the true area of contact is a very small percentage of the apparent contact area, thus, they dismissed Da Vinci's theory about the friction independency of the area in contact.

Their Theory indicated that the *true contact area* is formed by the asperities. As the normal force increases, more asperities come into contact and the average area of each asperity contact grows. The frictional force was shown to be dependent on the true contact area. Bowden and Tabor argued that within these asperities all of the dynamics of friction take place [3].

Bowden and Leben observed the welding in the photomicrographs, using the thermocouple effect between dissimilar metals.They found wide temperature fluctuations that are correlated with the stick-slip cycle, and they posited local melting of one rubbing metal as a mechanism for decreased friction during sliding. They also found that similar *stick-slip* occurs in many lubricated systems, even if there is no welding; and that no stick slip occurs when long chain fatty acids are used as a lubricant [1].

In 1951, Ernest Rabinowicz built a simple apparatus to determine the transition between static and dynamic friction [4]. The apparatus consisted of an inclined plane with metal surface and a 1kg load stationed on it, Figure 1.3 illustrates a sketch of the apparatus. He rolled a ball from a known distance to push the load, and then he measured the distance the load travelled. The relation, he found, between impulse, sliding distance, plate slope, and lubricants yielded information about the transition from dynamic to static friction [1].

In 1977, Phil R. Dahl presented an apparatus in his report [5] for studying friction in ball bearings. The apparatus consists of a bearing test fixture connected to a Torque meter from one side and to a drive servo from the other.

The bearing test fixture has two angular contact bearings back to back. A spacer between the two bearing outer races gives a load path through the races and assures separation of the inner races. The bearing inner races have a sliding fit to the fixture shaft, and are axially loaded by means of the inner race loading flange which has the pre-load applied via the springs between the sliding flanges. The preload is adjustable with the adjustment nut or the left end-bolt on the shaft that retains the left bearing inner race. The outer races were lightly fastened with small set screws to a mounting adapter that mounts on the head of the torque meter. A coupling on the inner race drive shaft was used to connect the bearing test fixture to the drive servo [5]. Figure 1.4 shows the apparatus used by Dahl.

Dahl work resulted in the discovery of an effect named after him "*Dahl Effect*" which states that the junction between two surfaces acts like a spring in static friction, which means that the friction force is a linear function of displacement, up to a critical (force) where the *break-away* occurs [1].

In 1991, Brian Armstrong-Helouvry used a PUMA (Particular Manipulator) robot arm to measure and study friction characteristics, and to get experimental data to help him design a model to simulate friction in servo-mechanism. He reported his work and results in his book [1].

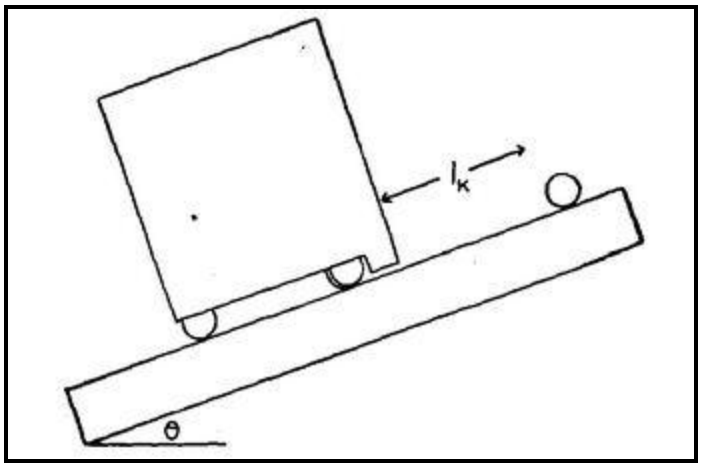


Figure 1.3 A sketch of the apparatus that Rabinowicz used in studying friction. He found that for a critical distance I_k the block will stay at rest if the rolling distance was smaller than I_k, and will be set into motion right down the plane if the rolling distance was greater than I_k.

Source: [4]

In 1994, Bernard Friedland and Avraham Harnoy presented an apparatus to measure friction in journal bearings in their paper [7] for the American Control Conference, Figure 1.5. The apparatus contains four equally loaded bearings, two of these bearings are pre-stressed towards the other two by means of thin elastic ring. The total friction torque of the bearings was measured by load cells which measure the friction torque while preventing the rotation of the external bearings hosing. The time-variable motion of the shaft was generated by a computer controlled D.C. servo motor.

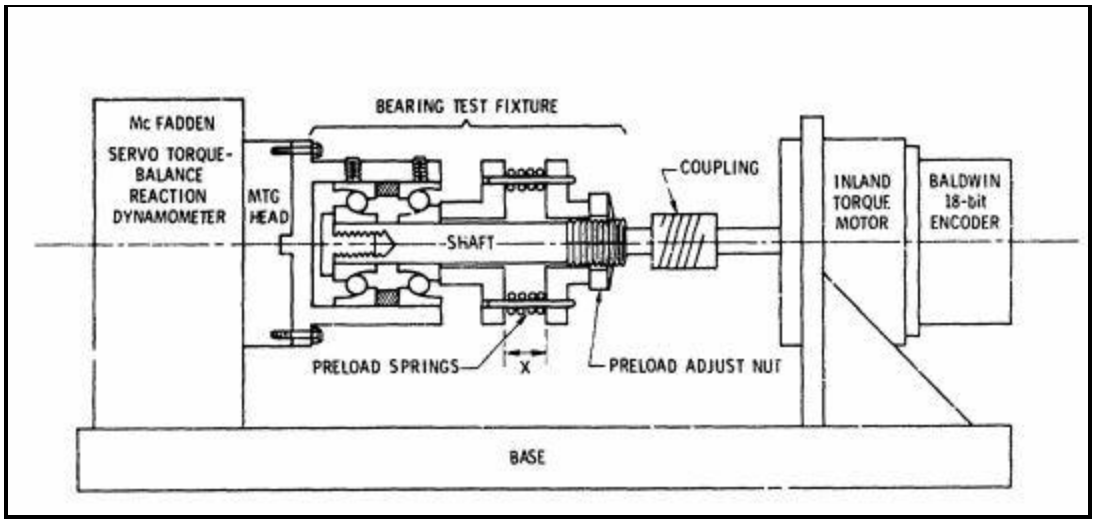


Figure 1.4 Dahl's apparatus for measuring ball bearings friction.

Source: [5].

Friedland and Harnoy used the apparatus to determine the friction between the shaft and four bearings in the presence of sinusoidal excitation. Figure 1.6 shows a sample from the results obtained using this apparatus.

The same paper [6] presented another apparatus. This apparatus, shown in the Figure 1.7, was designed to measure friction force in linear motion at very low velocity. It contains a linear motion sliding table, driven by a servo meter, and a ball screw drive to reduce the velocity. The line contact is created between a short, finally ground, cylindrical shaft and the flat friction surface. No results have been reported on this apparatus.

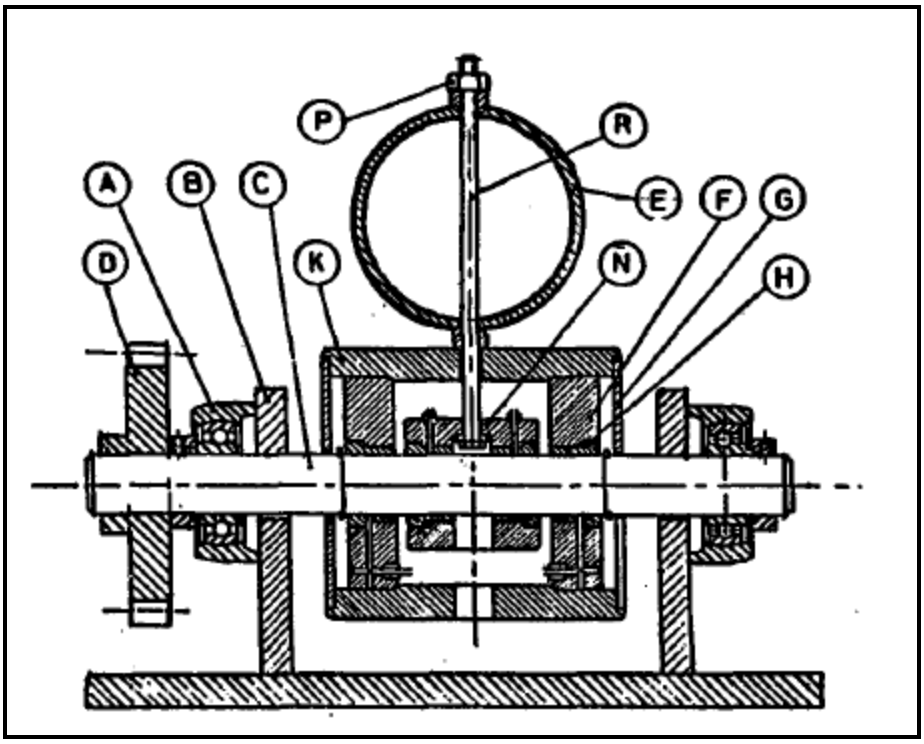


Figure 1.5 cross sectional view shows the basic components of Harnoy's apparatus for measuring friction effect in journal bearing.
Source: [5]

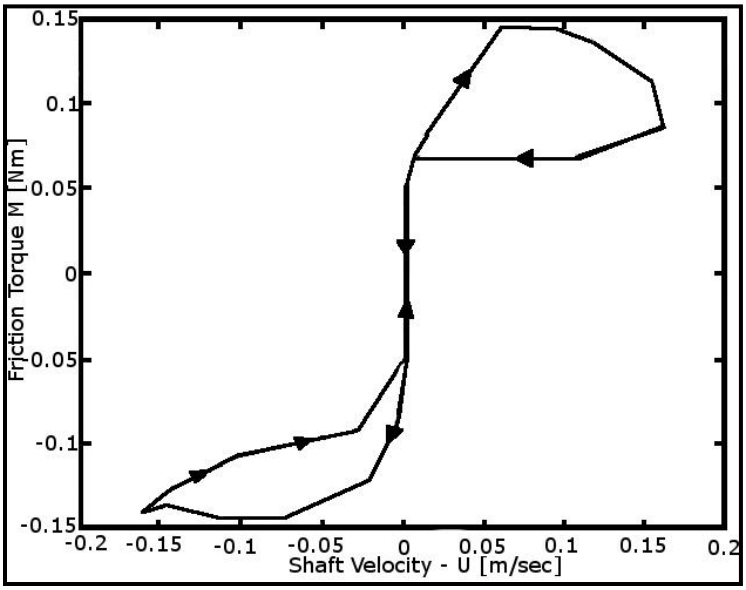


Figure 1.6 A sample of the results obtained by Harnoy's apparatus, it shows the friction hysteresis in journal bearing.

Source: [6]

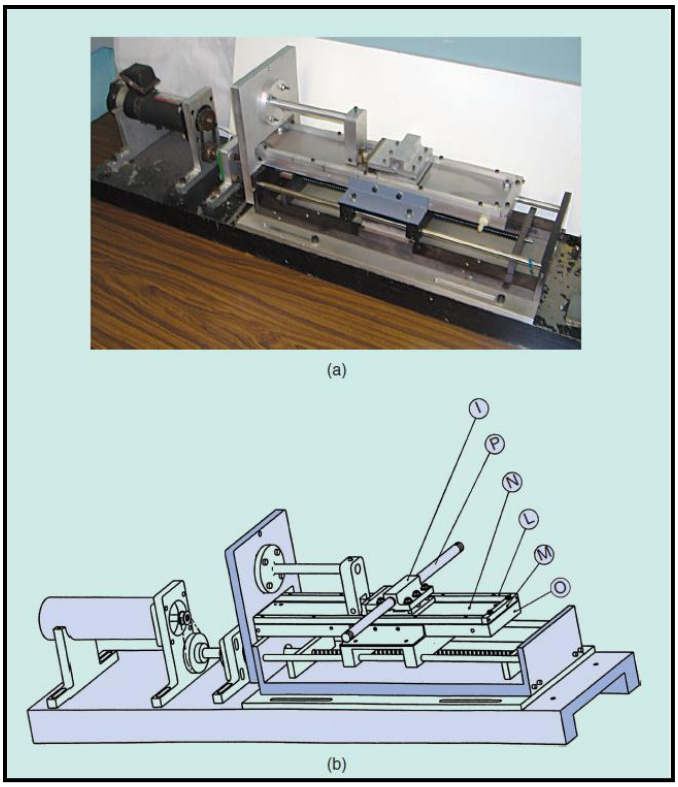


Figure 1.7 Harnoy's apparatus for measuring friction in linear motion.
Source [6]

1.2 An Apparatus for Dynamic Friction Measurement

All the experimental work presented in the last section illustrates the effort made by many scientists in order to obtain experimental data about friction. The experimental data is necessary to enhance our understanding of friction phenomenon and to design better models that simulate friction which will help reduce its side effects in real life applications. The basic and simple experiments that Da Vinci and Rabinowicz developed gave great information about static friction but little insight about the dynamic friction. The more complicated apparatuses always comprised a massive fixed base and a relatively light object moving on the base, although this configuration is necessary for friction measurements at substantial velocities [6], since it may be impractical to move

the base adequately. But keeping the base fixed leads to a problem associated with measuring dynamic friction, namely ensuring that the friction is the only force applied on the object [6].

To overcome this issue, Friedland suggested in a paper published in the Control System Magazine [6] that an apparatus with a moving base that has a light object free to move above it, such apparatus will be more appropriate to measure the dynamic friction in applications where low velocities and velocity reversals occur [7]. The configuration that he suggested is shown in Figure 1.8, and includes a reciprocating platform with a test object of mass *m* rests on it, the platform moves by some means of actuation and a non-contacting sensors is used to measure the velocity and acceleration of the object to The non-contacting sensors are to insure that no force but the friction force.

This configuration insures that the only force that makes the object sticks to the platform is friction, and in this case the friction acceleration would be equal to μg, where μ is the friction coefficient [6].

Based on Friedland's idea, this thesis presents an apparatus built to study dynamic friction in different conditions and for surfaces made of different materials, the implementation of such apparatus is presented in the next chapter of this thesis.

A Hall effect sensor has been used to measure the relative displacement of the Test Mass, and from the displacement, the relative velocity of the Test Mass was derived. The straight forward method to derive the velocity is by differentiating the displacement, but due to the low resolution of the DAQ used in this thesis, this was not practical. So an observer has been designed to estimate the velocity and overcome the quantization issue.

The friction force is measured using a MEMS accelerometer. The light weight, small dimensions and high accuracy of the MEMS accelerometers make them the most

suitable for the apparatus in this thesis.

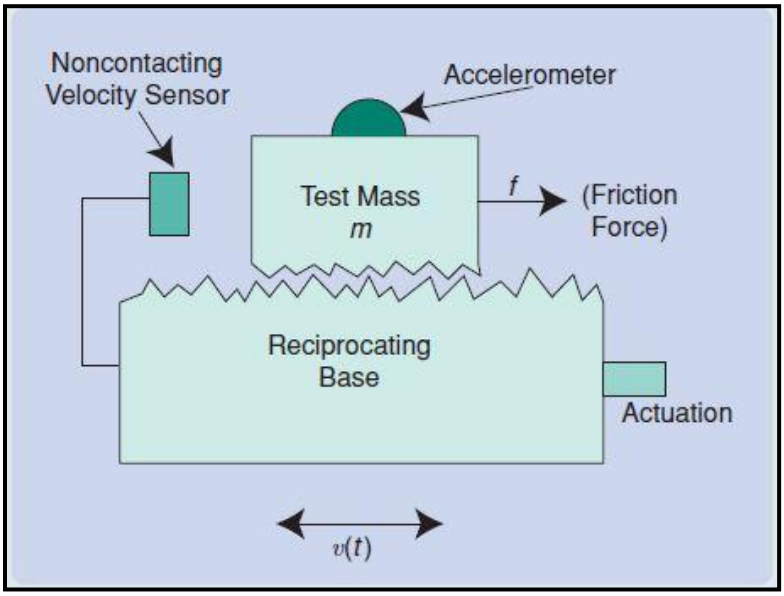


Figure 1.8 Conceptual representation of an apparatus for measuring friction. Friction is the only force acting on the upper object. This force is measured by the accelerometer.

Source: [6]

The actuation of the base is achieved by using two opposing solenoids. The actuator has the capability of high frequency operation which is needed for this apparatus. But the stray magnetic field of the solenoids has introduced noise that affected some of the components that has been used to implement this apparatus.

The apparatus has been built on a linear actuator developed by Alexander Rokhvarg in 1996 [8]. Rokhvarg constructed the apparatus to implement a control algorithm that he developed for a single-axis agnostic system, which consisted of a payload whose position is governed by the net force exerted by two opposed solenoids [8].

In order to use Rokhvarg's apparatus in the thesis herein, a number of modifications had to be made to the hardware and the software of that apparatus. These

changes can be summarized as follows:

- A platform is attached to the payload in order to fix a carrier of the Test Mass.
- Additional circuitry is added including the circuitry associated with sensors needed to measure the force of velocity and acceleration of the Test Mass.
- Rokhvarg's software is modified to accommodate a different data acquisition card.
- Rokhvarg's control algorithm, intended for a step input, is modified to permit a sinusoidal reference signal of arbitrary amplitude and frequency.

Chapter two of this thesis will explore in more details the implementation aspects of the apparatus built in this research.

CHAPTER 2

THE IMPLEMENTATION OF THE DYNAMIC FRICTION MEASUREMENT APPARATUS

The apparatus presented in Figure 2.1 is the implementation of the idea described in Chapter 1. The apparatus' hardware is constructed on a 0.5"thick aluminum plate that measures 12" by 20", while the software is implemented on two computers with two different data acquisition cards. The reason two computers have been used is further explained in the coming sections.

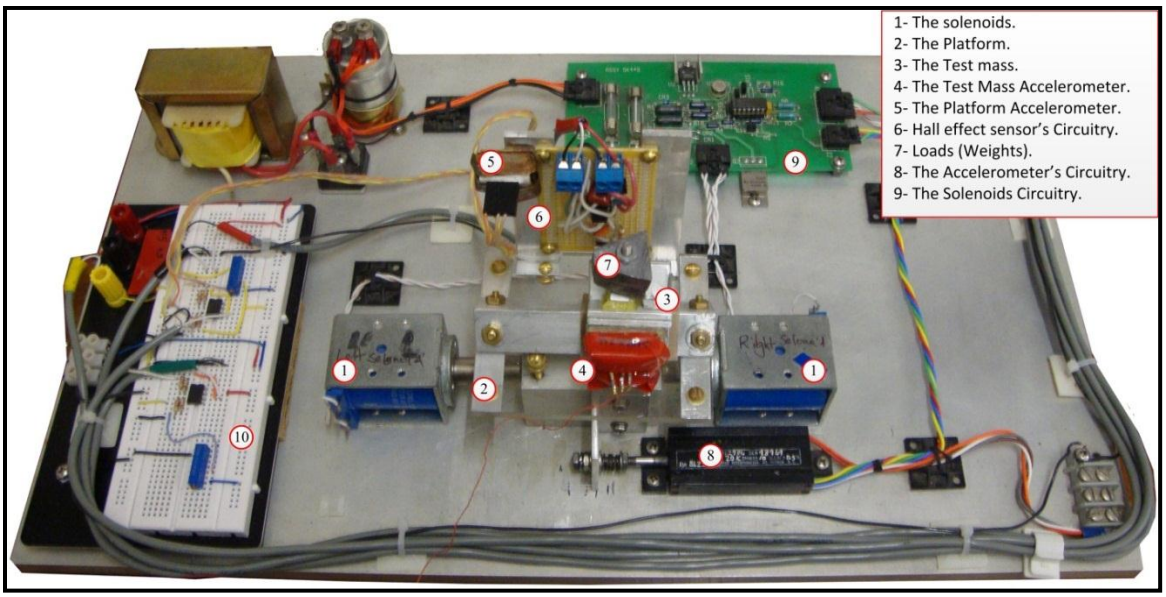


Figure 2.1 A photograph of the apparatus that shows the different components of the apparatus.

2.1 Hardware

Figure 2.2 shows a block diagram of the hardware of this apparatus, which comprises the following main subsystems:

- The actuation system that includes the solenoids and their associated circuitry.
- The measurement platform with the Test Mass.
- The instrumentation of the apparatus includes a Hall effect sensor with its circuitry, two accelerometers with their associated circuitry and a PCB board that has the power supply and signal conditioning circuit associated with the two solenoids.

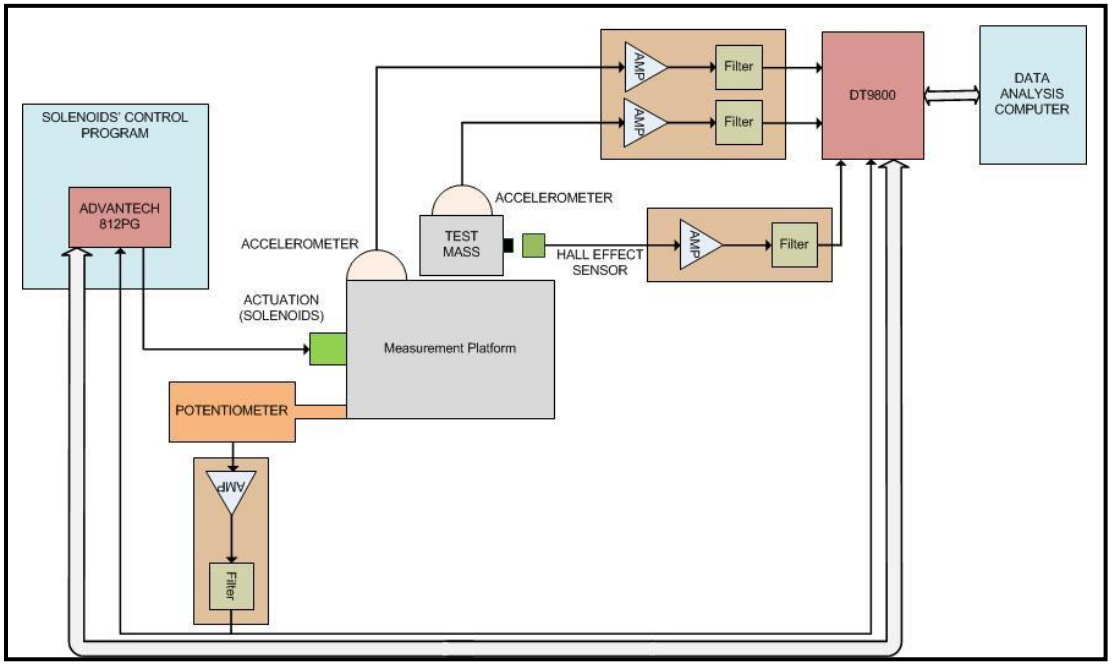


Figure 2.2 A block diagram of the apparatus different components and subsystems.

2.1.1 The Actuation and The Measurement Platform

The actuation system and the measurement platform are presented in Figure 2.3.

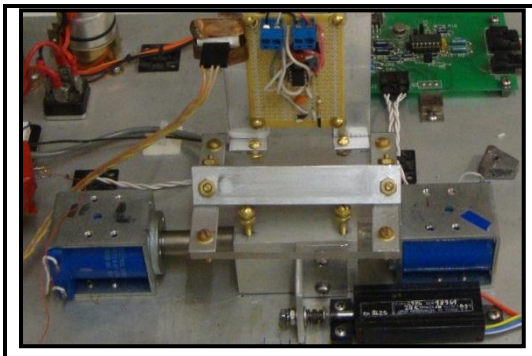
The two opposing solenoids used in this thesis are D-frame Intermittent Duty devices Part NO. 53719-87 manufactured by Deltrol Controls. They are capable of 1.0" travel and can develop forces in excess of 40 lb. The spacing between the solenoids is selected to allow 0.8" (0.02032 m) platform travel [8]. The solenoids provide a convenient means of direct unidirectional motion which is required in this apparatus, but they produce static magnetic noise that affects the measurements. Thus, to reduce the effects of the solenoids' magnetic field, some of the components of the apparatus have been shielded and, whenever possible, shielded wires have been used.

The solenoids are energized using a Voltage Controlled Current Source (I of V), because the force generated by the solenoids is a function of the input current [8].

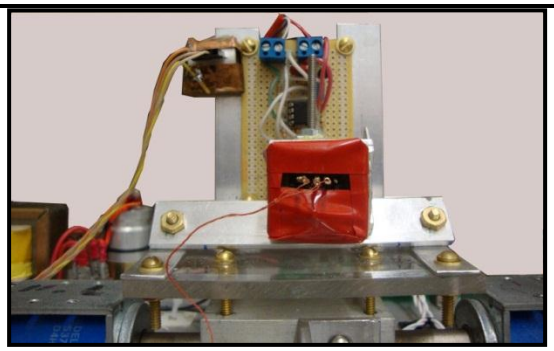
The measurement platform includes a cuboid base that is secured directly to the solenoid's plungers. On top of the base, a plastic plate has been fastened to provide a table for a triangular rail to carry the Test Mass and allow it only a one dimension motion. The rail and the Test Mass are made of Aluminum, but other different materials can be placed between the Test Mass and the rail to study friction for different materials. The measurement platform moves by forces generated by the opposed-solenoids actuation system.

On the top of the Test Mass, there is a screw that function as a holder to put external loads. This allows studying friction under different loads. The Test Mass also has an accelerometer attached to one side of it, and on the opposite side, four magnets have been attached. The magnets are used with Hall effect sensor system in measuring displacement.

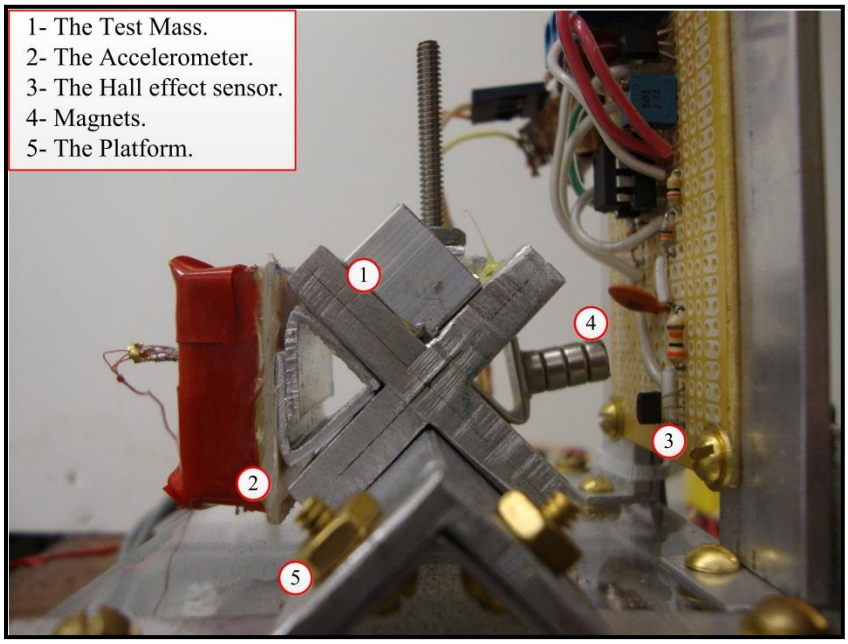
In order to prevent any outside force form affecting the Test Mass, the connections to the accelerometer on the Test Mass are made by means of 0.43 mm gauge wires which have negligible mechanical effect. Hence the only significant force acting on the Test Mass is the force of friction.



(a)



(b)



(c)

Figure 2.3 The measurement system consists of a platform driven by a pair of opposed solenoids and a Test Mass that is free to slide on the platform.

2.1.2 The Instrumentation and The Sensors

The friction force is measured by the accelerometer (Part No. ADXL203EB manufactured by Analog Devices) mounted on the Test Mass. The relative displacement between the Test Mass and the moving platform is measured by means of the Hall effect sensor system comprising a sensor mounted on the platform and four magnets on the Test Mass. The four magnets provide the magnetic flux needed to measure the Test Mass displacement. The Hall effect sensor part No. is SS496 manufactured by Honeywell Inc.

A linear potentiometer, whose plunger is fastened to the platform, is used to accomplish position feedback required by the solenoids control program. The potentiometer is a plastic film element device manufactured by Markite (type 2986). Its electrical travel is 1.0" and the mechanical travel is 1.3". A precision reference and scaling amplifier are used to interface with the potentiometer [8]. The 10.00 Volts reference was generated by an Analog Device Precession reference I.C. AD584ST, and the scaling amplifier is constructed with a precession quad Op-Amp manufactured by Linear Technology(P/NLT1014J), the gain of the amplifier was set so that the 0.0 to 10.0 Volt output corresponds to extreme Left to extreme Right payload positions respectively.

Three differential amplifiers have been designed and constructed on a prototype PCB board and a bread board. The amplifiers are used to scale and shift the output of the sensors to match the input range of the data acquisition cards which is $\pm$10V. All the amplifiers were constructed using an operational amplifiers manufactured by Analog Devices Inc. (P/T AD797AZN).

In order to limit the exposure of the analog to digital convertors to high frequencies an anti-aliasing Low-pass filter is used. For this apparatus a single stage RC filter have been used with a cut off frequency of 50Hz.

Appendix C lists all the schematics of the apparatus.

Two computers are used in this thesis because the opposed-solenoid actuator is an open-loop unstable system and require "hard" real-time control and relatively high sampling rate. The former cannot be provided under the Windows operating system. Hence the actuator-control is run on a 486 DX4 computer, equipped with an ADVATECH PCL 812 PG data acquisition card, under MSDOS 6.11 as a single interrupt service at a sampling rate of 0.8 milliseconds. The control algorithm is based on the algorithm developed by Rokhvarg [8]. Listing is given in Appendix B.

The second computer runs under Windows 7 and process the Graphical User Interface (GUI), which provides a capability for changing the frequency, the amplitude, the sample rate and the number of samples; and at the same time handling the sensors outputs, processing the data and displaying the experimental results. This computer is equipped with a DATA TRANSLATION DT9802 USB data acquisition system.

2.1.2.1 Hall Effect Sensor Calibration The Hall effect sensor is used in this research as a displacement sensor which is not its primary purpose of use, the sensor is used primarily to measure the magnetic flux. ,but when the Test Mass, with the magnets attached to it, moves relative to the platform, which has the Hall effect sensor fixed on it, the magnetic field applied on the Hall effect sensor will change. This change will produce a signal that represents the relative displacement of the Test Mass. Owing to the nonlinear magnetic field produced by the four magnets, the output signal of the Hall effect sensor is not proportional to the displacement and the sensor has to be calibrated.

To calibrate the hall effect sensor, the Test Mass has been moved for specific small distances (0.025") and each time the Test Mass moves the output of the hall effect sensor has been recorded. The final result is a set of points where each displacement

value represents a Hall effect sensor output value.

Figure 2.4 show a set of points that represent the Hall effect sensor outputs at a given displacement inputs. Not all the points have been used the first 10 points and the last 8 points had to be omitted in order to get a simple analytical function that describes the relation between the Hall effect sensor output and the Test Mass displacement.

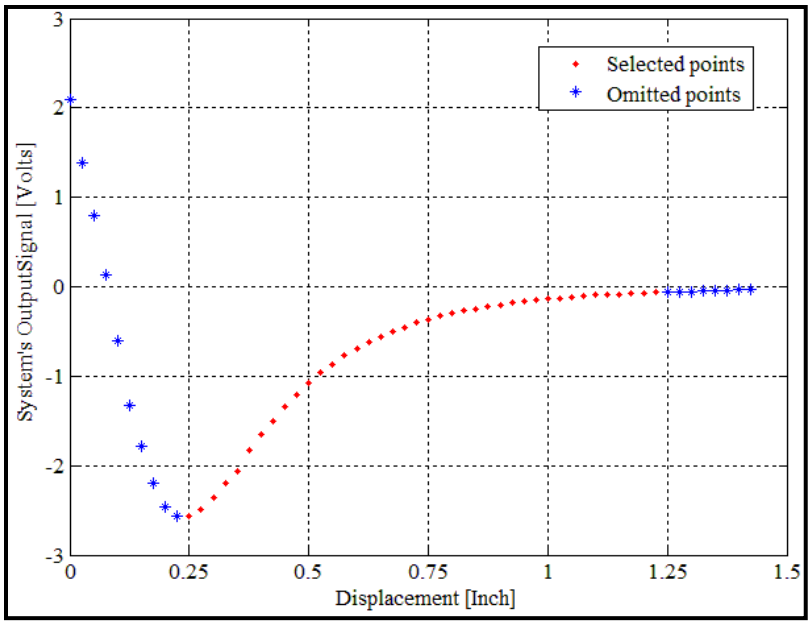


Figure 2.4 A set of points where each displacement value represents a Hall effect sensor output value.

After reversing the data points, displacement vs. voltage instead of voltage vs. displacement, and with the aid of MATLAB®, the best analytical function that represents the relation between the displacement and the sensor's output has been defined in equation 2.1.

$$d = 0.6338 \exp(10.86\, V) + 0.7051 \exp(0.9975\, V) \qquad (2.1)$$

Where:

d is the displacement in inches.

V is the Hall effect output voltage in Volts.

Figure 2.5 illustrates the fitting between the analytical function defined in equation 2.1 and the actual values of the Hall effect sensor signal obtained through the calibration.

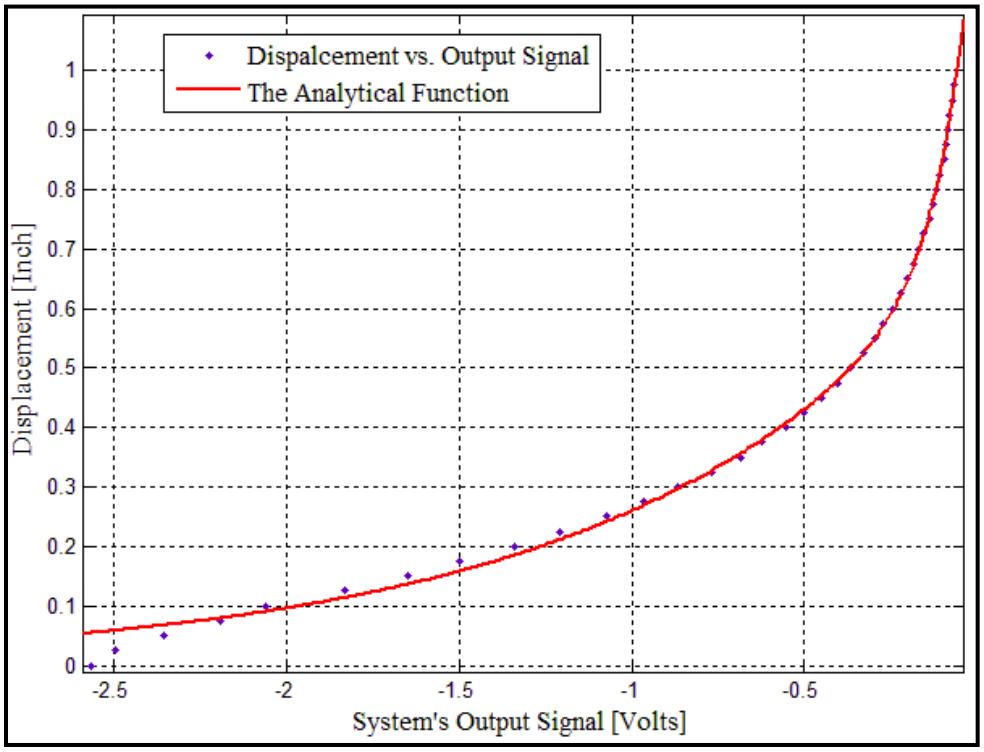


Figure 2.5 The Displacement plotted as a function of the Hall effect sensor output. The red curve represents the analytical function described in Equation 2.1, while the blue points represents the actual values of the hall effect sensor output for the displacement inputs.

2.2 Software

The software of this research comprises two programs: a graphical user interface for data processing and a control program that runs the control algorithm for the solenoids.

2.2.1 The Graphical User Interface (GUI)

A Graphical User Interface (GUI) is written and implemented using MATLAB®, Version 7.8.0.347. The GUI gathers all the inputs needed to operate the apparatus and sends them to the Platform control computer through the DT9802 card, and it collects all the signals from the sensors for processing and generating the desired outputs. In addition, all experimental data is saved in [*.mat] for subsequent analysis, if desirable.

Figure 2.6 shows a screen-shot of the GUI, The inputs of the GUI include the frequency, amplitude, sample rate and the Number of samples. The first two inputs are sent to the solenoids' control program to determine the frequency and the amplitude of the platform sinusoidal motion, while the sample rate and the number of sample are sent to the DT9802 card to determine the duration of the experiment.

Starting the apparatus is initiated by hitting the S*tart* button, where the GUI sends a signal to solenoid's control program to start moving the platform. When the experiment finishes, the GUI collects the signals from the sensors and send them to a MATLAB® Simulink model, where they are processed to get the friction coefficient and velocity signals. The GUI also displays the data collected from the sensors.

Although the motion of the platform is periodic, the Test Mass doesn't always move in a periodic pattern and sometimes it strays from the measurement range which produces corrupted data. To remedy these issues a duration field has been added to the

GUI, this field allows choosing a time portion of the experiment where the signals are clear and periodic.

For test and troubleshooting reasons a third field, called the variable plots, is added to plot the any signal that the experiment produces. The plotting can be versus time or sample's number.

The listing of the source code used to build the GUI is provided in Appendix A.

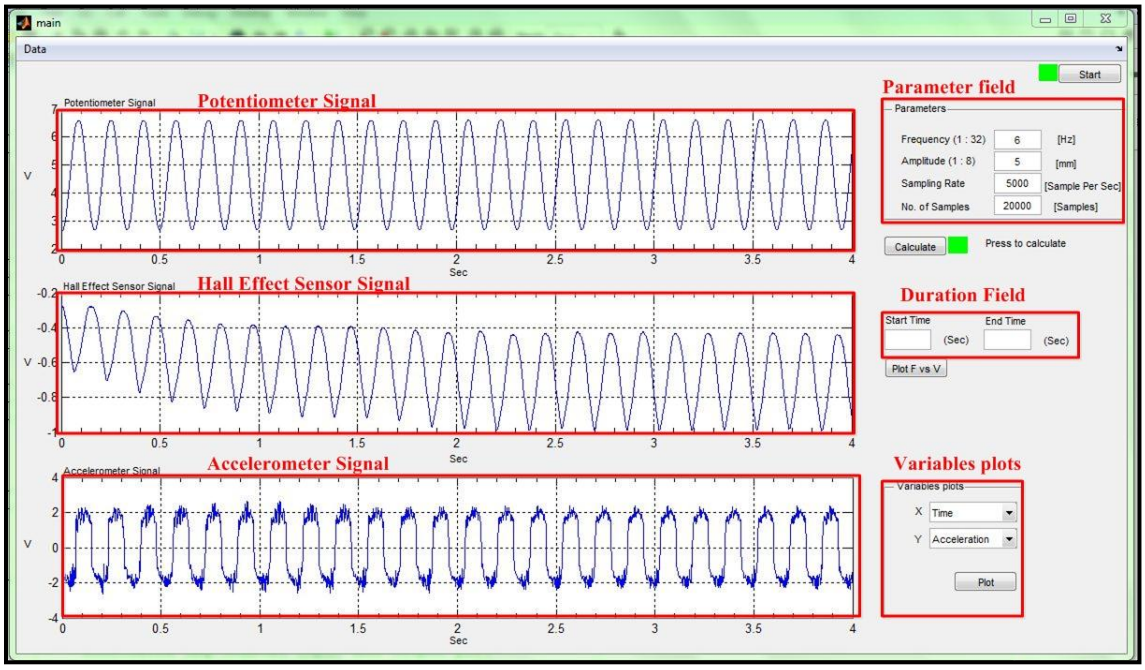


Figure 2.6 The graphical User Interface used in this research. The parameters field is for the conditions of the experiment, those include the frequency, amplitude, sample rate and the number of samples. The three axes fields are for plotting the raw data collected from the sensors. The duration field is for determining the time portion on which the output curve would be plot. The Variables plots field is for plotting any signal individually versus time or sample number.

The final output of the GUI is a Figure that shows the friction coefficient plotted versus the velocity the time period indicated in the duration field. Figure 2.7 shows a sample output of an experiment, Chapter 3 has more results that have been obtained by this apparatus.

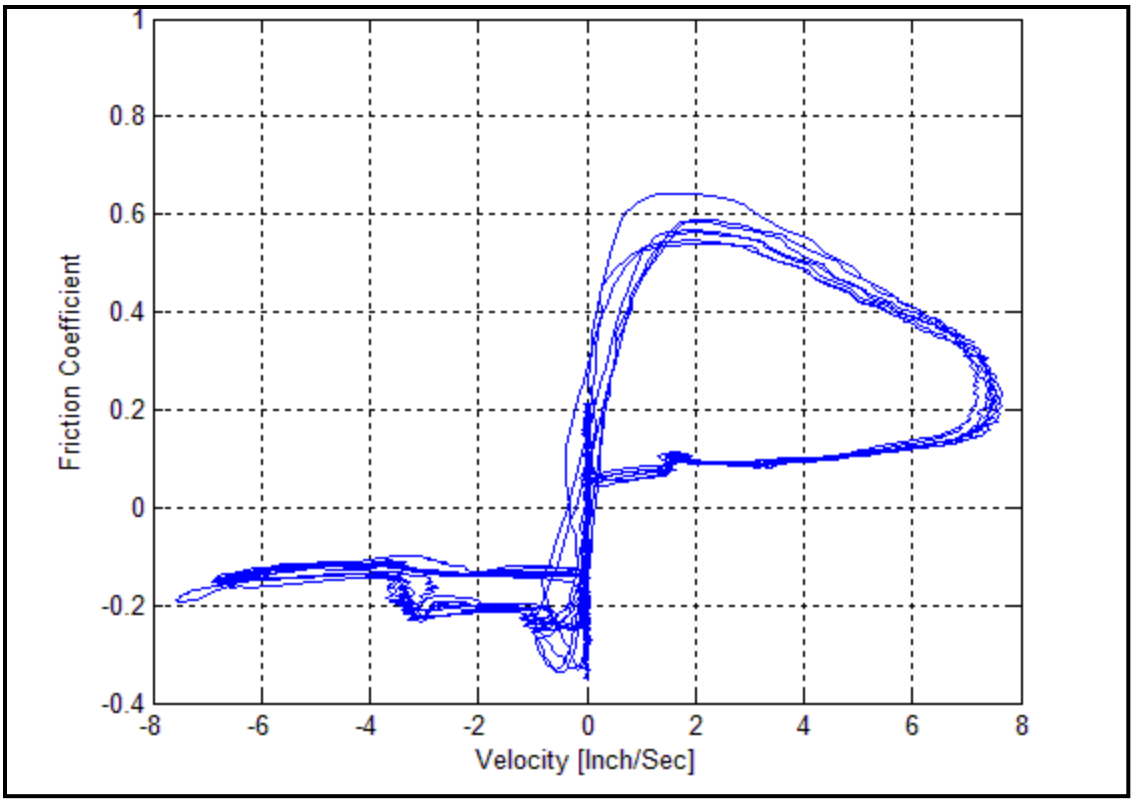


Figure 2.7 A sample of the final output of the GUI.

2.2.1.1 Velocity Estimation An observer has been used to get a smooth velocity estimate, because the 12-bit output does not have enough resolution to be differentiated.

According to theory in [9], if $y = q(x)$ is the 12-bit quantized relative displacement measurement after correction to compensate for the non-linearity, then the observer can be given by the equations:

$$\dot{\hat{x}} = \hat{v} + K_1 \left(y - q(\hat{x})\right) \tag{2.2}$$

$$\dot{\hat{v}} = K_2\left(y - q(\hat{x})\right) \tag{2.3}$$

Where:

- x is the displacement signal.
- $y = q(x)$ is the output of the quantizer.
- $\hat{v}$ is the desired smoothed velocity.
- $\hat{x}$ is the estimate of the displacement.
- K_1, K_2 are the observer's gains.

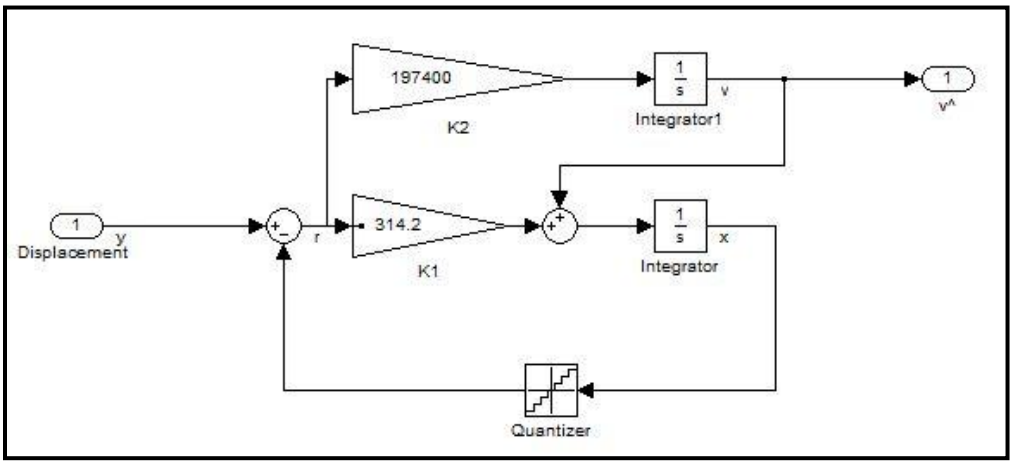


Figure 2.8 The observer that estimate velocity.

The gains K_1, K_2 are chosen, with the aid of MATLAB function fminisearch(), to minimize the residual:

$$r = y(x) - y(\hat{x}) \tag{2.4}$$

2.2.2 The Solenoids Control Program

The control program is written and implemented using Borland C++ version 3.0.

The program is constructed on the program Rokhvarg wrote in [4] with changes related to a different data acquisition card and the introduction of the control from the GUI. This program runs as a single interrupt service where all the interrupts of the processor are disabled to allocate all the computer resources to the program. The duration of running this program is decided by the GUI where it sends a start pulse when the

program should start and it remove the pulse when the program should be terminated. The program can always be terminated by hitting the ESC button.

CHAPTER 3

EXPERIMENTAL RESULTS

3.1 Calculating the Friction Coefficient

This section shows the derivation of the equation that has been used to calculate the friction coefficient from the measured acceleration of the Test Mass.

Figure 3.1 shows the free body diagram of the Test Mass. The forces acting on the Test Mass are the normal force N, the Test Mass weight *mg*, and friction *f*. The sum of on forces on the Y axis is:

$$N.\cos 45° = mg \qquad \ldots (3.1)$$

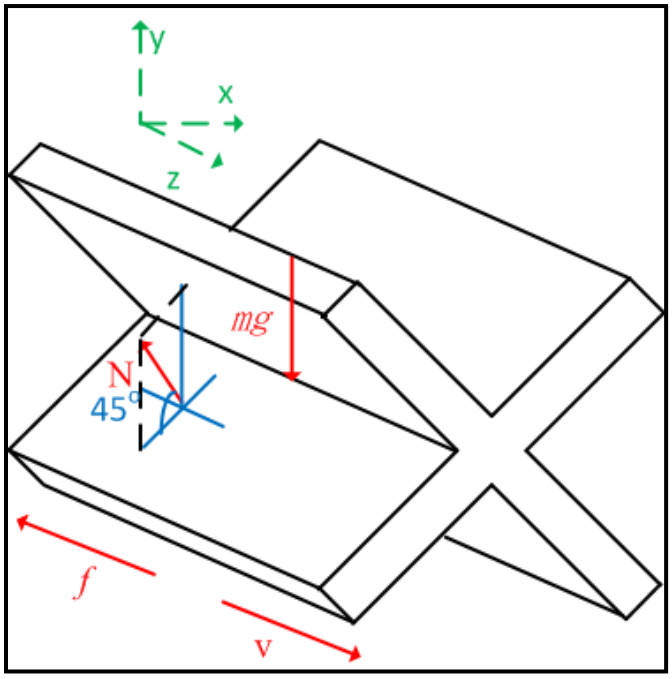


Figure 3.1 The free body diagram of the Test Mass. Only, the normal force N, the weight of the Test Mass, and friction are acting on the Test Mass.

The frictional force can be described [9] as:

$$f = \mu(v) \cdot N \quad \text{... (3.2)}$$

Where: μ is the friction coefficient, and in our case μ depends on the velocity and it's not always a constant.

From (3.1) and (3.2) we can derive:

$$\frac{f}{N} = \mu(v) \quad \text{... (3.3)}$$

But:

$$a = \frac{f}{m} \quad \text{... (3.4)}$$

a is the acceleration of the Test Mass.

Which makes (3.3)

$$\mu(v) = \frac{a.cos45°}{g.} \quad \text{... (3.5)}$$

The accelerometer measures the acceleration as a ratio of g (the gravitational constant).In other words:

$$a = a_{measured} \,.\, g \Rightarrow a_{measured} = \frac{a}{g} \quad \text{... (3.6)}$$

Substituting (3.6) in (3.5) will result in the final equation that will be used to calculate the friction coefficient:

$$\boxed{\mu(v) = a_{measured}.cos45°} \quad \ldots (3.7)$$

3.2 Averaging the Velocity and Friction Coefficient Data

Figure 3.3 shows data obtained during an arbitrary experiment. As the Figure illustrates that each of the Displacement (Test Mass), Velocity, and Friction Coefficient signals is periodic and follow the platform's motion frequency. Thus, in order to get better insight about the relation between the Fiction Coefficient and the Velocity, their signals have been averaged on a portion of time equals the Period (T) of the platform motion. Each of the signals has been averaged separately, and the averages are plotted to get another set of Figures. The script used in averaging the signals is listed in Table (3.1)

Table 3.1 The MATLAB Script Used to Average the Velocity and the Friction Coefficient Data

```
Nosamples1 = length(FrictionCoefficient);
Samples_Per_Period = floor((Samplerate)/Frequency);
No_of_cycles = floor(Nosamples1/Samples_Per_Period);


for I = 1: Samples_Per_Period
    Vel_Sum(I)=Velocity(I);
    Frction_Ceo_Sum(I)=FrictionCoefficient(I);
    for J = 1: (No_of_cycles - 1)
        Vel_Sum(I)=Vel_Sum(I)+...
            Velocity(I+(J*No_of_cycles));
        Frtion-Coe-Sum(I)= Frtion_Coe_Sum(I)+...
        FrictionCoefficient(I+(J* No_of_cycles));
    end
  Velavrg(I)=Vel_Sum(I)/No_of_cycles;
  FrtionCoiavrg(I)=Frtion_Coi_Sum(I)/No_of_cycles;
end
```

The algorithm behind the averaging is simple, the script takes the data vector of the Velocity and the Friction Coefficient, divide each one to portions of samples that matches the frequency of the platform motion. Then, the script sums the samples that fall in the same time slot.

Figure 3.2 shows that application of the algorithm on curve taken from the same experiment used to get the data in Figure 2.3.

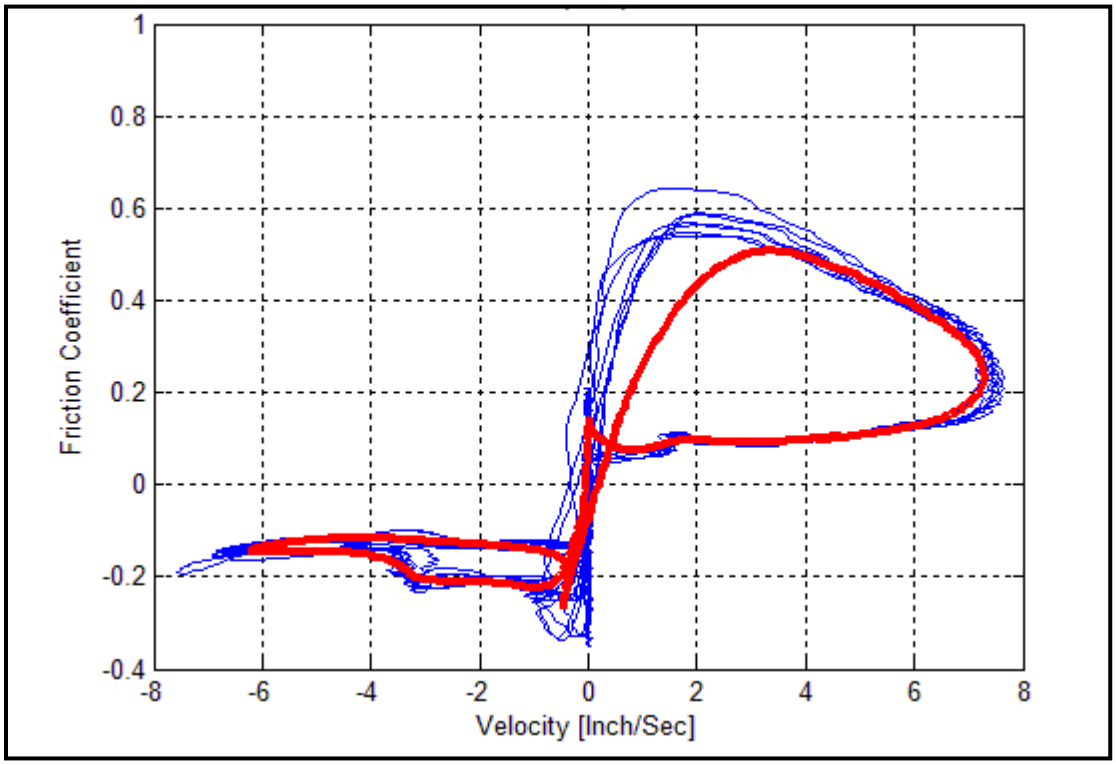


Figure 3.2 The averaging allows better insight about the relation between the Friction Coefficient and Velocity. The Blue curve is for a data taken from an experiment over one second period. The Red line is for the same data but it is averaged over that period.

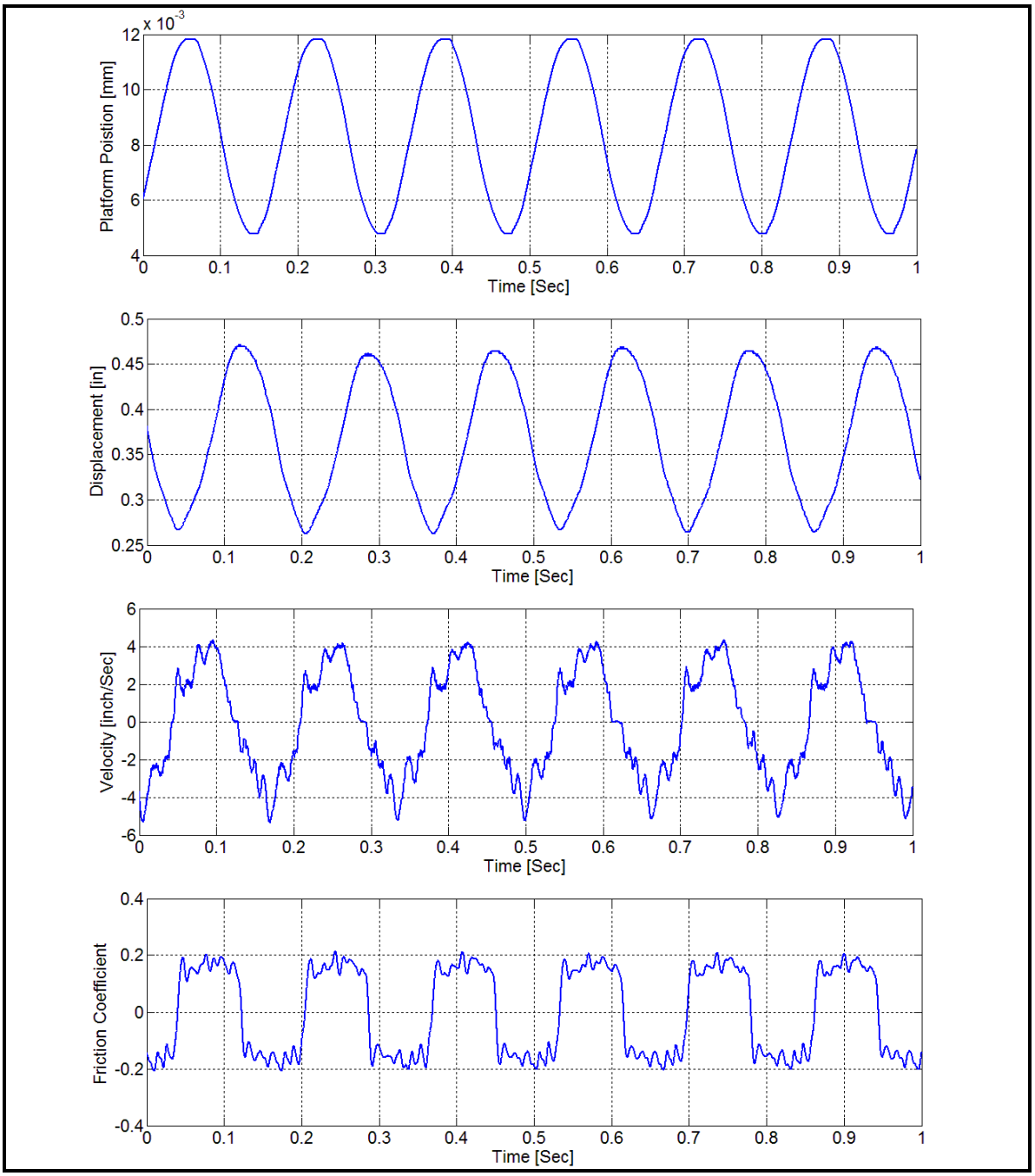


Figure 3.3 Position, Displacement, Velocity, and Friction Coefficient signals from an arbitrary experiment, taken for a 1 second period. Notice how all the signals are almost periodic and follow the position frequency.

3.3 The Velocity vs. Friction Coefficient Curves

This section presents the results obtained by the apparatus described in chapter two; the data is represented as curves of friction coefficient vs. velocity for different types of materials in different conditions. The experiments were carried out for polished Aluminum against Aluminum, Aluminum against Teflon, and Teflon against Teflon. The Aluminum against Aluminum experiments were carried out under dry conditions (no lubricants) and with lubricants, the lubricants used were the Silicon Spray manufactured by Radiator Specialty Company and the WD40 manufactured by WD40 Company.

Figures 3.4-3.48 shows the results in sets, each set (Figure) represents an experiment, the results are arranged in a way that allows comparing them for the same weight and material with the frequency fixed for horizontal results and the amplitude fixed for vertical results, after each set, three Figures that represent data taken for tow fixed conditions (e.g. amplitude and frequency) and variable third condition (e.g. loads).

The results, in general, for all experiments show acceptable repeatability and they share some characteristics that can be summarized in:

- There is a velocity memory which makes the friction values for increased velocities different from friction for decreased velocity which results in a hysteresis that is illustrated all over the curves shown in this chapter.
- There is a steep change in the sign of the friction coefficient near zero velocities which conforms to coulomb friction.
- Changing the external loads will have no effect on the value of the friction coefficient. For example, in the dry aluminum against aluminum experiment the friction coefficient is always (0.2-0.35) regardless of the load.

- For the same load and material, changing the amplitude has less effect than changing the frequency.

3.3.1 Aluminum against Aluminum (Dry Conditions)

The results of this experiment are presented in Figures 3.4 through Figure 3.12.

In this experiment, coulomb friction was the most observed type of friction, although for high frequencies the change of sign near zero velocity was not as abrupt as it should be in coulomb friction. See Figures 3.4, 3.6 and 3.8, for frequencies 7 and 8 Hz.This is due to the fact that for high frequencies the dynamic friction effects are more present, hence the change will be more subtle.

In Figure 3.4 for frequency 8 Hz and amplitude 7 mm there are signs of negative viscous friction, though that type of friction is only seen in lubricated contacts [1]. This may be due to the fact that because of the relative light weight of the Test Mass and high speed of the platform, a thin layer of air formed and caused the viscous friction.

3.3.2 Aluminum against Aluminum (with WD40)

The results of this experiment are presented in Figures 3.13 through Figure 3.21.

The hysteresis is more pronounced in this experiment and although the first expected outcome of introducing a lubricant between the two surfaces is the decrease of the friction coefficient, on the contrary, it increased. This is because the WD40 made the two surfaces stick together, and due to the platform motion, the Test Mass will gain momentum that exceeds the amount necessary for the bonds between the surfaces to break, hence the high friction coefficient. Another observation is that the sticking is directional, i.e. it depends on the motion direction.

3.3.3 Aluminum against Aluminum (with Silicon Spray)

The results of this experiment are presented in Figures 3.22 through Figure 3.30.

As in the WD40 experiment, sticking also occurred here, and the friction coefficient was not largely affected.

3.3.4 Teflon against Teflon

The results of this experiment are presented in Figures 3.31 through Figure 3.39.

The friction coefficient has decreased as expected, with most of the results showing coulomb friction with a friction coefficient of 0.2.

In most of the results that there have been minor hysteresis loop that has been attributed to static friction by Dahl [5] and Armstrong [1].

3.3.5 Teflon against Aluminum

The results of this experiment are presented in Figures 3.40 through Figure 3.48.

A clear decrease in the friction coefficient is observed, the friction coefficient is around 0.1. The results showed coulomb friction in most of the experiments. Figure 3.42 shows a Stribeck friction in the negative direction of velocity.

Table 3.2 summarizes the conditions under which the experimental results represented in Figures 3.4-3.48 were obtained.

Table 3.2 Summary of All the Experimental Results Shown in Figures 3.4 -3.31

Figure No.	Conditions					
	Materials	**Lubricants**	**Frequencies** (Hz)	**Loads** (g)	**Amplitudes** (mm)	**remarks**
3.4	Aluminum against Aluminum	No lubricants	5, 6, 7, 8	No load	4, 5, 6, 7	
3.5	Aluminum against Aluminum	No lubricants	5, 6, 7, 8	No load	4, 5, 6, 7	Averaged
3.6	Aluminum against Aluminum	No lubricants	5, 6, 7, 8	50g	4, 5, 6, 7	
3.7	Aluminum against Aluminum	No lubricants	5, 6, 7, 8	50g	4, 5, 6, 7	Averaged
3.8	Aluminum against Aluminum	No lubricants	4, 5, 6, 7	75g	4, 5, 6, 7	
3.9	Aluminum against Aluminum	No lubricants	4, 5, 6, 7	75g	4, 5, 6, 7	Averaged
3.10	Aluminum against Aluminum	No lubricants	8	No load	4, 5, 6, 7	Summary
3.11	Aluminum against Aluminum	No lubricants	5, 6, 7, 8	50 g	6	Summary
3.12	Aluminum against Aluminum	No lubricants	7	0, 50, 75	5	Summary
3.13	Aluminum against Aluminum	WD40	5, 6, 7, 8	No load	4, 5, 6, 7	
3.14	Aluminum against Aluminum	WD40	5, 6, 7, 8	No load	4, 5, 6, 7	Averaged
3.15	Aluminum against Aluminum	WD40	5, 6, 7, 8	50g	4, 5, 6, 7	

Table 3.2 (Continued A) Summary of All the Experimental Results Shown in Figures 3.4 -3.31

Figure No.	Conditions					
	Materials	**Lubricants**	**Frequencies**	**Loads**	**Amplitudes**	**remarks**
3.16	Aluminum against Aluminum	WD40	5, 6, 7, 8	50g	4, 5, 6, 7	Averaged
3.17	Aluminum against Aluminum	WD40	5, 6, 7, 8	75g	4, 5, 6, 7	
3.18	Aluminum against Aluminum	WD40	5, 6, 7, 8	75g	4, 5, 6, 7	Averaged
3.19	Aluminum against Aluminum	WD40	8	0	4, 5, 6, 7	Summary
3.20	Aluminum against Aluminum	WD40	5	0	4, 5, 6, 7	Summary
3.21	Aluminum against Aluminum	WD40	7	0, 50, 75	6	Summary
3.22	Aluminum against Aluminum	Silicon Spray	5, 6, 7, 8	No load	4, 5, 6, 7	
3.23	Aluminum against Aluminum	Silicon Spray	5, 6, 7, 8	No load	4, 5, 6, 7	Averaged
3.24	Aluminum against Aluminum	Silicon Spray	5, 6, 7, 8	50g	4, 5, 6, 7	
3.25	Aluminum against Aluminum	Silicon Spray	5, 6, 7, 8	50g	4, 5, 6, 7	Averaged
3.26	Aluminum against Aluminum	Silicon Spray	5, 6, 7, 8	75g	4, 5, 6, 7	
3.27	Aluminum against Aluminum	Silicon Spray	5, 6, 7, 8	75g	4, 5, 6, 7	Averaged

Table 3.2 (Continued B) Summary of All the Experimental Results Shown in Figures 3.4 -3.31

Figure No.	Conditions					
	Materials	**Lubricants**	**Frequencies**	**Loads**	**Amplitudes**	**remarks**
3.28	Aluminum against Aluminum	Silicon Spray	7	50	4, 5, 6, 7	Summary
3.29	Aluminum against Aluminum	Silicon Spray	5, 6, 7, 8	50	7	Summary
3.30	Aluminum against Aluminum	Silicon Spray	7	0, 50, 75	6	Summary
3.31	Teflon against Teflon	No lubricants	4, 5, 6, 7	No load	3, 4, 5, 6	
3.32	Teflon against Teflon	No lubricants	4, 5, 6, 7	No load	3, 4, 5, 6	Averaged
3.33	Teflon against Teflon	No lubricants	4, 5, 6, 7	50g	3, 4, 5, 6	
3.34	Teflon against Teflon	No lubricants	4, 5, 6, 7	50g	3, 4, 5, 6	Averaged
3.35	Teflon against Teflon	No lubricants	4, 5, 6	75g	3, 4, 5, 6	
3.36	Teflon against Teflon	No lubricants	4, 5, 6	75g	3, 4, 5, 6	Averaged
3.37	Teflon against Teflon	No lubricants	7	0	3, 4, 5, 6	Summary
3.38	Teflon against Teflon	No lubricants	4, 5, 6, 7	50	4	Summary
3.39	Teflon against Teflon	No lubricants	6	0, 50, 75	5	Summary

Table 3.2 (Continued C) Summary of All the Experimental Results Shown in Figures 3.4 -3.31

Figure No.	Conditions					
	Materials	**Lubricants**	**Frequencies**	**Loads**	**Amplitudes**	**remarks**
3.40	Aluminum against Teflon	No lubricants	4, 5, 6, 7	No load	4, 5, 6, 7	
3.41	Aluminum against Teflon	No lubricants	4, 5, 6, 7	No load	4, 5, 6, 7	Averaged
3.42	Aluminum against Teflon	No lubricants	4, 5, 6, 7	50g	4, 5, 6, 7	
3.43	Aluminum against Teflon	No lubricants	4, 5, 6, 7	50g	4, 5, 6, 7	Averaged
3.44	Aluminum against Teflon	No lubricants	4, 5, 6, 7	75g	3, 4, 5, 6	
3.45	Aluminum against Teflon	No lubricants	4, 5, 6, 7	75g	3, 4, 5, 6	Averaged
3.46	Aluminum against Teflon	No lubricants	7	50	4, 5, 6, 7	Summary
3.47	Aluminum against Teflon	No lubricants	4, 5, 6, 7	50	5	Summary
3.48	Aluminum against Teflon	No lubricants	6	0, 50, 75	4	Summary

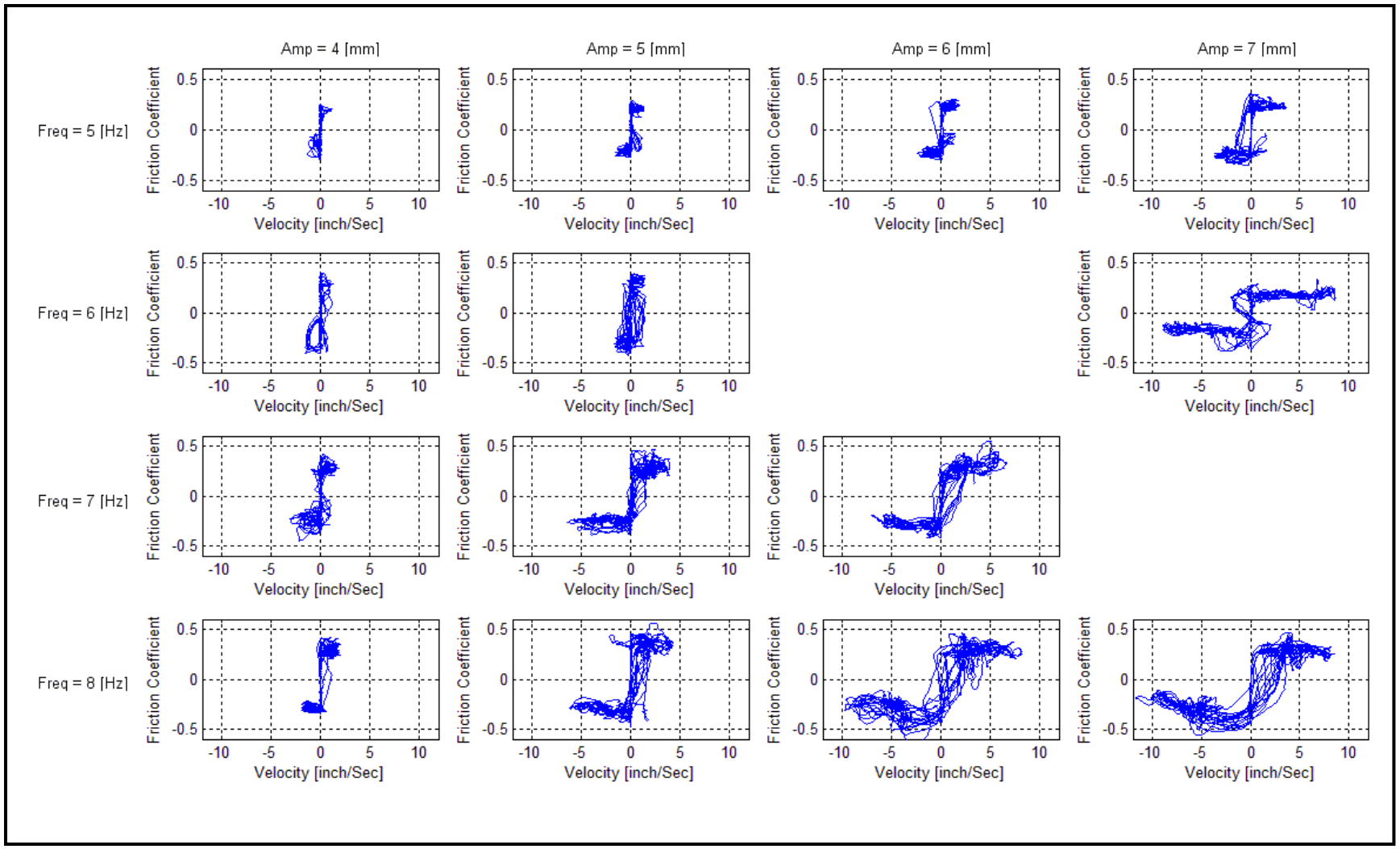


Figure 3.4 Experimental results for Aluminum against Aluminum, with no loads and with no lubricants.

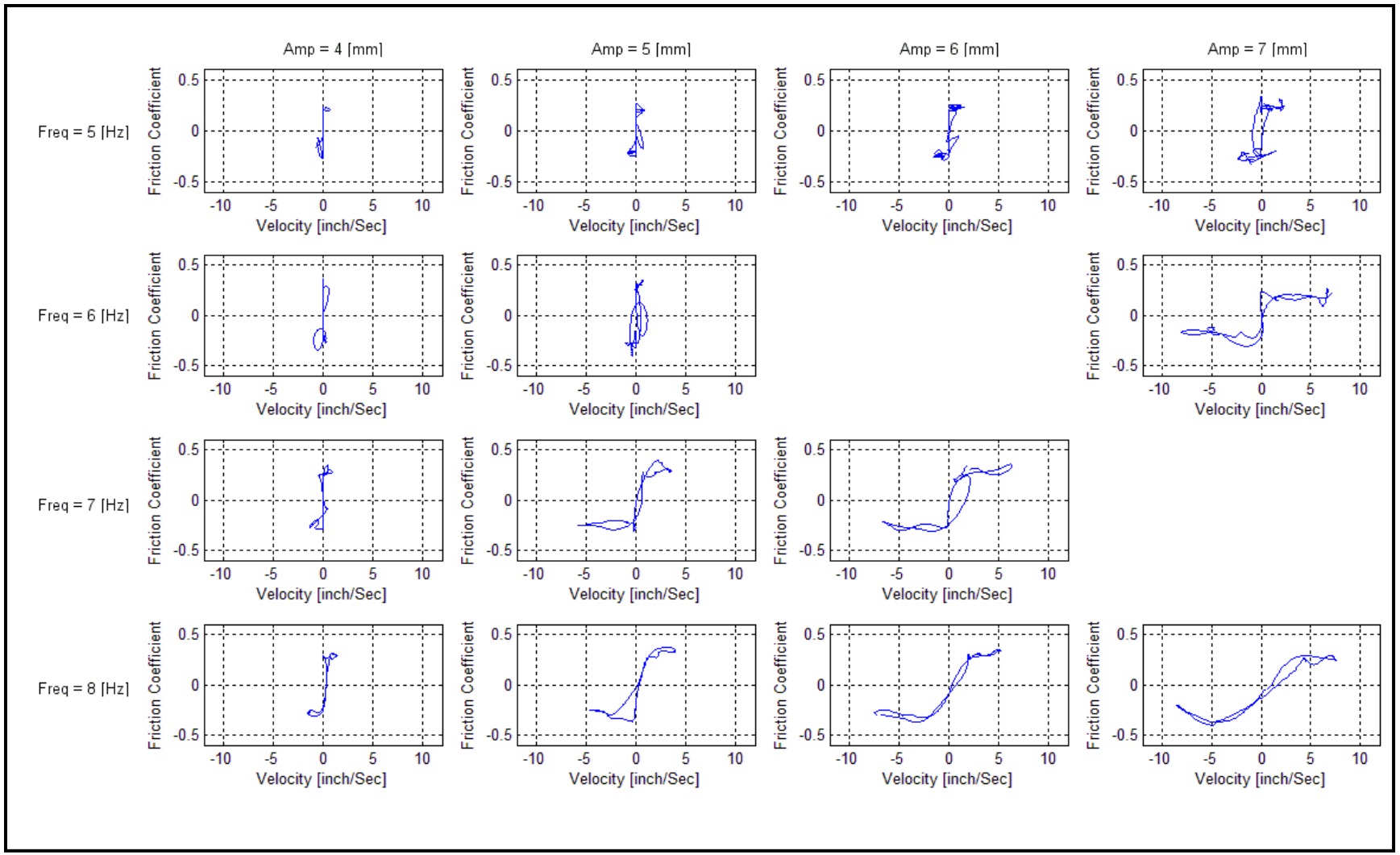


Figure 3.5 Averaged experimental results for Aluminum against Aluminum, with no loads and with no lubricants.

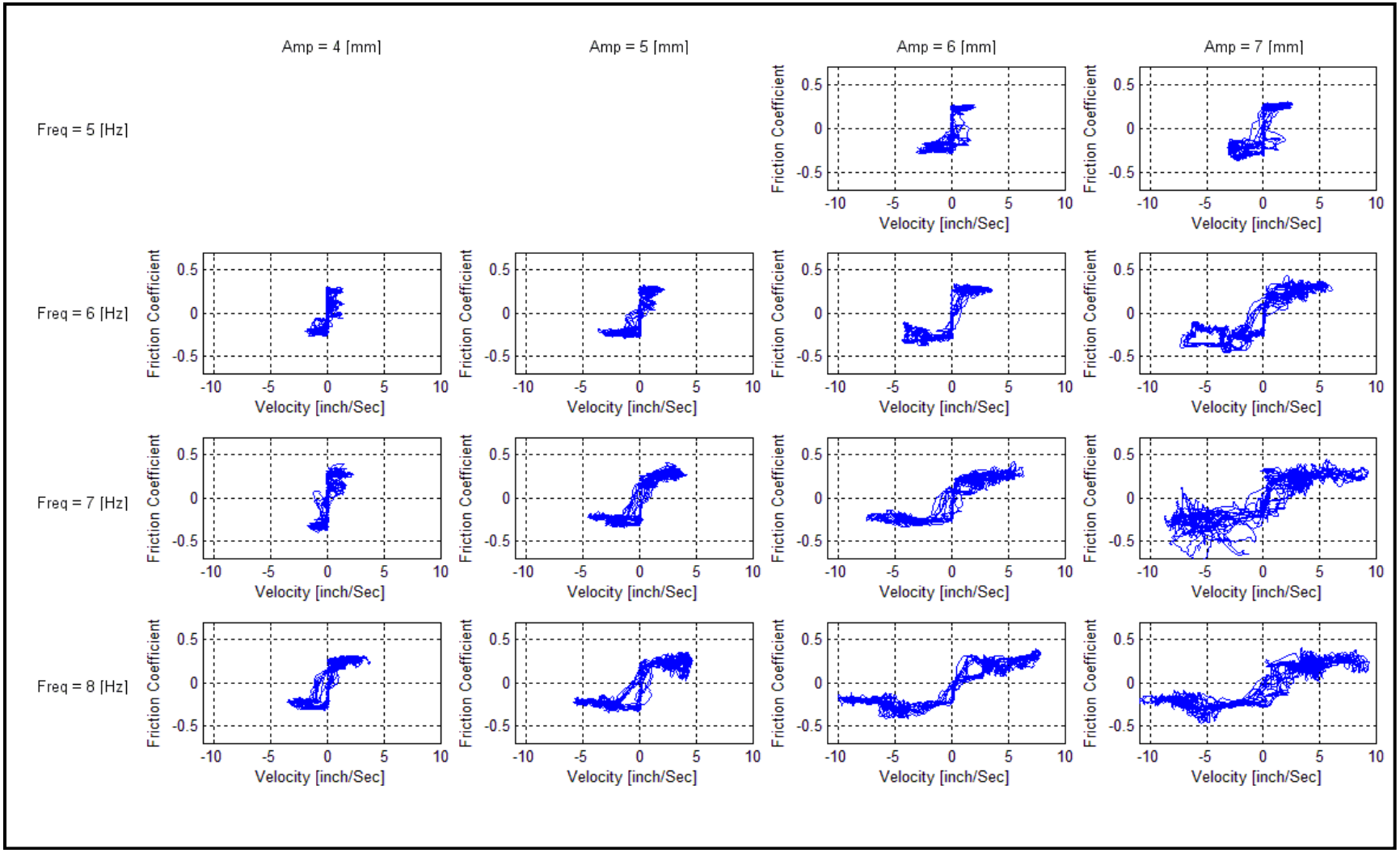


Figure 3.6 Experimental results for Aluminum against Aluminum, with 50g load and with no lubricants.

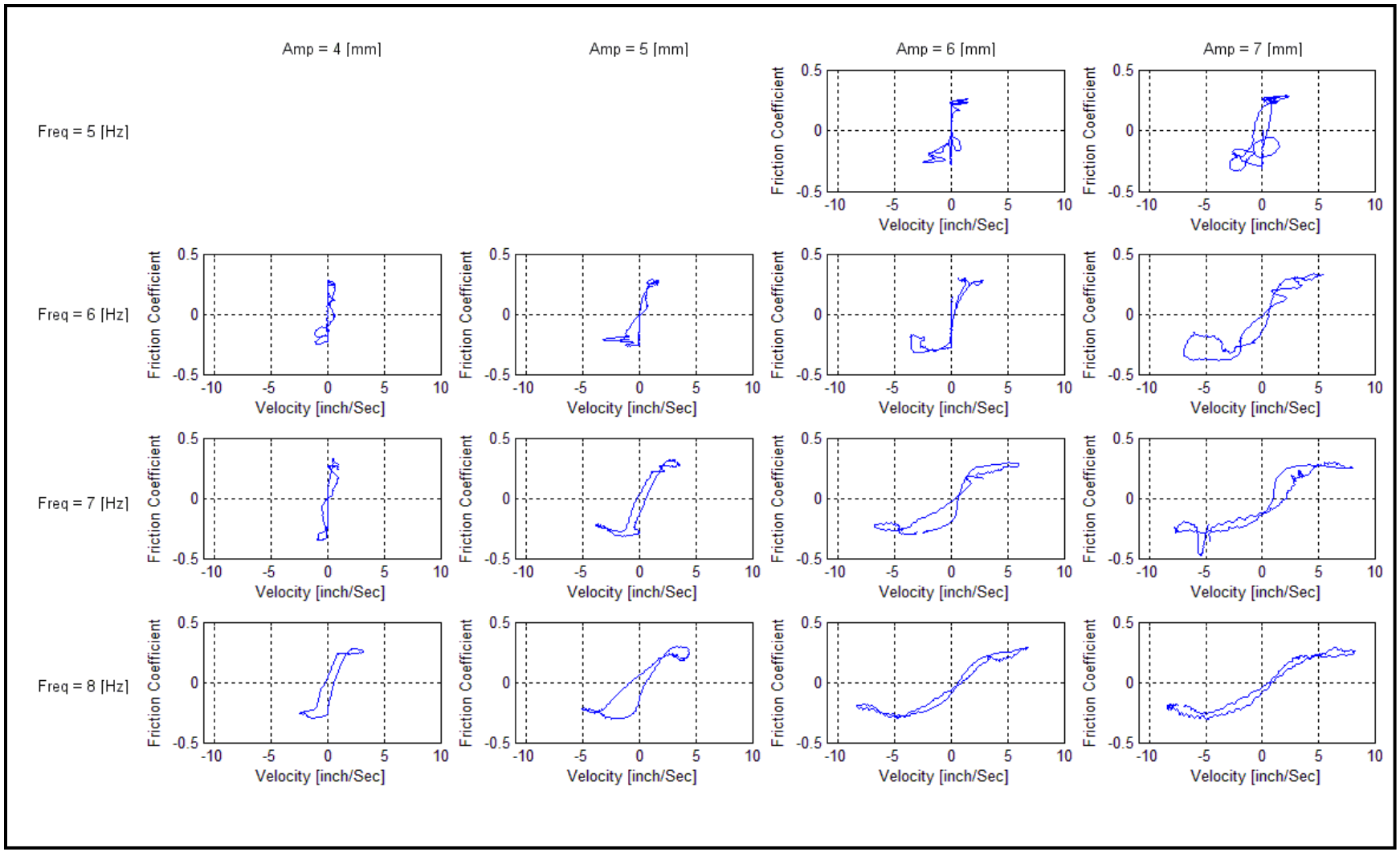


Figure 3.7 Averaged experimental results for Aluminum against Aluminum, with 50g load and with no lubricants.

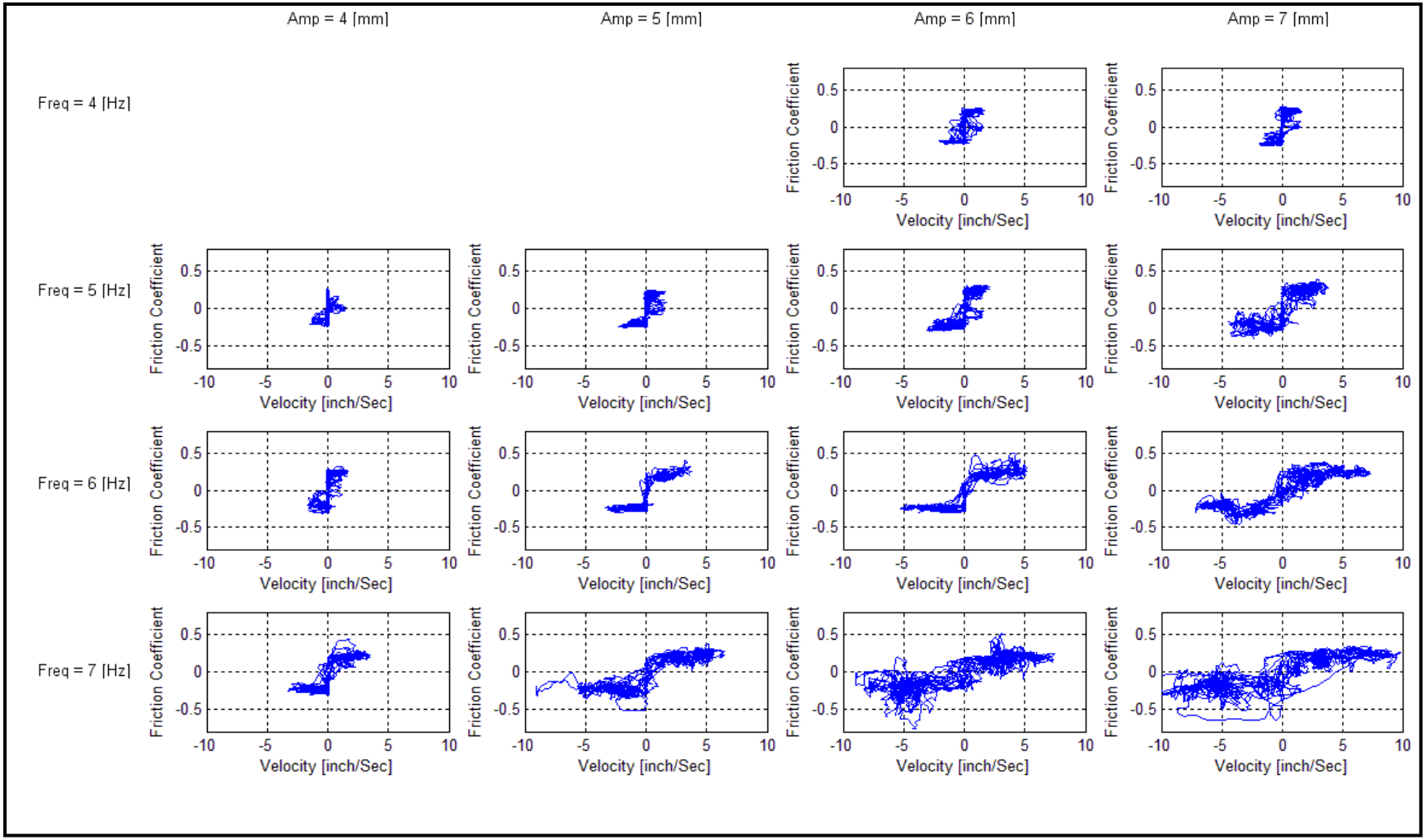


Figure 3.8 Experimental results for Aluminum against Aluminum, with 75g load and with no lubricants.

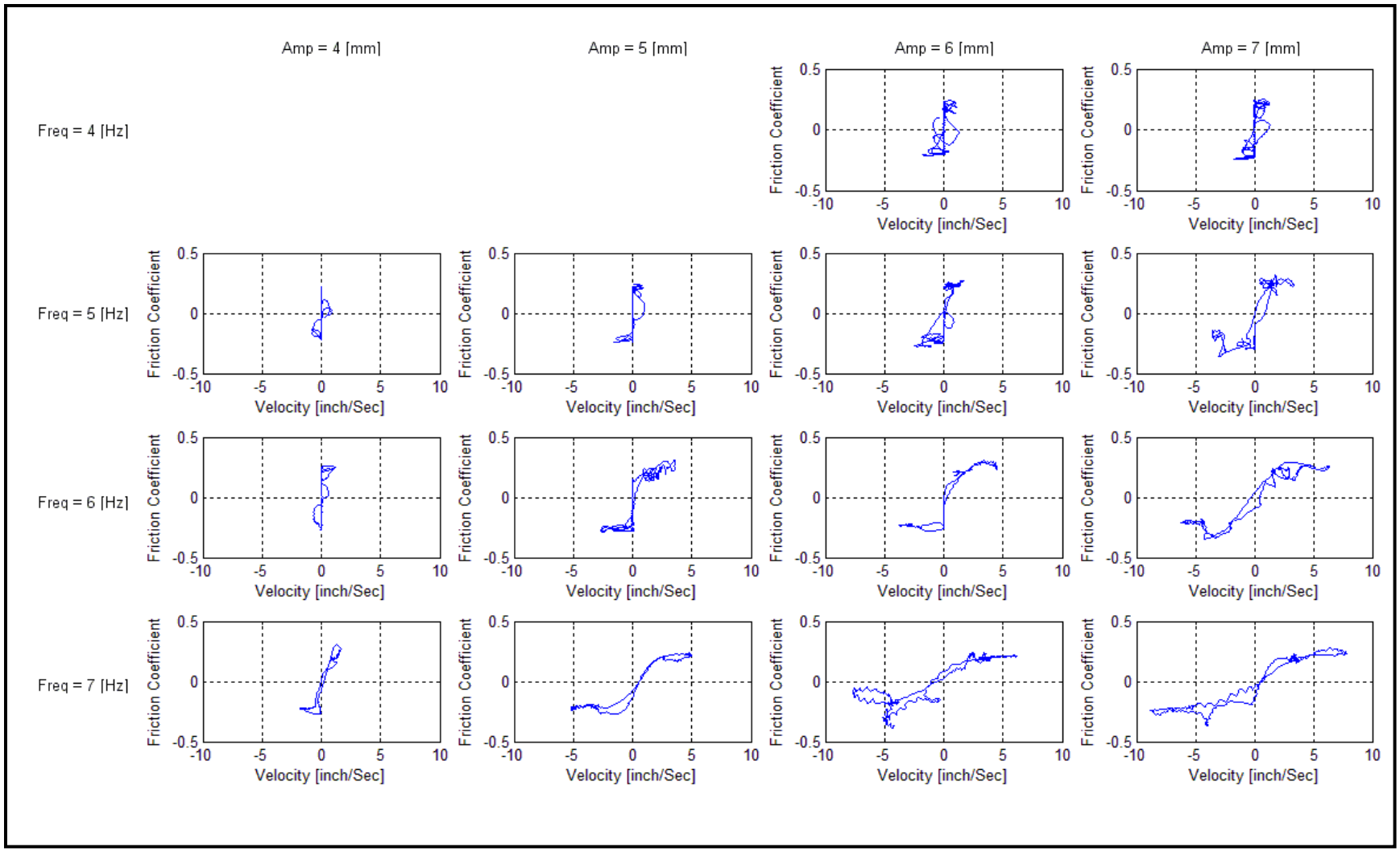


Figure 3.9 Averaged experimental results for Aluminum against Aluminum, with 75g load and with no lubricants.

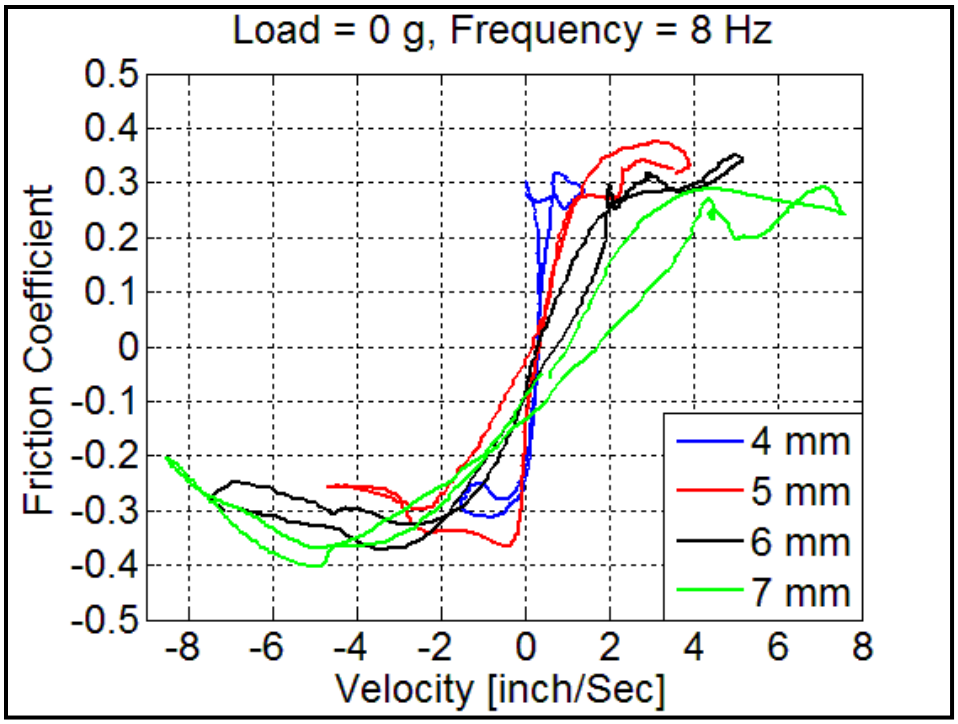


Figure 3.10 Aluminum – Aluminum in dry conditions (no lubrication), with no load, frequency of 8Hz and different amplitudes.

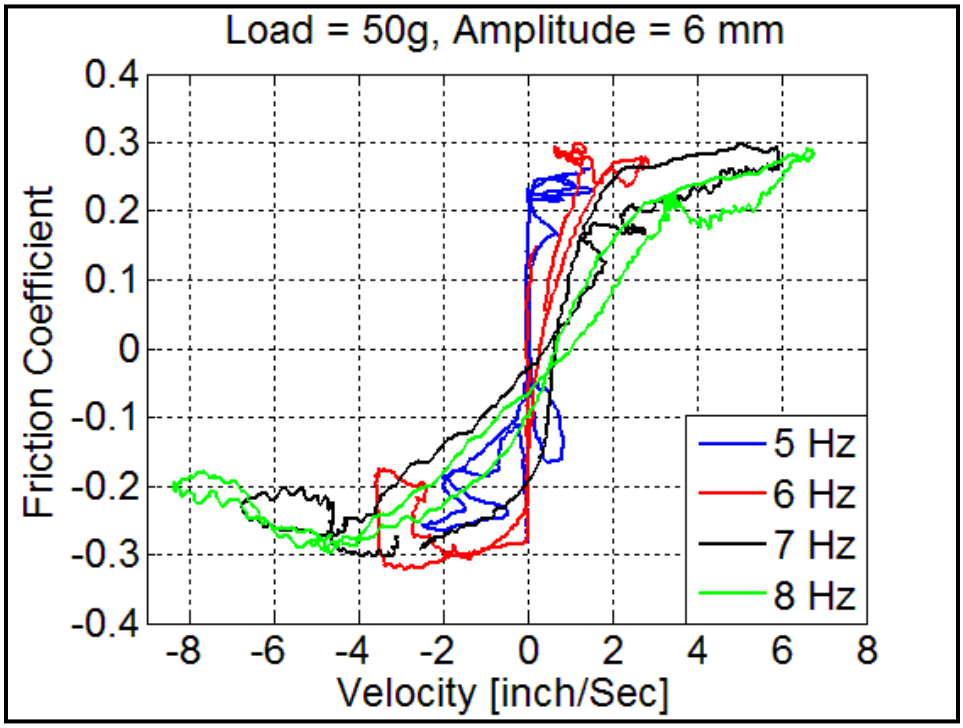


Figure 3.11 Aluminum – Aluminum in dry conditions (no lubrication), for a load of 50g, amplitude of 6 mm and different frequencies.

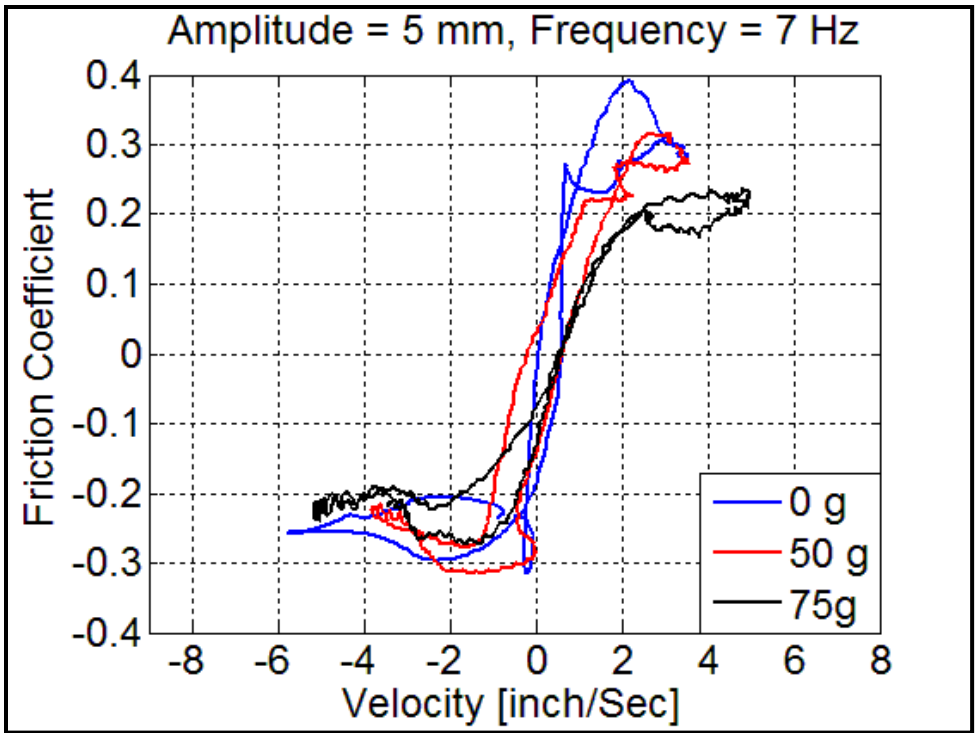


Figure 3.12 Aluminum – Aluminum in dry conditions (no lubrication), for amplitude of 5 mm, frequency of 7 Hz and different loads.

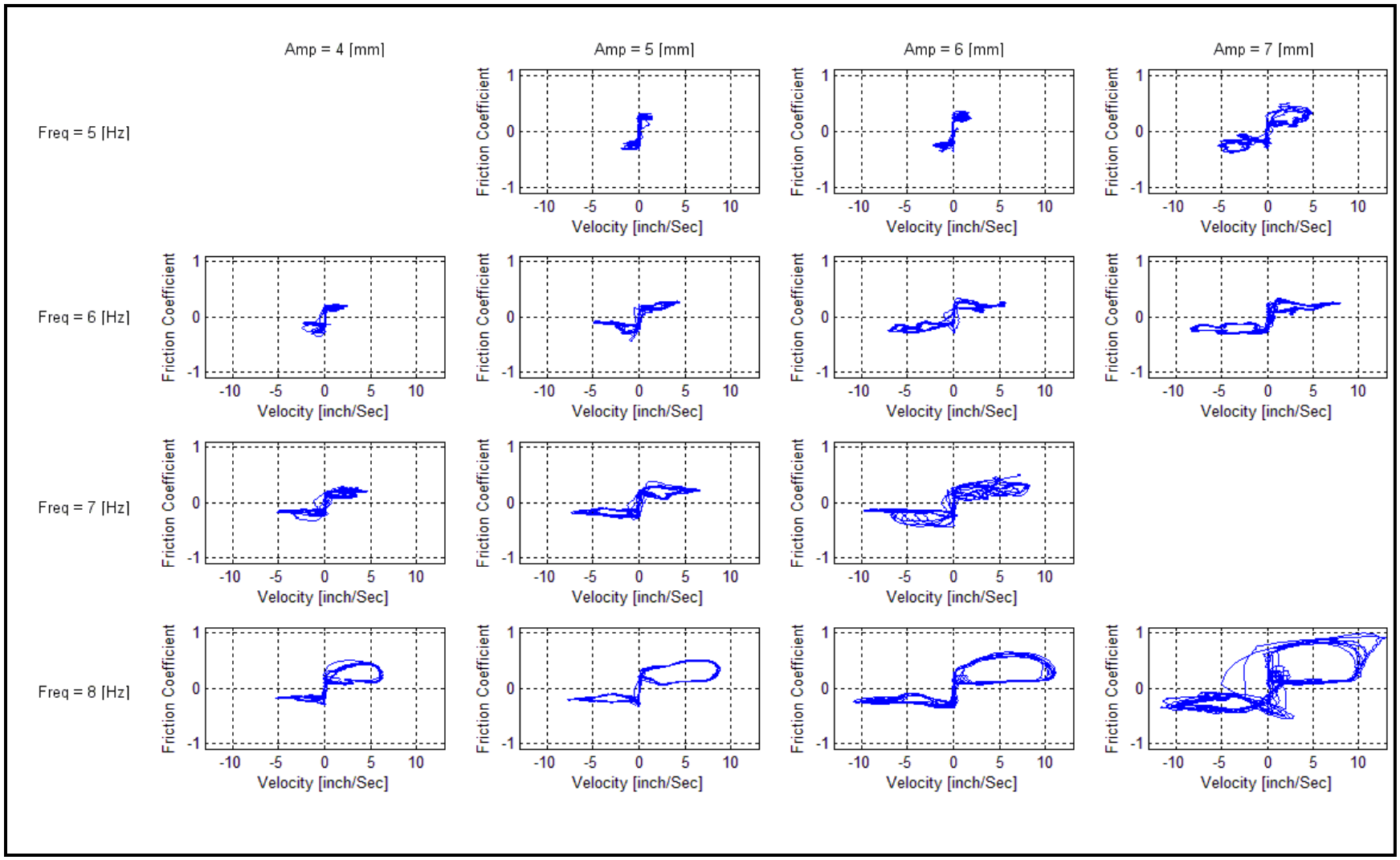


Figure 3.13 Experimental results for Aluminum against Aluminum, with no load and WD40 as a lubricant.

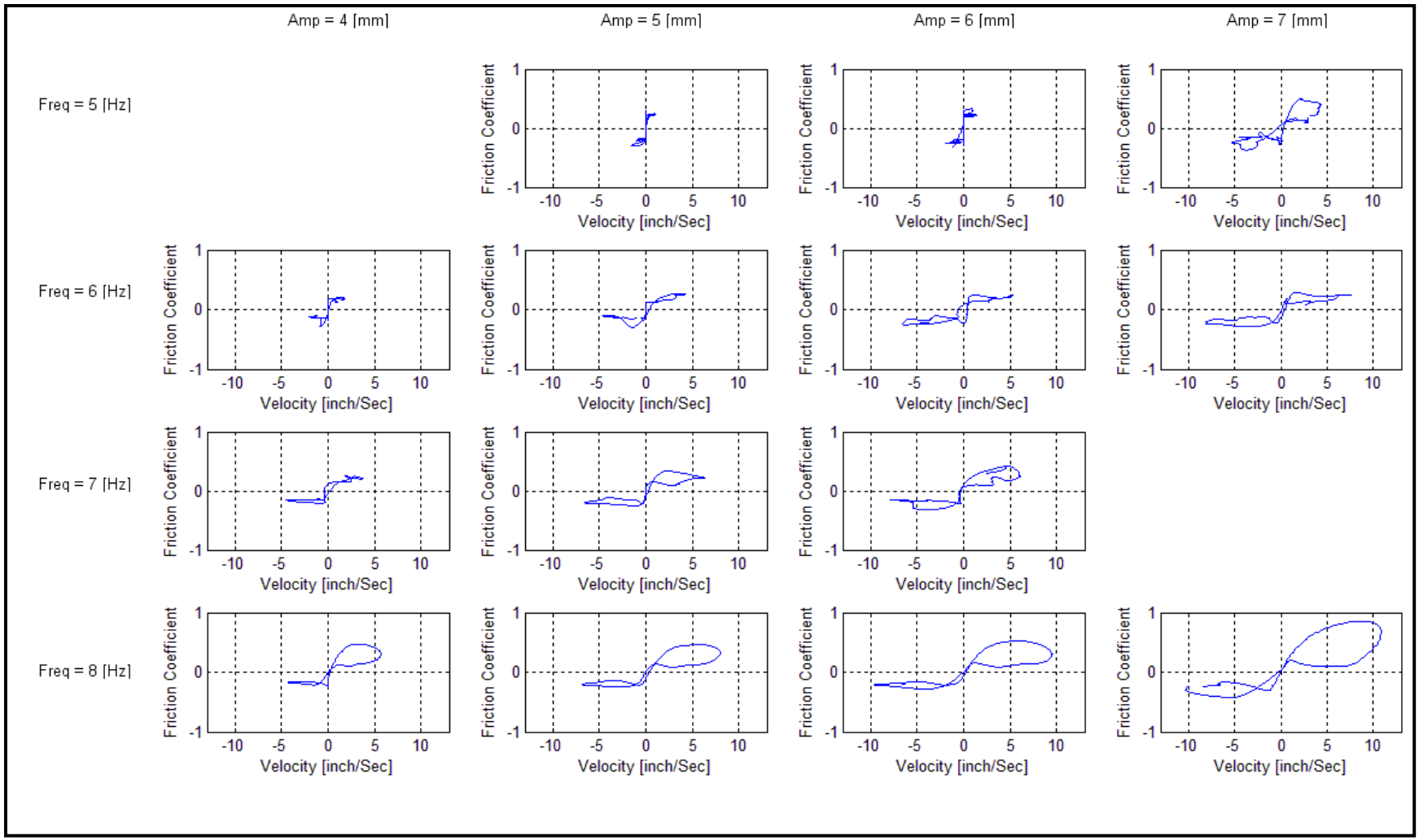


Figure 3.14 Averaged experimental results for Aluminum against Aluminum, with no load and WD40 as a lubricant.

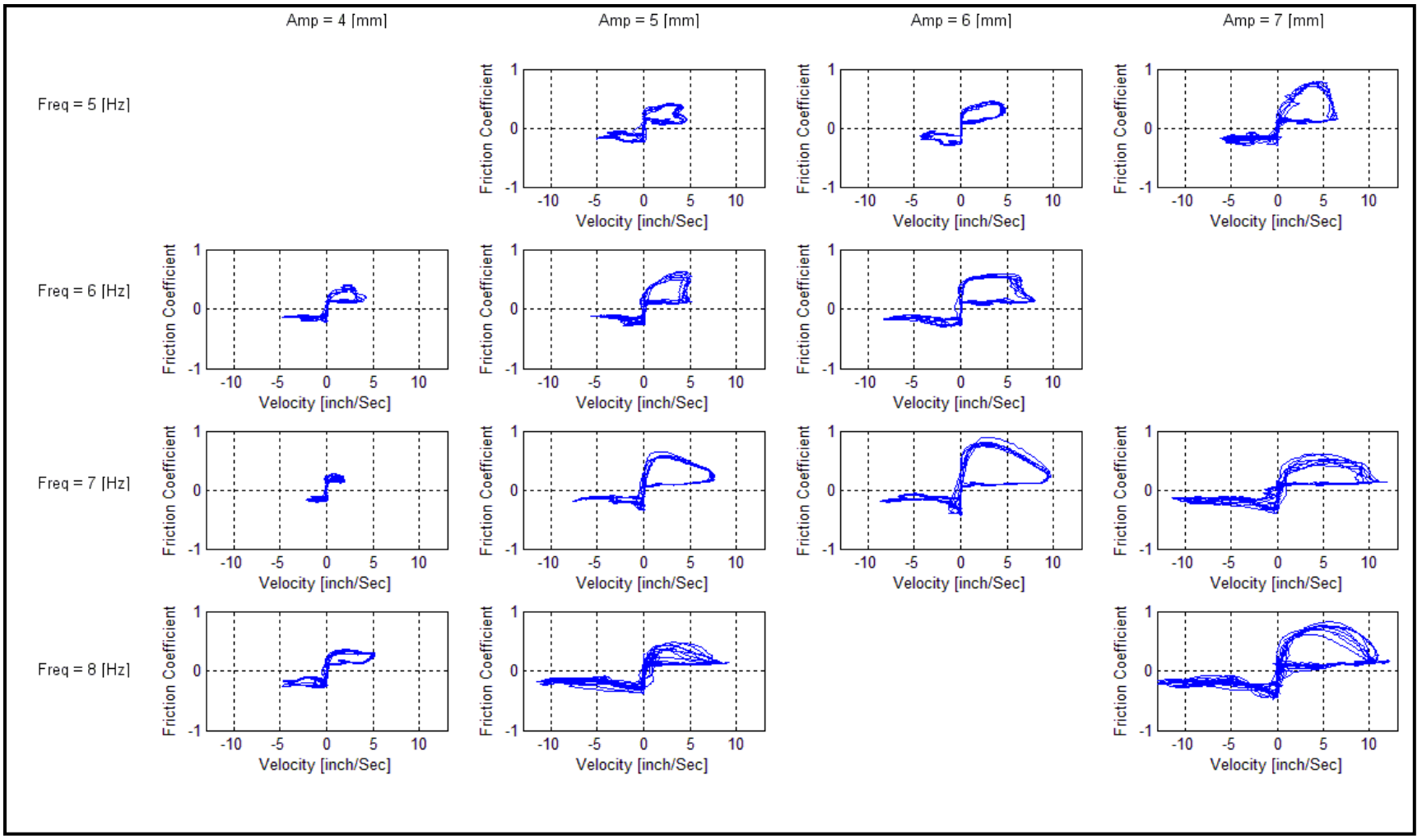


Figure 3.15 Experimental results for Aluminum against Aluminum, with 50g load and WD40 as a lubricant.

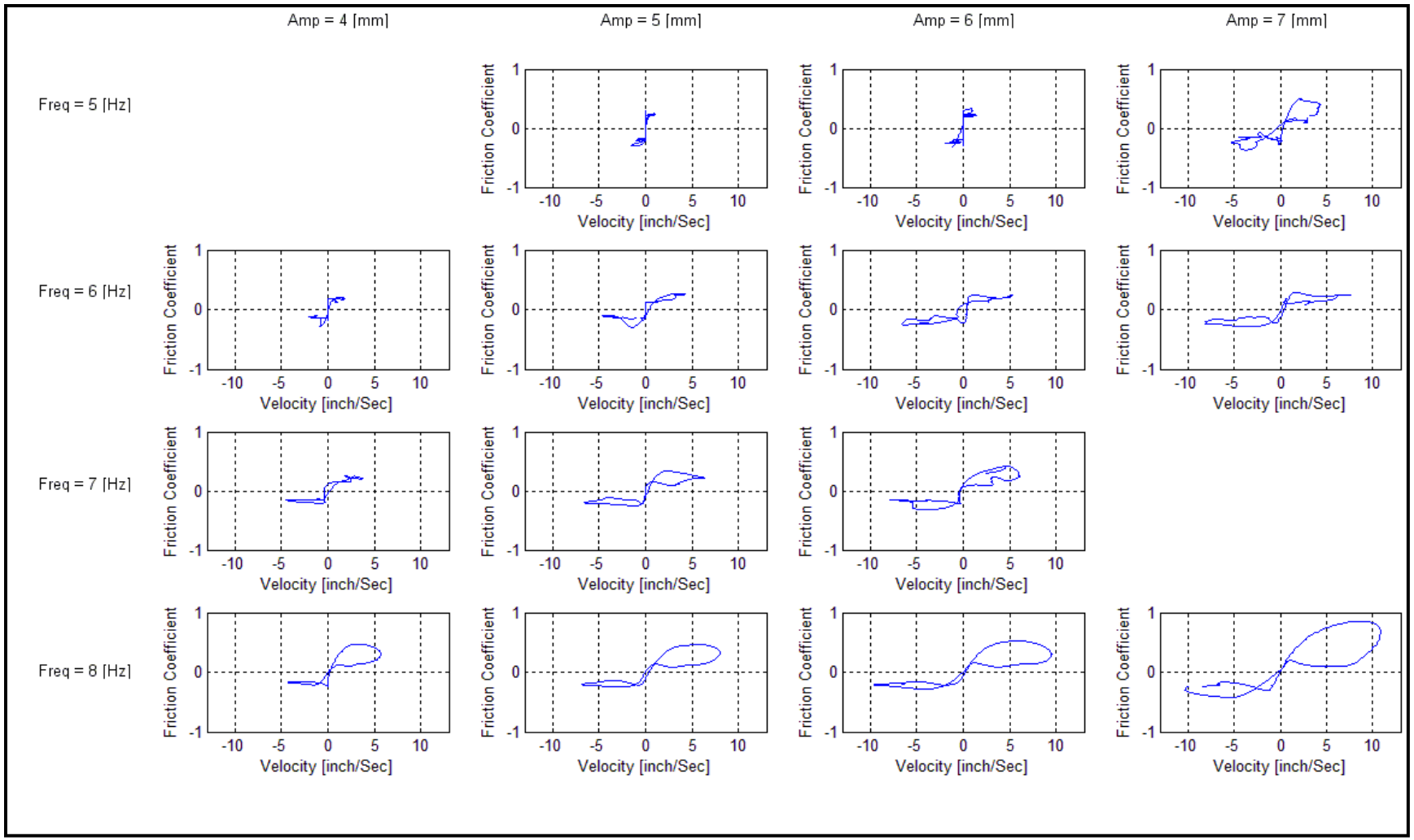


Figure 3.16 Averaged experimental results for Aluminum against Aluminum, with 50g load and WD40 as a lubricant.

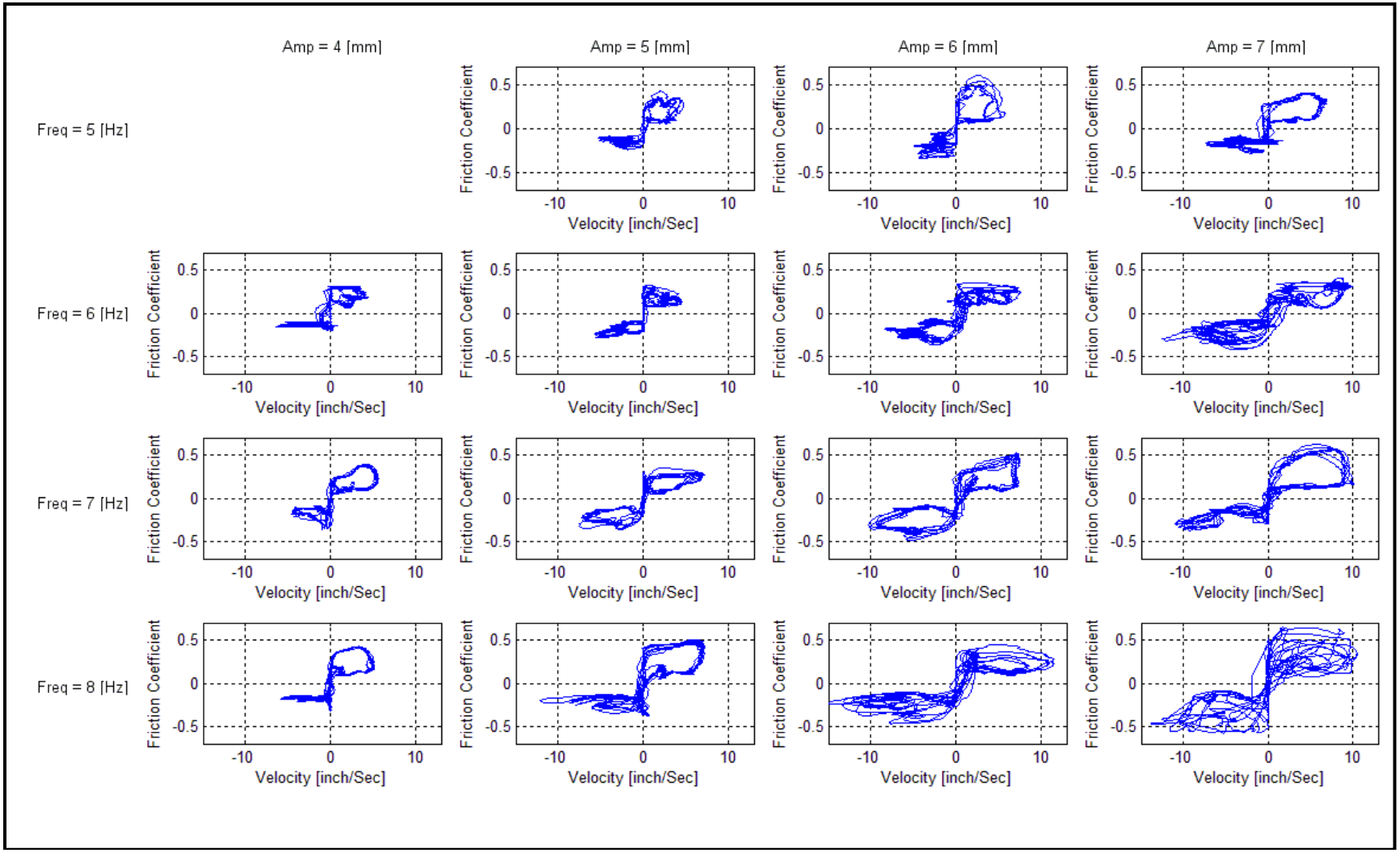


Figure 3.17 Experimental results for Aluminum against Aluminum, with 75g load and WD40 as a lubricant.

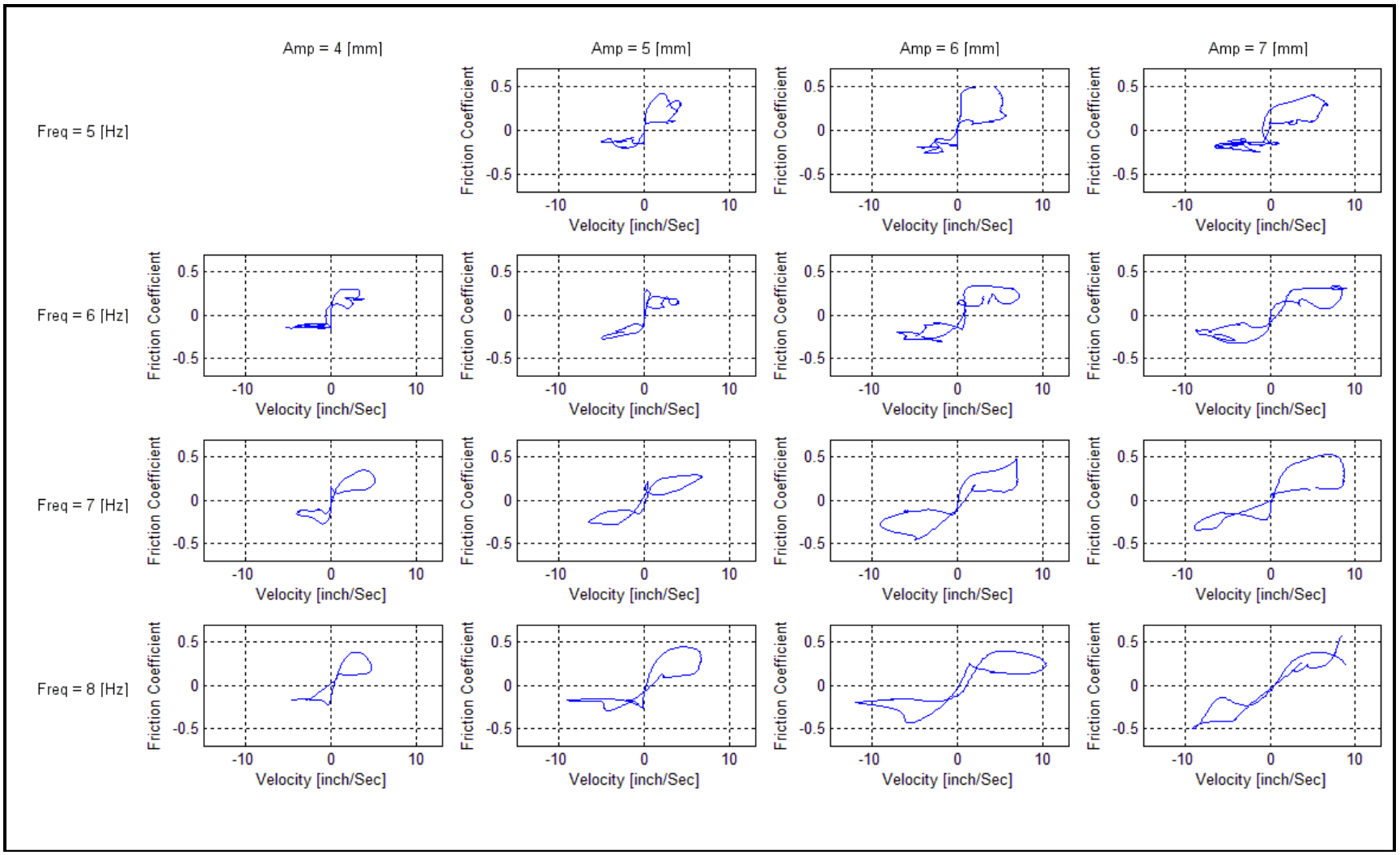


Figure 3.18 Averaged experimental results for Aluminum against Aluminum, with 75g load and WD40 as a lubricant.

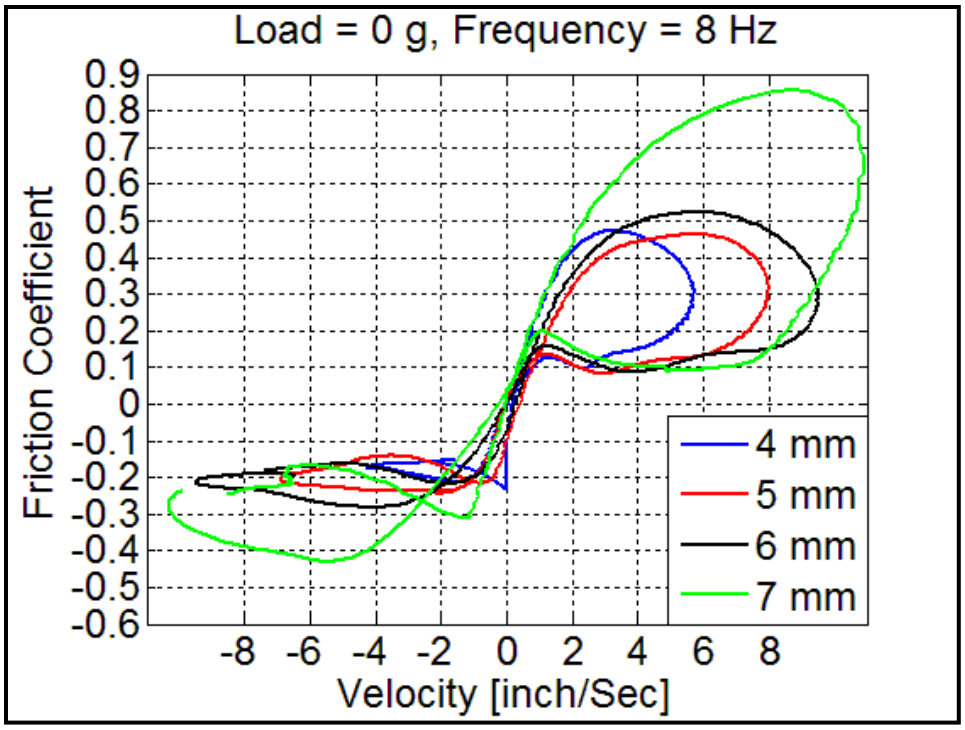


Figure 3.19 Aluminum – Aluminum with WD40 as a lubricant, with no load, frequency = 8 Hz and different amplitudes.

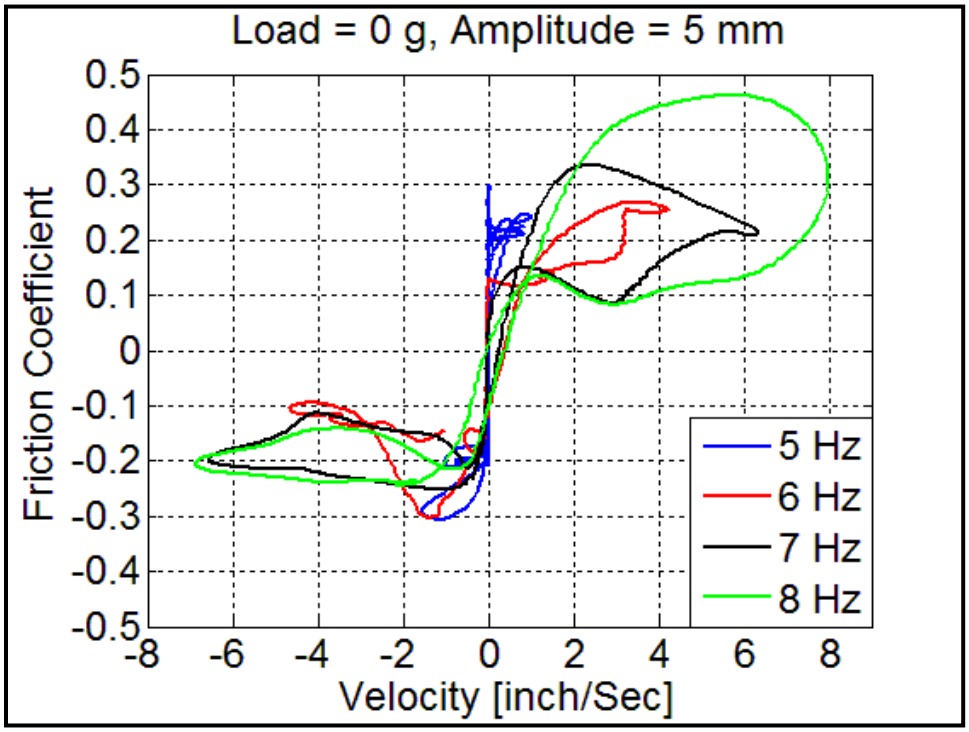


Figure 3.20 Aluminum – Aluminum with WD40 as a lubricant, with no load, amplitude = 5 mm and different frequencies.

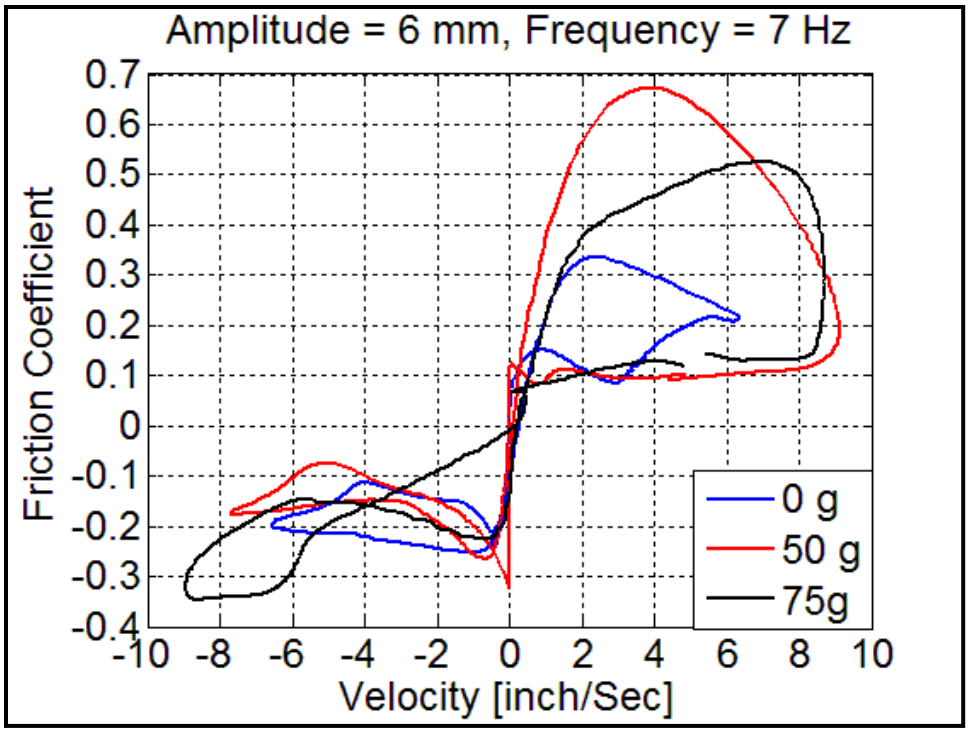


Figure 3.21 Aluminum – Aluminum with WD40 as a lubricant, with different loads, amplitude = 6 mm, frequency = 7 Hz.

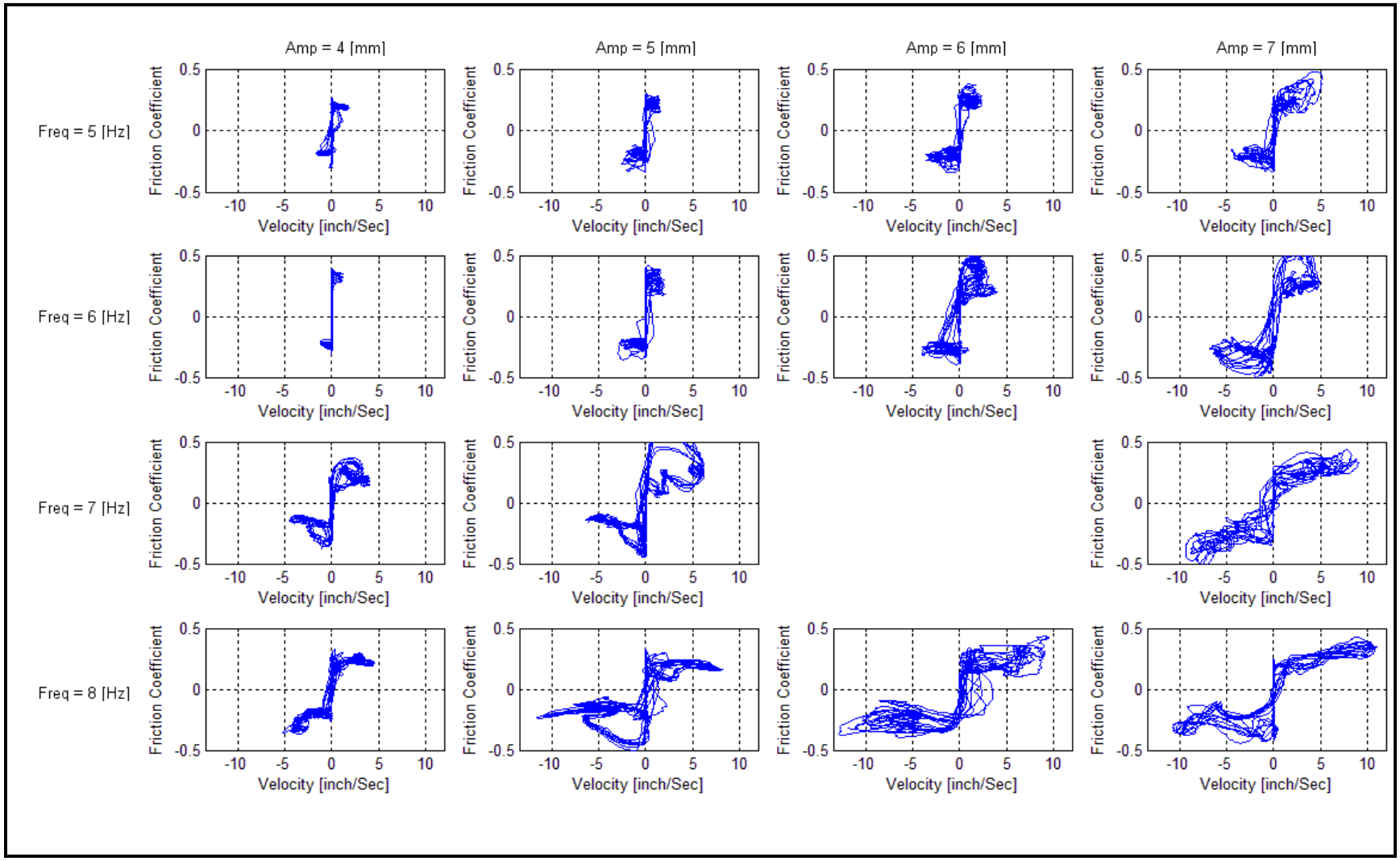


Figure 3.22 Experimental results for Aluminum against Aluminum with no load and Silicon Spray as a lubricant.

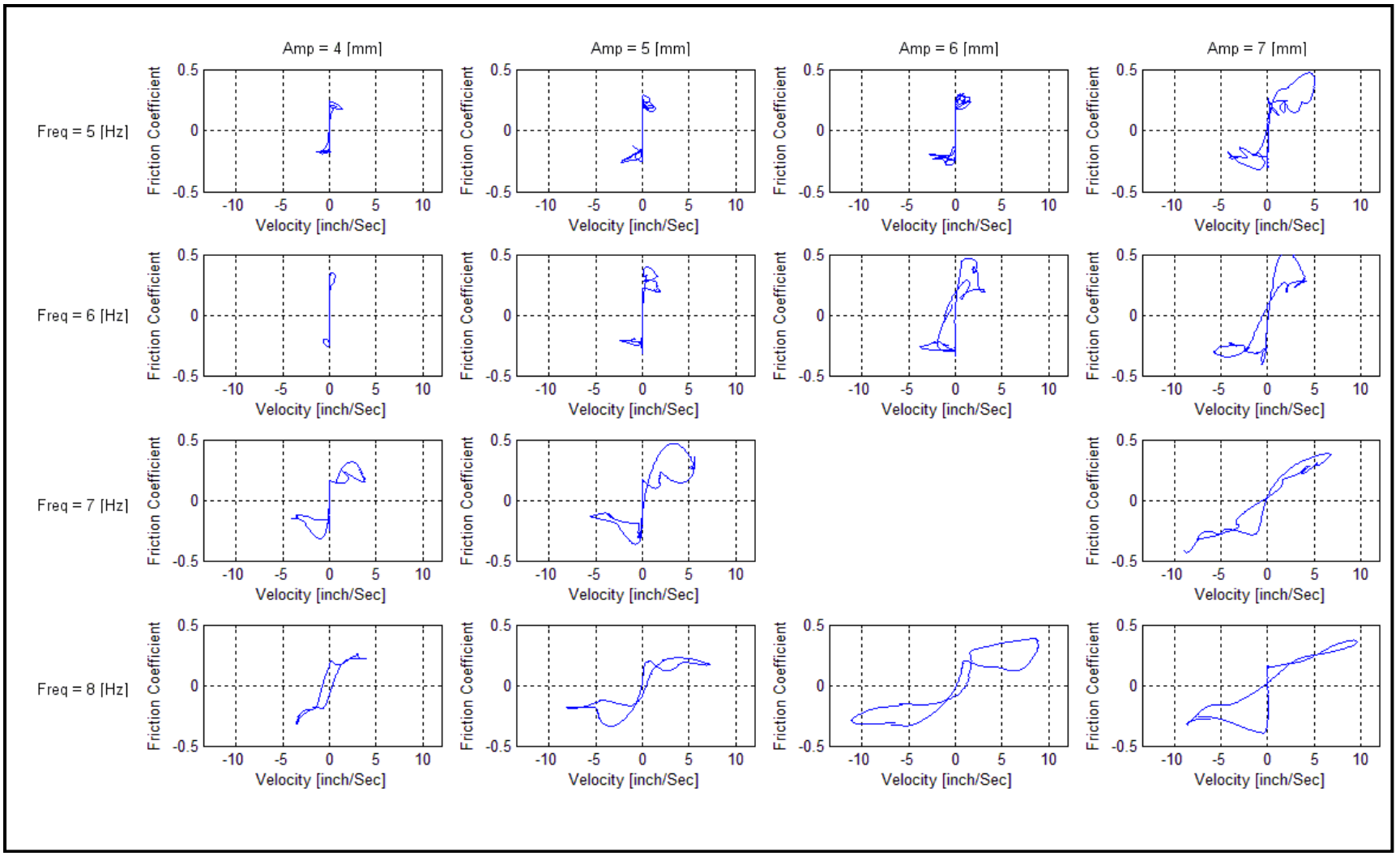


Figure 3.23 Averaged experimental results for Aluminum against Aluminum, with no load and Silicon Spray as a lubricant.

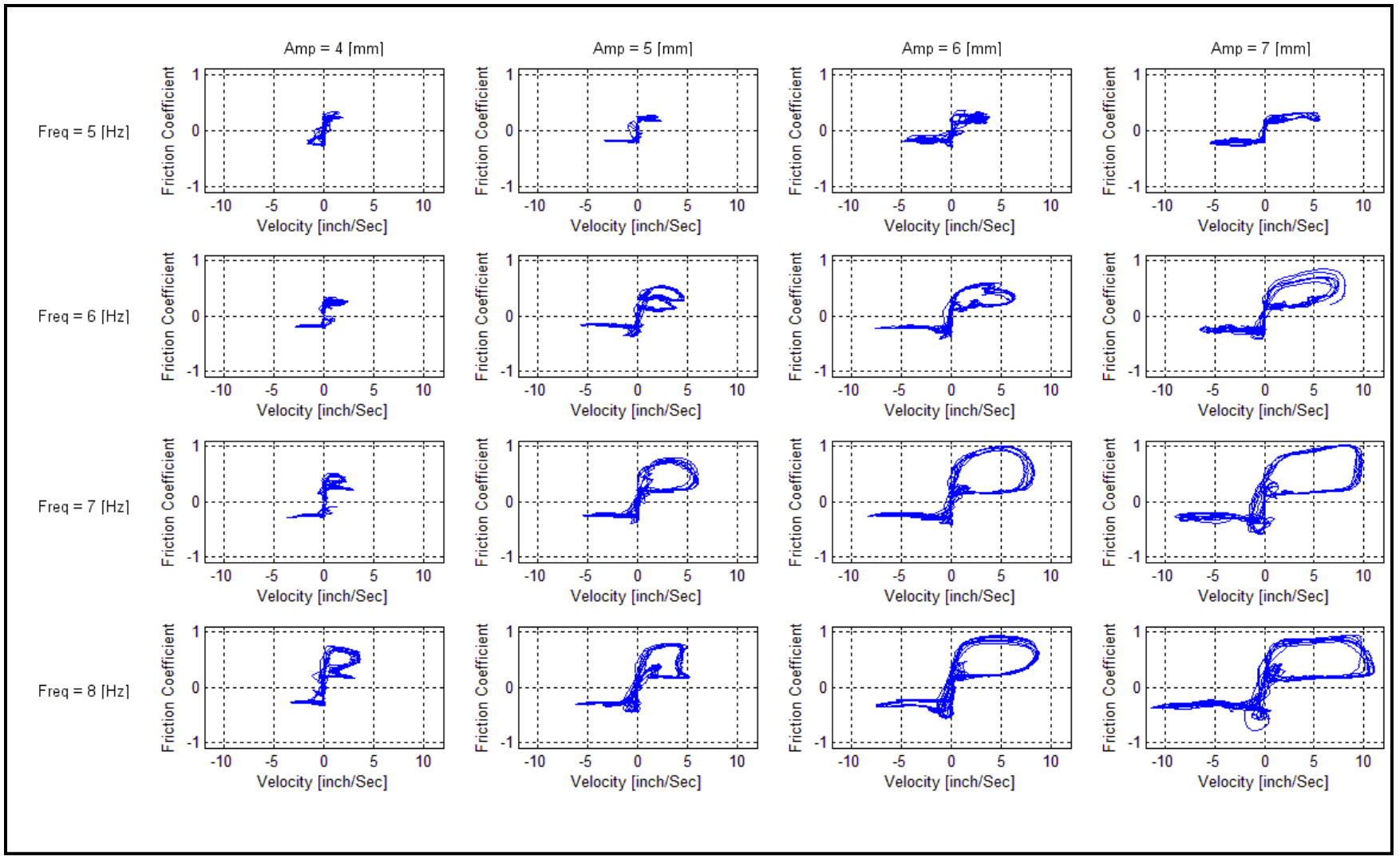


Figure 3.24 Experimental results for Aluminum against Aluminum, with 50g load and Silicon Spray as a lubricant.

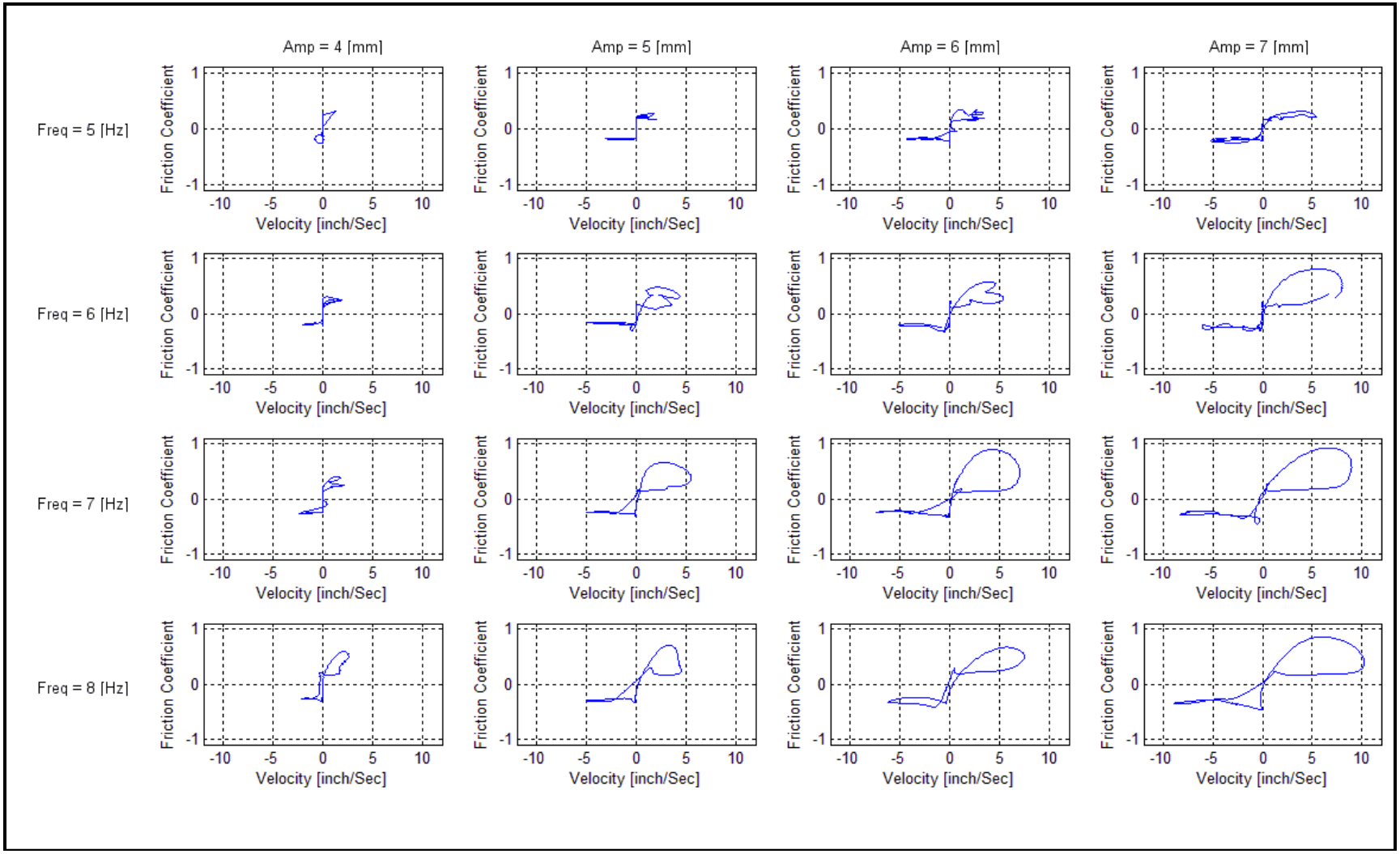


Figure 3.25 Averaged experimental results for Aluminum against Aluminum, with 50g load and Silicon Spray as a lubricant.

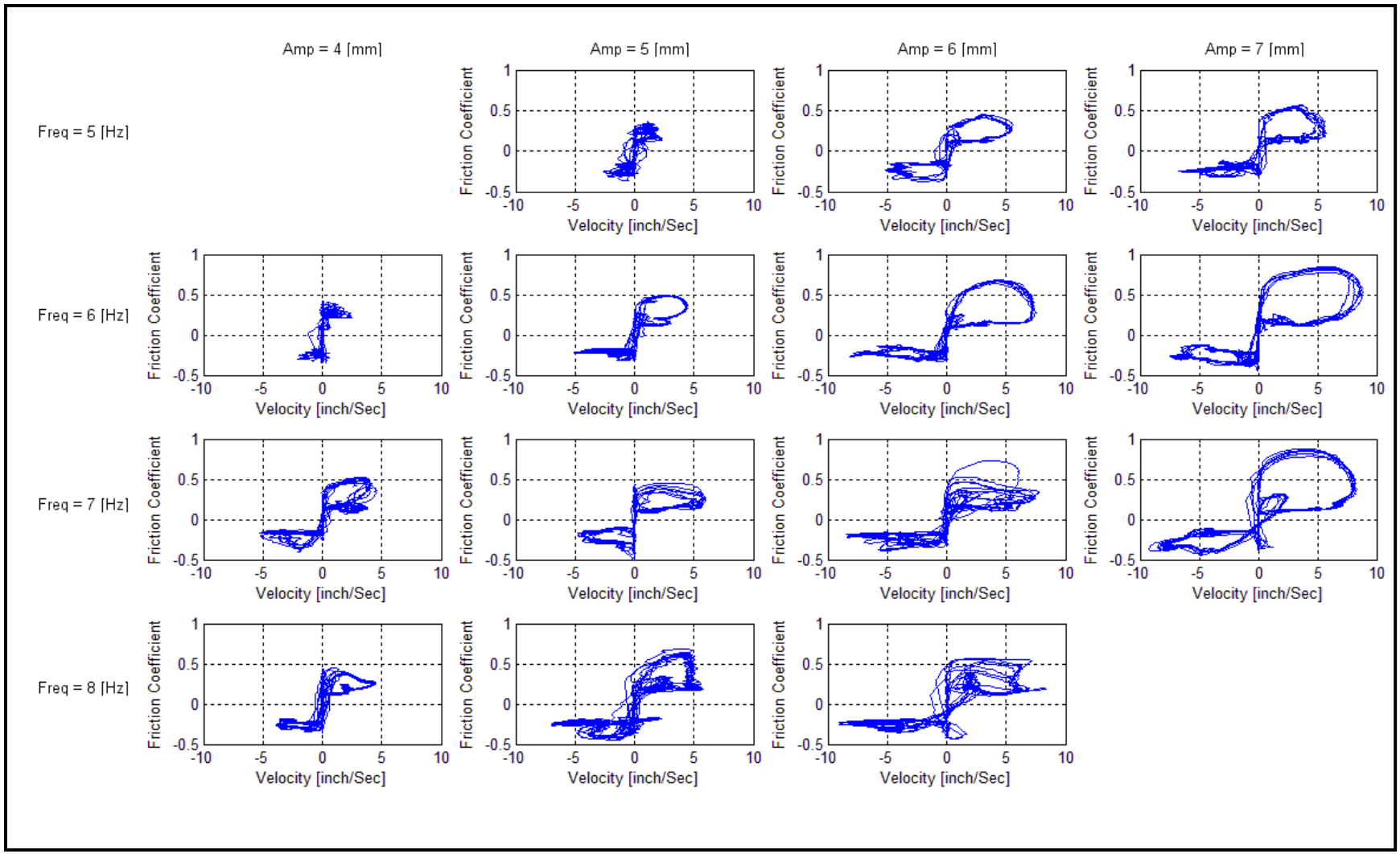


Figure 3.26 Experimental results for Aluminum against Aluminum, with 75g load and Silicon Spray as a lubricant.

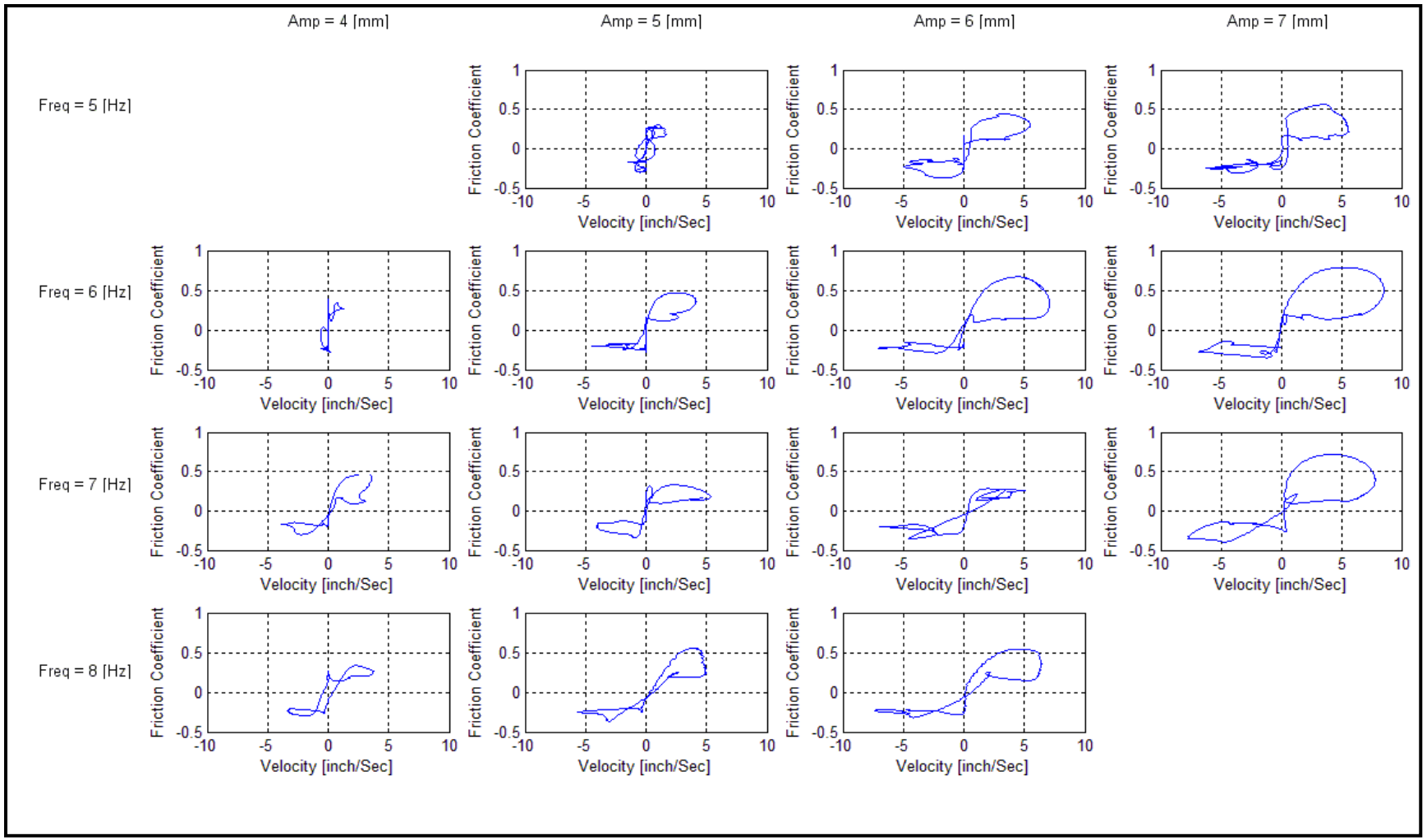


Figure 3.27 Averaged experimental results for Aluminum against Aluminum, with 75g load and Silicon Spray as a lubricant.

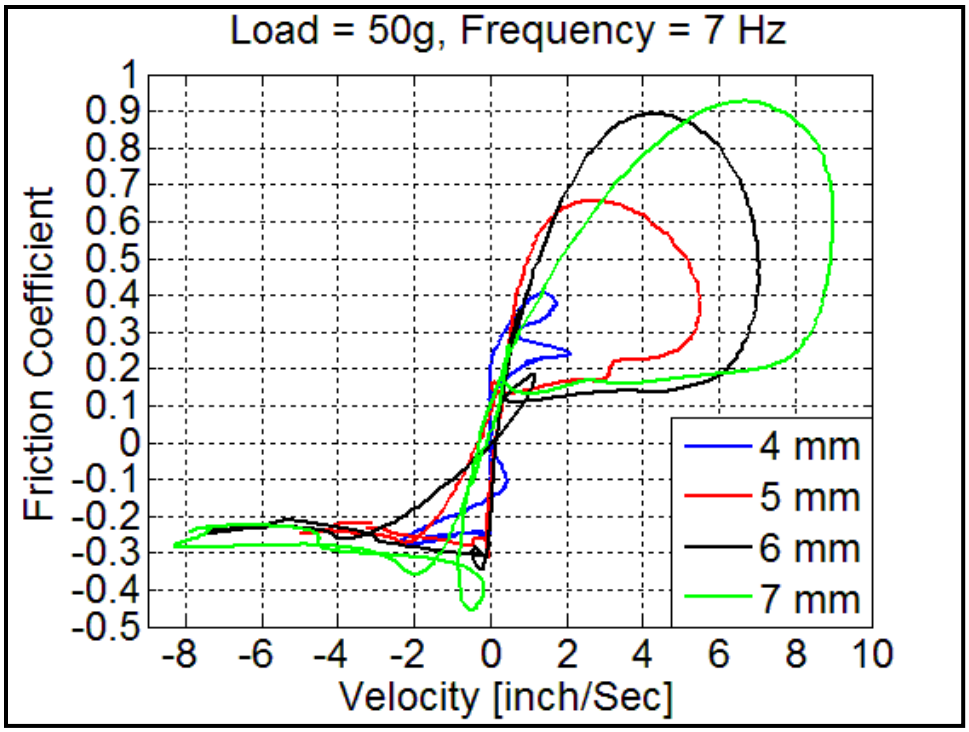


Figure 3.28 Aluminum – Aluminum with Silicon Spray as a lubricant, for with 50g load, frequency = 7 Hz and different amplitudes.

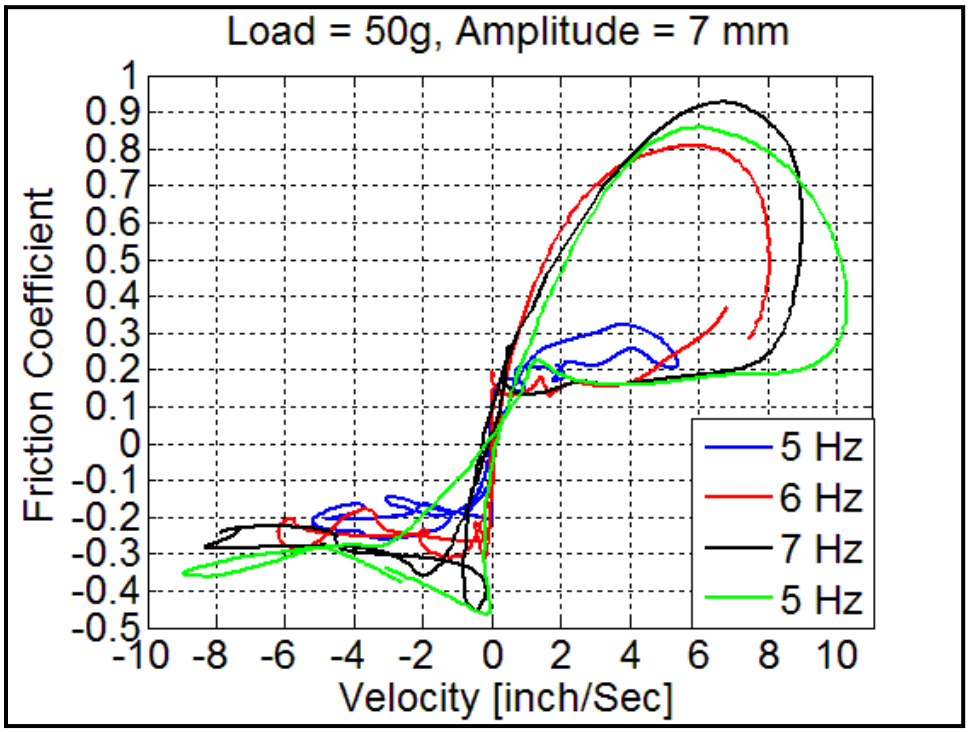


Figure 3.29 Aluminum – Aluminum with Silicon Spray as a lubricant, with 50g load, amplitude = 7 mm and different frequencies.

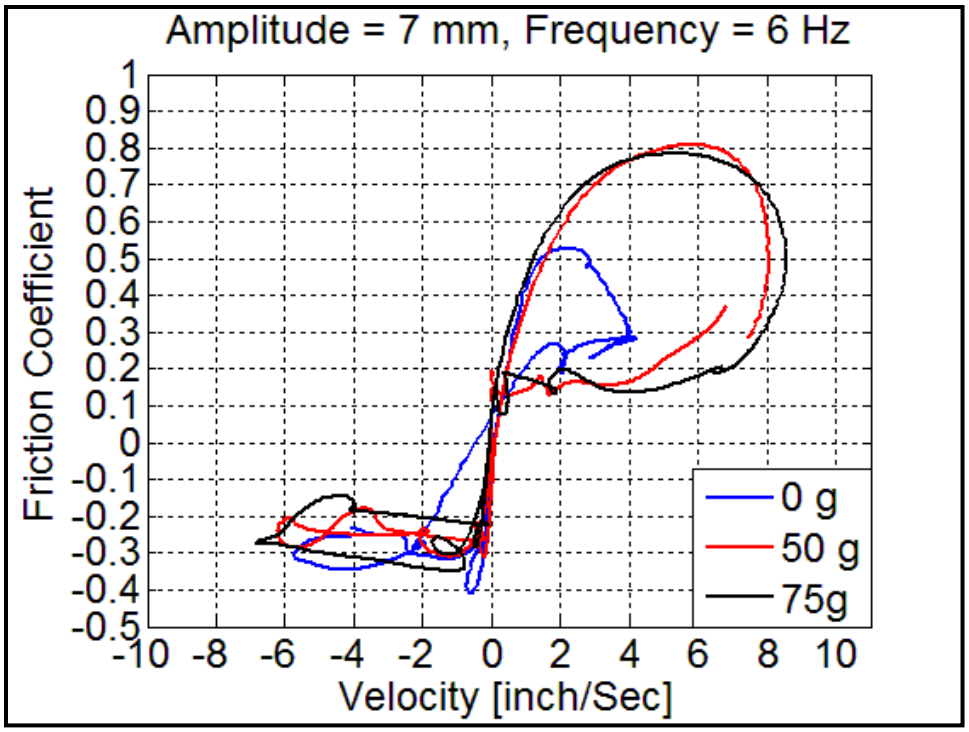


Figure 3.30 Aluminum – Aluminum with Silicon Spray as a lubricant, with different loads, amplitude = 7 mm, frequency = 6 Hz.

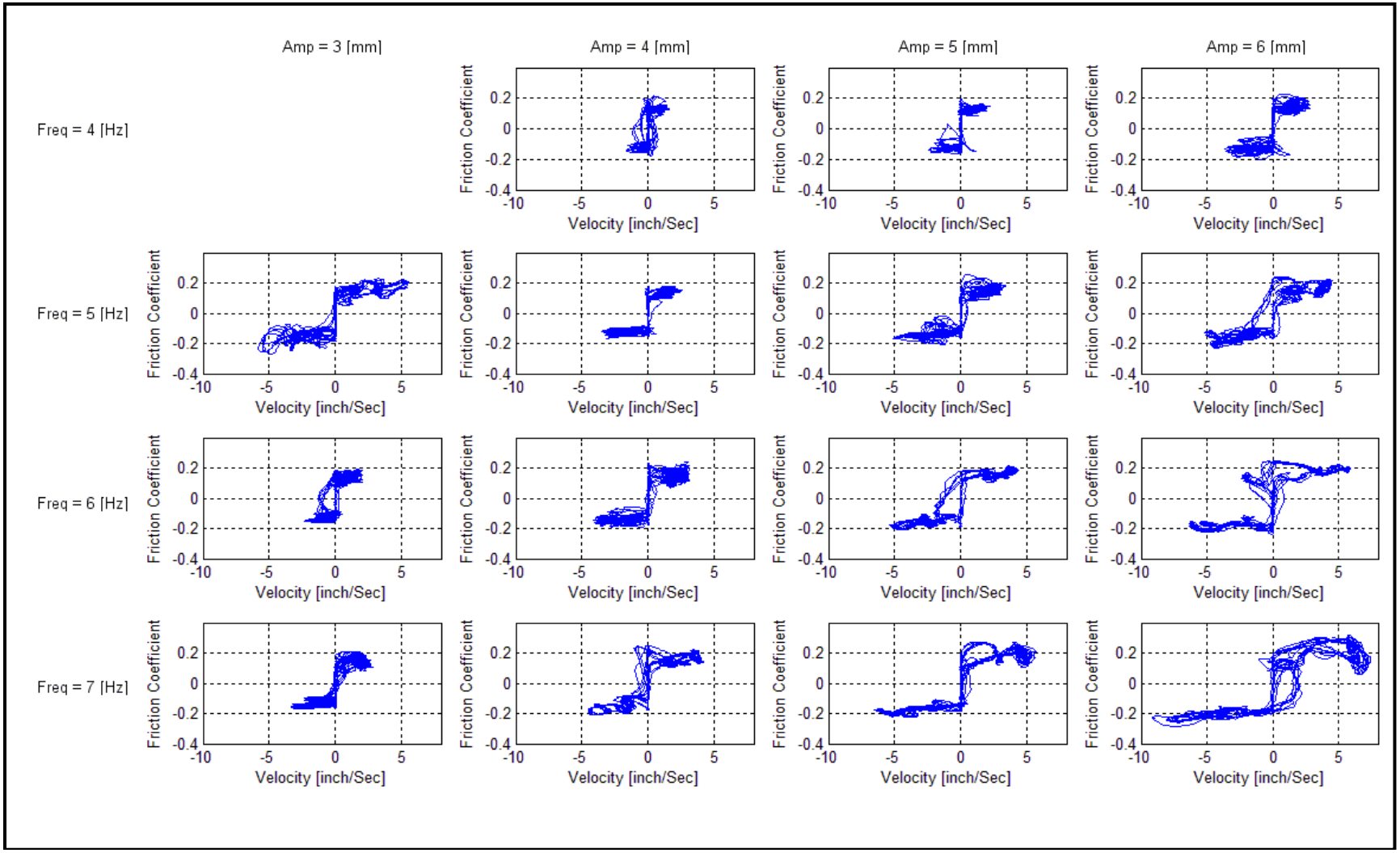


Figure 3.31 Experimental results for Teflon against Teflon with no load.

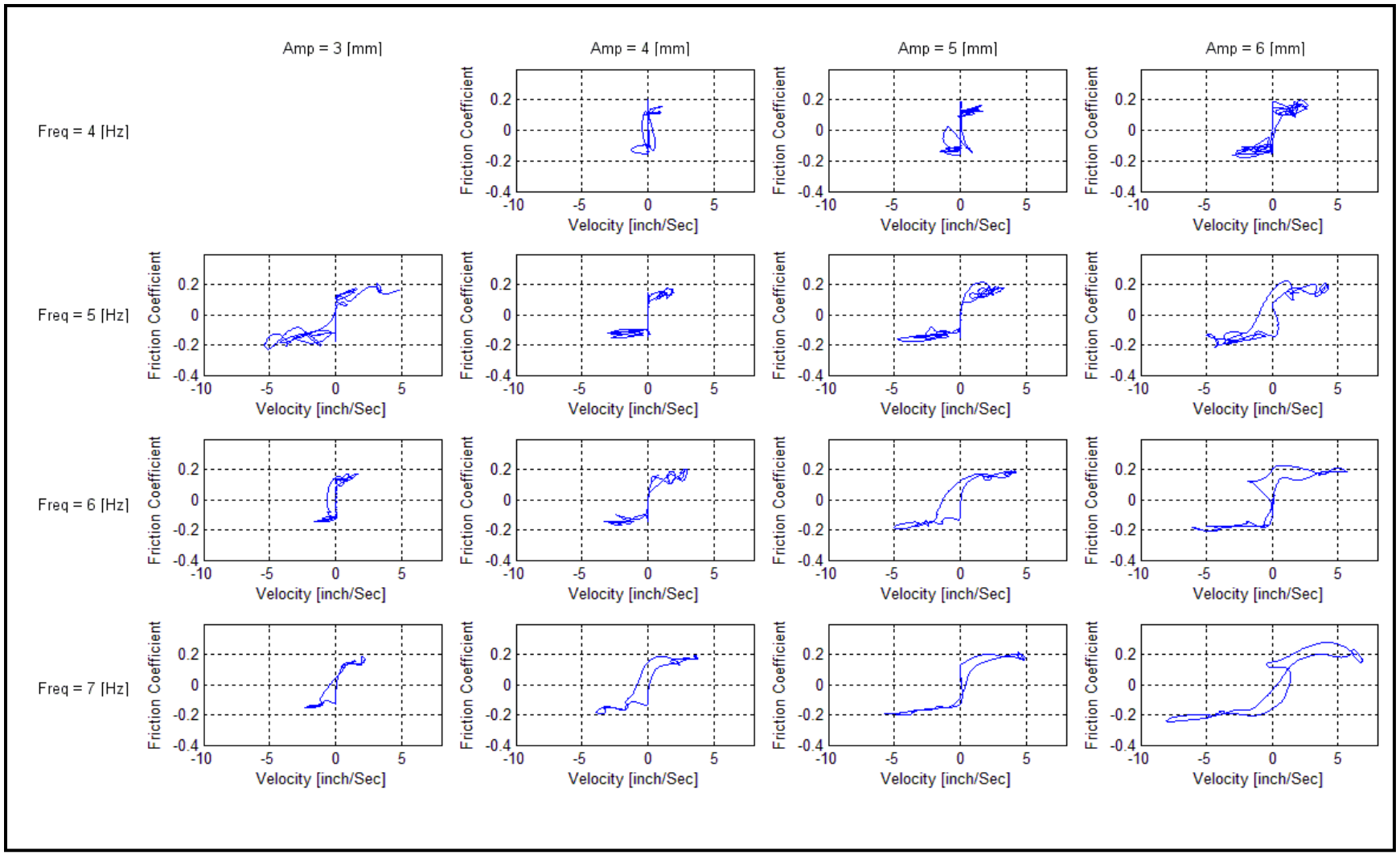


Figure 3.32 Averaged experimental results for Teflon against Teflon with no load.

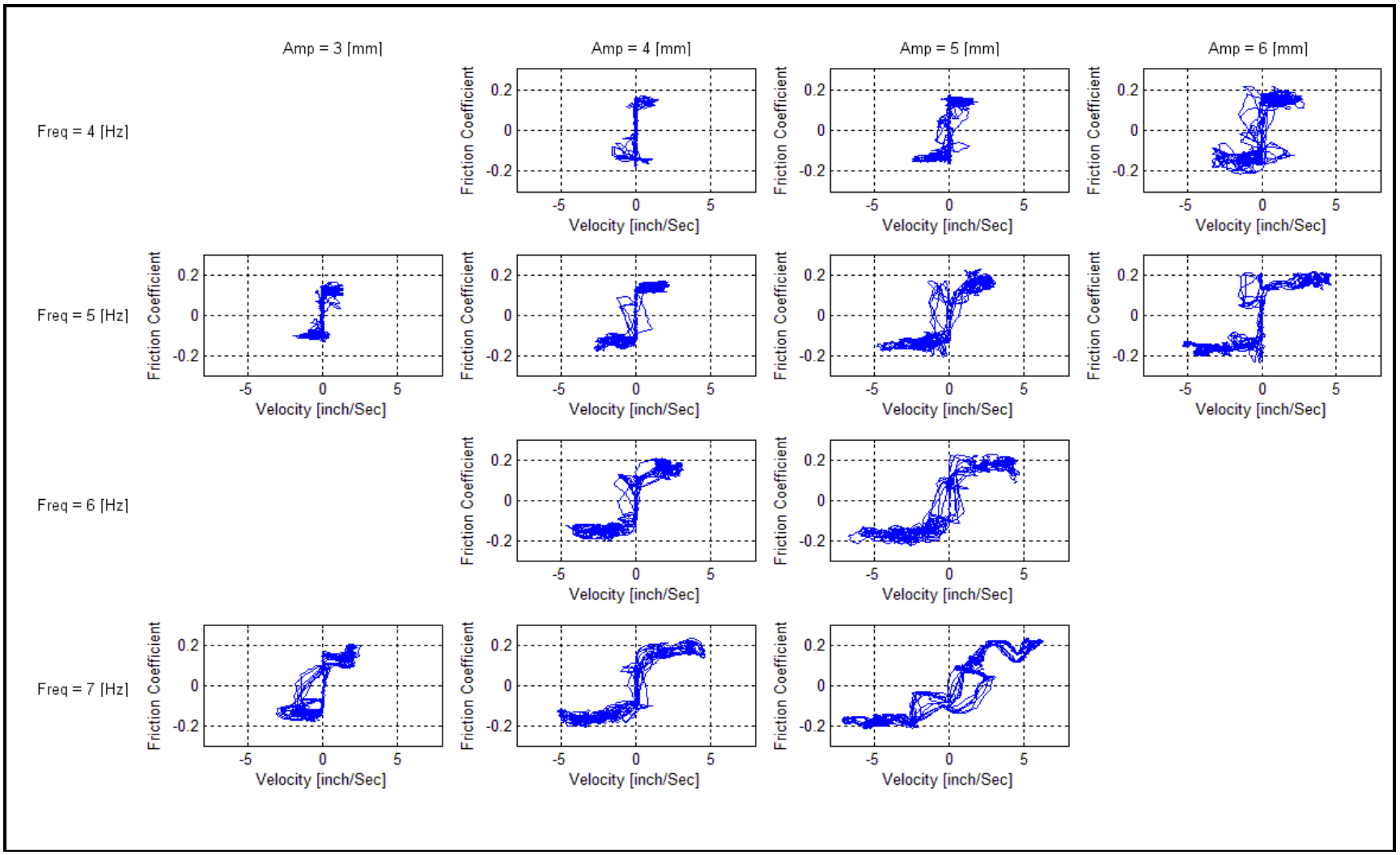


Figure 3.33 Experimental results for Teflon against Teflon with 50g load.

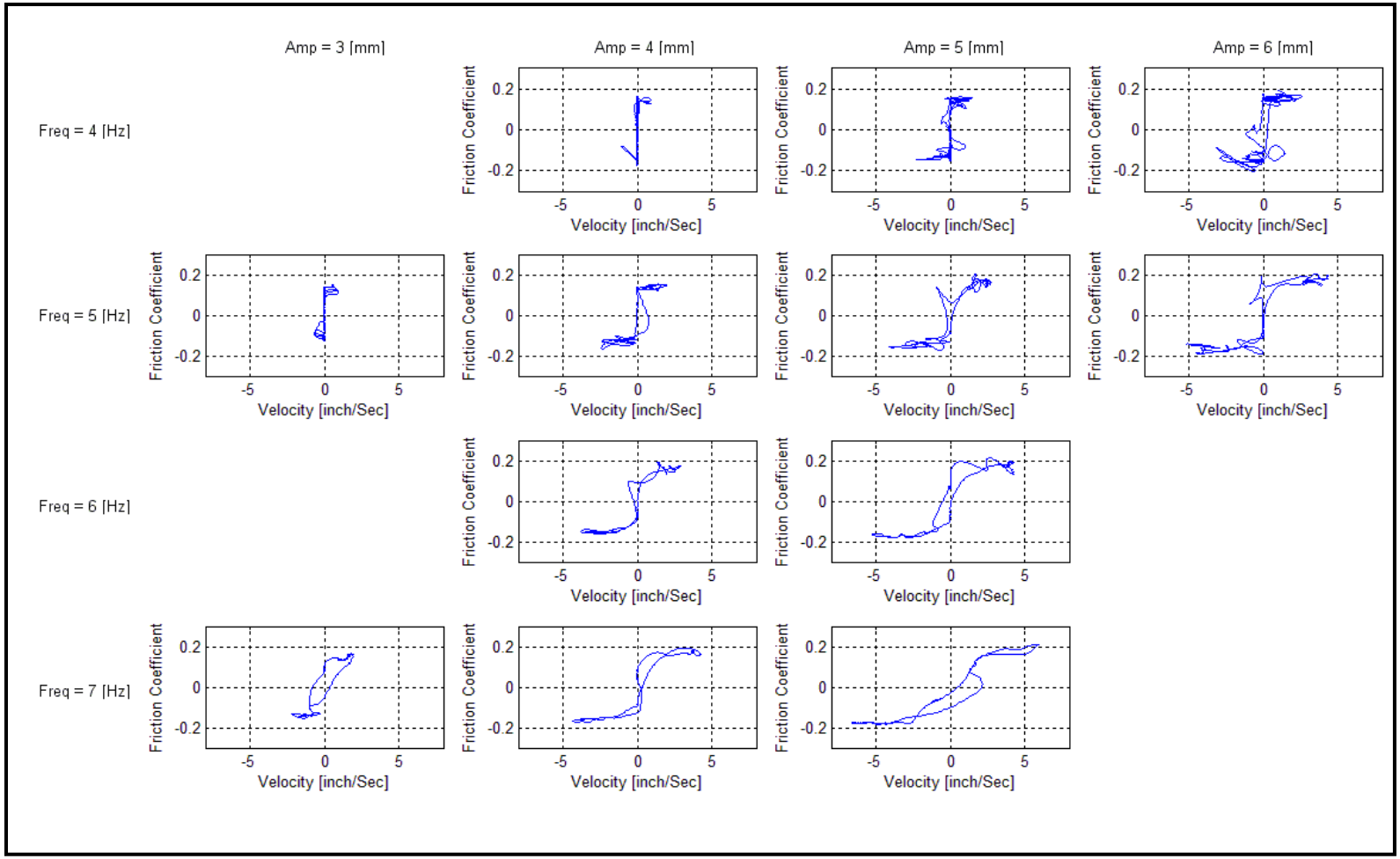


Figure 3.34 Averaged experimental results for Teflon against Teflon with 50g load.

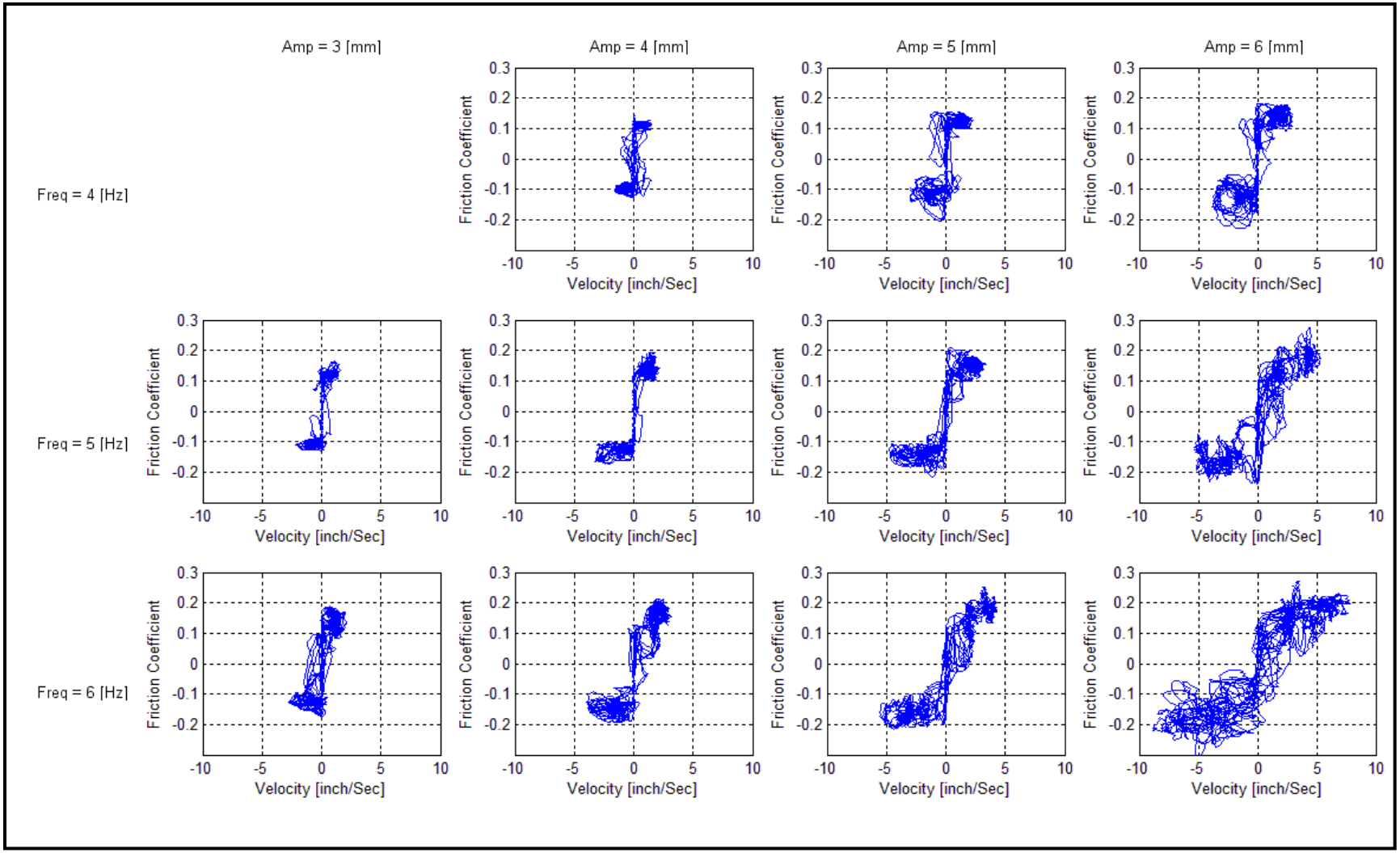


Figure 3.35 Experimental results for Teflon against Teflon with 75g load.

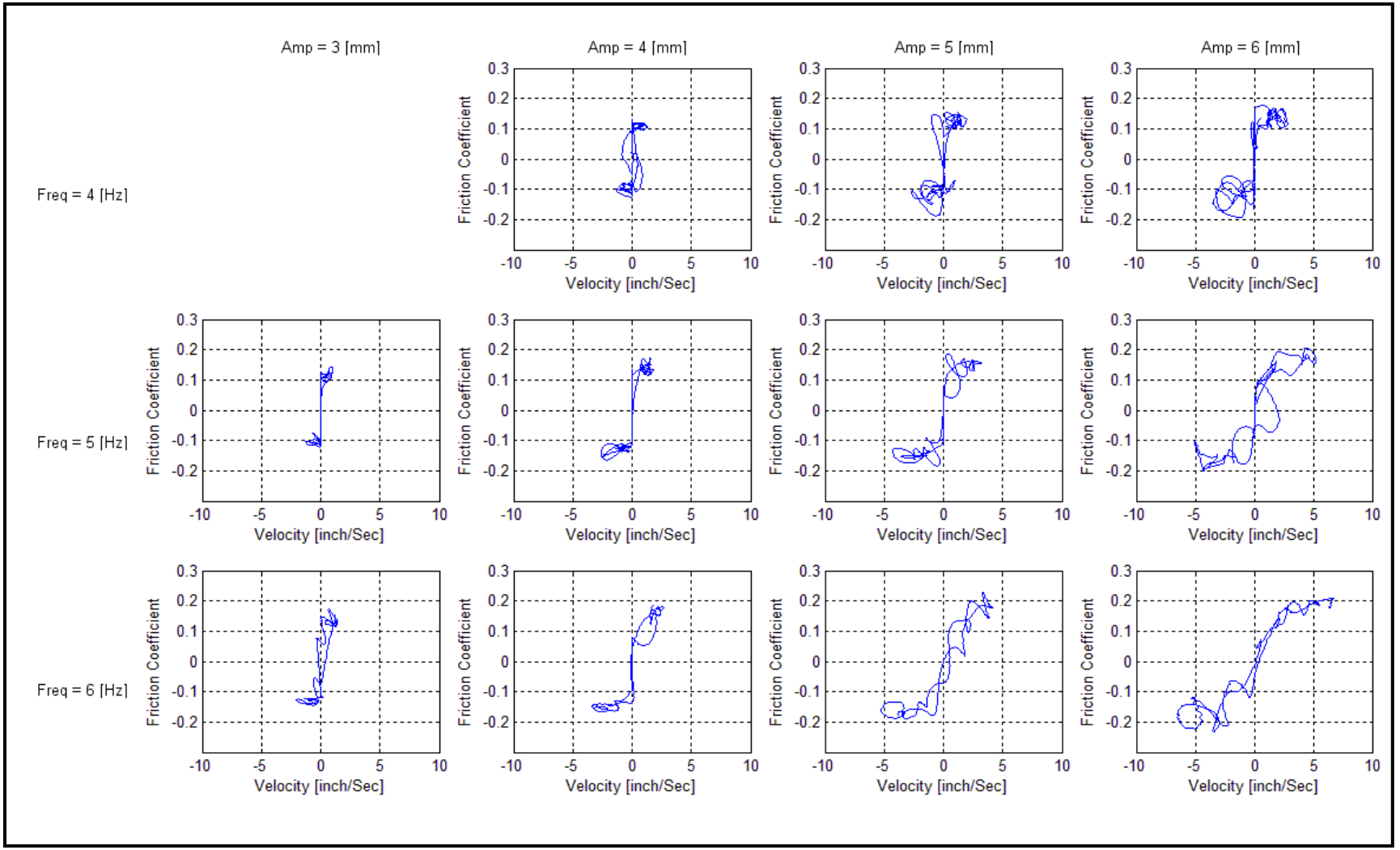


Figure 3.36 Averaged experimental results for Teflon against Teflon with 75g load.

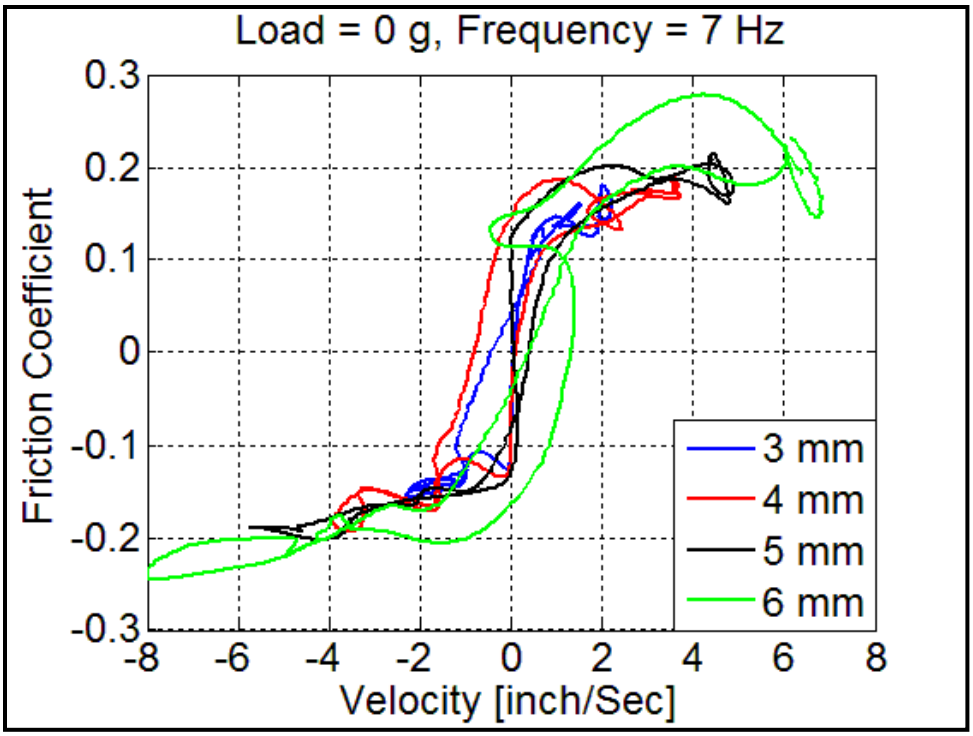


Figure 3.37 Teflon – Teflon with No load, frequency = 7 Hz, and different amplitudes.

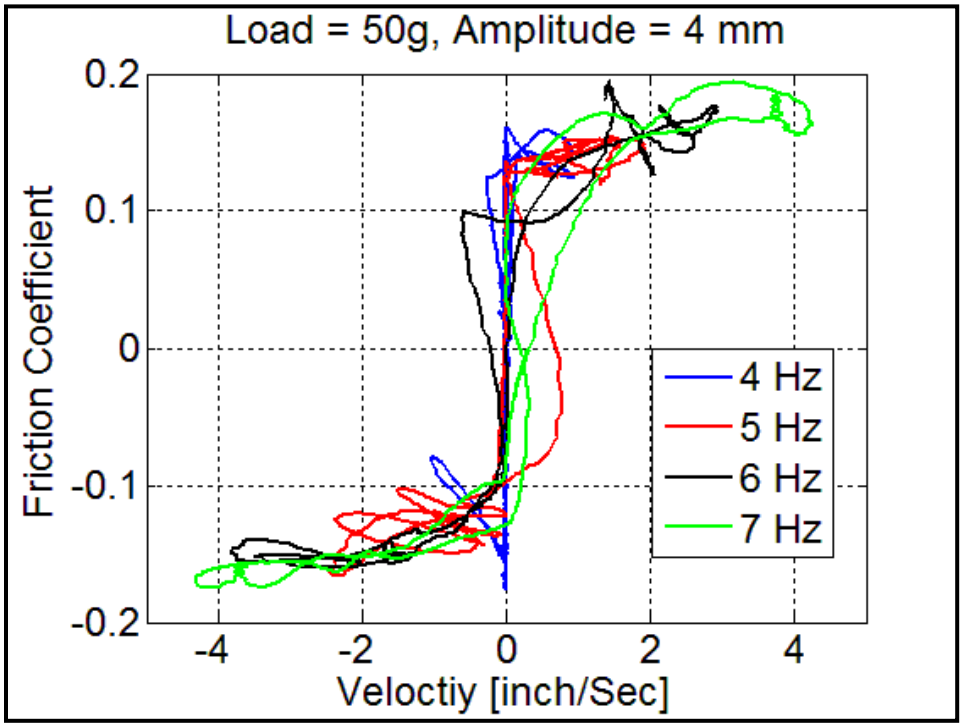


Figure 3.38 Teflon – Teflon with a 50g load, amplitude = 4 mm and different frequencies.

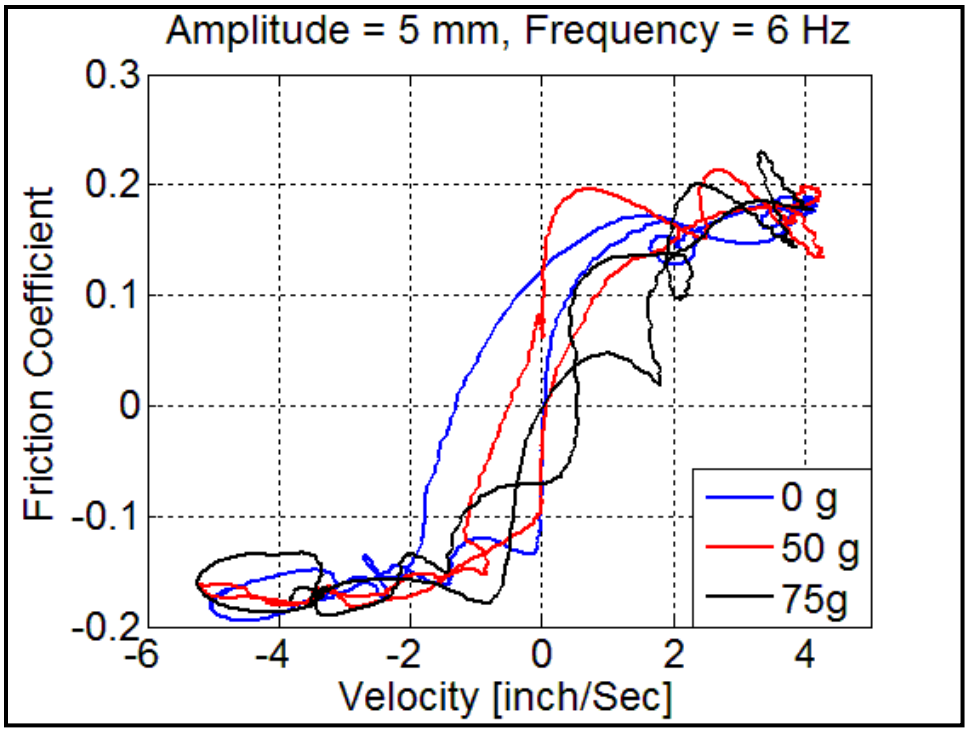


Figure 3.39 Teflon – Teflon with different loads, amplitude = 5 mm and frequency = 6 Hz.

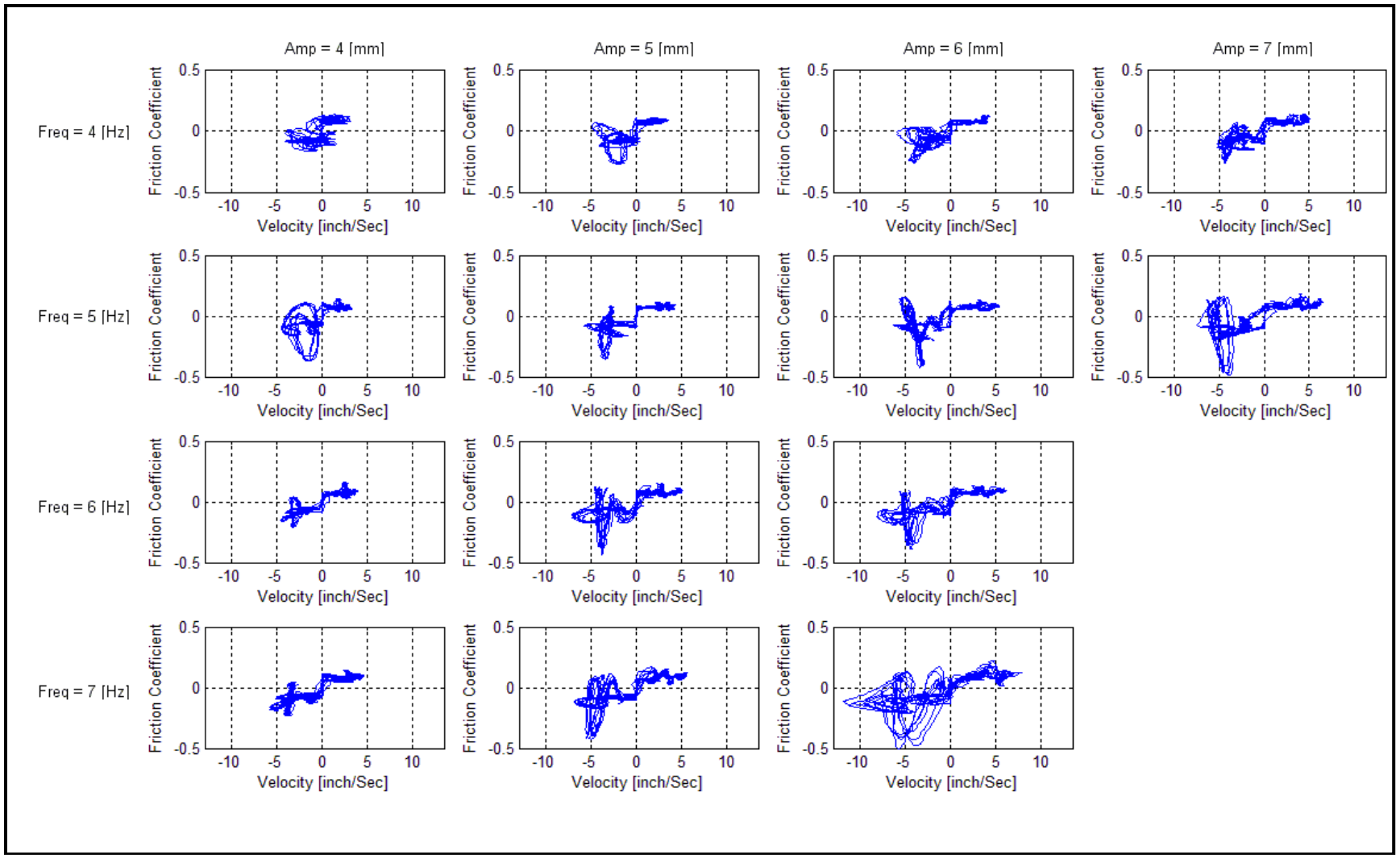


Figure 3.40 Experimental results for Aluminum against Teflon with no load.

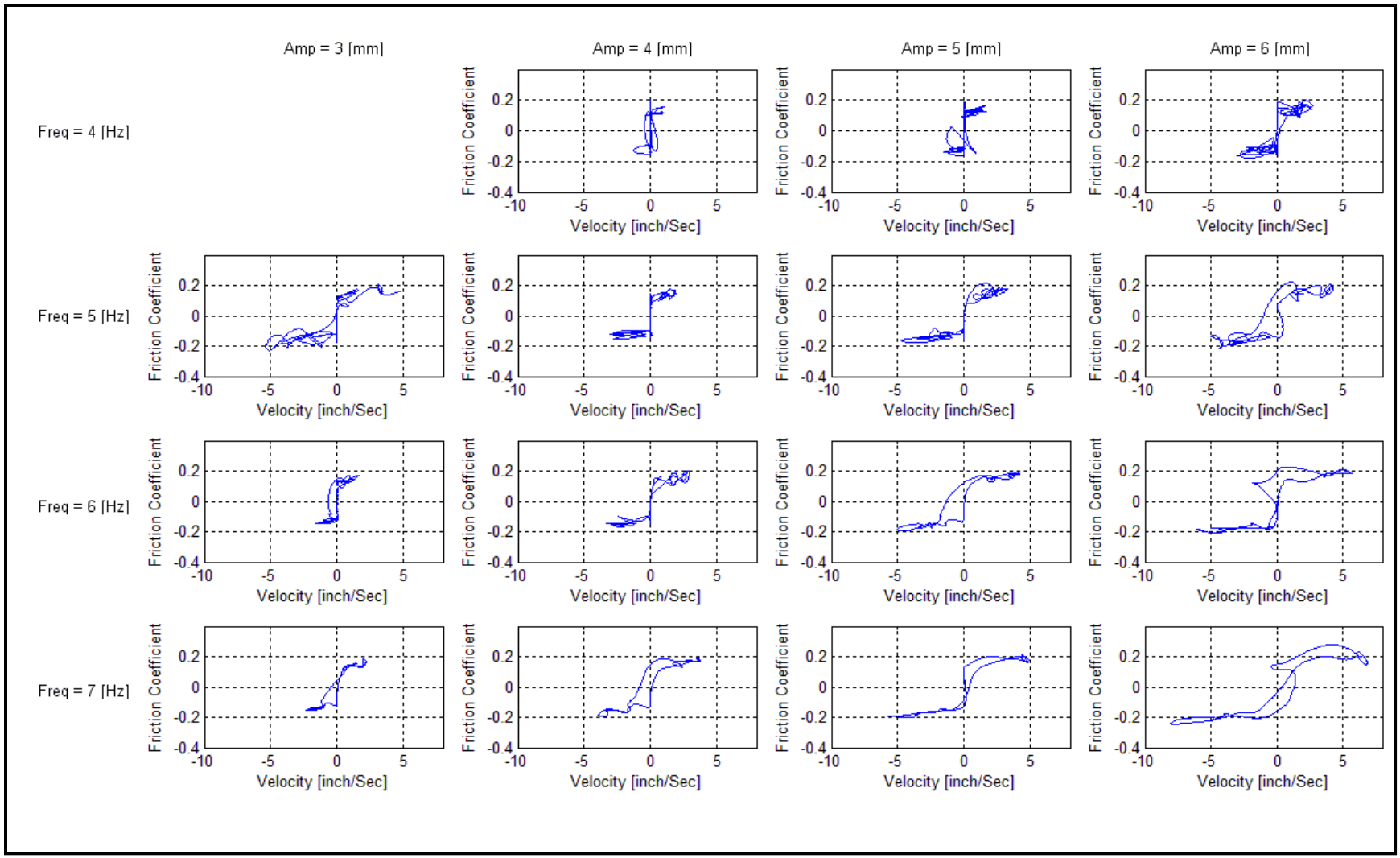


Figure 3.41 Averaged experimental results for Aluminum against Teflon with no load.

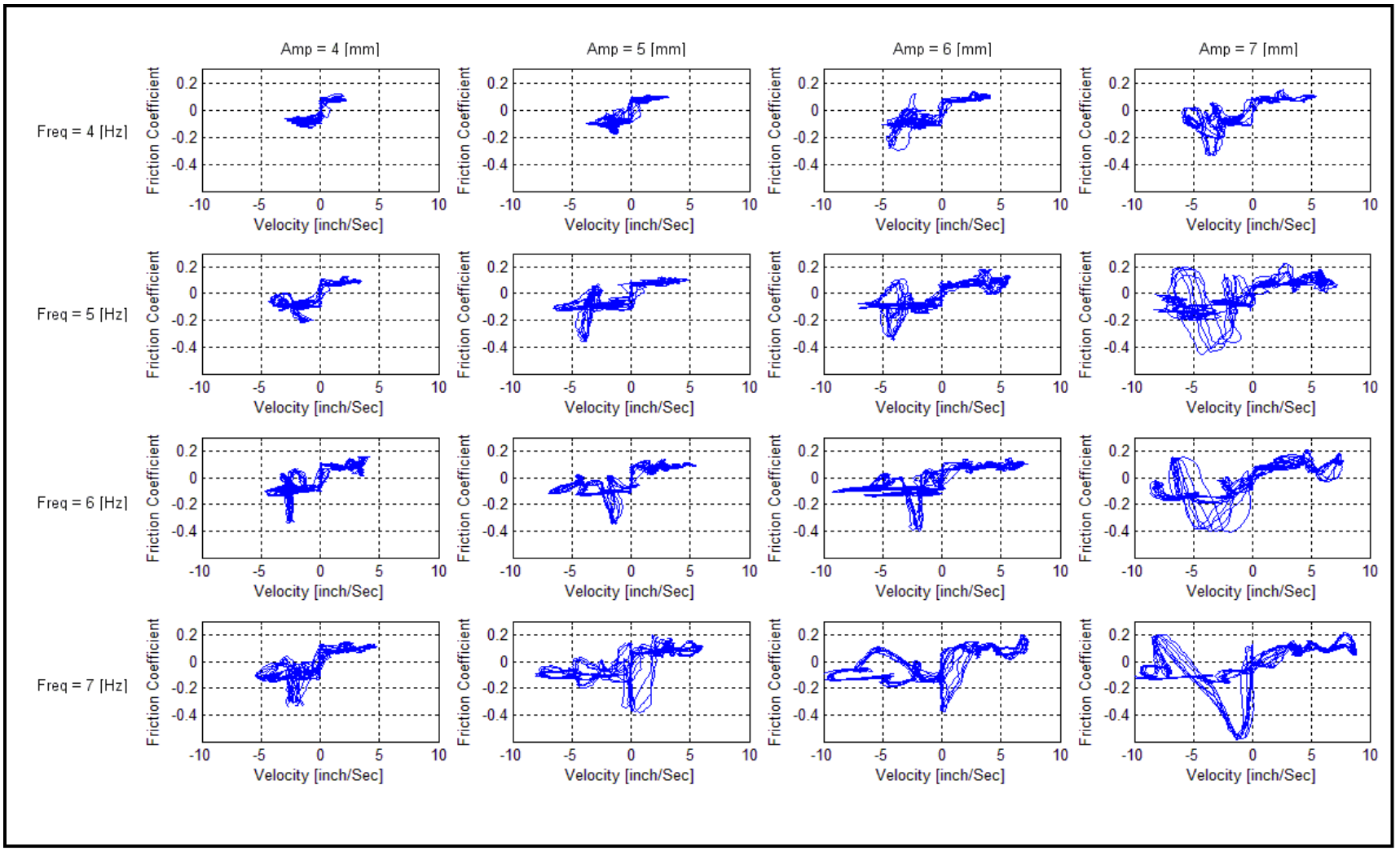


Figure 3.42 Experimental results for Aluminum against Teflon with 50g load.

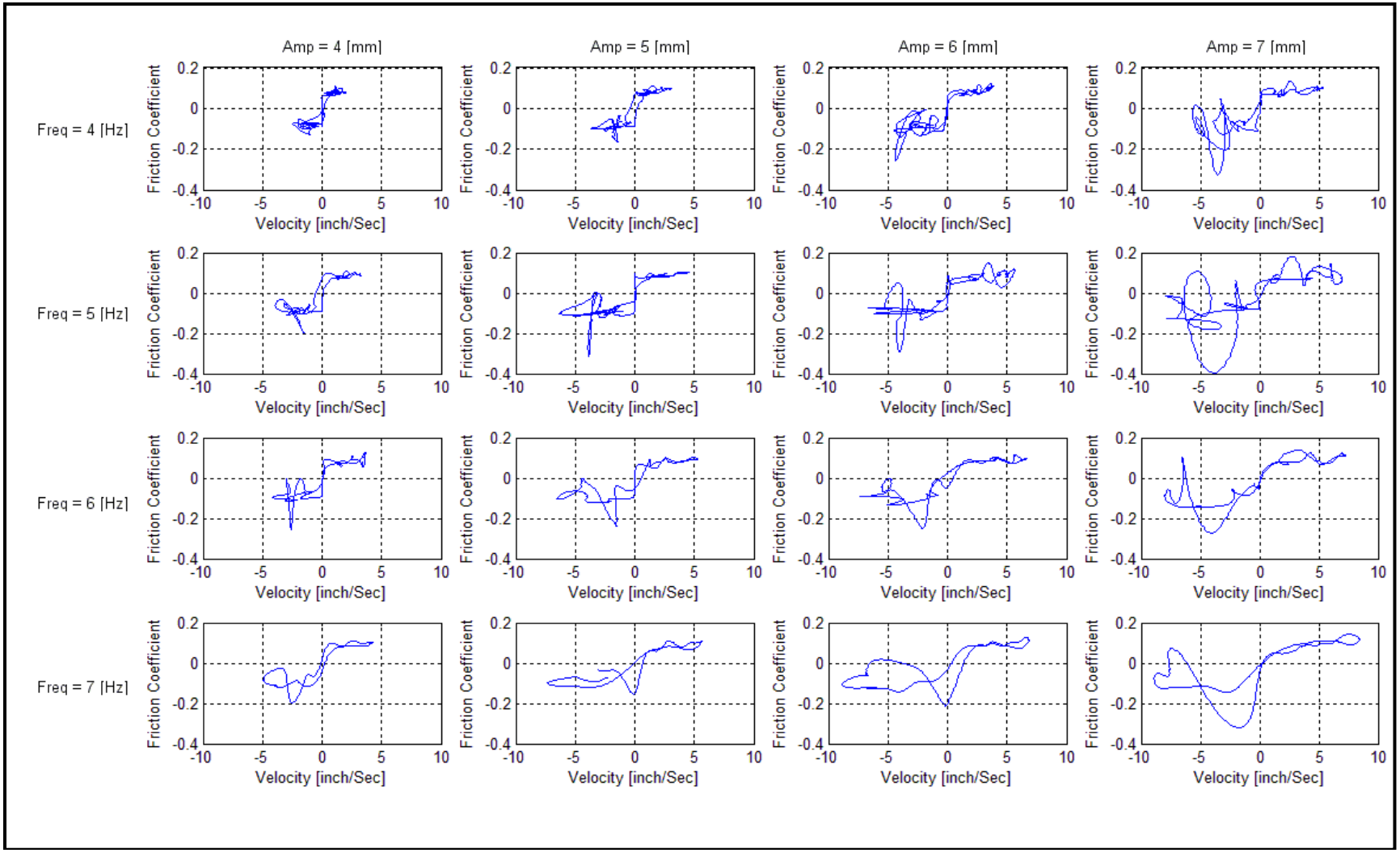


Figure 3.43 Averaged experimental results for Aluminum against Teflon with 50g load.

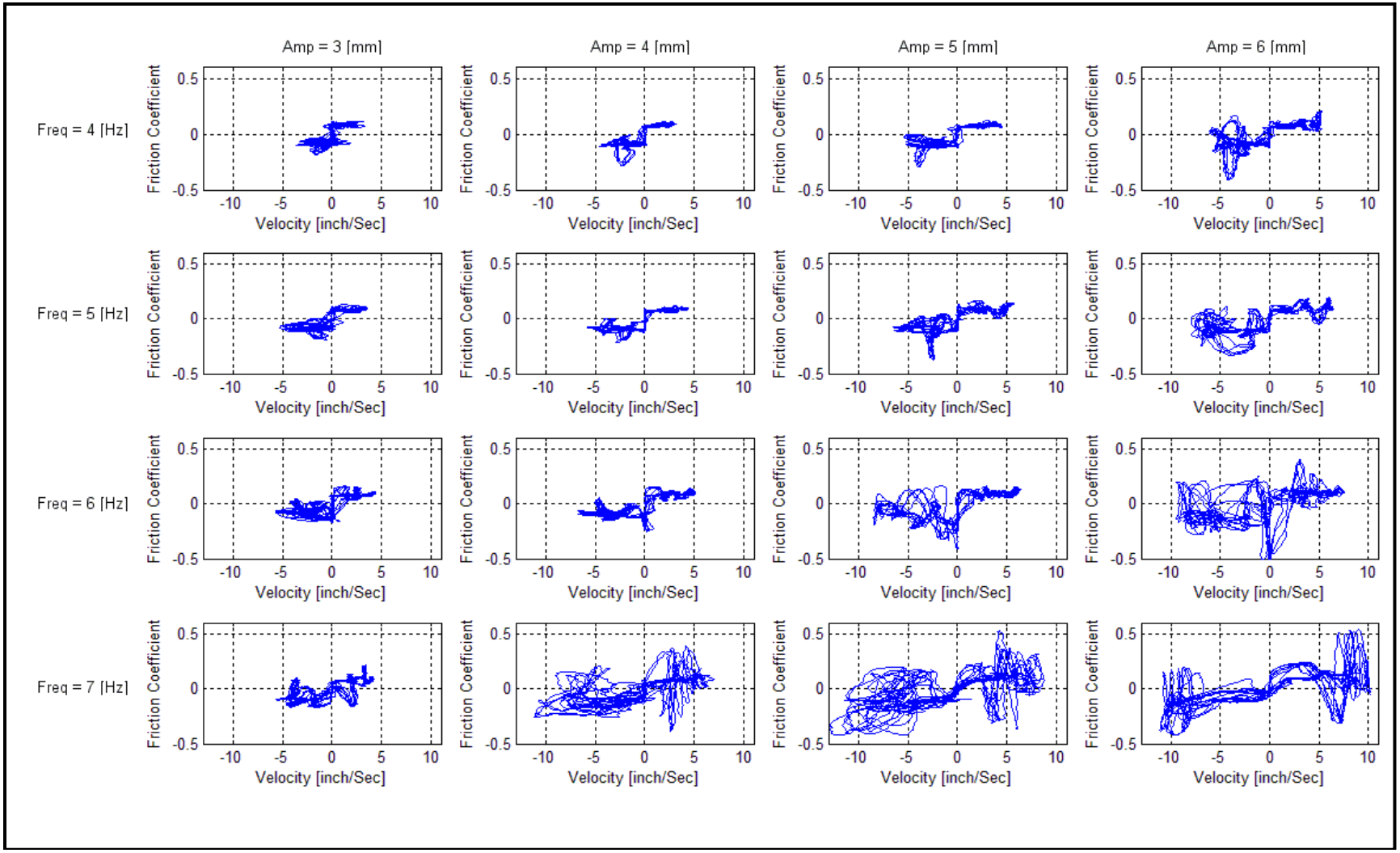


Figure 3.44 Experimental results for Aluminum against Teflon with 75g load.

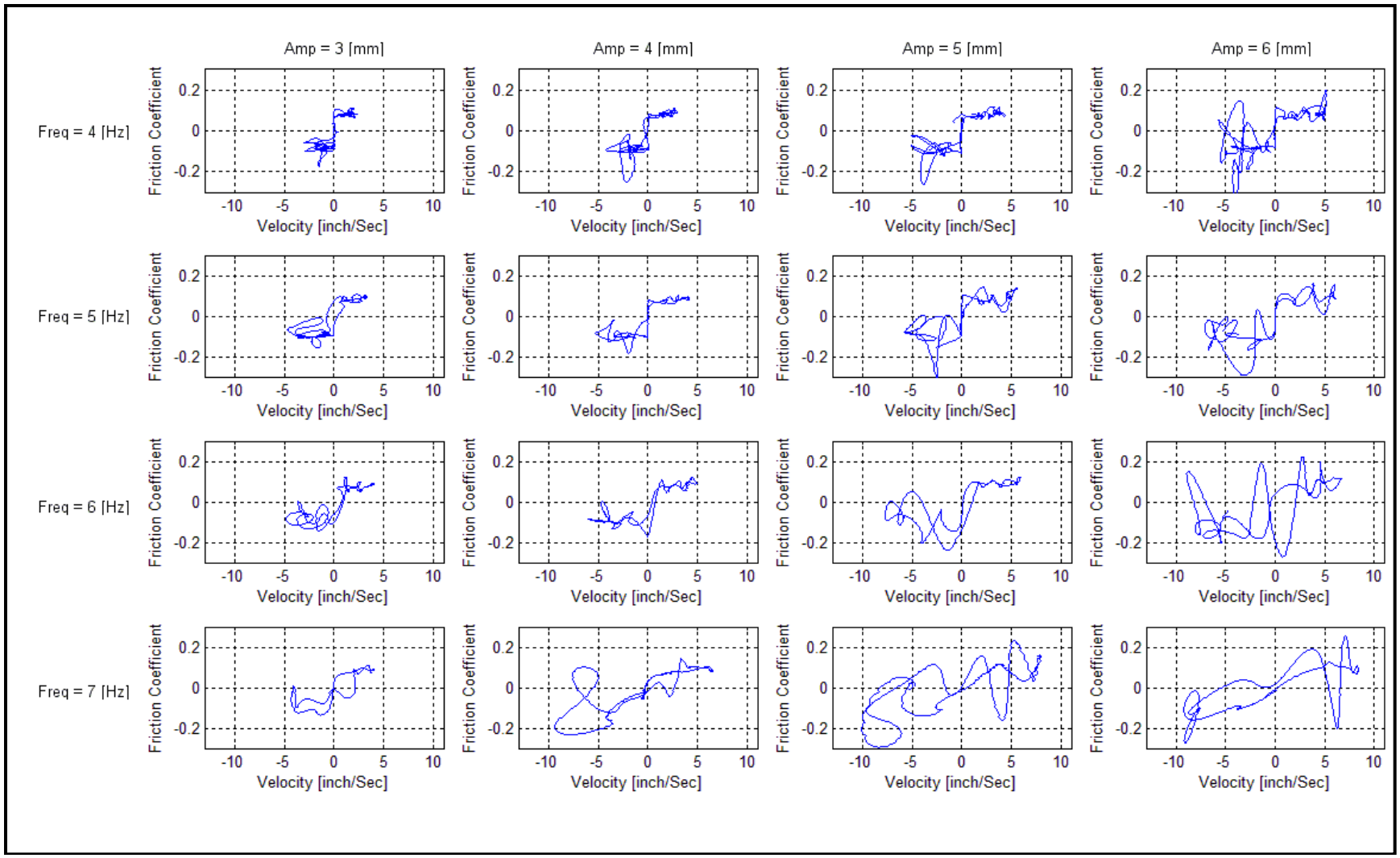


Figure 3.45 Averaged experimental results for Aluminum against Teflon with 75g load.

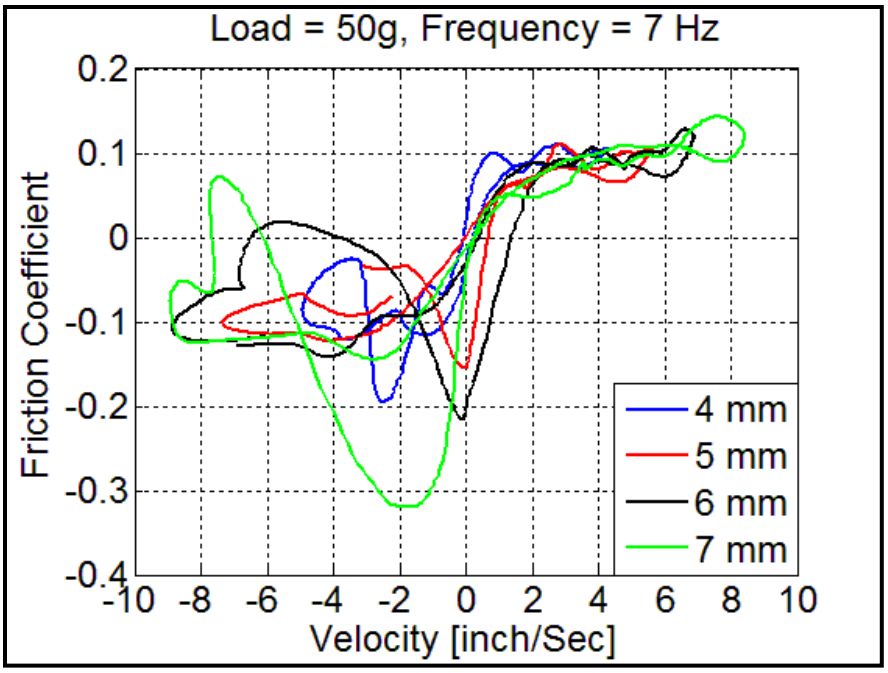


Figure 3.46 Aluminum - Teflon for a load of 50 g, frequency of 7 Hz and different amplitudes.

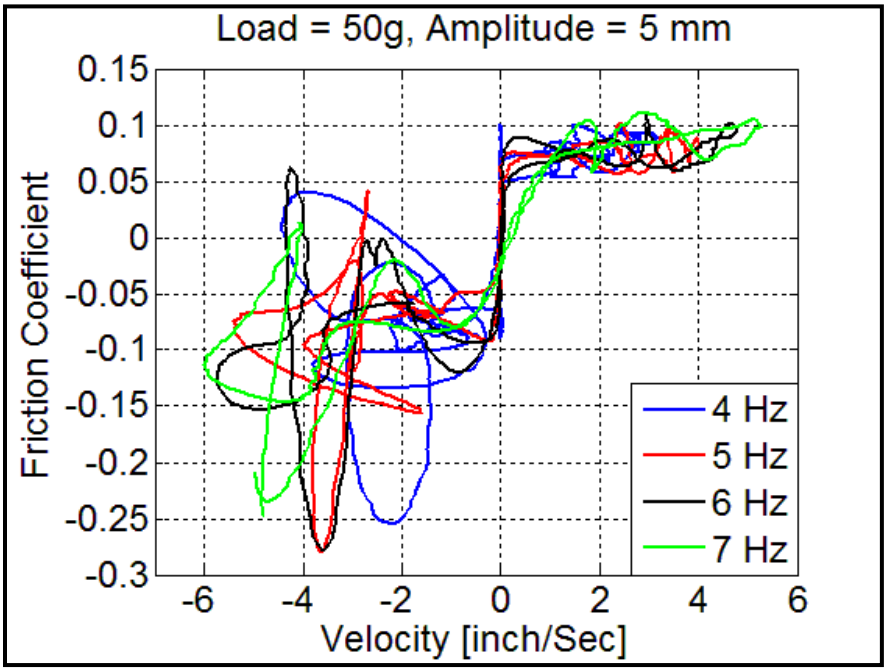


Figure 3.47 Aluminum - Teflon 50g load, amplitude of 5 mm and different frequencies.

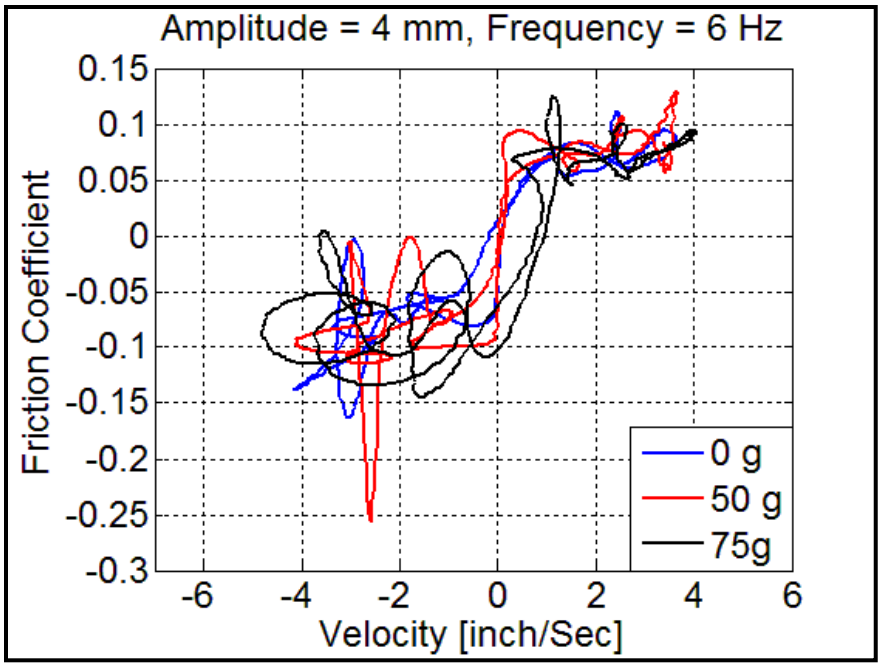


Figure 3.48 Aluminum – Teflon with different loads, amplitude = 4 mm, frequency = 6 Hz.

CHAPTER 4

CONCLUSIONS AND SUGGESTIONS FOR FUTURE STUDIES

The thesis presented herein establishes the feasibility of the concept mentioned in [6] through the building of an apparatus that uses electromagnetic actuation to move a platform with a Test Mass above it, and measurs the friction between the surfaces of the Test Mass and the platform, with a curve of velocity vs. friction coefficient as the final result.

As mentioned in Chapter 2, a Hall Effect sensor has been used to measure the relative displacement of the Test Mass, and using a Kalman filter observer, the velocity has been estimated from the displacement. Simultaneously, an accelerometer has been used to measure the acceleration of the Test Mass, from which the friction coefficient has been derived.

The Figures in Chapter 3 show that the apparatus produces results that conform to the theoretical considerations mentioned in [1], [6] and that justify the concept described in [6].

The apparatus has a wide room for improvements to make the results more clean, in other words more repeatable, those improvements can summarized in these points:

Actuator: The opposed-solenoid actuator, selected because it was available and appeared suitable for the application, is not without shortcomings. The stray magnetic field of the solenoids has an effect on the operation of the Hall effect sensor and the Accelerometer (The stray electric radiation from the solenoid electronics is one reason for the shielding of the accelerometer). Also, the hard nonlinearity restricts sinusoidal motion at low frequencies. Moreover, the stroke of the actuator is limited to l inch. A commercial, high-bandwidth actuator, possibly pneumatic or hydraulic, would be

preferable.

Motion Sensors: The Hall effect displacement sensor, although very inexpensive (under $10.) it is far from optimum for the application. A commercial capacitive or inductive sensor, or a laser sensor, would be preferable. What would be better is a direct sensor of velocity rather than a displacement sensor, since the goal of the experiment is the measurement of the friction force as a function of relative velocity. An acoustic Doppler velocity sensor might be feasible for this application.

Data Acquisition: The twelve-bit resolution of the data acquisition interface is a serious limitation of the accuracy with which the relative velocity can be derived from measurements of relative displacement. Resolution of 16 or, preferably, 24 bits, would alleviate the problem of deriving the velocity from displacement measurements, if direct sensing of the relative velocity is not feasible.

Fabrication: Bending and wobbling, etc. of the moving parts would be considerably reduced in an apparatus fabricated by skilled machinists, rather than by the author.

Notwithstanding the need for improvements the results that the apparatus produced shows that the apparatus works.

APPENDIX A

THE GUI MATLAB SOURCE CODES

This Appendix presents the listings of the MATLAB® source code for the Graphical User Interface used in this thesis, the code have been developed using MathWorks® MATLAB® Version 7.8.0.347. The flow chart that describes the logical flow of this GUI is presented in Figure A.1

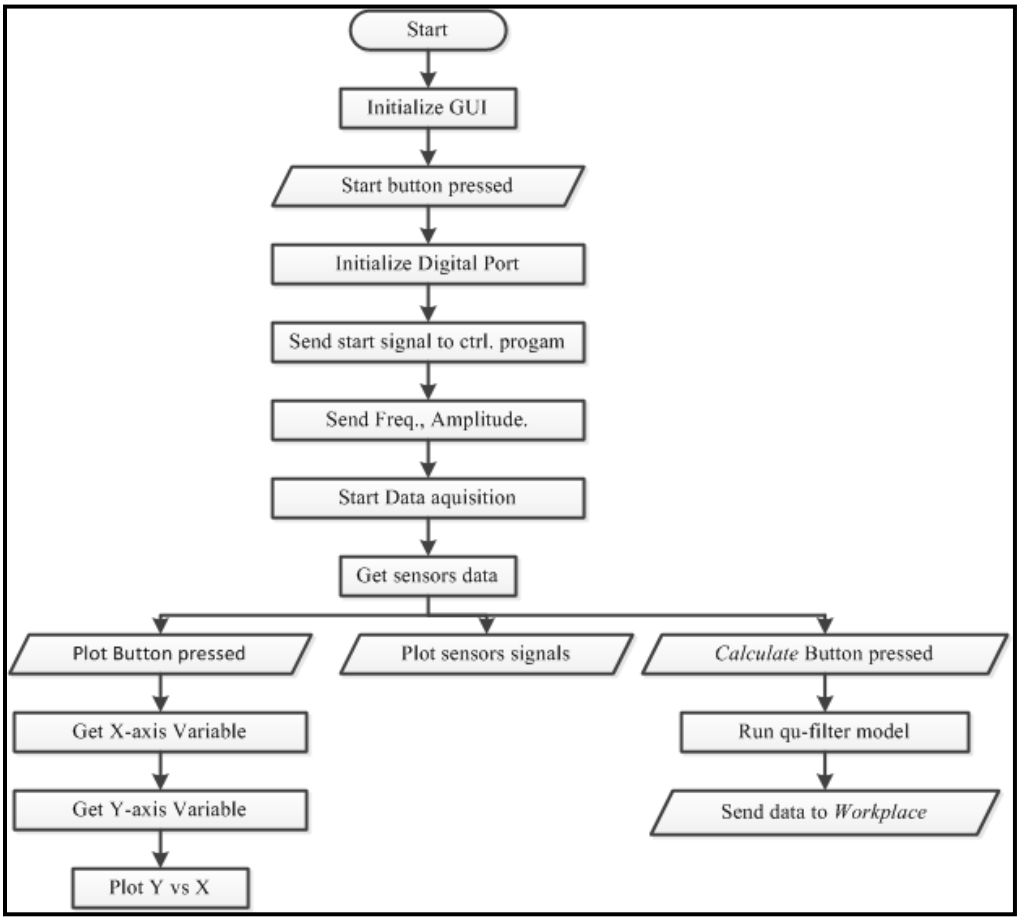


Figure A.1 The flow chart of the graphical user interface used in this thesis.

```
% Initialize the Graphical user ineterface%

function varargout = main_program(varargin)
    gui_Singleton = 1;
    gui_State = struct('gui_Name',       mfilename, ...
                       'gui_Singleton',  gui_Singleton, ...
                       'gui_OpeningFcn',
@main_program_OpeningFcn, ...
                       'gui_OutputFcn',
@main_program_OutputFcn, ...
                       'gui_LayoutFcn',
@main_program_LayoutFcn, ...
                       'gui_Callback',   []);
    if nargin && ischar(varargin{1})
        gui_State.gui_Callback = str2func(varargin{1});
    end

    if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State,...
                                  varargin{:});
    else
        gui_mainfcn(gui_State, varargin{:});
    end

function main_program_OpeningFcn(hObject, eventdata,
handles,...
                              varargin)
        handles.output = hObject;
        handles.amplitudev = 1;
        handles.Sampleratev = 5000;
        handles.Nosamplesv = 20000
        handles.frequencyv = 6;

        guidata(hObject, handles);

function varargout = main_program_OutputFcn(hObject,
eventdata,...
                                           handles)
        varargout{1} = handles.output;

%%-----------------------------------------------------------
function start_Callback(hObject, eventdata, handles)
    %Initialze DAQ digital input and output port
        set(handles.start,'Enable','off');
        set(handles.Start_Panel,'BackgroundColor','r');
        evalin('base','clear all')
        set(handles.calculatepanel,'String',...
```

```
        'Press to calculate');
    Digitaloutput=digitalio('dtol');
    Digitalinput=digitalio('dtol');
    addline(Digitalinput,0:1,0,'in');
    addline(Digitaloutput,0:7,1,'out');

%send GUI ready to the base control program
    putvalue(Digitaloutput,128);

%wait for ready signal
    z = getvalue(Digitalinput)
    while(z ~= [1 0])
        z = getvalue(Digitalinput);
    end
        putvalue(Digitaloutput,0);

%send the Frequency to the control Program
    Frequency =handles.frequencyv;
    FreqBinValue = dec2binvec(Frequency,6);
    FreqPluseReady =[FreqBinValue 1 1];
    putvalue(Digitaloutput,FreqPluseReady);

%wait for ready signal
    z = getvalue(Digitalinput);
    while(z ~= [0 1])
        z = getvalue(Digitalinput);
    end
    putvalue(Digitaloutput,0);

%send the Amplitude signal to the control program
    Amplitude =handles.amplitudev;
    AmpBinValue = dec2binvec(Amplitude,6);
    AmpPluseReady =[AmpBinValue 0 1];
    putvalue(Digitaloutput,AmpPluseReady);

%wait for ready signal and send start signal
    while(z ~= [1 1])
        z = getvalue(Digitalinput)
    end
    putvalue(Digitaloutput,129);

%Initialze DAQ Analog input port
    DAQInput= analoginput('dtol');
    DAQInput.inputtype='Differential';
    addchannel(DAQInput,1:4);
    DAQInput.samplerate =handles.Sampleratev;
    DAQInput.SamplesPerTrigger=handles.Nosamplesv;
```

```
        DAQInput.BufferingMode='Auto';

    %get data from the DAQ
        start(DAQInput);
        data = getdata(DAQInput);

        putvalue(Digitaloutput,0);

    %send data to workspace
     time = 0:1/handles.Sampleratev:(handles.Nosamplesv *
...
            1/handles.Sampleratev)-1/handles.Sampleratev;
        time=time';
        data = [time data];
        handles.datav = data;
        guidata(hObject,handles);
        assignin('base','data',data);
        assignin('base','Samplerate',handles.Sampleratev);
        assignin('base','Frequency',handles.frequencyv);
        assignin('base','Amplitude',handles.amplitudev);
        assignin('base','Nosamples',handles.Nosamplesv);

    %plot data
        plot(handles.axes1,data(:,1),data(:,2));
        set(handles.axes1,'XMinorTick','on')
        grid(handles.axes1,'on');

        plot(handles.axes2,data(:,1),data(:,3));
        set(handles.axes2,'XMinorTick','on')
        grid(handles.axes2,'on');

        plot(handles.axes3,data(:,1),data(:,4));
        set(handles.axes3,'XMinorTick','on')
        grid(handles.axes3,'on');
        set(handles.start,'Enable','on');
        set(handles.Start_Panel,'BackgroundColor','g');

    %objects callback functions
%%------------------------------------------------------------
function frequency_Callback(hObject, eventdata, handles)
  handles.frequencyv = str2double(get(hObject,'String'));
        guidata(hObject,handles);
%%------------------------------------------------------------
function frequency_CreateFcn(hObject, eventdata, handles)
  if ispc && isequal(get(hObject,'BackgroundColor'), ...
              get(0,'defaultUicontrolBackgroundColor'))
            set(hObject,'BackgroundColor','white');
```

```
  end
%%-------------------------------------------------------------
function amplitude_Callback(hObject, eventdata, handles)
  handles.amplitudev= str2double(get(hObject,'String'));
  guidata(hObject,handles);
%%-------------------------------------------------------------
function amplitude_CreateFcn(hObject, eventdata, handles)
  if ispc && isequal(get(hObject,'BackgroundColor'),...
               get(0,'defaultUicontrolBackgroundColor'))
           set(hObject,'BackgroundColor','white');
  end
%-------------------------------------------------------------
function sample_rate_Callback(hObject, eventdata, handles)
handles.Sampleratev= str2double(get(hObject,'String'));
        guidata(hObject,handles);
%-------------------------------------------------------------
function sample_rate_CreateFcn(hObject, eventdata, handles)

 if ispc && isequal(get(hObject,'BackgroundColor'), ...
               get(0,'defaultUicontrolBackgroundColor'))
           set(hObject,'BackgroundColor','white');
 end
%-------------------------------------------------------------
function No_of_samples_Callback(hObject,eventdata, handles)
 handles.Nosamplesv= str2double(get(hObject,'String'));
        guidata(hObject,handles);
%-------------------------------------------------------------
function No_of_samples_CreateFcn(hObject,eventdata,handles)

if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%-------------------------------------------------------------
function Yvaraiable_Callback(hObject, eventdata, handles)
        str = get(hObject,'String');
        val = get(hObject,'Value');
        switch str{val}
            case 'Acceleration'
       evalin('base','Yvar =
Acceleration(2:Nosamples+1);');
            case 'Velocity'
       evalin('base','Yvar =velocity(2:Nosamples+1);');
            case 'Distance'
       evalin('base','Yvar =Distance(2:Nosamples+1);');
            case 'Halleffect Sensor Output (V)'
       evalin('base','Yvar = data(:,3);');
```

```
            case 'Potetiometer OutPut (V)'
        evalin('base','Yvar = data(:,2);');
            case 'Accelermeter Output (V)'
        evalin('base','Yvar = data(:,4);');
    end
%--------------------------------------------------------
function Yvaraiable_CreateFcn(hObject, eventdata, handles)
 if ispc && isequal(get(hObject,'BackgroundColor'),...
             get(0,'defaultUicontrolBackgroundColor'))
             set(hObject,'BackgroundColor','white');
 end
%--------------------------------------------------------
function Xvariable_Callback(hObject, eventdata, handles)
        str = get(hObject,'String');
        val = get(hObject,'Value');
        switch str{val}
                case 'Time'
            evalin('base','Xvar = data(:,1);');
                case 'Samples'
            evalin('base','Xvar = 1:Nosamples');
        end
%--------------------------------------------------------
function Xvariable_CreateFcn(hObject, eventdata, handles)
 if ispc && isequal(get(hObject,'BackgroundColor'),...
            get(0,'defaultUicontrolBackgroundColor'))
            set(hObject,'BackgroundColor','white');
 end
%--------------------------------------------------------
function plot_Callback(hObject, eventdata, handles)
        evalin('base','figure;');
        evalin('base','plot(Xvar,Yvar)');
% --------------------------------------------------------
function load_Callback(hObject, eventdata, handles)
        set(handles.calculatepanel,'String',...
            'Press to calculate');
        evalin('base','clear all');
        OriginalFolder = pwd;
        [file filepath]  = uigetfile('*.mat');
        if ~isequal(file, 0)
            cd (filepath);
            load (file);
            cd(OriginalFolder);

        handles.datav = data;
        handles.frequencyv  = Frequency;
        handles.amplitudev  = Amplitude;
        handles.Sampleratev = Samplerate;
```

```
        handles.Nosamplesv  = Nosamples;
        guidata(hObject,handles);
        assignin('base','data',data);
        assignin('base','Samplerate',Samplerate);
        assignin('base','Frequency',Frequency);
        assignin('base','Amplitude',Amplitude);
        assignin('base','Nosamples',Nosamples);

        set(handles.amplitude,'string',Amplitude);
        set(handles.frequency,'string',Frequency);
        set(handles.sample_rate,'string',Samplerate);
        set(handles.No_of_samples,'string',Nosamples);

        plot(handles.axes1,data(:,1),data(:,2));
        set(handles.axes1,'XMinorTick','on')
        grid(handles.axes1,'on');

        plot(handles.axes2,data(:,1),data(:,3));
        set(handles.axes2,'XMinorTick','on')
        grid(handles.axes2,'on');

        plot(handles.axes3,data(:,1),data(:,4));
        set(handles.axes3,'XMinorTick','on')
        grid(handles.axes3,'on');
        end
% --------------------------------------------------------
function save_Callback(hObject, eventdata, handles)
        data = handles.datav ;
        Frequency = handles.frequencyv  ;
        Amplitude = handles.amplitudev  ;
        Samplerate = handles.Sampleratev;
        Nosamples = handles.Nosamplesv;
        OriginalFolder = pwd;
        [file filepath] = uiputfile('*.*','Save As');
        if ~isequal(file,0)
            cd(filepath);
  save(file,'data','Frequency','Nosamples','Samplerate',...
                'Amplitude');
            figure(1);
            hgsave(file);
        end
        cd(OriginalFolder);
% --------------------------------------------------------
function close_Callback(hObject, eventdata, handles)
        selection = ...
           questdlg('close the program?',...
                    'Close ....',...
```

```
                    'Yes','No','Yes');
        if strcmp(selection,'No')
            return;
        end
        close;
%--------------------------------------------------------
function calculate_Callback(hObject, eventdata, handles)
 set(handles.calculatepanel,'String','Calculating....');
 set(handles.Calculatepanel1,'BackgroundColor','r');
 evalin('base',...
     'sim(''Data_Handling'',Nosamples/Samplerate)');
 set(handles.calculatepanel,'String','Done..:)');
 set(handles.Calculatepanel1,'BackgroundColor','g');
%--------------------------------------------------------
function start_time_Callback(hObject, eventdata, handles)
        get(hObject,'String');
        starttime = str2double(get(hObject,'String'));
        assignin('base','starttime',starttime);
        evalin('base','starts = (Samplerate *
starttime)+1;');
%--------------------------------------------------------
function start_time_CreateFcn(hObject, eventdata, handles)
 if ispc && isequal(get(hObject,'BackgroundColor'),...
     get(0,'defaultUicontrolBackgroundColor'))
     set(hObject,'BackgroundColor','white');
 end
%--------------------------------------------------------
function end_time_CreateFcn(hObject, eventdata, handles)
 if ispc && isequal(get(hObject,'BackgroundColor'),...
                get(0,'defaultUicontrolBackgroundColor'))
            set(hObject,'BackgroundColor','white');
 end
%--------------------------------------------------------
function end_time_Callback(hObject, eventdata, handles)
        get(hObject,'String');
        endtime = str2double(get(hObject,'String'));
        assignin('base','endtime',endtime);
        evalin('base','ends = (Samplerate * endtime)+1;')
%--------------------------------------------------------
function plotfvsv_Callback(hObject, eventdata, handles)
        evalin('base','figure(1)');
        evalin('base',...
'plot(velocity(starts:ends),FrictionCoefficient(starts:ends
));');
        evalin('base','hold on');
        evalin('base','grid on');
        evalin('base','xlabel(''Velocity [Ince/Sec]'')');
```

```
        evalin('base','ylabel(''Friction Coefficient'')');
        evalin('base','hold off');


%--------------------------------------------------------------
function Data_Callback(hObject, eventdata, handles)


% --- Creates and returns a handle to the GUI figure.
function h1 = main_program_LayoutFcn(policy)


persistent hsingleton;
if strcmpi(policy, 'reuse') & ishandle(hsingleton)
    h1 = hsingleton;
    return;
end

appdata = [];
appdata.GUIDEOptions = struct(...
    'active_h', [], ...
    'taginfo', struct(...
    'figure', 2, ...
    'axes', 4, ...
    'edit', 8, ...
    'pushbutton', 5, ...
    'text', 28, ...
    'uipanel', 6, ...
    'popupmenu', 4), ...
    'override', 1, ...
    'release', 13, ...
    'resize', 'simple', ...
    'accessibility', 'callback', ...
    'mfile', 1, ...
    'callbacks', 1, ...
    'singleton', 1, ...
    'syscolorfig', 1, ...
    'blocking', 0, ...
    'lastSavedFile', 'C:', ... %change this to the
    'lastFilename', 'C:\main.fig');% locatio on you file
appdata.lastValidTag = 'figure1';
appdata.GUIDELayoutEditor = [];
appdata.initTags = struct(...
    'handle', [], ...
    'tag', 'figure1');

h1 = figure(...
```

```
'Units','characters',...
'Color',[0.941176470588235 0.941176470588235 ...
       0.941176470588235],...
'Colormap',[0 0 0.5625;0 0 0.625;0 0 0.6875;0 0 0.75;
   0 0 0.8125;0 0 0.875;0 0 0.9375;0 0 1;
   0 0.0625 1;0 0.125 1;0 0.1875 1;0 0.25 1;
   0 0.3125 1;0 0.375 1;0 0.4375 1;0 0.5 1;
   0 0.5625 1;0 0.625 1;0 0.6875 1;0 0.75 1;
   0 0.8125 1;0 0.875 1;0 0.9375 1;0 1 1;
   0.0625 1 1;0.125 1 0.9375;0.1875 1 0.875;0.25 1 0.8125;
   0.3125 1 0.75;0.375 1 0.6875;0.4375 1 0.625;
   0.5 1 0.5625;0.5625 1 0.5;0.625 1 0.4375;
   0.6875 1 0.375;0.75 1 0.3125;0.8125 1 0.25;
   0.875 1 0.1875;0.9375 1 0.125;1 1 0.0625;
   1 1 0;1 0.9375 0;1 0.875 0;1 0.8125 0;
   1 0.75 0;1 0.6875 0;1 0.625 0;1 0.5625 0;
   1 0.5 0;1 0.4375 0;1 0.375 0;1 0.3125 0;
   1 0.25 0;1 0.1875 0;1 0.125 0;1 0.0625 0;
   1 0 0;0.9375 0 0;0.875 0 0;0.8125 0 0;0.75 0 0;
   0.6875 0 0;0.625 0 0;0.5625 0 0],...
'IntegerHandle','off',...
'InvertHardcopy',get(0,'defaultfigureInvertHardcopy'),...
'MenuBar','none',...
'Name','main',...
'NumberTitle','off',...
'PaperPosition',get(0,'defaultfigurePaperPosition'),...
'Position',[103.8 12.2307692307692 239
49.3076923076923],...
'HandleVisibility','callback',...
'UserData',[],...
'Tag','figure1',...
'Visible','on',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'axes1';

h2 = axes(...
'Parent',h1,...
'FontUnits','pixels',...
'Position',[0.0401673640167364 0.684867394695788 ...
0.712133891213389 0.235569422776911],...
'CameraPosition',[0.5 0.5 9.16025403784439],...
'CameraPositionMode',...
get(0,'defaultaxesCameraPositionMode'),...
'Color',get(0,'defaultaxesColor'),...
'ColorOrder',get(0,'defaultaxesColorOrder'),...
```

```
'FontSize',13.3333333333333,...
'LooseInset',[0.138705357142857 0.261148148148148 ...
0.101361607142857 0.178055555555556],...
'XColor',get(0,'defaultaxesXColor'),...
'YColor',get(0,'defaultaxesYColor'),...
'ZColor',get(0,'defaultaxesZColor'),...
'Tag','axes1',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

h3 = get(h2,'title');

set(h3,...
'Parent',h2,...
'Units','data',...
'FontUnits','points',...
'BackgroundColor','none',...
'Color',[0 0 0],...
'DisplayName',blanks(0),...
'EdgeColor','none',...
'EraseMode','normal',...
'DVIMode','auto',...
'FontAngle','normal',...
'FontName','Helvetica',...
'FontSize',10,...
'FontWeight','normal',...
'HorizontalAlignment','center',...
'LineStyle','-',...
'LineWidth',0.5,...
'Margin',2,...
'Position',[0.5 1.04304635761589 1.00005459937205],...
'Rotation',0,...
'String',blanks(0),...
'Interpreter','tex',...
'VerticalAlignment','bottom',...
'ButtonDownFcn',[],...
'CreateFcn', {@local_CreateFcn, [], ''} ,...
'DeleteFcn',[],...
'BusyAction','queue',...
'HandleVisibility','off',...
'HelpTopicKey',blanks(0),...
'HitTest','on',...
'Interruptible','on',...
'SelectionHighlight','on',...
'Serializable','on',...
'Tag',blanks(0),...
'UserData',[],...
'Visible','on',...
```

```
'XLimInclude','on',...
'YLimInclude','on',...
'ZLimInclude','on',...
'CLimInclude','on',...
'ALimInclude','on',...
'IncludeRenderer','on',...
'Clipping','off');

h4 = get(h2,'xlabel');

set(h4,...
'Parent',h2,...
'Units','data',...
'FontUnits','points',...
'BackgroundColor','none',...
'Color',[0 0 0],...
'DisplayName',blanks(0),...
'EdgeColor','none',...
'EraseMode','normal',...
'DVIMode','auto',...
'FontAngle','normal',...
'FontName','Helvetica',...
'FontSize',10,...
'FontWeight','normal',...
'HorizontalAlignment','center',...
'LineStyle','-',...
'LineWidth',0.5,...
'Margin',2,...
'Position',[0.49882491186839 -0.155629139072848 ...
1.00005459937205],...
'Rotation',0,...
'String',blanks(0),...
'Interpreter','tex',...
'VerticalAlignment','cap',...
'ButtonDownFcn',[],...
'CreateFcn', {@local_CreateFcn, [], ''} ,...
'DeleteFcn',[],...
'BusyAction','queue',...
'HandleVisibility','off',...
'HelpTopicKey',blanks(0),...
'HitTest','on',...
'Interruptible','on',...
'SelectionHighlight','on',...
'Serializable','on',...
'Tag',blanks(0),...
'UserData',[],...
'Visible','on',...
```

```
'XLimInclude','on',...
'YLimInclude','on',...
'ZLimInclude','on',...
'CLimInclude','on',...
'ALimInclude','on',...
'IncludeRenderer','on',...
'Clipping','off');

h5 = get(h2,'ylabel');

set(h5,...
'Parent',h2,...
'Units','data',...
'FontUnits','points',...
'BackgroundColor','none',...
'Color',[0 0 0],...
'DisplayName',blanks(0),...
'EdgeColor','none',...
'EraseMode','normal',...
'DVIMode','auto',...
'FontAngle','normal',...
'FontName','Helvetica',...
'FontSize',10,...
'FontWeight','normal',...
'HorizontalAlignment','center',...
'LineStyle','-',...
'LineWidth',0.5,...
'Margin',2,...
'Position',[-0.0334900117508813 0.493377483443709...
1.00005459937205],...
'Rotation',90,...
'String',blanks(0),...
'Interpreter','tex',...
'VerticalAlignment','bottom',...
'ButtonDownFcn',[],...
'CreateFcn', {@local_CreateFcn, [], ''} ,...
'DeleteFcn',[],...
'BusyAction','queue',...
'HandleVisibility','off',...
'HelpTopicKey',blanks(0),...
'HitTest','on',...
'Interruptible','on',...
'SelectionHighlight','on',...
'Serializable','on',...
'Tag',blanks(0),...
'UserData',[],...
'Visible','on',...
```

```
'XLimInclude','on',...
'YLimInclude','on',...
'ZLimInclude','on',...
'CLimInclude','on',...
'ALimInclude','on',...
'IncludeRenderer','on',...
'Clipping','off');

h6 = get(h2,'zlabel');

set(h6,...
'Parent',h2,...
'Units','data',...
'FontUnits','points',...
'BackgroundColor','none',...
'Color',[0 0 0],...
'DisplayName',blanks(0),...
'EdgeColor','none',...
'EraseMode','normal',...
'DVIMode','auto',...
'FontAngle','normal',...
'FontName','Helvetica',...
'FontSize',10,...
'FontWeight','normal',...
'HorizontalAlignment','right',...
'LineStyle','-',...
'LineWidth',0.5,...
'Margin',2,...
'Position',[-0.0569917743830787 1.32781456953642...
1.00005459937205],...
'Rotation',0,...
'String',blanks(0),...
'Interpreter','tex',...
'VerticalAlignment','middle',...
'ButtonDownFcn',[],...
'CreateFcn', {@local_CreateFcn, [], ''} ,...
'DeleteFcn',[],...
'BusyAction','queue',...
'HandleVisibility','off',...
'HelpTopicKey',blanks(0),...
'HitTest','on',...
'Interruptible','on',...
'SelectionHighlight','on',...
'Serializable','on',...
'Tag',blanks(0),...
'UserData',[],...
'Visible','off',...
```

```
'XLimInclude','on',...
'YLimInclude','on',...
'ZLimInclude','on',...
'CLimInclude','on',...
'ALimInclude','on',...
'IncludeRenderer','on',...
'Clipping','off');

appdata = [];
appdata.lastValidTag = 'axes2';

h7 = axes(...
'Parent',h1,...
'FontUnits','pixels',...
'Position',[0.0401673640167364 0.374414976599064...
0.712133891213389 0.235569422776911],...
'CameraPosition',[0.5 0.5 9.16025403784439],...
'CameraPositionMode',...
get(0,'defaultaxesCameraPositionMode'),...
'Color',get(0,'defaultaxesColor'),...
'ColorOrder',get(0,'defaultaxesColorOrder'),...
'FontSize',13.3333333333333,...
'LooseInset',[0.138705357142857 0.261148148148148...
0.101361607142857 0.178055555555556],...
'XColor',get(0,'defaultaxesXColor'),...
'YColor',get(0,'defaultaxesYColor'),...
'ZColor',get(0,'defaultaxesZColor'),...
'Tag','axes2',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

h8 = get(h7,'title');

set(h8,...
'Parent',h7,...
'Units','data',...
'FontUnits','points',...
'BackgroundColor','none',...
'Color',[0 0 0],...
'DisplayName',blanks(0),...
'EdgeColor','none',...
'EraseMode','normal',...
'DVIMode','auto',...
'FontAngle','normal',...
'FontName','Helvetica',...
'FontSize',10,...
'FontWeight','normal',...
'HorizontalAlignment','center',...
```

```
'LineStyle','-',...
'LineWidth',0.5,...
'Margin',2,...
'Position',[0.5 1.04304635761589 1.00005459937205],...
'Rotation',0,...
'String',blanks(0),...
'Interpreter','tex',...
'VerticalAlignment','bottom',...
'ButtonDownFcn',[],...
'CreateFcn', {@local_CreateFcn, [], ''} ,...
'DeleteFcn',[],...
'BusyAction','queue',...
'HandleVisibility','off',...
'HelpTopicKey',blanks(0),...
'HitTest','on',...
'Interruptible','on',...
'SelectionHighlight','on',...
'Serializable','on',...
'Tag',blanks(0),...
'UserData',[],...
'Visible','on',...
'XLimInclude','on',...
'YLimInclude','on',...
'ZLimInclude','on',...
'CLimInclude','on',...
'ALimInclude','on',...
'IncludeRenderer','on',...
'Clipping','off');

h9 = get(h7,'xlabel');

set(h9,...
'Parent',h7,...
'Units','data',...
'FontUnits','points',...
'BackgroundColor','none',...
'Color',[0 0 0],...
'DisplayName',blanks(0),...
'EdgeColor','none',...
'EraseMode','normal',...
'DVIMode','auto',...
'FontAngle','normal',...
'FontName','Helvetica',...
'FontSize',10,...
'FontWeight','normal',...
'HorizontalAlignment','center',...
'LineStyle','-',...
```

```
'LineWidth',0.5,...
'Margin',2,...
'Position',[0.49882491186839 -0.155629139072848...
1.00005459937205],...
'Rotation',0,...
'String',blanks(0),...
'Interpreter','tex',...
'VerticalAlignment','cap',...
'ButtonDownFcn',[],...
'CreateFcn', {@local_CreateFcn, [], ''} ,...
'DeleteFcn',[],...
'BusyAction','queue',...
'HandleVisibility','off',...
'HelpTopicKey',blanks(0),...
'HitTest','on',...
'Interruptible','on',...
'SelectionHighlight','on',...
'Serializable','on',...
'Tag',blanks(0),...
'UserData',[],...
'Visible','on',...
'XLimInclude','on',...
'YLimInclude','on',...
'ZLimInclude','on',...
'CLimInclude','on',...
'ALimInclude','on',...
'IncludeRenderer','on',...
'Clipping','off');

h10 = get(h7,'ylabel');

set(h10,...
'Parent',h7,...
'Units','data',...
'FontUnits','points',...
'BackgroundColor','none',...
'Color',[0 0 0],...
'DisplayName',blanks(0),...
'EdgeColor','none',...
'EraseMode','normal',...
'DVIMode','auto',...
'FontAngle','normal',...
'FontName','Helvetica',...
'FontSize',10,...
'FontWeight','normal',...
'HorizontalAlignment','center',...
'LineStyle','-',...
```

```
'LineWidth',0.5,...
'Margin',2,...
'Position',[-0.0334900117508813 0.493377483443709...
1.00005459937205],...
'Rotation',90,...
'String',blanks(0),...
'Interpreter','tex',...
'VerticalAlignment','bottom',...
'ButtonDownFcn',[],...
'CreateFcn', {@local_CreateFcn, [], ''} ,...
'DeleteFcn',[],...
'BusyAction','queue',...
'HandleVisibility','off',...
'HelpTopicKey',blanks(0),...
'HitTest','on',...
'Interruptible','on',...
'SelectionHighlight','on',...
'Serializable','on',...
'Tag',blanks(0),...
'UserData',[],...
'Visible','on',...
'XLimInclude','on',...
'YLimInclude','on',...
'ZLimInclude','on',...
'CLimInclude','on',...
'ALimInclude','on',...
'IncludeRenderer','on',...
'Clipping','off');

h11 = get(h7,'zlabel');

set(h11,...
'Parent',h7,...
'Units','data',...
'FontUnits','points',...
'BackgroundColor','none',...
'Color',[0 0 0],...
'DisplayName',blanks(0),...
'EdgeColor','none',...
'EraseMode','normal',...
'DVIMode','auto',...
'FontAngle','normal',...
'FontName','Helvetica',...
'FontSize',10,...
'FontWeight','normal',...
'HorizontalAlignment','right',...
'LineStyle','-',...
```

```
'LineWidth',0.5,...
'Margin',2,...
'Position',[-0.0569917743830787 2.64569536423841...
1.00005459937205],...
'Rotation',0,...
'String',blanks(0),...
'Interpreter','tex',...
'VerticalAlignment','middle',...
'ButtonDownFcn',[],...
'CreateFcn', {@local_CreateFcn, [], ''} ,...
'DeleteFcn',[],...
'BusyAction','queue',...
'HandleVisibility','off',...
'HelpTopicKey',blanks(0),...
'HitTest','on',...
'Interruptible','on',...
'SelectionHighlight','on',...
'Serializable','on',...
'Tag',blanks(0),...
'UserData',[],...
'Visible','off',...
'XLimInclude','on',...
'YLimInclude','on',...
'ZLimInclude','on',...
'CLimInclude','on',...
'ALimInclude','on',...
'IncludeRenderer','on',...
'Clipping','off');

appdata = [];
appdata.lastValidTag = 'axes3';

h12 = axes(...
'Parent',h1,...
'FontUnits','pixels',...
'Position',[0.0410041841004184 0.060842433697348...
0.712133891213389 0.237129485179407],...
'CameraPosition',[0.5 0.5 9.16025403784439],...
'CameraPositionMode',...
get(0,'defaultaxesCameraPositionMode'),...
'Color',get(0,'defaultaxesColor'),...
'ColorOrder',get(0,'defaultaxesColorOrder'),...
'FontSize',13.3333333333333,...
'LooseInset',[0.138705357142857 0.260184501845019...
0.101361607142857 0.17739852398524],...
'XColor',get(0,'defaultaxesXColor'),...
'YColor',get(0,'defaultaxesYColor'),...
```

```
'ZColor',get(0,'defaultaxesZColor'),...
'Tag','axes3',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

h13 = get(h12,'title');

set(h13,...
'Parent',h12,...
'Units','data',...
'FontUnits','points',...
'BackgroundColor','none',...
'Color',[0 0 0],...
'DisplayName',blanks(0),...
'EdgeColor','none',...
'EraseMode','normal',...
'DVIMode','auto',...
'FontAngle','normal',...
'FontName','Helvetica',...
'FontSize',10,...
'FontWeight','normal',...
'HorizontalAlignment','center',...
'LineStyle','-',...
'LineWidth',0.5,...
'Margin',2,...
'Position',[0.5 1.04276315789474 1.00005459937205],...
'Rotation',0,...
'String',blanks(0),...
'Interpreter','tex',...
'VerticalAlignment','bottom',...
'ButtonDownFcn',[],...
'CreateFcn', {@local_CreateFcn, [], ''} ,...
'DeleteFcn',[],...
'BusyAction','queue',...
'HandleVisibility','off',...
'HelpTopicKey',blanks(0),...
'HitTest','on',...
'Interruptible','on',...
'SelectionHighlight','on',...
'Serializable','on',...
'Tag',blanks(0),...
'UserData',[],...
'Visible','on',...
'XLimInclude','on',...
'YLimInclude','on',...
'ZLimInclude','on',...
'CLimInclude','on',...
'ALimInclude','on',...
```

```
'IncludeRenderer','on',...
'Clipping','off');

h14 = get(h12,'xlabel');

set(h14,...
'Parent',h12,...
'Units','data',...
'FontUnits','points',...
'BackgroundColor','none',...
'Color',[0 0 0],...
'DisplayName',blanks(0),...
'EdgeColor','none',...
'EraseMode','normal',...
'DVIMode','auto',...
'FontAngle','normal',...
'FontName','Helvetica',...
'FontSize',10,...
'FontWeight','normal',...
'HorizontalAlignment','center',...
'LineStyle','-',...
'LineWidth',0.5,...
'Margin',2,...
'Position',[0.49882491186839 -0.154605263157895...
1.00005459937205],...
'Rotation',0,...
'String',blanks(0),...
'Interpreter','tex',...
'VerticalAlignment','cap',...
'ButtonDownFcn',[],...
'CreateFcn', {@local_CreateFcn, [], ''} ,...
'DeleteFcn',[],...
'BusyAction','queue',...
'HandleVisibility','off',...
'HelpTopicKey',blanks(0),...
'HitTest','on',...
'Interruptible','on',...
'SelectionHighlight','on',...
'Serializable','on',...
'Tag',blanks(0),...
'UserData',[],...
'Visible','on',...
'XLimInclude','on',...
'YLimInclude','on',...
'ZLimInclude','on',...
'CLimInclude','on',...
'ALimInclude','on',...
```

```
'IncludeRenderer','on',...
'Clipping','off');

h15 = get(h12,'ylabel');

set(h15,'Parent',h12,'Units','data','FontUnits','points','B
ackgroundColor','none','Color',[0 0 0],
'DisplayName',blanks(0),'EdgeColor','none',...
'EraseMode','normal',...
'DVIMode','auto',...
'FontAngle','normal',...
'FontName','Helvetica',...
'FontSize',10,...
'FontWeight','normal',...
'HorizontalAlignment','center',...
'LineStyle','-',...
'LineWidth',0.5,...
'Margin',2,...
'Position',[-0.0334900117508813 0.490131578947369...
1.00005459937205],...
'Rotation',90,...
'String',blanks(0),...
'Interpreter','tex',...
'VerticalAlignment','bottom',...
'ButtonDownFcn',[],...
'CreateFcn', {@local_CreateFcn, [], ''} ,...
'DeleteFcn',[],...
'BusyAction','queue',...
'HandleVisibility','off',...
'HelpTopicKey',blanks(0),...
'HitTest','on',...
'Interruptible','on',...
'SelectionHighlight','on',...
'Serializable','on',...
'Tag',blanks(0),...
'UserData',[],...
'Visible','on',...
'XLimInclude','on',...
'YLimInclude','on',...
'ZLimInclude','on',...
'CLimInclude','on',...
'ALimInclude','on',...
'IncludeRenderer','on',...
'Clipping','off');

h16 = get(h12,'zlabel');
```

```
set(h16,...
'Parent',h12,...
'Units','data',...
'FontUnits','points',...
'BackgroundColor','none',...
'Color',[0 0 0],...
'DisplayName',blanks(0),...
'EdgeColor','none',...
'EraseMode','normal',...
'DVIMode','auto',...
'FontAngle','normal',...
'FontName','Helvetica',...
'FontSize',10,...
'FontWeight','normal',...
'HorizontalAlignment','right',...
'LineStyle','-',...
'LineWidth',0.5,...
'Margin',2,...
'Position',[-0.0581668625146886 3.95065789473684...
1.00005459937205],...
'Rotation',0,...
'String',blanks(0),...
'Interpreter','tex',...
'VerticalAlignment','middle',...
'ButtonDownFcn',[],...
'CreateFcn', {@local_CreateFcn, [], ''} ,...
'DeleteFcn',[],...
'BusyAction','queue',...
'HandleVisibility','off',...
'HelpTopicKey',blanks(0),...
'HitTest','on',...
'Interruptible','on',...
'SelectionHighlight','on',...
'Serializable','on',...
'Tag',blanks(0),...
'UserData',[],...
'Visible','off',...
'XLimInclude','on',...
'YLimInclude','on',...
'ZLimInclude','on',...
'CLimInclude','on',...
'ALimInclude','on',...
'IncludeRenderer','on',...
'Clipping','off');

appdata = [];
appdata.lastValidTag = 'start';
```

```
h17 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'FontUnits','pixels',...
'Callback',...
@(hObject,eventdata)main_program('start_Callback',...
hObject,eventdata,guidata(hObject)),...
'FontSize',10.6666666666667,...
'Position',[0.938075313807531 0.96411856474259...
0.0577405857740586 0.0343213728549142],...
'String','Start',...
'TooltipString','Start the experiment',...
'Tag','start',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'uipanel1';

h18 = uipanel(...
'Parent',h1,...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'Title','Parameters',...
'Tag','uipanel1',...
'Clipping','on',...
'Position',[0.77907949790795 0.731669266770671...
0.220083682008368 0.201248049921997],...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'frequency';

h19 = uicontrol(...
'Parent',h18,...
'Units','normalized',...
'FontUnits','pixels',...
'BackgroundColor',[1 1 1],...
'Callback',...
@(hObject,eventdata)main_program('frequency_Callback',...
hObject,eventdata,guidata(hObject)),...
'FontSize',10.6666666666667,...
'Position',[0.459459459459459 0.672566371681416...
0.196911196911197 0.194690265486726],...
'String','6',...
'Style','edit',...
'CreateFcn', ...
```

```
{@local_CreateFcn,...
@(hObject,eventdata)main_program('frequency_CreateFcn',...
hObject,eventdata,guidata(hObject)), appdata} ,...
'Tag','frequency');

appdata = [];
appdata.lastValidTag = 'amplitude';

h20 = uicontrol(...
'Parent',h18,...
'Units','normalized',...
'FontUnits','pixels',...
'BackgroundColor',[1 1 1],...
'Callback',...
@(hObject,eventdata)main_program('amplitude_Callback',...
hObject,eventdata,guidata(hObject)),...
'FontSize',10.6666666666667,...
'Position',[0.459459459459459 0.460176991150442...
0.196911196911197 0.194690265486726],...
'String','4',...
'Style','edit',...
'CreateFcn',...
{@local_CreateFcn,...
@(hObject,eventdata)main_program('amplitude_CreateFcn',...
hObject,eventdata,guidata(hObject)), appdata} ,...
'Tag','amplitude');

appdata = [];
appdata.lastValidTag = 'sample_rate';

h21 = uicontrol(...
'Parent',h18,...
'Units','normalized',...
'FontUnits','pixels',...
'BackgroundColor',[1 1 1],...
'Callback',...
@(hObject,eventdata)main_program('sample_rate_Callback',...
hObject,eventdata,guidata(hObject)),...
'FontSize',10.6666666666667,...
'Position',[0.459459459459459 0.247787610619469...
0.196911196911197 0.194690265486726],...
'String','5000',...
'Style','edit',...
'CreateFcn',...
{@local_CreateFcn,...
@(hObject,eventdata)main_program('sample_rate_CreateFcn',..
.
```

```
hObject,eventdata,guidata(hObject)), appdata} ,...
'Tag','sample_rate');

appdata = [];
appdata.lastValidTag = 'No_of_samples';

h22 = uicontrol(...
'Parent',h18,...
'Units','normalized',...
'FontUnits','pixels',...
'BackgroundColor',[1 1 1],...
'Callback',...
@(hObject,eventdata)main_program('No_of_samples_Callback',.
..
hObject,eventdata,guidata(hObject)),...
'FontSize',10.6666666666667,...
'Position',[0.459459459459459 0.0353982300884956...
0.196911196911197 0.194690265486726],...
'String','20000',...
'Style','edit',...
'CreateFcn', {@local_CreateFcn,...
@(hObject,eventdata)main_program('No_of_samples_CreateFcn',
...
hObject,eventdata,guidata(hObject)), appdata} ,...
'Tag','No_of_samples');

appdata = [];
appdata.lastValidTag = 'text1';

h23 = uicontrol(...
'Parent',h18,...
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'HorizontalAlignment','left',...
'Position',[0.0694980694980695 0.699115044247788...
0.38996138996139 0.132743362831858],...
'String','Frequency (1 : 32) ',...
'Style','text',...
'Tag','text1',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text2';

h24 = uicontrol(...
'Parent',h18,...
```

```
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'HorizontalAlignment','left',...
'Position',[0.0694980694980695 0.486725663716814 ...
0.38996138996139 0.141592920353982],...
'String','Amplitude (1 : 8)     ',...
'Style','text',...
'Tag','text2',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text3';

h25 = uicontrol(...
'Parent',h18,...
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'HorizontalAlignment','left',...
'Position',[0.0694980694980695 0.274336283185841...
0.38996138996139 0.132743362831858],...
'String','Sampling Rate',...
'Style','text',...
'Tag','text3',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text4';

h26 = uicontrol(...
'Parent',h18,...
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'HorizontalAlignment','left',...
'Position',[0.0694980694980695 0.0530973451327434...
0.38996138996139 0.132743362831858],...
'String','No. of Samples',...
'Style','text',...
'Tag','text4',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text7';

h27 = uicontrol(...
```

```
'Parent',h18,...
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'Position',[0.656370656370656 0.699115044247788...
0.204633204633205 0.132743362831858],...
'String','[Hz]',...
'Style','text',...
'Tag','text7',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text8';

h28 = uicontrol(...
'Parent',h18,...
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'Position',[0.656370656370658 0.469026548672567...
0.200772200772201 0.132743362831858],...
'String','[mm]',...
'Style','text',...
'Tag','text8',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text9';

h29 = uicontrol(...
'Parent',h18,...
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'Position',[0.664092664092664 0.247787610619469...
0.324324324324324 0.132743362831858],...
'String','[Sample Per Sec]',...
'Style','text',...
'Tag','text9',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text10';

h30 = uicontrol(...
'Parent',h18,...
'Units','normalized',...
```

```
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'Position',[0.694980694980696 0.0530973451327434...
0.200772200772201 0.132743362831858],...
'String','[Samples]',...
'Style','text',...
'Tag','text10',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'Data';

h31 = uimenu(...
'Parent',h1,...
'Callback',...
@(hObject,eventdata)main_program('Data_Callback',...
hObject,eventdata,guidata(hObject)),...
'Label','Data',...
'Tag','Data',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'load';

h32 = uimenu(...
'Parent',h31,...
'Callback',...
@(hObject,eventdata)main_program('load_Callback',...
hObject,eventdata,guidata(hObject)),...
'Label','Load',...
'Tag','load',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'save';

h33 = uimenu(...
'Parent',h31,...
'Callback',...
@(hObject,eventdata)main_program('save_Callback',...
hObject,eventdata,guidata(hObject)),...
'Label','Save',...
'Tag','save',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'close';
```

```
h34 = uimenu(...
'Parent',h31,...
'Callback',...
@(hObject,eventdata)main_program('close_Callback',...
hObject,eventdata,guidata(hObject)),...
'Label','Close',...
'Separator','on',...
'Tag','close',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );
appdata = [];
appdata.lastValidTag = 'uipanel3';
h35 = uipanel(...
'Parent',h1,...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'Title','Variables plots',...
'Tag','uipanel3',...
'Clipping','on',...
'Position',[0.780753138075314 0.0608424336973479...
0.125523012552301 0.234009360374415],...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'plot';

h36 = uicontrol(...
'Parent',h35,...
'Units','normalized',...
'FontUnits','pixels',...
'Callback',...
@(hObject,eventdata)main_program('plot_Callback',...
hObject,eventdata,guidata(hObject)),...
'FontSize',10.6666666666667,...
'Position',[0.5 0.201492537313433 0.472602739726027...
0.164179104477612],...
'String','Plot',...
'Tag','plot',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );
 appdata = [];
appdata.lastValidTag = 'text12';

h37 = uicontrol(...
'Parent',h35,...
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
```

```
'Position',[0.191780821917808 0.805970149253731...
0.0958904109589041 0.104477611940299],...
'String','X',...
'Style','text',...
'Tag','text12',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );
appdata = [];
appdata.lastValidTag = 'text13';

h38 = uicontrol(...
'Parent',h35,...
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'Position',[0.164383561643836 0.574626865671642 ...
0.143835616438356 0.104477611940299],...
'String','Y',...
'Style','text',...
'Tag','text13',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'Xvariable';

h39 = uicontrol(...
'Parent',h35,...
'Units','normalized',...
'FontUnits','pixels',...
'BackgroundColor',[1 1 1],...
'Callback',...
@(hObject,eventdata)main_program('Xvariable_Callback',...
hObject,eventdata,guidata(hObject)),...
'FontSize',10.6666666666667,...
'Position',[0.321917808219178 0.783582089552238 ...
0.650684931506849 0.149253731343284],...
'String',{  'Time'; 'Samples'; blanks(0) },...
'Style','popupmenu',...
'Value',1,...
'CreateFcn', {@local_CreateFcn,...
@(hObject,eventdata)main_program('Xvariable_CreateFcn',...
hObject,eventdata,guidata(hObject)), appdata} ,...
'Tag','Xvariable');
appdata = [];
appdata.lastValidTag = 'Yvaraiable';

h40 = uicontrol(...
'Parent',h35,...
```

```
'Units','normalized',...
'FontUnits','pixels',...
'BackgroundColor',[1 1 1],...
'Callback',...
@(hObject,eventdata)main_program('Yvaraiable_Callback',...
hObject,eventdata,guidata(hObject)),...
'FontSize',10.6666666666667,...
'Position',[0.321917808219178 0.567164179104478...
0.650684931506849 0.149253731343284],...
'String',{  'Acceleration'; 'Velocity'; 'Distance';...
'Halleffect Sensor Output (V)'; ...
'Potetiometer OutPut (V)'; 'Accelermeter Output (V)' },...
'Style','popupmenu',...
'Value',1,...
'CreateFcn', {@local_CreateFcn,...
@(hObject,eventdata)main_program('Yvaraiable_CreateFcn',...
hObject,eventdata,guidata(hObject)), appdata} ,...
'Tag','Yvaraiable');
appdata = [];
appdata.lastValidTag = 'calculate';
h41 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'FontUnits','pixels',...
'Callback',...
@(hObject,eventdata)main_program('calculate_Callback',...
hObject,eventdata,guidata(hObject)),...
'FontSize',10.6666666666667,...
'Position',[0.780753138075314 0.672386895475819 ...
0.0577405857740586 0.0343213728549142],...
'String','Calculate',...
'TooltipString',...
'Calculte actual acceleration, distance and velocity',...
'Tag','calculate',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'start_time';

h42 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'FontUnits','pixels',...
'BackgroundColor',[1 1 1],...
'Callback',...
@(hObject,eventdata)main_program('start_time_Callback',...
hObject,eventdata,guidata(hObject)),...
```

```
'FontSize',10.6666666666667,...
'Position',[0.781589958158996 0.513260530421217 ...
0.0426778242677824 0.0343213728549142],...
'String',blanks(0),...
'Style','edit',...
'TooltipString','The begining of the plot',...
'CreateFcn', {@local_CreateFcn, ...
@(hObject,eventdata)main_program('start_time_CreateFcn',...
hObject,eventdata,guidata(hObject)), appdata} ,...
'Tag','start_time');

appdata = [];
appdata.lastValidTag = 'end_time';

h43 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'FontUnits','pixels',...
'BackgroundColor',[1 1 1],...
'Callback',...
@(hObject,eventdata)main_program('end_time_Callback',...
hObject,eventdata,guidata(hObject)),...
'FontSize',10.6666666666667,...
'Position',[0.871129707112971 0.513260530421217...
0.0426778242677824 0.0343213728549142],...
'String',blanks(0),...
'Style','edit',...
'TooltipString','The end of the plot',...
'CreateFcn', {@local_CreateFcn,...
@(hObject,eventdata)main_program('end_time_CreateFcn',...
hObject,eventdata,guidata(hObject)), appdata} ,...
'Tag','end_time');

appdata = [];
appdata.lastValidTag = 'text14';

h44 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'Position',[0.780753138075314 0.553822152886116 ...
0.0435146443514644 0.0218408736349454],...
'String','Start Time',...
'Style','text',...
'Tag','text14',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );
```

```
appdata = [];
appdata.lastValidTag = 'text15';

h45 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'Position',[0.868619246861925 0.55226209048362 ...
0.0435146443514644 0.0218408736349454],...
'String','End Time',...
'Style','text',...
'Tag','text15',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text16';

h46 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'Position',[0.824267782426778 0.519500780031201 ...
0.0435146443514644 0.0218408736349454],...
'String','(Sec)',...
'Style','text',...
'Tag','text16',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text17';

h47 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'Position',[0.913807531380753 0.517940717628705 ...
0.0435146443514644 0.0218408736349454],...
'String','(Sec)',...
'Style','text',...
'Tag','text17',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
```

```
appdata.lastValidTag = 'plotfvsv';

h48 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'FontUnits','pixels',...
'Callback',...
@(hObject,eventdata)main_program('plotfvsv_Callback',...
hObject,eventdata,guidata(hObject)),...
'FontSize',10.6666666666667,...
'Position',[0.781589958158996 0.466458658346334 ...
0.0577405857740586 0.0343213728549142],...
'String','Plot Mu vs V',...
'TooltipString',...
'Plot the Friction Coefficient vs the Velocity',...
'Tag','plotfvsv',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'calculatepanel';

h49 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'Position',[0.858577405857741 0.683307332293292 ...
0.100418410041841 0.0218408736349454],...
'String','press to calculate',...
'Style','text',...
'Tag','calculatepanel',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'Start_Panel';

h50 = uipanel(...
'Parent',h1,...
'FontUnits','pixels',...
'BorderType','line',...
'FontSize',10.6666666666667,...
'HighlightColor',[0.941176470588235 ...
0.941176470588235 0.941176470588235],...
'ShadowColor',[0.941176470588235 0.941176470588235 ...
0.941176470588235],...
'Title',blanks(0),...
'Tag','Start_Panel',...
```

```
'Clipping','on',...
'BackgroundColor',[0 1 0],...
'Position',[0.919665271966527 0.967238689547582 ...
0.0192468619246862 0.031201248049922],...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'Calculatepanel1';

h51 = uipanel(...
'Parent',h1,...
'FontUnits','pixels',...
'BorderType','line',...
'FontSize',10.6666666666667,...
'HighlightColor',[0.941176470588235 0.941176470588235 ...
0.941176470588235],...
'ShadowColor',[0.941176470588235 0.941176470588235 ...
0.941176470588235],...
'Title',blanks(0),...
'Tag','Calculatepanel1',...
'Clipping','on',...
'BackgroundColor',[0 1 0],...
'Position',[0.838493723849372 0.675507020280812 ...
0.0192468619246862 0.031201248049922],...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text19';

h52 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'HorizontalAlignment','left',...
'Position',[0.0418410041841004 0.920436817472699 ...
0.112133891213389 0.0218408736349454],...
'String','Potentiometer Signal',...
'Style','text',...
'Tag','text19',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text20';

h53 = uicontrol(...
'Parent',h1,...
```

```
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'HorizontalAlignment','left',...
'Position',[0.0410041841004184 0.609984399375975 ...
0.120502092050209 0.0218408736349454],...
'String','Hall Effect Sensor Signal',...
'Style','text',...
'Tag','text20',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text21';

h54 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'HorizontalAlignment','left',...
'Position',[0.0410041841004184 0.297971918876755...
0.115481171548117 0.0218408736349454],...
'String','Accelerometer Signal',...
'Style','text',...
'Tag','text21',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text22';

h55 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'Position',[0.375732217573222 0.318252730109205...
0.0435146443514644 0.0218408736349454],...
'String','Sec',...
'Style','text',...
'Tag','text22',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text23';

h56 = uicontrol(...
'Parent',h1,...
```

```
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'Position',[0.375732217573222 0.634945397815913...
0.0435146443514644 0.0218408736349454],...
'String','Sec',...
'Style','text',...
'Tag','text23',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text24';

h57 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'FontUnits','pixels',...
'FontSize',10.6666666666667,...
'Position',[0.375732217573222 0.0109204368174727...
0.0435146443514644 0.0218408736349454],...
'String','Sec',...
'Style','text',...
'Tag','text24',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text25';

h58 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'FontUnits','pixels',...
'CData',[],...
'FontSize',10.6666666666667,...
'Position',[-0.000836820083682008 0.790951638065523...
0.0200836820083682 0.0280811232449298],...
'String','V',...
'Style','text',...
'UserData',[],...
'Tag','text25',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text26';

h59 = uicontrol(...
'Parent',h1,...
```

```
'Units','normalized',...
'FontUnits','pixels',...
'CData',[],...
'FontSize',10.6666666666667,...
'Position',[-0.000836820083682008 0.168486739469579...
0.0200836820083682 0.0280811232449298],...
'String','V',...
'Style','text',...
'UserData',[],...
'Tag','text26',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text27';

h60 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'FontUnits','pixels',...
'CData',[],...
'FontSize',10.6666666666667,...
'Position',[-0.000836820083682008 0.47581903276131...
0.0200836820083682 0.0280811232449298],...
'String','V',...
'Style','text',...
'UserData',[],...
'Tag','text27',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );


hsingleton = h1;


%  Set application data first then calling the CreateFcn.
function local_CreateFcn(hObject, eventdata, ...
    createfcn, appdata)

if ~isempty(appdata)
   names = fieldnames(appdata);
   for i=1:length(names)
       name = char(names(i));
       setappdata(hObject, name, getfield(appdata,name));
   end
end

if ~isempty(createfcn)
   if isa(createfcn,'function_handle')
```

```
        createfcn(hObject, eventdata);
    else
        eval(createfcn);
    end
end



function varargout = gui_mainfcn(gui_State, varargin)

gui_StateFields =  {'gui_Name'
    'gui_Singleton'
    'gui_OpeningFcn'
    'gui_OutputFcn'
    'gui_LayoutFcn'
    'gui_Callback'};
gui_Mfile = '';
for i=1:length(gui_StateFields)
    if ~isfield(gui_State, gui_StateFields{i})
        error('MATLAB:gui_mainfcn:FieldNotFound',...
'Could not find field %s in the gui_State struct',...
gui_StateFields{i}, gui_Mfile);
    elseif isequal(gui_StateFields{i}, 'gui_Name')
        gui_Mfile = [gui_State.(gui_StateFields{i}), '.m'];
    end
end

numargin = length(varargin);

if numargin == 0

    gui_Create = true;
elseif local_isInvokeActiveXCallback(gui_State,
varargin{:})
    % MAIN_PROGRAM(ACTIVEX,...)
    vin{1} = gui_State.gui_Name;
    vin{2} = [get(varargin{1}.Peer, 'Tag'), '_',
varargin{end}];
    vin{3} = varargin{1};
    vin{4} = varargin{end-1};
    vin{5} = guidata(varargin{1}.Peer);
    feval(vin{:});
    return;
elseif local_isInvokeHGCallback(gui_State, varargin{:})
    %
MAIN_PROGRAM('CALLBACK',hObject,eventData,handles,...)
    gui_Create = false;
```

```
else
    % MAIN_PROGRAM(...)
    % create the GUI and hand varargin to the openingfcn
    gui_Create = true;
end

if ~gui_Create

    designEval = false;
    if (numargin>1 && ishghandle(varargin{2}))
        fig = varargin{2};
        while ~isempty(fig) && ~ishghandle(fig,'figure')
            fig = get(fig,'parent');
        end

        designEval = isappdata(0,'CreatingGUIDEFigure')...
            || isprop(fig,'__GUIDEFigure');
    end

    if designEval
        beforeChildren = findall(fig);
    end

    varargin{1} = gui_State.gui_Callback;
    if nargout
        [varargout{1:nargout}] = feval(varargin{:});
    else
        feval(varargin{:});
    end


    if designEval && ishghandle(fig)
        set(setdiff(findall(fig),beforeChildren),...
            'Serializable','off');
    end
else
    if gui_State.gui_Singleton
        gui_SingletonOpt = 'reuse';
    else
        gui_SingletonOpt = 'new';
    end


    gui_Visible = 'auto';
    gui_VisibleInput = '';
    for index=1:2:length(varargin)
        if length(varargin) == index ||...
```

```
            ~ischar(varargin{index})
        break;
    end

    % Recognize 'visible' P/V pair
 len1 = min(length('visible'),length(varargin{index}));
  len2 = min(length('off'),length(varargin{index+1}));
    if ischar(varargin{index+1}) && ...
strncmpi(varargin{index},'visible',len1) && len2 > 1
        if strncmpi(varargin{index+1},'off',len2)
            gui_Visible = 'invisible';
            gui_VisibleInput = 'off';
        elseif strncmpi(varargin{index+1},'on',len2)
            gui_Visible = 'visible';
            gui_VisibleInput = 'on';
        end
    end
end


% Do feval on layout code in m-file if it exists
gui_Exported = ~isempty(gui_State.gui_LayoutFcn);
   setappdata(0,genvarname(['OpenGuiWhenRunning_',...
       gui_State.gui_Name]),1);
if gui_Exported
    gui_hFigure = feval(gui_State.gui_LayoutFcn,...
        gui_SingletonOpt);
   if isempty(gui_VisibleInput)
        gui_VisibleInput = get(gui_hFigure,'Visible');
    end
    set(gui_hFigure,'Visible','off')
    movegui(gui_hFigure,'onscreen');
else
    gui_hFigure = local_openfig(gui_State.gui_Name,...
        gui_SingletonOpt, gui_Visible);

    if isappdata(gui_hFigure, 'InGUIInitialization')
        delete(gui_hFigure);
      gui_hFigure = local_openfig(gui_State.gui_Name...
            , gui_SingletonOpt, gui_Visible);
    end
end
if isappdata(0, genvarname(['OpenGuiWhenRunning_',...
        gui_State.gui_Name]))
    rmappdata(0,genvarname(['OpenGuiWhenRunning_',...
        gui_State.gui_Name]));
end
```

```
setappdata(gui_hFigure,'InGUIInitialization',1);


gui_Options = getappdata(gui_hFigure,'GUIDEOptions');

gui_Options.singleton = gui_State.gui_Singleton;

if ~isappdata(gui_hFigure,'GUIOnScreen')

    if gui_Options.syscolorfig
        set(gui_hFigure,'Color',...
            get(0,'DefaultUicontrolBackgroundColor'));
    end

    data = guidata(gui_hFigure);
    handles = guihandles(gui_hFigure);
    if ~isempty(handles)
        if isempty(data)
            data = handles;
        else
            names = fieldnames(handles);
            for k=1:length(names)
                data.(char(names(k)))=...
                    handles.(char(names(k)));
            end
        end
    end
    guidata(gui_hFigure, data);
end
for index=1:2:length(varargin)
    if length(varargin) == index ||...
            ~ischar(varargin{index})
        break;
    end
    len1 = min(length('visible'),...
        length(varargin{index}));
    if ~strncmpi(varargin{index},'visible',len1)
        try set(gui_hFigure, varargin{index},...
                varargin{index+1}), catch break, end
    end
end
gui_HandleVisibility = get(gui_hFigure,...
    'HandleVisibility');
if strcmp(gui_HandleVisibility, 'callback')
    set(gui_hFigure,'HandleVisibility', 'on');
end
feval(gui_State.gui_OpeningFcn, gui_hFigure,...
```

```
        [], guidata(gui_hFigure), varargin{:});
    if isscalar(gui_hFigure) && ishghandle(gui_hFigure)

    guidemfile('restoreToolbarToolPredefinedCallback',...
        gui_hFigure);


        set(gui_hFigure,'HandleVisibility',...
            gui_HandleVisibility);
        if ~gui_Exported
            gui_hFigure = ...
local_openfig(gui_State.gui_Name, 'reuse',gui_Visible);
        elseif ~isempty(gui_VisibleInput)
            set(gui_hFigure,'Visible',gui_VisibleInput);
        end
        if strcmpi(get(gui_hFigure, 'Visible'), 'on')
            figure(gui_hFigure);

            if gui_Options.singleton
                setappdata(gui_hFigure,'GUIOnScreen', 1);
            end
        end
       if isappdata(gui_hFigure,'InGUIInitialization')
            rmappdata(gui_hFigure,'InGUIInitialization');
        end


        gui_HandleVisibility = ...
            get(gui_hFigure,'HandleVisibility');
        if strcmp(gui_HandleVisibility, 'callback')
            set(gui_hFigure,'HandleVisibility', 'on');
        end
        gui_Handles = guidata(gui_hFigure);
    else
        gui_Handles = [];
    end

    if nargout
        [varargout{1:nargout}] = ...
        feval(gui_State.gui_OutputFcn,...
        gui_hFigure, [], gui_Handles);
    else
        feval(gui_State.gui_OutputFcn,...
            gui_hFigure, [], gui_Handles);
    end

    if isscalar(gui_hFigure) && ...
```

```
            ishghandle(gui_hFigure)
        set(gui_hFigure,'HandleVisibility', ...
            gui_HandleVisibility);
    end
end

function gui_hFigure = local_openfig(name,...
    singleton, visible)


if nargin('openfig') == 2
    gui_OldDefaultVisible = get(0,'defaultFigureVisible');
    set(0,'defaultFigureVisible','off');
    gui_hFigure = openfig(name, singleton);
    set(0,'defaultFigureVisible',gui_OldDefaultVisible);
else
    gui_hFigure = openfig(name, singleton, visible);
end

function result =...
    local_isInvokeActiveXCallback(gui_State, varargin)

try
    result = ispc && iscom(varargin{1}) ...
             && isequal(varargin{1},gcbo);
catch
    result = false;
end

function result = ...
    local_isInvokeHGCallback(gui_State, varargin)

try
    fhandle = functions(gui_State.gui_Callback);
    result = ...
 ~isempty(findstr(gui_State.gui_Name,fhandle.file)) || ...
             (ischar(varargin{1}) ...
             && isequal(ishghandle(varargin{2}), 1) ...
             && (~isempty(strfind(varargin{1},...
             [get(varargin{2}, 'Tag'), '_'])) || ...
                ~isempty(strfind(varargin{1},...
                '_CreateFcn'))) );
catch
    result = false;
end
```

APPENDIX B

THE SOLENOIDS' CONTROL PROGRAM

This Appendix presents a listing of the source code for the program used to control the solenoids motion. The program is written and compiled using Borland C++ 3.0 platform and it works under MS-DOS. The flow chart describing the logical flow of the program is shown in Figure B.1.

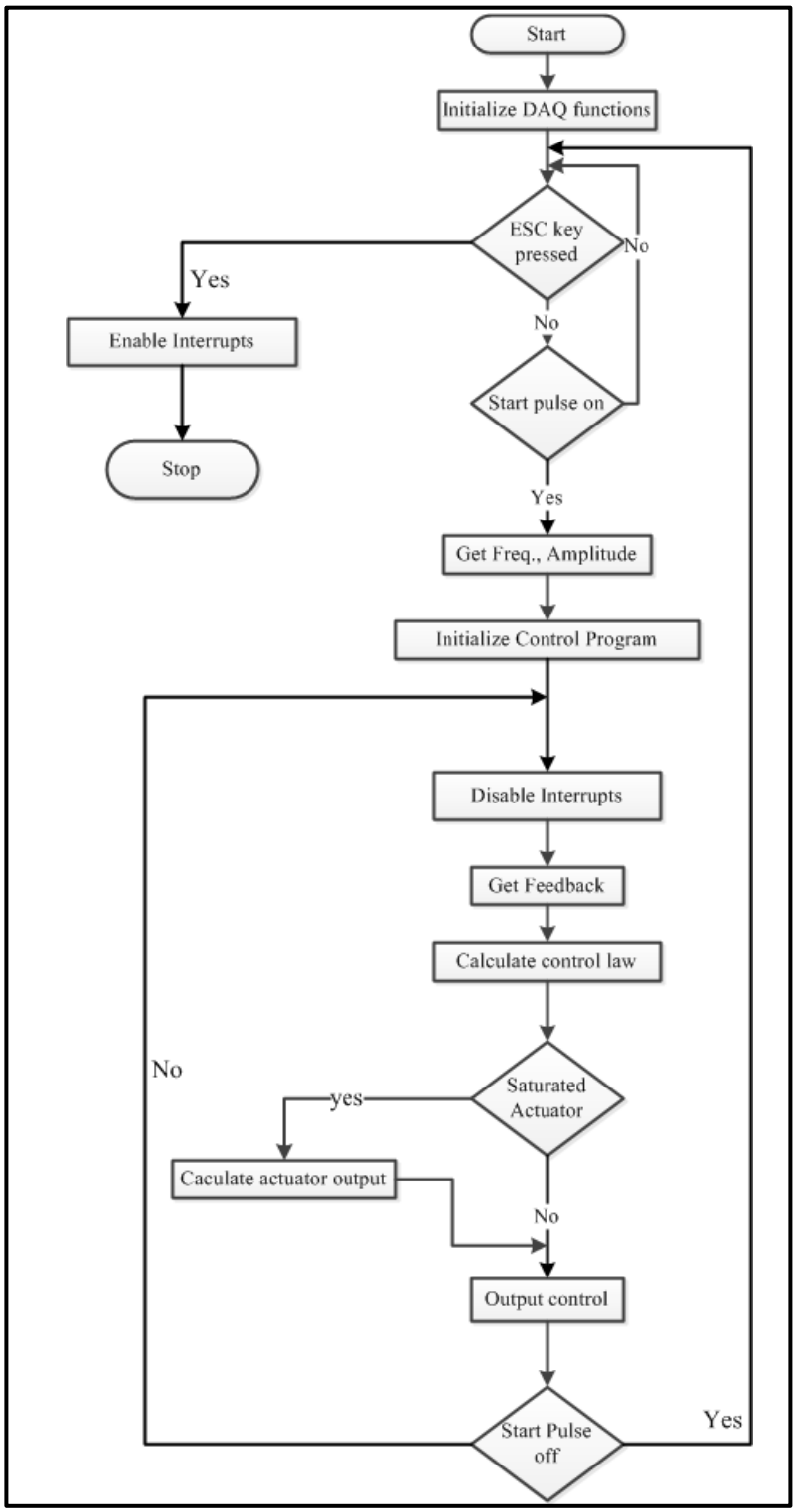


Figure B.1 Flow chart of the solenoids' control program.

```
/* ControlProgram.c source code */

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <dos.h>
#include <math.h>


extern pcl812(int, unsigned int *); // this is from 812cs.lib so you need to link
it

#define Tension                     2.0
#define MaxCurrent          4.0
#define AmpScale            0.4
#define ClockRate           1250
#define PotGain                     555.2941
#define PotOffset           0.039
#define TravelLength 0.017
#define Mass                0.36877
#define RefreshTime         3
#define g1                          7360.0        // FSEB gain
#define g2                          101.21        // FSEB gain
#define k1                          0.96881       // Kalman filter gain
#define k2                          409.25        // Kalman filter gain
#define k3                          33175.0       // Kalman filter gain
#define ArraySize           3750.0                // check the size

int toggle = 1;
unsigned int CycleCount,                          // auxilary varialbles
            MaxCycle,
            N,
            M;

float Feedback ;
double  Position,
              Force,
              RefPosition,
              ForceRight,
              ForceLeft,
              ExoForce,
              Error,
              CurrentRight,
              CurrentLeft,
              PositionError,
              PositionRight,
              PositionLeft,
              Position_1,
              Velocity,
              L1,
              L2,
              L3,
              L4,
              L5;
unsigned int param[60];                           // for DAQ intialization.
```

```
/********* function getvalue() is to get the feedback from the potentiometer
**********/

float getvalue (unsigned int Channel){
              unsigned int param[60];
              unsigned int data[10];
              unsigned int far * dat;
              float  DataBuf;

              dat = data;

              param[0]  = 0;
              param[1]  = 0x220;
              param[4]  = 2;
              param[5]  = 5;
              param[6]  = 100;
              param[7]  = 0;
              param[8]  = 0;
              param[10] = FP_OFF(dat);
              param[11] = FP_SEG(dat);
              param[12] = 0;
              param[13] = 0;
              param[14] = 1;
              param[15] = Channel;
              param[16] = Channel;
              param[17] = 0;


              pcl812(9, param);         /* Func 9 : Pacer trigger A/D conversion
*/
              if (param[45] != 0) {     /*          with interrupt data transfer
*/
                     printf("\nA/D PACER TRIGGER WITH INTERRUPT DATA TRANSFER
FAILED !");
                     exit(1);
                                              }
              do {
                     pcl812(10, param);            /* Func 10 : Check interrupt
status */
                     }while((param[46] & 1) != 0);      /* 0 : not active, 1 :
active       */

              DataBuf = data[0] & 0xFFF;
              DataBuf =( (10 - (-10)) * DataBuf / 4096) + (-10);

return(DataBuf);
} //getvalue


/********* function putvalue() is to send the control signal to the solenoids
**********/

void putvalue(unsigned int Channel, double output)
{
       unsigned int param[60];
       unsigned int far * dat;
       unsigned int data[100];
       unsigned int DigitalValue;
       DigitalValue =(output*4095)/10 ;
```

```
	dat = data;
	param[0] = 0;
	param[1] = 0x220;
	param[20] = FP_OFF(dat);
	param[21] = FP_SEG(dat);
	param[22] = 0;
	param[23] = 0;
	param[24] = 1;
	param[25] = Channel;
	param[26] = Channel;
	data[0] = DigitalValue;


	pcl812(13, param);	/* Func 13 : "N" times of D/A output	*/
	if (param[45] != 0) {
		printf(" D/A OUTPUT FAILED !");
		exit(1);
                                        }

}//putvalue


/********* function get_digital_value() is to get the control orders from the GUI
******/

unsigned char get_digital_value()
{
	unsigned int param[60];
	unsigned  char data[10];
	unsigned  char far * dat;
	dat = data;
	param[0] = 0;
	param[1] = 0x220;
	param[5]  = 50;
	param[6]  = 100;
	param[27] = FP_OFF(dat);
	param[28] = FP_SEG(dat);
	param[29] = 0;
	param[30] = 0;
	param[31] = 1;
	param[32] = 0;


	pcl812(20, param);
	if (param[45] != 0) {
	printf(" DIGITAL INPUT INITIALIZATION FAILED !");
	exit(1);
	}

	pcl812(21, param);
	if (param[45] != 0) {
	printf(" DIGITAL INPUT FAILED !");
	exit(1);
	}

	return data[0];
}//getdigitalvalue
```

```
/********* function put_digital_value() is to send feedback to the GUI **********/

void put_digital_value(char value){
unsigned int param[60];
unsigned char data[1];
unsigned char far * dat;

       dat = data;
       param[0] = 0;
       param[1] = 0x220;
       param[33] = FP_OFF(dat);
       param[34] = FP_SEG(dat);
       param[35] = 0;
       param[36] = 0;
       param[37] = 1;
       param[38] = 0;


       pcl812(28, param);        /* Func 28 : Digital output initialization*/
       if (param[45] != 0) {
           printf(" DIGITAL OUTPUT INITIALIZATION FAILED !");
           exit(1);
           }

       data[0]=value;

       pcl812(29, param);        /* Func 29 : "N" times of digital output  */
       if (param[45] != 0) {
           printf(" DIGITAL OUTPUT FAILED !");
           exit(1);
           }

}


/********************************* check for ESC
************************************/

void chkstatus (void){
        int ch;
        if(kbhit()){
                    ch = getch();
                    if (ch == 0x1B){
                    putvalue(0,0.0);
                    putvalue(1,0.0);
                    clrscr();
                    put_digital_value(0);
                    printf("THE PROGRAM HAS ENDED");
                    exit(1);
                    }
             }//if(kbhit)
}//chk statuse


/*******************************
main************************************************/

void main(void)
```

```
{
	unsigned int data[10];	/* Conversion data buffer
*/
	unsigned int far * dat;
	unsigned int Frequency;
	unsigned char Amplitudemm;
	float Amplitude;
	unsigned char start, nextstep;
	int ch;
	int i;
	int z;
	clock_t start1, finish1;
	double duration;
	clrscr();

	param[0]  = 0;		/* Board number				*/
	param[1]  = 0x220;	/* Base I/O address			*/
	param[4]  = 2;		/* IRQ level   : IRQ2			*/
	param[5]  = 16;		/* Pacer rate = 2M / (50 * 100) = 400 Hz  */
	param[6]  = 100;
	param[7]  = 0;		/* Trigger mode, 0 : pacer trigger	*/
	param[8]  = 0;		/* Non-cyclic				*/
	param[10] = FP_OFF(dat);  /* Offset of A/D data buffer A	*/
	param[11] = FP_SEG(dat);  /* Segment of A/D data buffer A	*/
	param[12] = 0;		/* Data buffer B address, if not used,	*/
	param[13] = 0;		/* must set to 0.			*/
	param[14] = 1;		/* A/D conversion number		*/
	param[15] = 1;		/* A/D conversion start channel		*/
	param[16] = 1;		/* A/D conversion stop channel		*/
	param[17] = 0;		/* Overall gain code, 0 : +/- 5V	*/
	dat = data;


	pcl812(3, param);	/* Func 3 : Hardware initialization	*/
	if (param[45] != 0) {
	printf(" DRIVER INITIALIZATION FAILED !");
	exit(1);
					}
	pcl812(12, param);	/* Func 12 : D/A initialization		*/
	if (param[45] != 0) {
	printf(" D/A INITIALIZATION FAILED !");
	exit(1);
					}
	pcl812(4, param);	/* Func 4 : A/D initialization		*/
	if (param[45] != 0) {
	printf("\nA/D INITIALIZATION FAILED !");
	exit(1);
					}

	while(1)
	{
		chkstatus();
		gotoxy(24,11);
		printf("--------------------------------");
		gotoxy(24,12);
		printf("PLEASE HIT START ON THE MATLAB PC");
		gotoxy(24,13);
		printf("           <<<OR>>>             ");
		gotoxy(24,14);
```

```
cprintf("       Press ESC to exit");
gotoxy(24,15);
printf("--------------------------------");
chkstatus();
start =get_digital_value();
while(start != 128){
             start =get_digital_value();
             chkstatus();
                                  }
put_digital_value(1);
while((start&0xC0) != 0xC0){
             start =get_digital_value();
             chkstatus();
                    }

Frequency = get_digital_value();
Frequency = Frequency & 0x3F;
put_digital_value(2);
while((start&0xC0) != 0x80){
       start =get_digital_value();
       chkstatus();
       }

Amplitudemm = get_digital_value();
Amplitudemm = Amplitudemm&0x3F;
Amplitude = 0.001 * Amplitudemm;
put_digital_value(3);

while(start!=129 )
{
       start =get_digital_value();
       chkstatus();
}

while(start == 129)
{                            //while (start  129-1)
       clrscr();
       chkstatus();
       disable();
       gotoxy(12,8);
       cprintf("FRICTION LINEAR MEASURMENT SYSTEM ");
       gotoxy(12,10);
       printf("Frequency = %d     [Hz]",Frequency);
       gotoxy(12,11);
       printf("Amplitude = %.3f [m]",Amplitude);

       gotoxy(30,24);
       cprintf("Press ESC to exit");

       ExoForce=0.0;
       Position_1= 0.0;
       Velocity= 0.0;

       L1=k1 + k2/ClockRate + k3/pow(ClockRate,2)/2/Mass;
       L2=k2 + k3/ClockRate/Mass;
       L3=k3;
       L4=1/pow(ClockRate,2)/2/Mass;
       L5=1/ClockRate/Mass;
```

```
                    Feedback = getvalue(1);
                    Feedback = (Feedback-PotOffset)/PotGain;
                    while (start==129)
                    {
      //start(129-2)
                           chkstatus();
                           Feedback = getvalue(1);
                           Feedback = (Feedback-PotOffset)/PotGain;

                           /* calculate the kalman filter coefficient*/
                           RefPosition =  (Amplitude*
sin(2*3.14*Frequency*CycleCount*0.00079375)+0.008);
                           Error = (Position_1-Position);
                           Position =
Position+Velocity/ClockRate+L4*(Force+ExoForce)+L1*Error;
                           Velocity = Velocity+L5*(Force+ExoForce)+L2*Error;
                           ExoForce = ExoForce+L3*Error;
                           Position_1=Feedback;


                           /* calculate the control (the force)*/
                           Force = - g1*(Position-RefPosition) - g2*Velocity -
ExoForce;
                           if (Force > 0 )
                           {
                                  ForceRight = Tension;
                                  ForceLeft = Force+Tension;
                           }
                           else
                           {
                                  ForceRight = -Force+Tension;
                                  ForceLeft = Tension;
                           }

                           PositionRight = Feedback + Velocity/ClockRate;
                           if (PositionRight < 0.0) PositionRight = 0.0;
                           if (PositionRight > TravelLength) PositionRight =
TravelLength;
                           PositionLeft = TravelLength - PositionRight;


                           /* calculate output signals*/
CurrentRight=sqrt(fabs((pow(ForceRight, 2)+45.699*ForceRight) *
(47.003*pow(PositionRight,2) + 0.4312*PositionRight+0.000204)));
                           if (CurrentRight > MaxCurrent)
                           {
                                  CurrentRight = MaxCurrent;
Force= -Tension + 22.835 - sqrt(fabs(521.423+pow(MaxCurrent,2)
/(47.003*pow(PositionRight,2) + 0.4312*PositionRight+0.000204)));
                           }

CurrentLeft=sqrt(fabs((pow(ForceLeft, 2) +
45.699*ForceLeft)*(47.003*pow(PositionLeft,2) + 0.4312*PositionLeft+0.000204)));
                           if (CurrentLeft > MaxCurrent)
                           {
                                  CurrentLeft = MaxCurrent;
Force= -Tension + 22.835 - sqrt(fabs(521.423+pow(MaxCurrent,2) /
(47.003*pow(PositionLeft,2) + 0.4312*PositionLeft+0.000204)));
                           }
```

```
                              /* send the control signals*/
                              putvalue(1,CurrentLeft/AmpScale);//needs to be replaced
                              putvalue(0,CurrentRight/AmpScale);//needs to be
replaced
                              start =get_digital_value();
                              CycleCount++;
                        }//start(129-2)
                }//while(129-1)
                putvalue(0,0.0);
                putvalue(1,0.0);
                put_digital_value(0);
                clrscr();

        }//while(1)

}//main
```

APPENDIX C

SCHEMATICS OF THE APPARATUS

The information presented in this Appendix describes the electronic interface circuitry that was designed specifically for this thesis and is intended to supplement the data presented in the hardware description in section 2.1.

The interface circuitry consists of three differential amplifiers and three filters. Two amplifiers are designated for the accelerometers and one for the Hall Effect sensor. Although the schematic in Figure C.2 shows one combined circuitry, the interface was built on a prototype PCB and a bread board.

Resistors R1, R2, R3, and R4 with IC1 (AD797 op-amp) and the variable resistor R5 form the differential amplifier used for the Platform Accelerometer, the gain of this amplifier is R1/R2. The Test Mass amplifier consists of resistors R6, R7, R8, and R9 with IC2 (AD797 op-amp) and the variable resistor R10, the gain of this amplifier is R8/R6. The $\pm$15 supply voltages and the 5.00 Volt reference are supplied through an independent power supply. Both amplifiers are constructed on a bread board, see Figure C.1 (a).

The Hall effect's circuitry was built on a prototype PCB, see Figure C.1 (b), and consists of a differential amplifier and consists of R11, R12, R13, and R14 with IC3 (A D797 op-amp). The $\pm$15 Volts supply voltages are routed from the accelerometers' circuit and the 5.00 Volts reference is provided through a 78L05Z voltage regulator. An anti-aliasing low pass filter has been connected to the output of the Hall effect sensor. The filter is one stage RC circuit (R17, C9) with a cutoff frequency of approximately 53 Hz.

The Accelerometer is mounted on a small PCB (15mm x 15 mm), Figure C.1 (c).

On the PCB board, there are three capacitors C1, C2, and C3. Capacitor C1 is to decouple the accelerometer from the power supply. Capacitors C2, C3 are to set the bandwidth of the accelerometer at 50Hz. Capacitors C6, C5, and C4 have the role as capacitors C1, C2, and C3 respectively. Both accelerometers' circuits have been placed in copper boxes to keep them protected from outside noise.

Figure C-3 shows the schematic of the PCB designed and built by Alexander Rochvarg as an electronic interface to control the solenoids. The circuit consists of the power supply, two voltage controlled current sources, precision reference, and a scaling amplifier [4].

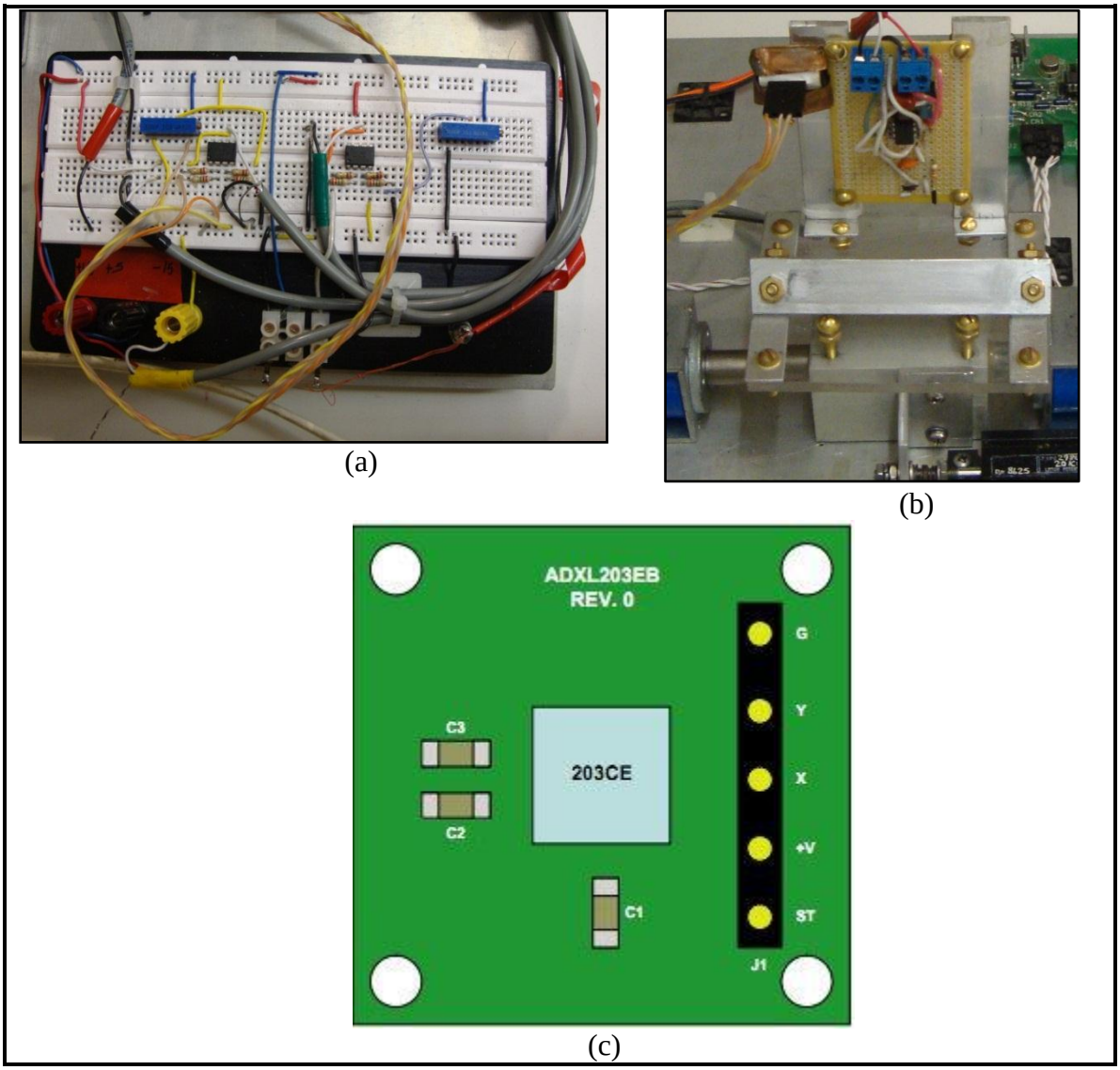


Figure C.1 (a) The bread board with the Accelerometers' circuitry, (b) The prototype PCB with the Hall effect sensor's circuitry, (c) The accelerometer printed circuit.

Source (c): [10].

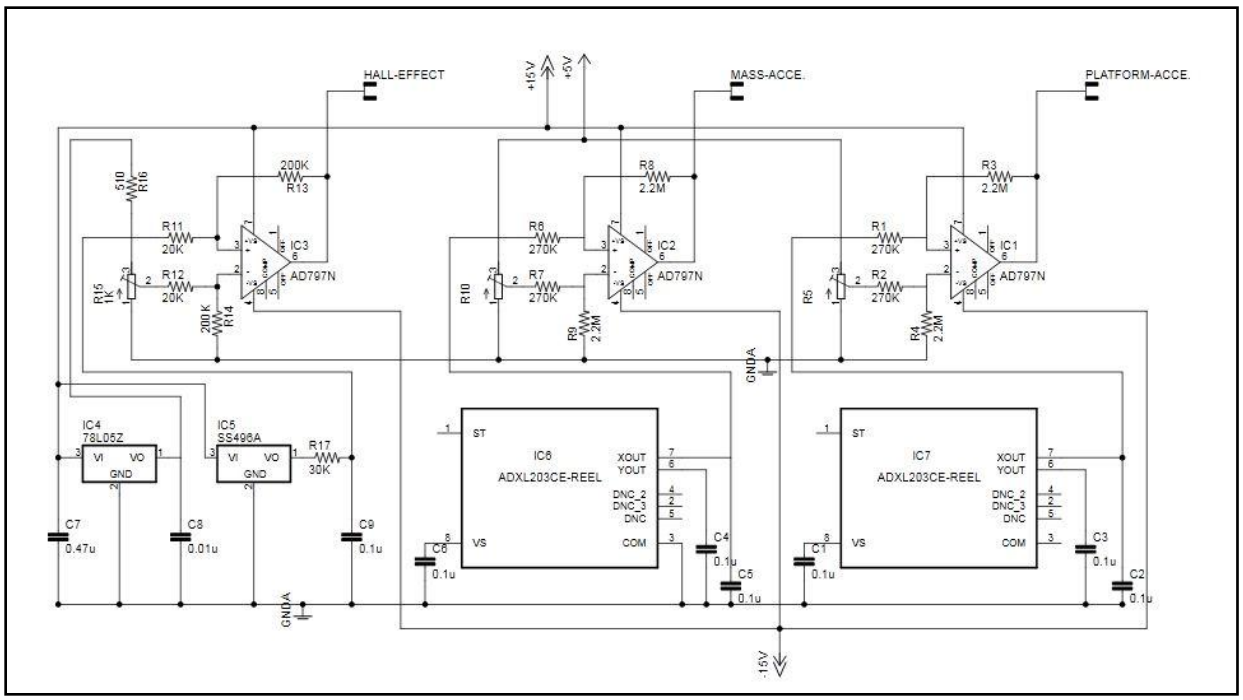


Figure C.2 Schematic of the electronic interface.

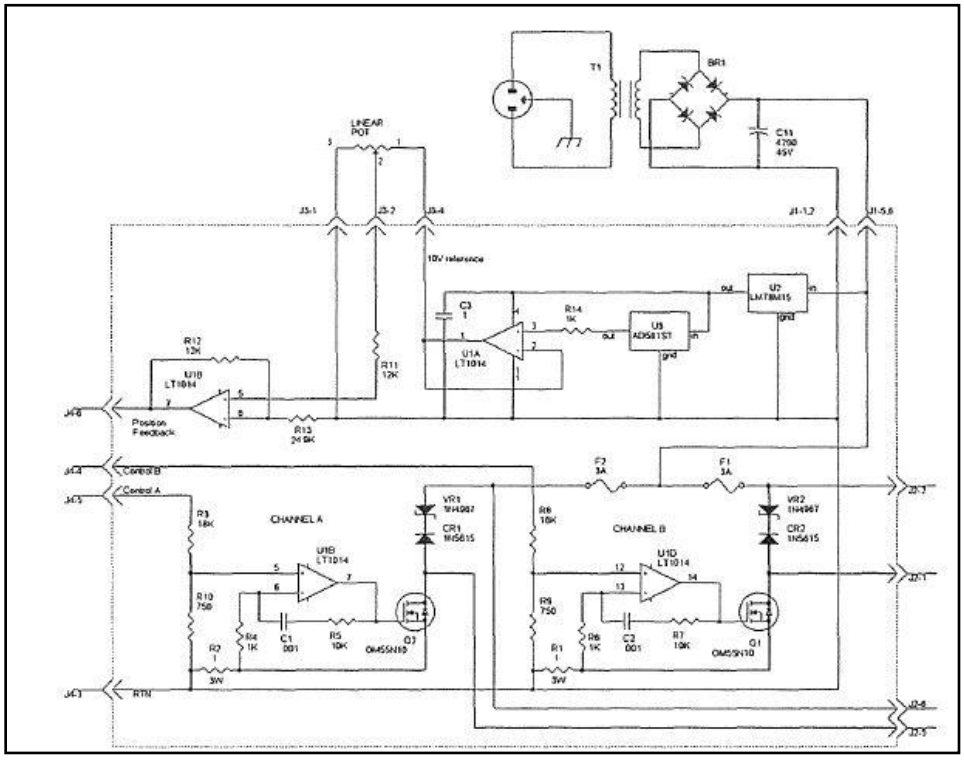


Figure C.3 Schematic of the solenoids' electronic interface.

Source: [8]

APPENDIX D

SHORT GUIDE TO OPERATE THE APPARATUS

This section presents a short guide on how to connect and operate the apparatus.

Requirements:

Before starting the apparatus the user needs these things:

To run the graphical user interface of this thesis the user needs:

1- A personal computer that has windows 7 and MATLAB 7.8.0.347 properly installed. This computer will run the graphical user interface (main.m)

2- A personal computer that has MS-DOS 5.00 properly installed. This computer will run the control program.

3- The MATLAB® m-file main.m which is listed in Appendix A.

4- The control program which is listed in Appendix B.

5- DATA TRANSLATION DT9802 data acquisition card.

6- DT9802 Driver which is found on the "Data Acquisition Omni Software" CD that comes with card.

7- DATA TRANSLATION DAQ adapter for MATLAB® which can be downloaded from [6]

8- Advantech PCL812PG Data acquisition card.

Connections:

The connections between the different components of the apparatus are illustrated in Figure E.1.

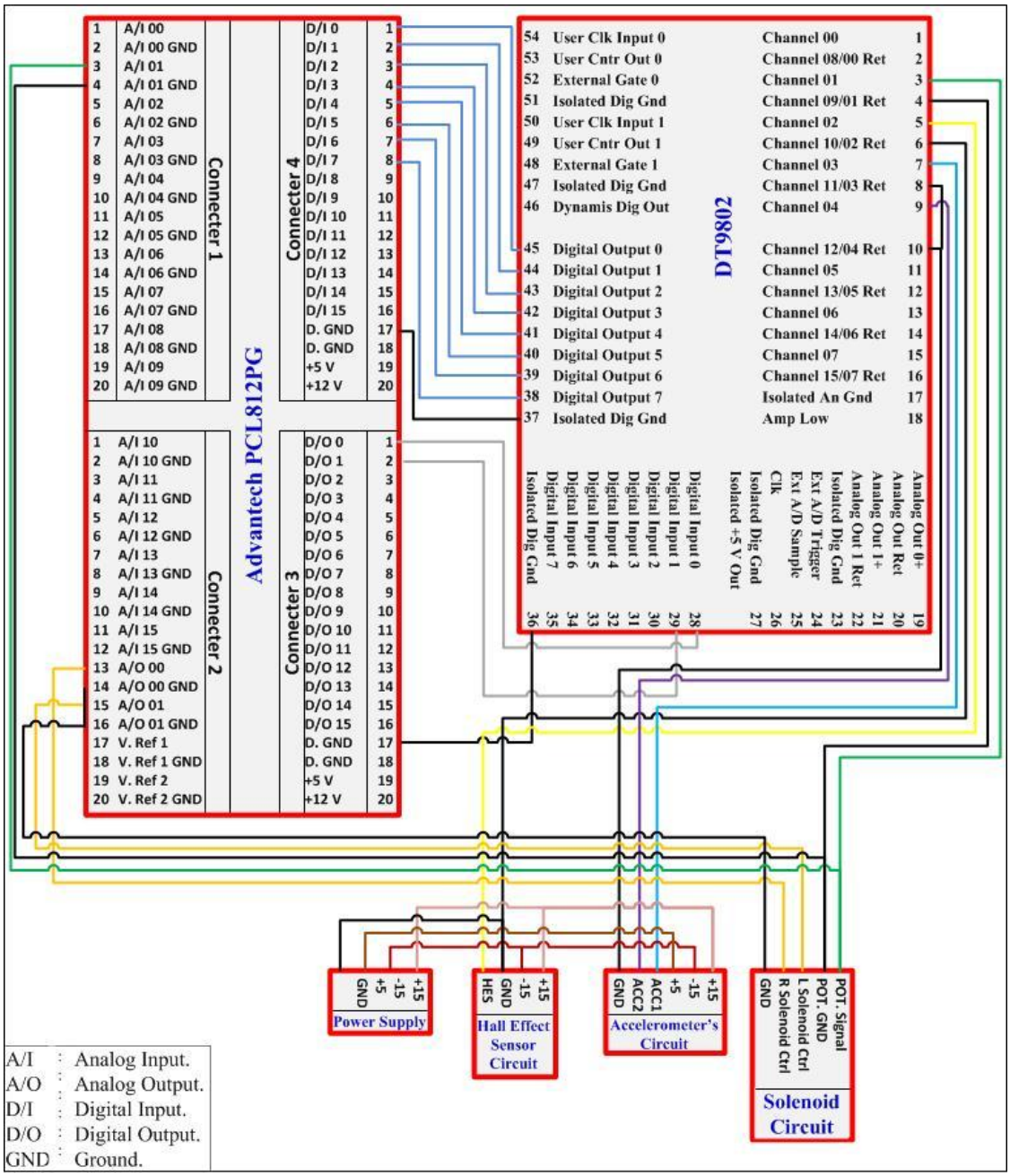


Figure D.1 The connections between the different components of the apparatus.

Calibration of the Hall Effect Sensor:

1- Remove the two hex nuts on the right side of the triangular rail, but do not remove the screws because they will be used to fix the micrometer. This step is shown in Figure E.2.

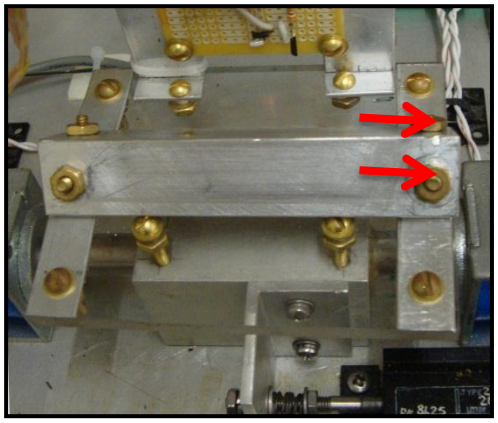


Figure D.2 Remove the hex nuts.

2- Install the micrometer on the triangular rail. Figure E.3 illustrates this step.

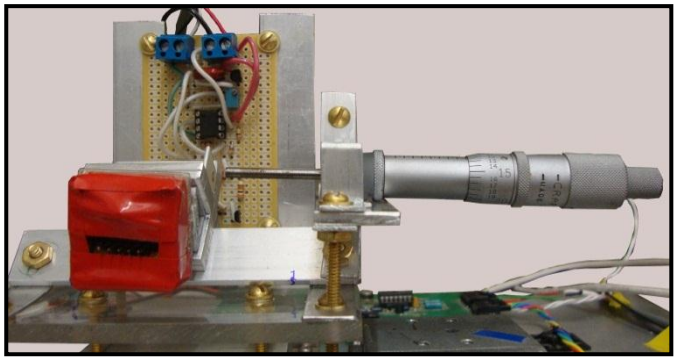


Figure D.3 The micrometer installed on the triangular rail.

3- Run the program caliber.m on MATLAB®. The program source code is listed in Table E.1. At the end of this step, the user will have two vectors one for distances and the other for the Hall effect sensor output values.

4- Use MATLAB® curve fitting tool to derive the analytical function that describes the relation between the distance and the Hall Effect system output.

5- Put the analytical function in the MATALB® model ("Data_Handling Model") associated with the graphical user interface.

Table 3 Caliber.m: a MATLAB® Program to Facilitate the Calibration Process

```
% This program requires the DT9802 to be connected
% Connect the Hall Effect Sensor to analog channel 2.
% Use the micrometer to push the Test Mass for 0.025"
% distances
%%
clear all;
clc;

startDistance = 0.0; % the beginning distance
DAQInput= analoginput('dtol');
addchannel(DAQInput,2); % connect to channel 2
DAQInput.inputtype='Differential';
for i = 1 : 58
    x(i) = (i -1) *0.025 + startDistance;
    for j= 1 : 3
    HESensorOutput(i,j) = getsample(DAQInput)
    end
    disp('Please move the Test Mass, then press any key');
    pause
end
figure;
plot (x , HESensorOutput(1,:));
hold on;
plot (x , HESensorOutput(2,:),'r');
plot (x , HESensorOutput(3,:),'g');
xlabel('Distance [Inch]');
ylable('The Hall effect sensor output');

HESensorOutput1= HESensorOutput(1,:); %you can choose any
                                      %of the three columns
                                      %of HESensorOutput
```

Starting the apparatus:

1- Connect the apparatus according to Figure E.1.

2- Start the GUI by running the main_program.m file. (copy the file listed in Appendix A, and paste it on a MATLAB® editor, save it under the name main_program.m then hit F5 or the Run button)

3- Start the solenoid's control program. (Compile the control program, listed in Appendix B using any C++ programing platforms, then take the execution file (.exe file) and run it on an MS-DOS 5.0 computer.

4- Calibrate the apparatus. (optional)

5- Check the outputs of the Accelerometers systems and the Hall effect system, they should all be zero. If not, adjust the variables resistors until the output is zero.

6- On the GUI, fill the parameter field with the desired value. Please, make sure the values are integers and in the range indicated beside each input field.

7- Hit the *Start* button. When the platform is moving this button will be disabled.

8- When the platform stops moving. Hit the *Calculate* button to get the velocity and acceleration signals. When the calculation stops the phrase beside the calculate button will change to "done".

9- Fill the desired period for plotting the friction coefficient vs. velocity curve. Hit the plot button, the curve will be shown on a new Figure window.

10- To save the data from the experiment, go to the *Data* menu tab > save.

11- To load preveously saved data, go to the *Data* menu tab > load.

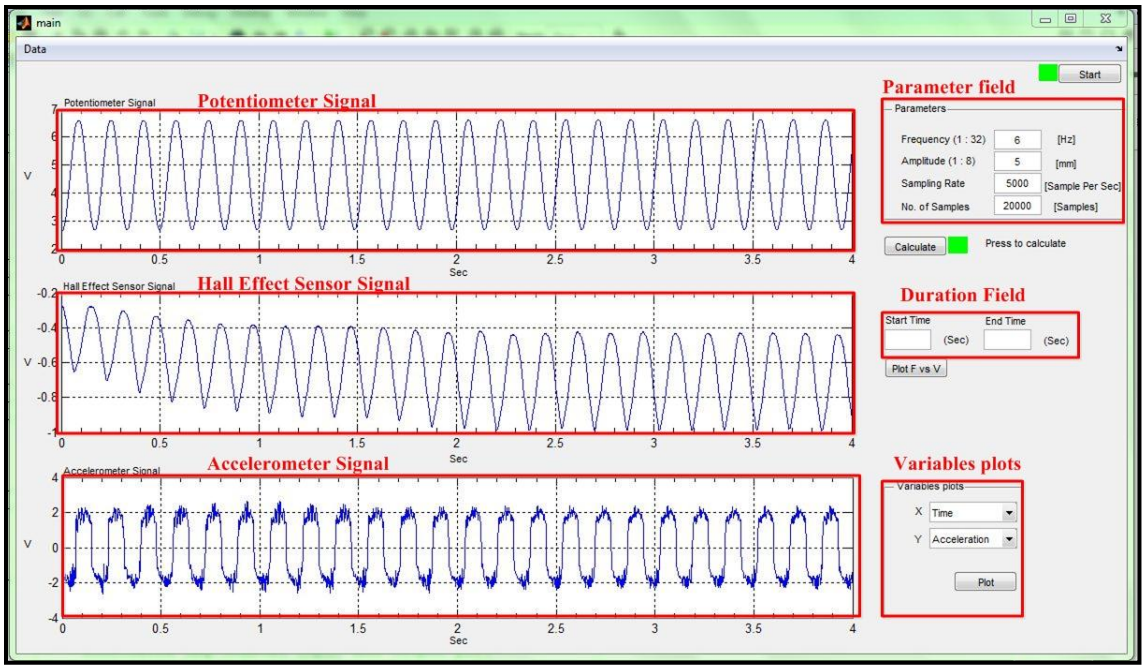


Figure D.4 The graphical User Interface used in this research. The parameters field is for the conditions of the experiment, those include the frequency, amplitude, sample rate and the number of samples. The three axes fields are for plotting the raw data collected from the sensors. The duration field is for determining the time portion on which the output curve would be plot. The Variables plots field is for plotting any signal individually versus time or sample number.

REFERENCES

[1] B. Armstrong-Helouvry, "Friction in Machines", in *Control of Machines with Friction*, Boston, MA: Kluwer Academic press, 1991, ch.2, pp. 5-42.

[2] H. J Güntherodt H. J Güntherodt .,H. Burkhart ,M. Guggisberg , T. Gyalog , Meyer E., Benedikt W. , (2011), *Friction Module* [Online], Retrieved June, 16, 2011, http://www.nano-world.org/frictionmodule/content/0200makroreibung/0400historisch/?lang=en

[3] http://www.tribology-abc.com (2011), *History of Friction Science* [Online], Retrieved June 16, 2011,http://www.tribology-abc.com/abc/history.htm

[4] E. Rabinowicz , "The Nature of the Static and Kinetic Coefficients of Friction", *J. Applied Physics*, Vol. 22, Iss. 11, pp. 1373-1379, Nov. 1951.

[5] Dahl P. R., "Measurement of Solid Friction Parameters of Ball Bearings", The Aerospace Corporation, El Segundo, CA. Tech. Rep. TR-077(2991-03)-3, 1977.

[6] A. Harnoy , B. Friedland , S. Cohn, "Modeling and Measuring Friction Effects. Physics, Apparatus, and Experiment*". IEEE Control Syst. Mag.*, Vol. 28, No. 6, Dec. 2008. pp. 82-91.

[7] B. Friedland et al. "Apparatus for Empirical Determination of Dynamic Friction,*"* in *Proc. of the American Control Conf.*, Baltimore, MD. June 1994, pp. 545-550.

[8] A. Rokhvarg "Single-Axes Agnostic Motion Control System with Electro-Magnetic Actuators". M.S. Thesis, Dept. of ECE, NJIT, Newark, NJ., 1996.

[9] B. Friedland , "Parasitic Effects," in *Advanced Control Systems Design*, Prentice Hall, Upper Saddle River, NJ. 1996, ch. 7, sec. 1, pp. 169-198.

[10] Analog Devices Inc., (2004), Dual Axis Accelerometer Evaluation Board, Retrieved June 2011, http://www.analog.com/static/imported-files/eval_boards/535395787ADXL203EB_0.pdf