Winter 1-16-2019

Design of Vibration Class Laboratory Assignments: Focusing on Broader Application

Ethan Wayne Cating

Design of Vibration Class Laboratory Assignments:

Focusing on Broader Application

A Thesis

Submitted to the Faculty

of

Rose-Hulman Institute of Technology

by

Ethan Wayne Cating

In Partial Fulfillment of the Requirements for the Degree

of

Master of Science in Mechanical Engineering

February 2019

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Final Examination Report

Ethan Cating — Name

Mechanical Engineering — Graduate Major

Thesis Title **Design of Vibration Class Laboratory Assignments: Focusing on a Broader Application**

DATE OF EXAM: January 16, 2019

EXAMINATION COMMITTEE:

	Thesis Advisory Committee	Department
Thesis Advisor:	Daniel Kawano	ME
	Phillip Cornwell	ME
	Jeffery Leader	MA

PASSED X **FAILED** _____

Abstract

Cating, Ethan Wayne

M.S.M.E.

Rose-Hulman Institute of Technology

February 2019

Design of Vibration Class Laboratory Assignments: Focusing on Broader Application

Thesis Advisor: Dr. Daniel Kawano

Courses in vibration taught at Rose-Hulman Institute of Technology are designed to blend classroom discussion of the science of vibration with practical, hands-on applications. The laboratory portion of the undergraduate course in vibration focuses on two areas: (1) system identification for single- and multi-degree-of-freedom systems in the frequency domain through sine-sweep shake testing; and (2) identifying natural frequencies and mode shapes for continuous systems via experimental modal analysis using frequency response data collected through a roving hammer test. Modal analysis is performed using an in-house MATLAB-based program known as Easy Modal Analysis Program (EMAP).

The goals of this thesis are two-fold: (1) to create new laboratory exercises to allow students to explore elements of base motion, rotating unbalance, and structural health monitoring; and (2) to expand the functionality of EMAP to allow students to explore the damping characteristics of a system via different methods. The development of these additional laboratory exercises and the expanded functionality of EMAP serve to broaden the students' practical knowledge of topics in

vibration measurement and analysis.

Keywords: “vibration”, “modal analysis”, “frequency response”, “base motion”, “rotating unbalance”

Dedication

This thesis is dedicated to my Lord and Savior Jesus Christ, my beautiful and devoted wife, Shiloh, our wonderful daughter Essie, and my family and friends. Without their support and encouragement this would not have been possible. Thank you all so very much.

Acknowledgements

I would like to extend my sincerest gratitude to my advisor, Dr. Daniel Kawano, on his assistance, mentorship, and patience through this process.

Table of Contents

Contents

List of Figures

List of Tables

List of Abbreviations

CMIF	Complex Mode Indicator Function
DAQ	Data Acquisition
ECP	Education Control Products
EEP	EMAP Expansion Pack
EMAP	Easy Model Analysis Program
FBD	Free Body Diagram
FRF	Frequency Response Function
GUI	Graphical User Interface
GUIDE	Graphical User Interface Development Environment
I/O	Input and Output
KD	Kinematic Diagram
MATLAB	Matrix Laboratory
PWM	Pulse Width Modulation
SHM	Structural Health Monitoring
SISO	Single-input Single-output
TMD	Tuned-Mass Damper

List of Symbols

English Symbols

cm	Centimeters
ft	Feet
g	Grams
Hz	Hertz
in	Inch
lb	Pounds
oz	Ounces
s, sec	Seconds

Greek Symbols

ϕ	Phase Offset
ω	a) Input Frequency
	b) Angular Velocity
ω_n	Natural Frequency
$\omega_n^{(1)}$	Lower Half-Band Power Frequency
$\omega_n^{(2)}$	Upper Half-Band Power Frequency
π	3.14159265

θ	Angular Position
ζ_n	Modal Damping Ratio

Mathematical Symbols

A_i, B_i, C_i, & D_i	Polynomial Coefficients
E	Modulus of Elasticity
e	Eccentricity of Rotating Offset Mass
$\underline{\mathbf{f}}(t)$	Forcing Input Vector
h	X-axis Offset of Circle Center
I	Rotational Moment of Inertia
$\underline{\mathbf{K}}$	Stiffness Matrix
k	a) Y-axis Offset of Circle Center
	b) Stiffness of Building Support Beams
k_d	Stiffness of Tuned-Mass Damper
k_{eq}	Equivalent Support Beam Stiffness
L	Length of Support Beams
$\underline{\mathbf{M}}$	Mass Matrix
$\underline{\mathbf{m}}$	Forcing Input Mass Vector
m, m_{floor}	Mass of Building Floor
m_2	Equivalent mass of building story, TMD weight and cart
m_d	Mass of Tuned-Mass Damper
m_e	Mass of Eccentric Motor Mass

m_{eq}	a) Equivalent Mass of Floor and Support Beams
	b) Equivalent Mass of Motor & Housing & Cart System
m_{beam}	Mass of Building Support Beams
P_i	Polynomial Equation
r	a) Radius of Circle
	b) Radius of Offset Mass
	c) Ratio of Input Frequency to Natural Frequency
t	Time
$\underline{\mathbf{u}}$	Position Related Frequency Vector
$\underline{\ddot{\mathbf{u}}}$	Acceleration Related Frequency Vector
x	a) Real FRF Data for Circle Fit
	b) X-axis Spline Point Data
X	Amplitude of Movement from Rotational Unbalance
$\ddot{x_1}$	Acceleration of First Floor of Building
x_2	Position of Second Floor of Building
x_d	Position of Tuned-Mass Damper of Building
$\underline{\mathbf{x}}_{ss}$	Steady-State Position Response Vector
$\underline{\ddot{\mathbf{x}}}$	Acceleration Vector
$\ddot{x_1}$	Acceleration of First Floor of Building
$\ddot{x_2}$	Acceleration of Second Floor of Building
$\ddot{x_d}$	Acceleration of Tuned-Mass Damper of Building
y	a) Y-axis Spline Point Data

	b) Position of First Floor of Building
	c) Position of Base Motion Input to Building
y_0	Base Motion Input Amplitude
$\bar{y}$	Centroid of Rotating Mass
$\ddot{y}$	Acceleration of Base Motion Input to Building
z	Z-axis Spline Point Data

1 Introduction

One of the greatest strengths that Rose-Hulman offers in its undergraduate engineering curriculum is its ability to focus a student's education on visual and hands-on learning. The Department of Mechanical Engineering's introductory vibrations class, EM406 - Vibration Analysis, is no exception. The goal of the vibrations course is to help give the student a better understanding of how a system is affected by oscillatory motion or harmonic forcing, how the system reacts, and how the system can be changed to better suit itself to those conditions.

In general, the course lays the foundation for theoretically analyzing single- and multi-degree-of-freedom models in both free and forced motion. Other concepts taught include vibration isolation, vibration absorption, rotating unbalance, and the use of vibration measurement instrumentation. Students are then able to apply this knowledge to laboratory assignments. Many classes teach students how to apply equations to given situations, but engineering problems cannot always be initially solved on paper and are not usually ideal.

The goal of this thesis is to increase hands-on learning by connecting real world applications in vibration analysis through strategically designed lab assignments. The labs focus the students to: (1) identify the use for vibration analysis in the problem; (2) analytically model the behavior of the system; (3) test the system, visually see what the vibration is doing to the system, and record data with the necessary instrumentation; and (4) compare the experimental and analytical results. The thesis was partitioned into two particular areas. The first was to expand the functionality of the MATLAB-based EMAP (Easy Modal Analysis Program) software used to perform experimental modal analysis in the lab. The second concerns the design of several labs that advance students'

hands-on learning of topics in vibration.

In Chapter 2, the expanded functionality of EMAP is explained and shown in the created application EEP (EMAP Expansion Pack). The program gives students the ability to visualize different methods of SISO (Single-input, Single-output) damping estimation. It allows the user the ability to look at every FRF (Frequency Response Function) collected at different points on a structure and at different natural frequencies. It also plots mode shapes of the object that was modally tested with splines to enhance their appearance compared to EMAP's default plotting.

In Chapter 3, base motion is studied by examining the vibration of a model building. For this, existing laboratory hardware (a mass-spring-damper apparatus designed by Education Control Products (ECP)) is utilized. The hardware is repurposed into a shaker table to visually and experimentally find the response of a two-story structure from base motion. An accelerometer is placed on each building floor for data collection. The real-time data recording program RT Pro along with the Photon II DAQ (Data Acquisition) unit is used to record the accelerometer data from each floor during testing. The base input position is also recorded from encoders on the shaker. An analytical model is created to compare with the experimental results. Finally, experimental and analytical mode shapes are also compared.

In Chapter 4, a tuned-mass damper (TMD) is attached to the second floor of the same model building from Chapter 3. The TMD is used to help remove excessive vibration from the building while shaking at its natural frequency. Different TMD weights are tested and the reduction in acceleration amplitudes is compared. The actual stiffnesses of the springs in the TMD are calculated based on the TMD weight with the greatest reduction in amplitude. The change in frequency response is analyzed by the changes in the frequency response function (FRF) of the model. The same procedure and data collection techniques are used as in Chapter 3.

In Chapter 5, two labs are discussed. Both of these labs were less developed because of time constraints and issues identified during testing. The applications of the labs are in rotational unbalance and structural health monitoring. The rotational unbalance lab makes use of the ECP station for its built-in positional encoders. A video game controller rumble motor, with offset eccentric plates, runs and forces the ECP cart to move from the motor's induced vibration. The recorded amplitude data is then compared with the calculated amplitude. The recorded amplitude data is also used to find the mass of the eccentric plates on the motor and compare to the actual motor offset mass. The structural health monitoring lab is based on using a black box structure that is intentionally flawed. A default configuration and several flawed configurations are tested and studied via modal analysis to try to identify the flaws or possible causes in the different frequency responses. Details on the default configuration would be known, but not for the other configurations. Suggestions for potential improvement(s) are given for each.

2 Modal Analysis Applications

2.1 Easy Modal Analysis Program

One of the major tools for use by students to analyze the frequency response of lightly damped structures is an in-house MATLAB program. This program is called the Easy Modal Analysis Program or EMAP [1]. EMAP is the last step of the experimental modal analysis test procedure in analyzing the vibrational excitation of a structure. This process starts with the structure being tap tested with a roving hammer. That hammer has a sensor attached to measure the force of impact into the structure. Simultaneously, an accelerometer placed on the structure senses the excitation from the hammer impacts at different points of the structure. The accelerometer on the structure remains stationary. This is a SISO (Single-input, Single-output) test setup. During tap testing, a Photon II unit is used to analyze the data from the analog sensor signals. After several physical points on the structure have been tested and recorded, the data is output from RT Pro. The data output from RT Pro are generated FRF (Frequency Response Function) files from each test point. EMAP uses each FRF file and calculates them into the CMIF (Complex Mode Indicator Function) for the structure. The CMIF is useful for the identification of modes and mode shapes over a matrix set of FRFs. The CMIF is calculated by taking the singular value decomposition of the absolute value of all of the FRF data. After this point students are able to see the peaks of the CMIF designating the structure's natural frequencies. The corresponding mode shapes are computed via peak picking of the imaginary part of the FRFs.

2.2 EMAP Expansion Pack

EEP (EMAP Expansion Pack) expands on the functionality of EMAP. EEP lets the user select the frequency range of interest from the CMIF plot, usually around a peak denoting a natural frequency, for further inspection. Within this range the user is then able to look at each FRF independently. This helps students see how a few methods of parameter estimation of a structure are analyzed. All methods are SISO specific tools geared to looking specifically at an FRF. All of these methods use exactly the same data, but are graphically represented in different ways. These tools are a basic magnitude plot, Co-Quad plot (real and imaginary plots), and circle-fit method. Data may not always be ideal enough for the computer to calculate parameters. EEP has a secondary peak-picking routine to aid in identifying uncertain changes in FRF data sets. Lastly, EEP also generates a graph of the mode shape of the natural frequency selected within the range. It plots the same mode shapes next to each other, but each uses a different line fit technique. Creating more aesthetic and realistic graphs was an interest and not a major goal of this project.

2.2.1 Using EEP

EEP is designed for use after modal testing is completed and the data is compiled in EMAP. The MATLAB Graphical User Interface Development Environment (GUIDE) is used to create the EEP program. GUIDE allows user interfaces to be graphically and interactively designed and then automatically generates the MATLAB code to run it and to later modify its behavior. When EEP is run in MATLAB, either from the command line or opening the GUIDE figure, the initial window opened is the same as shown in Figure 2.1. The only button that is enabled to use is to let the user select the .mat file for the structure they are studying. This .mat file is the same file that is

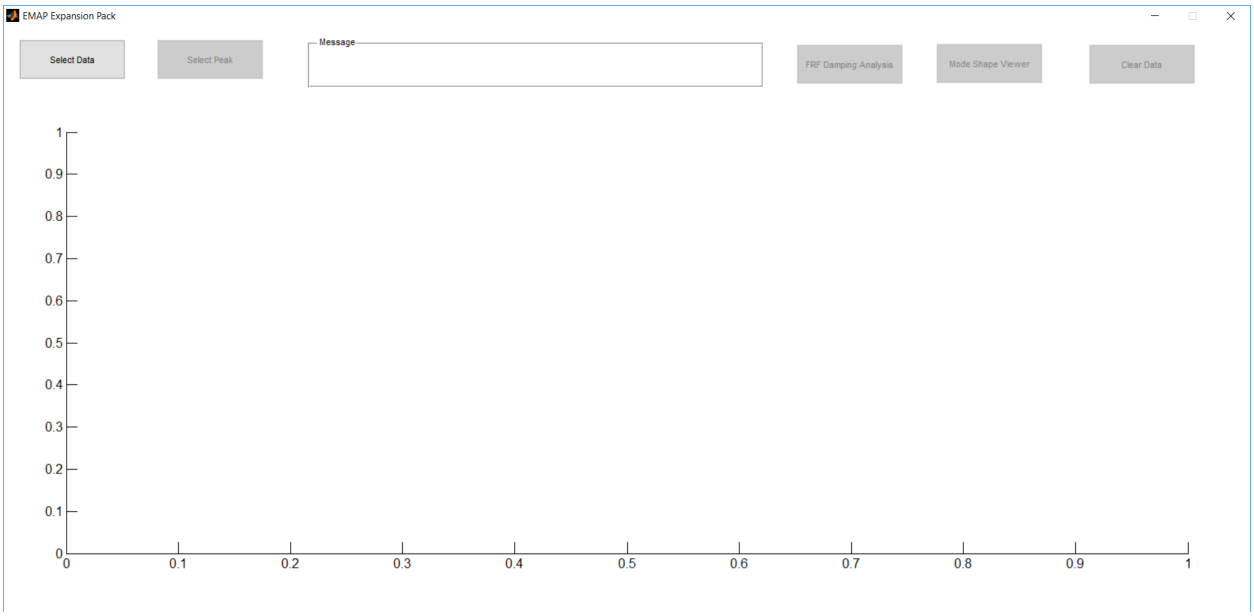


Figure 2.1: EEP start window.

generated after running the accelerometer data through EMAP. These files contain all of the CMIF, FRF, and mode shape data needed for use in EEP. While working with EEP a set of messages at the top of each window used help the user follow through the EEP workflow. Figure 2.2 shows the user selection window that pops-up after pressing the 'Select Data' button.

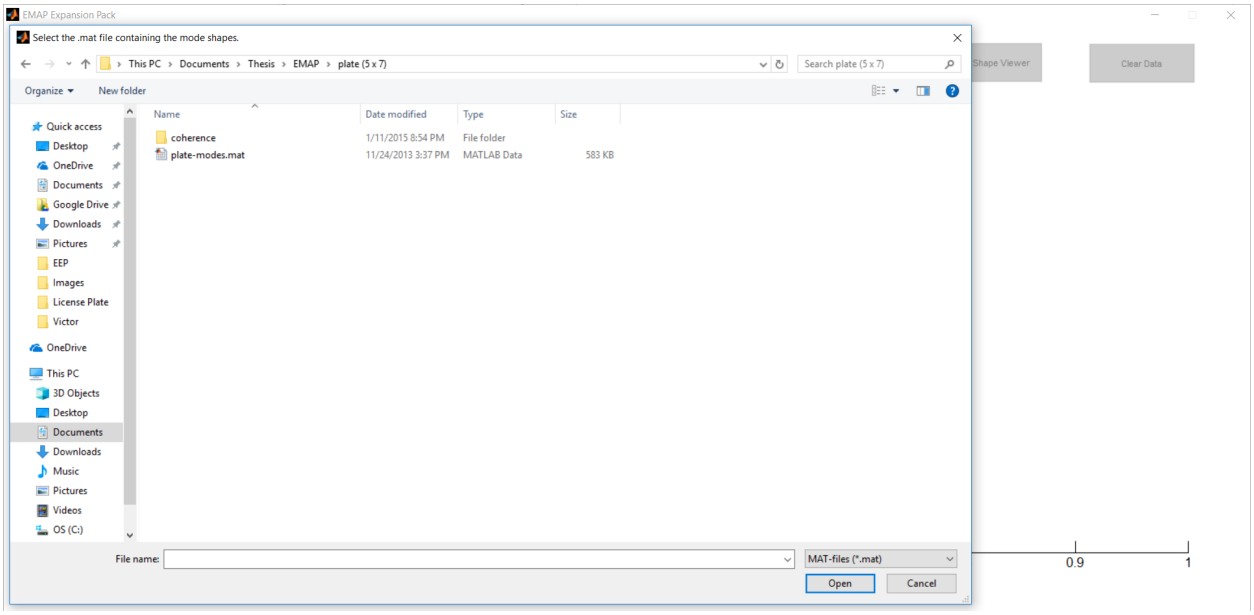


Figure 2.2: User data selection pop-up window.

This window allows the user to search through folders on the computer until the .mat file of structure data is found and selected. If nothing is selected or an error occurs, the user will be returned to the start window. Once the user has selected the desired .mat file the previously empty axes in the EEP window plots the CMIF from the data, shown in Figure 2.3.

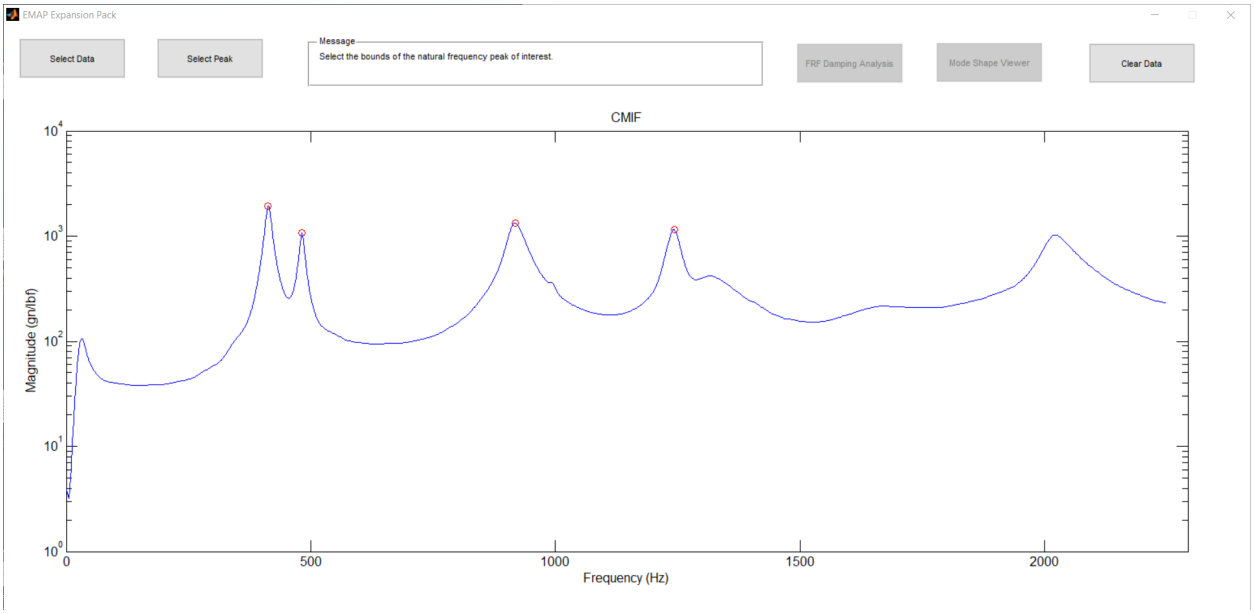


Figure 2.3: CMIF plot created with ‘Select Peak’ button enabled.

All data in the .mat file is loaded into EEP for its use. This is done by loading it into the Matlab workspace in the background. It can be accessed from the command line if desired. EEP talks to the workspace when the application is open. The CMIF peaks have a red circle overlay, denoting the natural frequency identified in EMAP. The user then needs to click the second button from the left, ‘Select Peak’. Once clicked a cross-hair takes the place of the mouse arrow and spans the entire application window as shown in Figure 2.4.

At that point the user just needs to click on the left side of the peak and then the right. The order of selection is not critical. Therefore the user can also select the frequency range from right to left. After both left and right bounding points around the peak of interest are selected, two

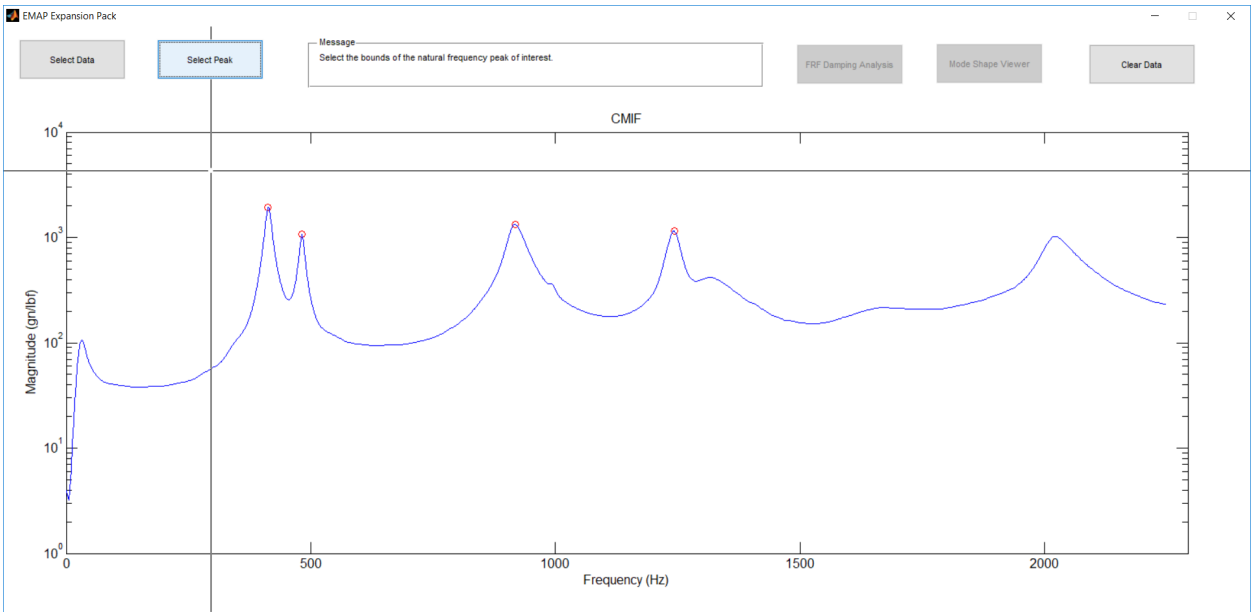


Figure 2.4: Cross-hair selection tool for user boundary placement.

dashed vertical lines are plotted on top of the CMIF. The lines shown in Figure 2.5 are plotted on the same selected points for the user to visualize the range to be analyzed. If the range was selected incorrectly or the user wishes to re-select them, the ‘Select Data’ button needs to be pressed again. The dashed lines will be removed and the cross-hairs will reappear to repeat that process.

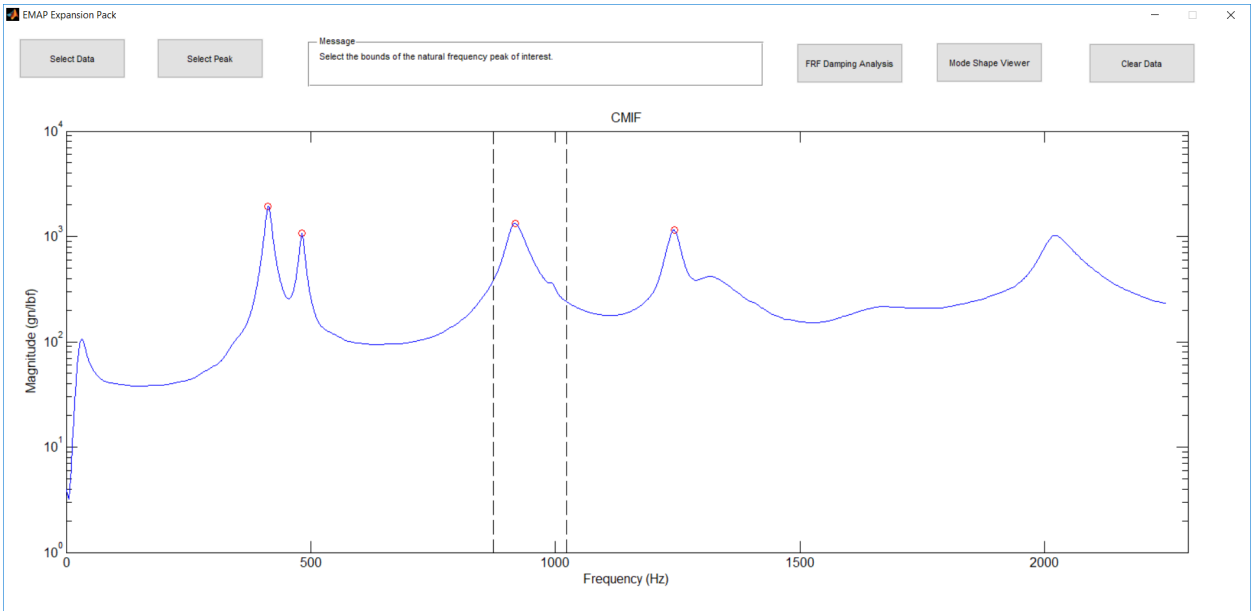


Figure 2.5: Frequency range that was selected is plotted on CMIF and remaining buttons enabled.

If the desired range of analysis has been selected the user then selects either the ‘FRF Damping Analysis’ or ‘Mode Shape Viewer’ button. For this set of instructions the ‘FRF Damping Analysis’ is selected first. Figure 2.6 is the pop-up window that is generated once the button is clicked on.

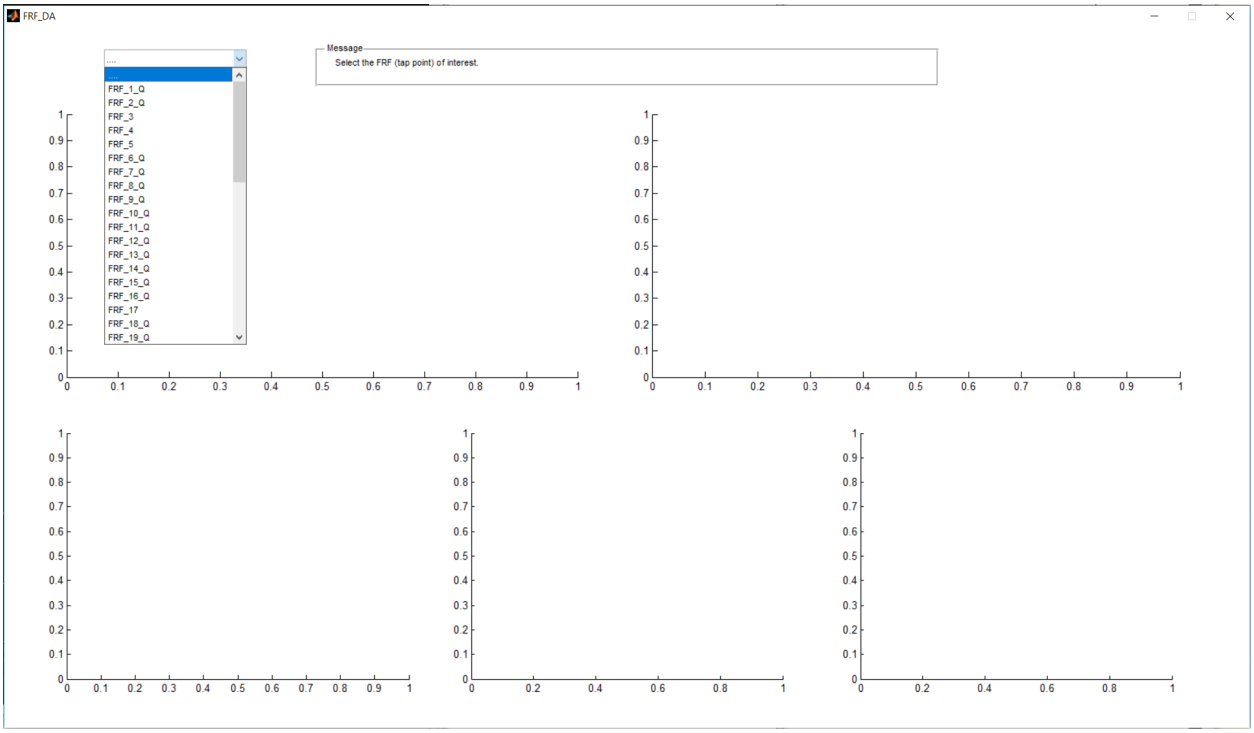


Figure 2.6: Drop-down list of each FRF for viewing.

All graphs are unpopulated with a generic axis for each. The user is able to select, at the top left corner of the window, a drop-down list of each FRF data set. In the drop-down list some FRF’s are suffixed by a “Q”. The “Q” stands for quarantine, as an indicator of the FRF data. That data has traits that make the program’s damping calculations less desirable for use. This is discussed further in Section 2.2.5 and is only noted here for the user. After the desired FRF is selected, all of the graphs are populated with the FRF data inside the range selected from the CMIF plot and is shown in Figure 2.7.

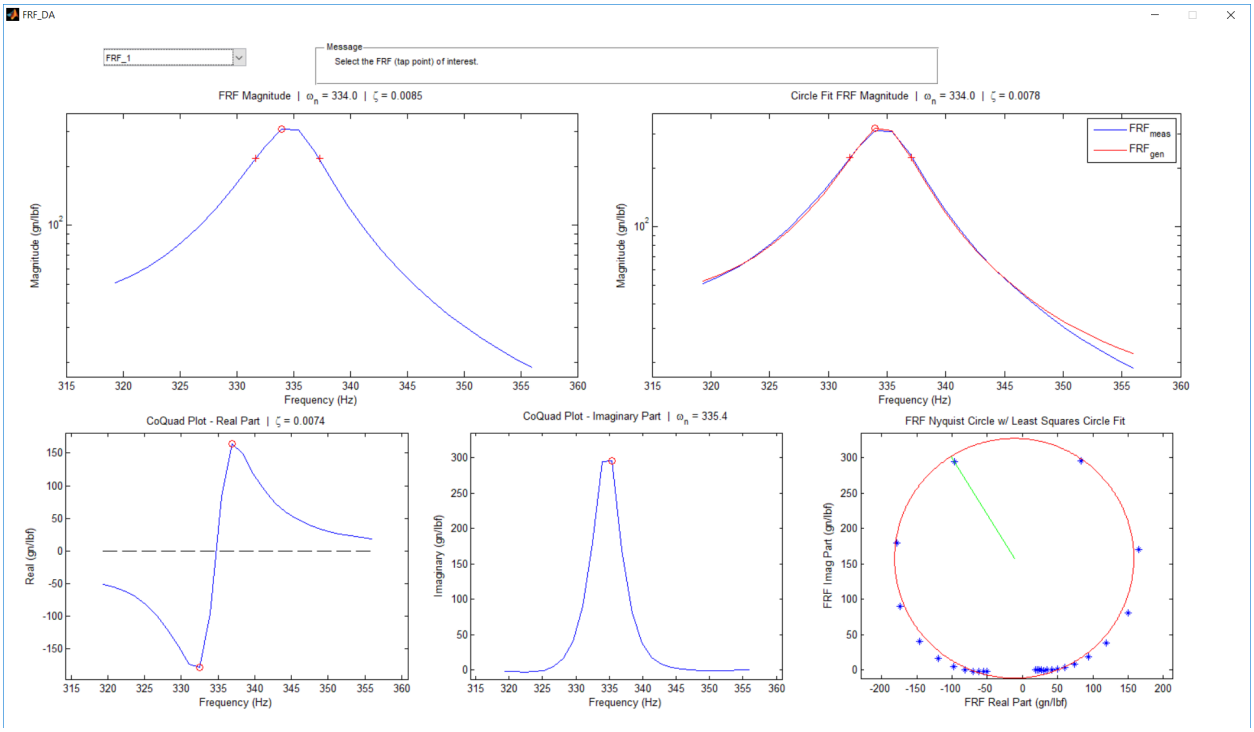


Figure 2.7: Normal FRF window with plots updated.

The top left plot is the Magnitude (absolute values of real and imaginary data) with respect to frequency. The two plots on the bottom left and bottom center are the Co-Quad plots. These are just the real and imaginary parts, respectively, of the FRF plotted with respect to frequency. The bottom right plot is a circle-fit of the FRF data, plotted on the real and imaginary axis. The top right plot is a Magnitude plot of the circle-fit data overlayed on top of the original FRF Magnitude data.

2.2.2 FRF Magnitude Plot

The main method of parameter estimation in modal analysis comes from using the magnitude plot of an FRF as shown in Figure 2.8. The peak of the FRF is the natural frequency of the mode of interest. The natural frequency is found by finding the peak of the FRF range. MATLAB makes it easy to find the peak by using the command:

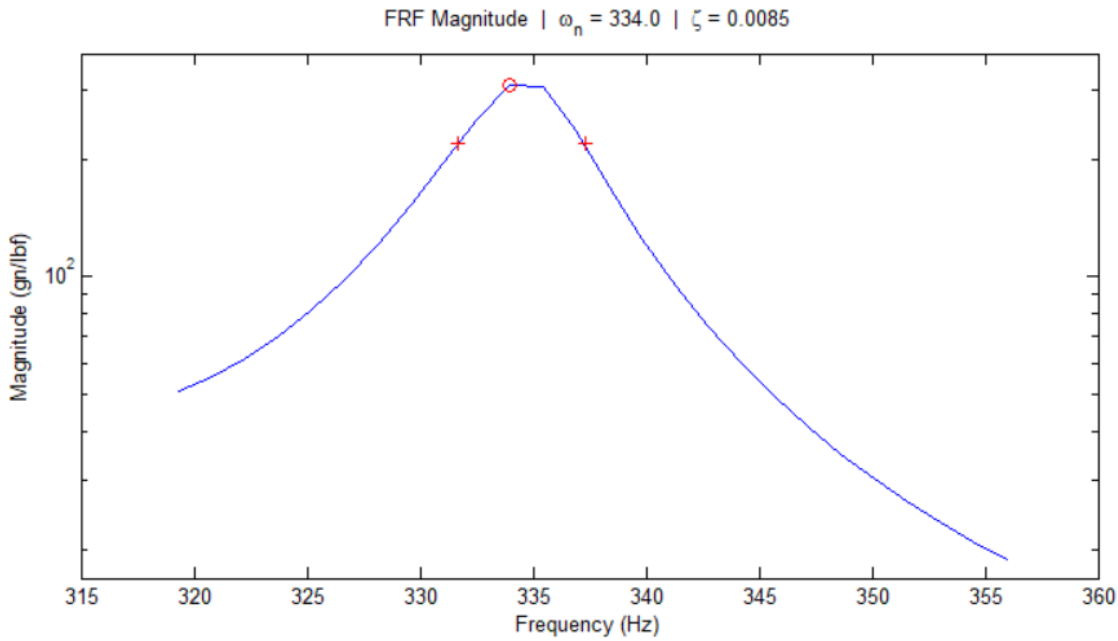


Figure 2.8: Clean FRF plot.

```
[C,I] = max(A);
```

where A is the input array, C is the largest elements of the dimensions asked of A, and I is the index locations for the maximum elements in the array. As long as the frequency array is equal to the length of the magnitude array, the peak frequency can be computed.

The damping ratio is calculated by the peak-pick method, Co-Quad method, and circle-fit method. Each is calculated slightly differently, but each calculate the damping ratio with the same equation, Equation (2.1) [2]:

$$\zeta_n = \frac{\omega_n^{(2)} - \omega_n^{(1)}}{2\omega_n}, \tag{2.1}$$

where ζ_n is the modal damping ratio value, ω_n is the natural frequency, $\omega_n^{(2)}$ is the upper half-band power frequency, and $\omega_n^{(1)}$ is the lower half-band power frequency. The half-band power frequencies are found by taking the magnitude of the peak at the natural frequency, dividing by

$\sqrt{2}$ and interpolating on the FRF curve to find where the frequency intersects the half-band power magnitudes on each side of the natural frequency peak.

2.2.3 Circle Fit Method

The same FRF data used in the previous magnitude plot is used, but plotted on a real part versus imaginary part in Figure 2.9. The original FRF data is shown in blue points, that nearly

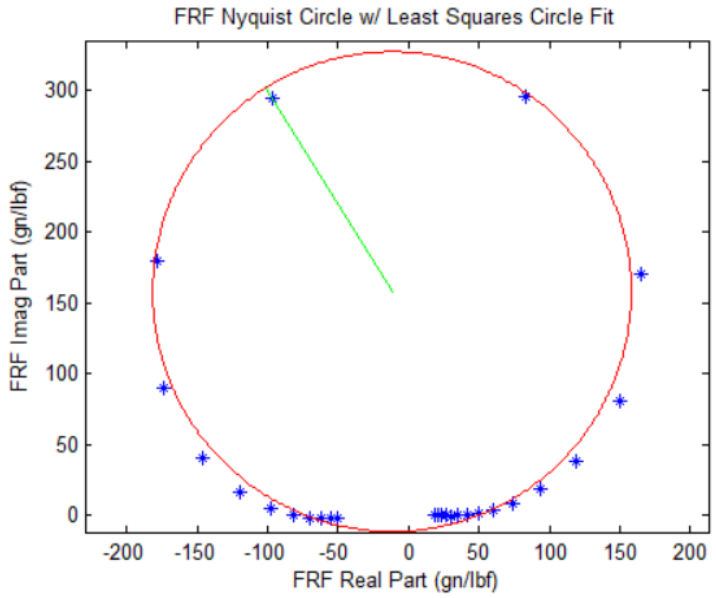


Figure 2.9: Circle fit overlay on original FRF point data.

resembles a circle. The red circle plotted is a least squares fit to the data. The least squares circle fit algorithm minimizes the deviation of the circle away from each point. The equation minimized is for a circle with its center not at the center of a plot axis. The equation is

$$(x-h)^2+(y-k)^2=r^2 \tag{2.2}$$

where x is the real part of FRF data, y is the imaginary part of FRF data, h is the x-axis offset of the circle's center, k is the y-axis offset of the circle's center, and r is the radius of the fitted circle. The

green line is to show the center of the circle to the peak of the FRF data. There are larger distances between points closer to the peak than at the ends of the FRF. A second FRF magnitude plot is made that includes the original FRF data and a generated FRF from the circle-fit data as shown in Figure 2.10.

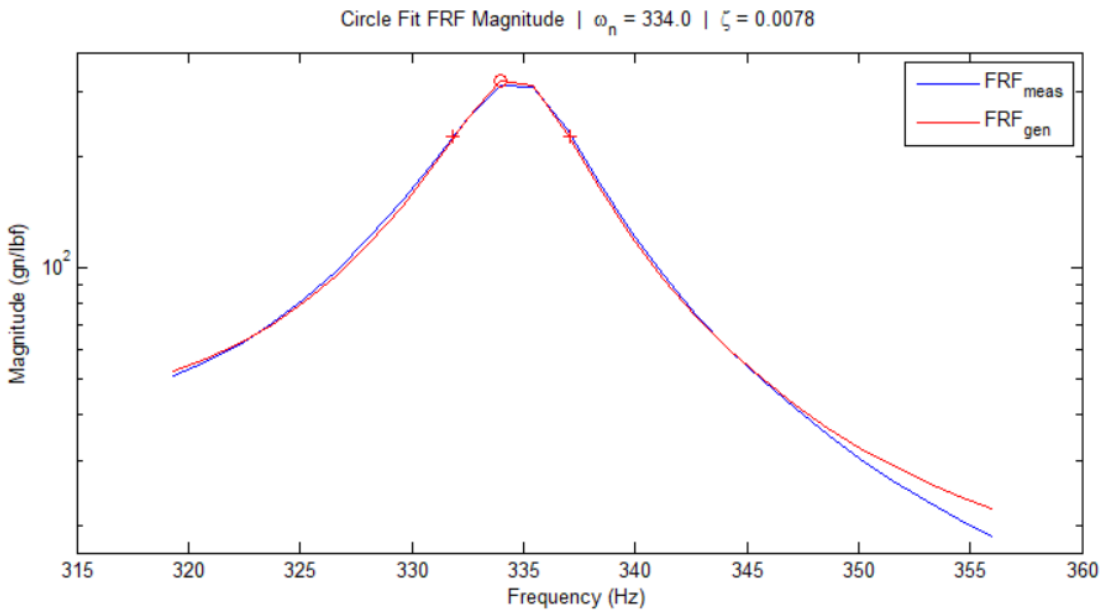


Figure 2.10: FRF generated from circle fit plotted with original FRF.

The frequency data from the original FRF is input into the circle-fit equation and then the magnitude is taken from that. The larger gaps in data points, in Figure 2.9, are noticeable around the peak of the magnitude plot. They are the same on both graphs because of using the same frequency points. The circle-fit FRF closely follows the original FRF. The circle-fit method uses the same technique by using an arc on the fitted circle to the real and imaginary data. The damping ratios are not exact as they are calculated from a generated FRF. The damping ratio is computed in different ways for comparison, but a "true" ζ is difficult to determine.

2.2.4 Co-Quad Plot

The Co-Quad [3] plots view the real and imaginary parts of the FRF in separate graphs, shown in Figure 2.11. The peak of the imaginary part plot is right around the natural frequency.

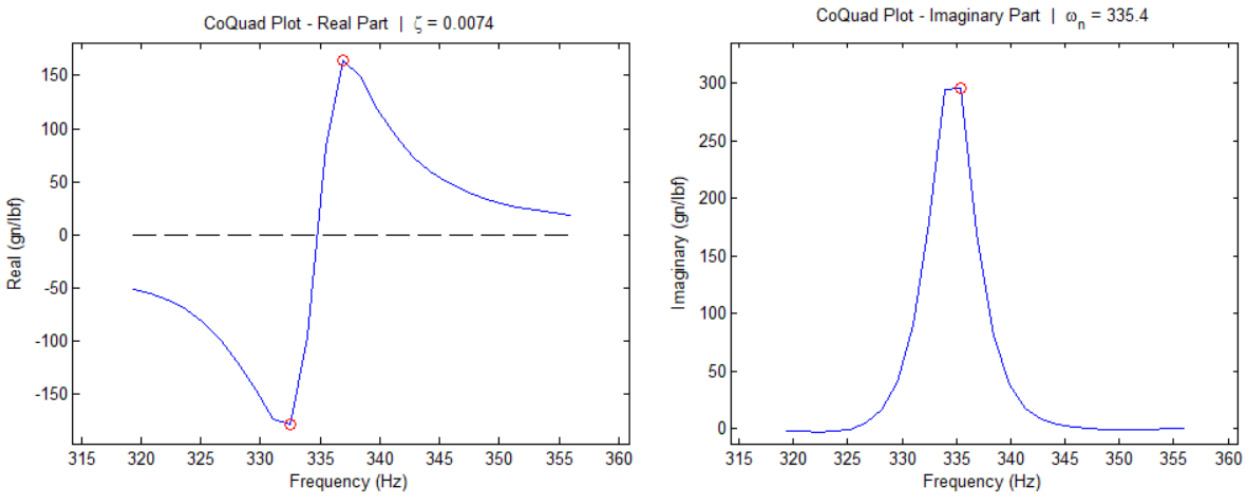


Figure 2.11: Co-Quad real and imaginary plots.

The negative and positive peaks of the real part plot are the half-band power points. The damping ratio is calculated from exact data points than by interpolation, but was also calculated with Equation (2.1) for the FRF Magnitude plot.

2.2.5 Uncertain FRF Data

A secondary aspect of peak picking, added in the FRF Damping Analysis, involves computing the natural frequency while looking at incremental changes in slope. The code for peak_pick.m is given in Appendix A.1 and allows a data peak to be counted as long as the previous point was smaller and the following point is also smaller. Although the code is simple, it is useful for giving some indication of whether computing the damping ratio will be difficult if not incorrect. This can become an issue if there are multiple peaks in the FRF of interest. This section gives an example

of how multiple peaks and potentially bad data affect finding the natural frequency and computing the damping ratio. After tap testing a plate, three different FRFs were observed with increasingly worsening data. This data was from a 5" x 7" x $\frac{1}{8}$" steel plate previously tested by Dr. Kawano.

The following two figures, Figures 2.12 and 2.13 respectively, are the CMIF of the plate that was studied and the third mode shape of the plate. The third natural frequency was selected as it had an extra small peak to the right of the main peak. This can be seen highlighted in Figure 2.12 by a black square. The mode shape for the plate was also included to show which points were

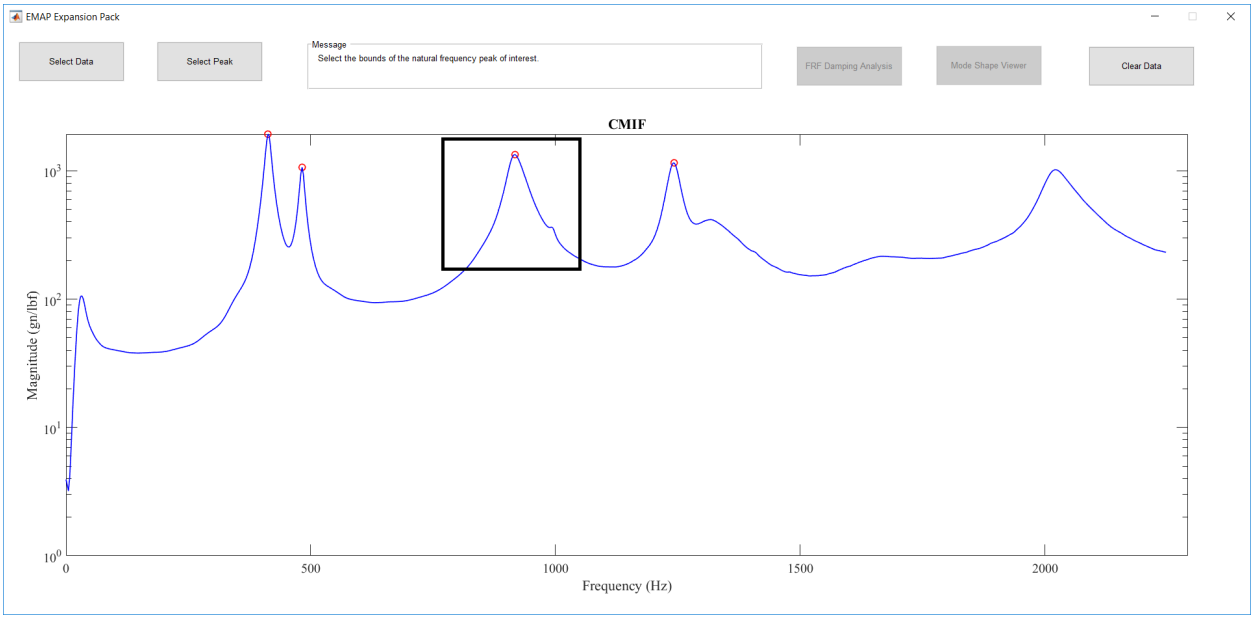


Figure 2.12: Third natural frequency selected in CMIF.

being discussed. The points of interest were points 1, 2, and 3 (See Figure 2.13). The order of their analysis is backwards though, starting with point 3 and ending with point 1. The gap between each point is 1" along the edge of the plate. The steel plate is assumed lightly damped.

Figure 2.14, the FRF of point 3, is the cleanest FRF of the three. It clearly has one peak that correlates to the natural frequency being observed. The data is clear and distinct. The small rise at the bottom right is small enough to not affect the parameter estimation. This is because the

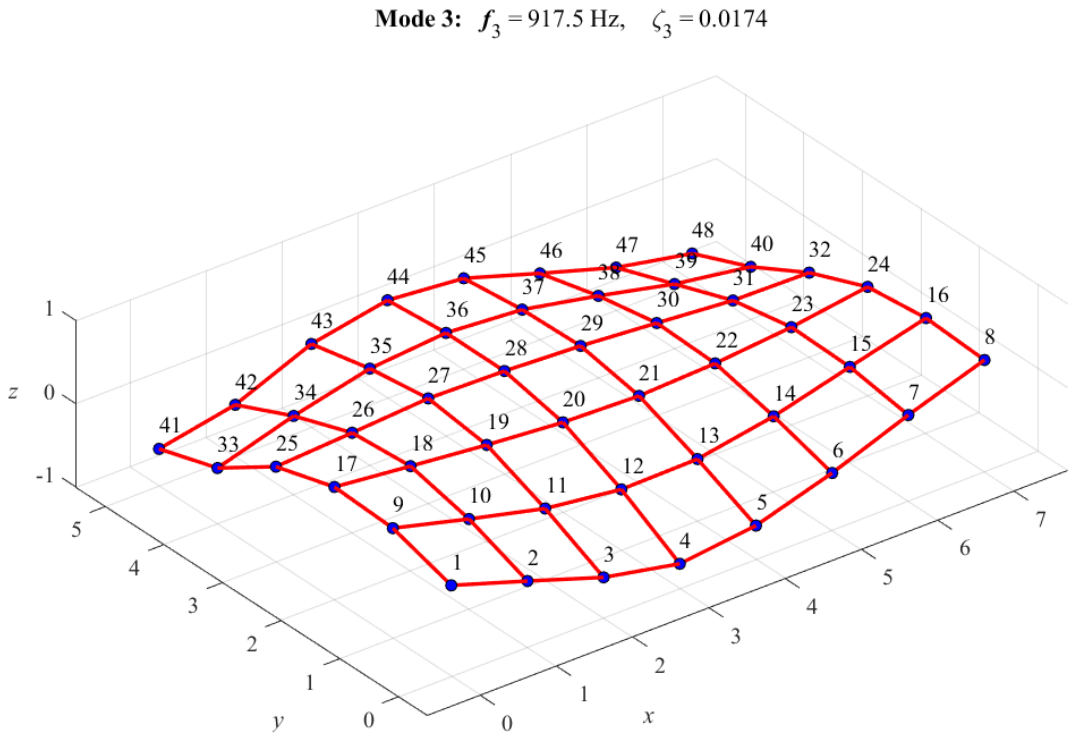


Figure 2.13: Third mode shape of plate with test points identified.

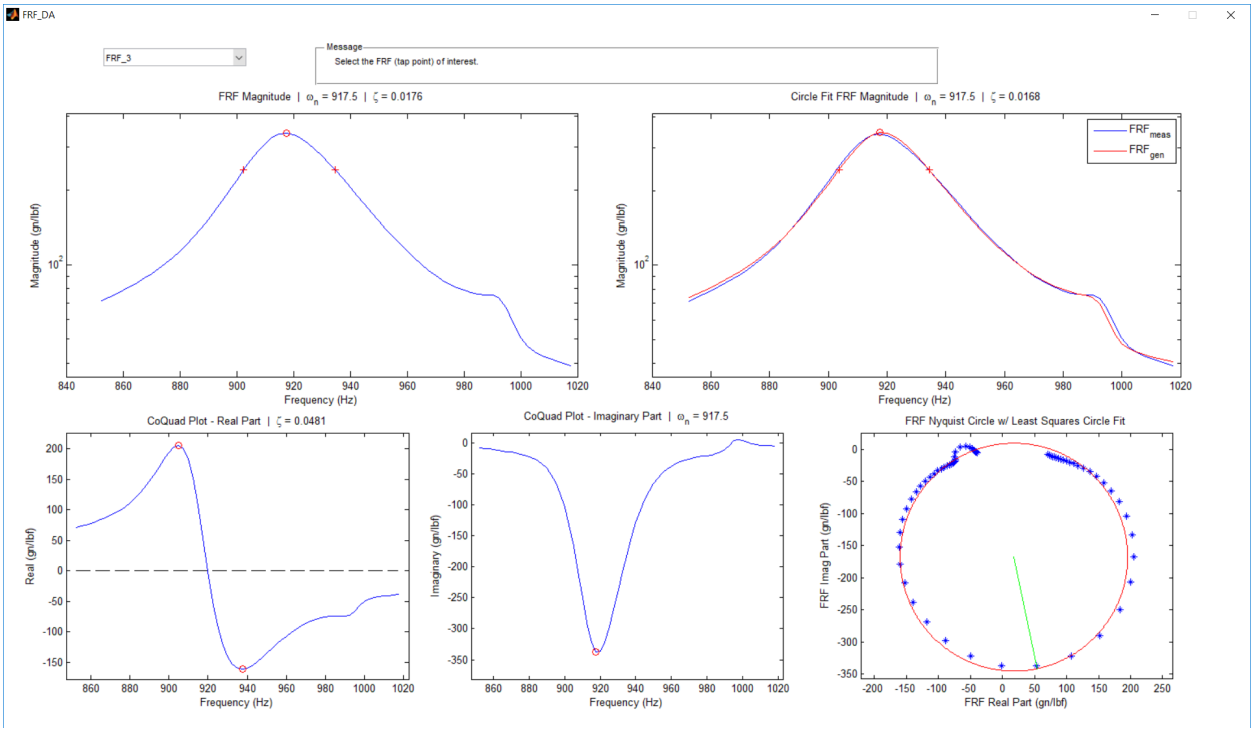


Figure 2.14: FRF data with small peak visible, but not found by program.

rise does not create another peak and is far enough away from the natural frequency peak to not interfere with the damping ratio calculation.

Figure 2.15, the FRF of point 2, shows where the small peak in Figure 2.14 has turned into a second distinct peak. This second peak could be another natural frequency of the plate, but not fully shown by the CMIF. Its magnitude is still smaller than the peak at the natural frequency so the program is still able to find the actual maximum point. The data and parameter estimation came out fine in the magnitude plot, but the circle-fit generated FRF does not fit the original FRF data as soon as it gets close to the second peak. The second peak has started to greatly affect the damping estimation from the Co-Quad plot.

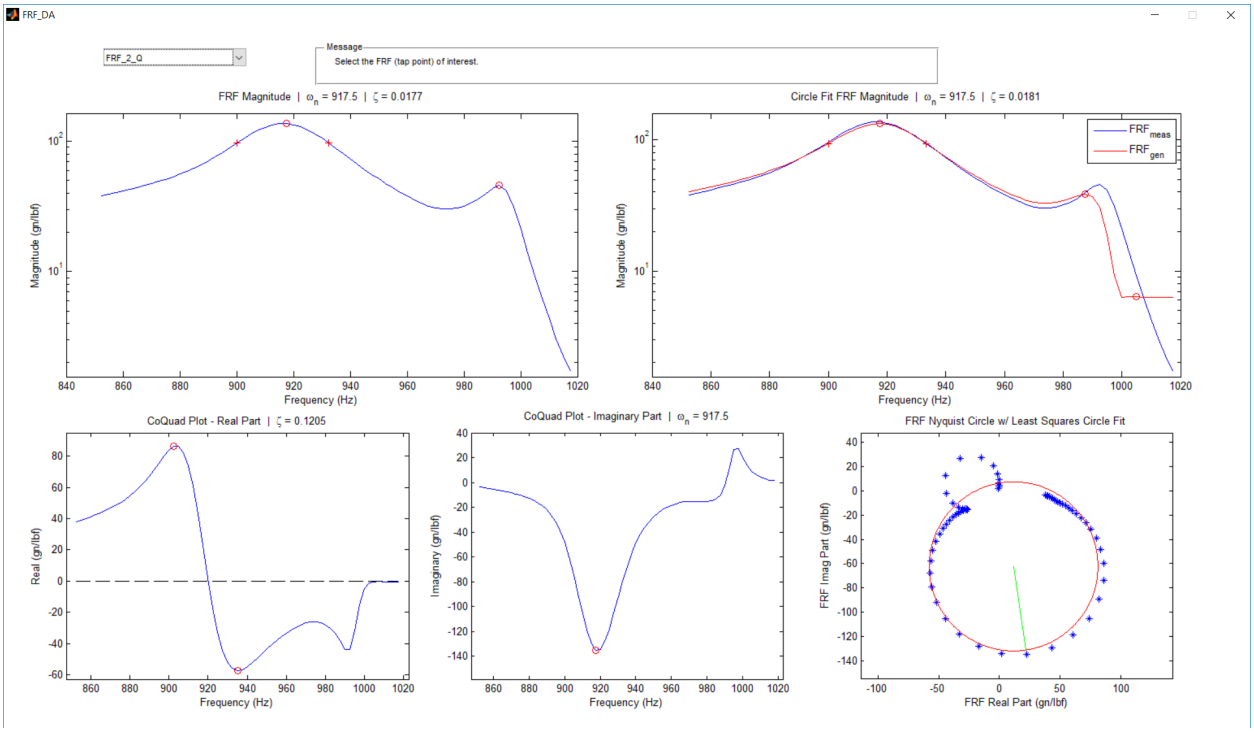


Figure 2.15: FRF data showing two peaks found.

The drop tab that displays FRF_2_Q is a hint that the data may be questionable, where the "Q" attached at the end stands for quarantine. This quarantine is for drawing attention to the data

set, but not for removing the validity of it. The "Q" comes about from the peak pick program discussed at the introduction of this section. If multiple peaks are found by the algorithm, that data set is automatically denoted as quarantined.

Figure 2.16, the FRF of point 1, seems to actually have poor quality data. This data set is also denoted with the quarantine "Q". The small peak is nearly as great in magnitude as the natural

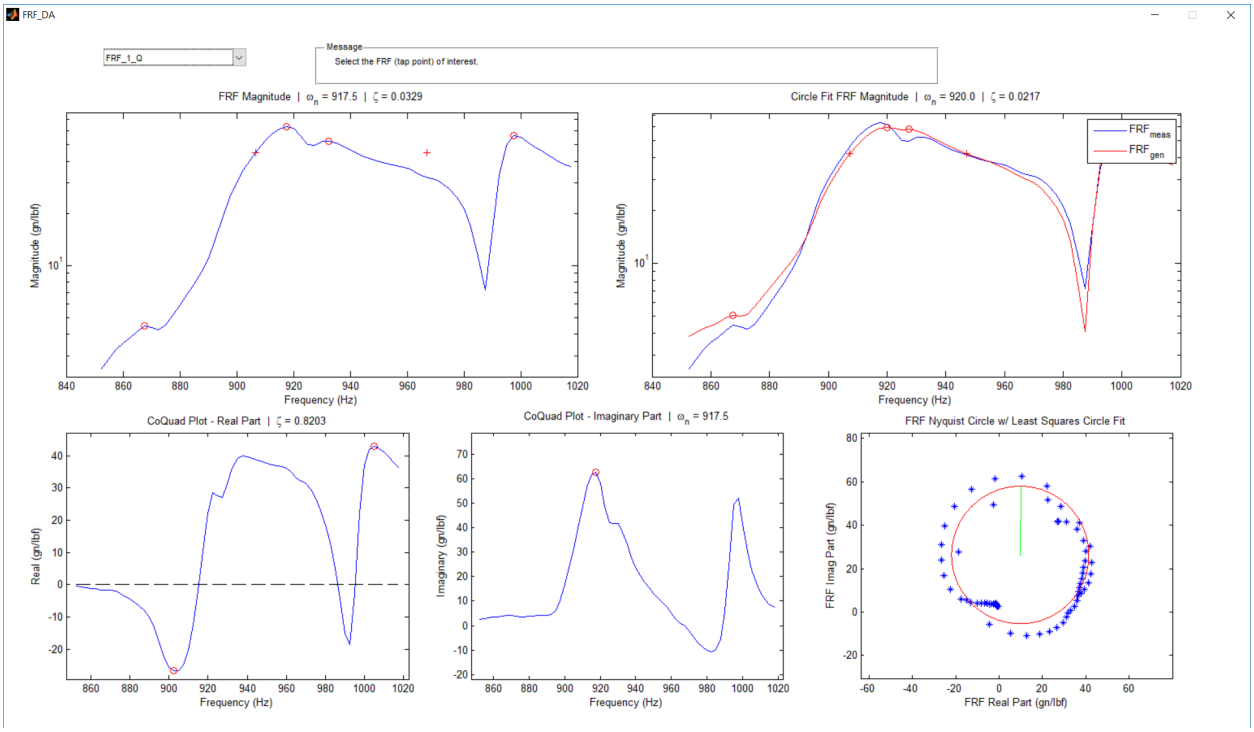


Figure 2.16: Poor FRF data showing four peaks found.

frequency peak and two other small peaks formed. The data from point 2 is still smooth while this data set is visibly more erratic. The program can still find the natural frequency, but the damping ratio values are considerably different. The damping ratio is twice as high as the previous two FRF data sets and the upper half band power point in Figure 2.16 does not fall on the FRF curve. The damping ratio for the Co-Quad plot is over twenty times higher than the ratio from the magnitude plot. It may not be fully understood as to why a particular FRF has such poor data, but it is not

relevant to the use of EEP. The main benefit is that the user is able to be better informed with the data collected and decide how beneficial the quarantined data is.

2.2.6 Spline Mode Shapes

Currently, all reputable modal analysis programs include the ability to plot mode shapes of the structure being tested. As long as all geometry points are recorded and input into the program correctly, a mesh of points should give an accurate representation of the structure. When each point is multiplied by the amplitude of the motion of the mode shape and saved in several frames, it will move according to the corresponding natural frequency of that mode. All of these plots and animations currently use point to point lines to create the mesh.

Plots for comparison, not animation, was the focus of this part of EEP. The 'Mode Shape Viewer' button in EEP generates a second window with a still shot of the mode shape of the structure at the natural frequency selected. Two identical plots are generated for the user to compare, the first of which is Figure 2.17. The top plot is the general point-to-point plot given from EMAP and most every modal analysis program in industry. The bottom plot of the mesh is done with piecewise cubic splines. The splines have the form of Equation (2.3):

$$P_i(x) = A_i + B_i(x - x_i) + C_i(x - x_i)^2 + D_i(x - x_i)^3 \tag{2.3}$$

where $P_i(x)$ is the function of the cubic polynomial and A_i, B_i, C_i, and D_i are the polynomial coefficients being solved for. The subintervals of $(x - x_i)^n$ set the range of function, where x is the vector of points to plot the spline and x_i are the points the spline is being fitted to. The splines are created from point to point with their curvature matching at each point or node. This is a condition for the splines as they need to be twice differentiable at each point. The calculation

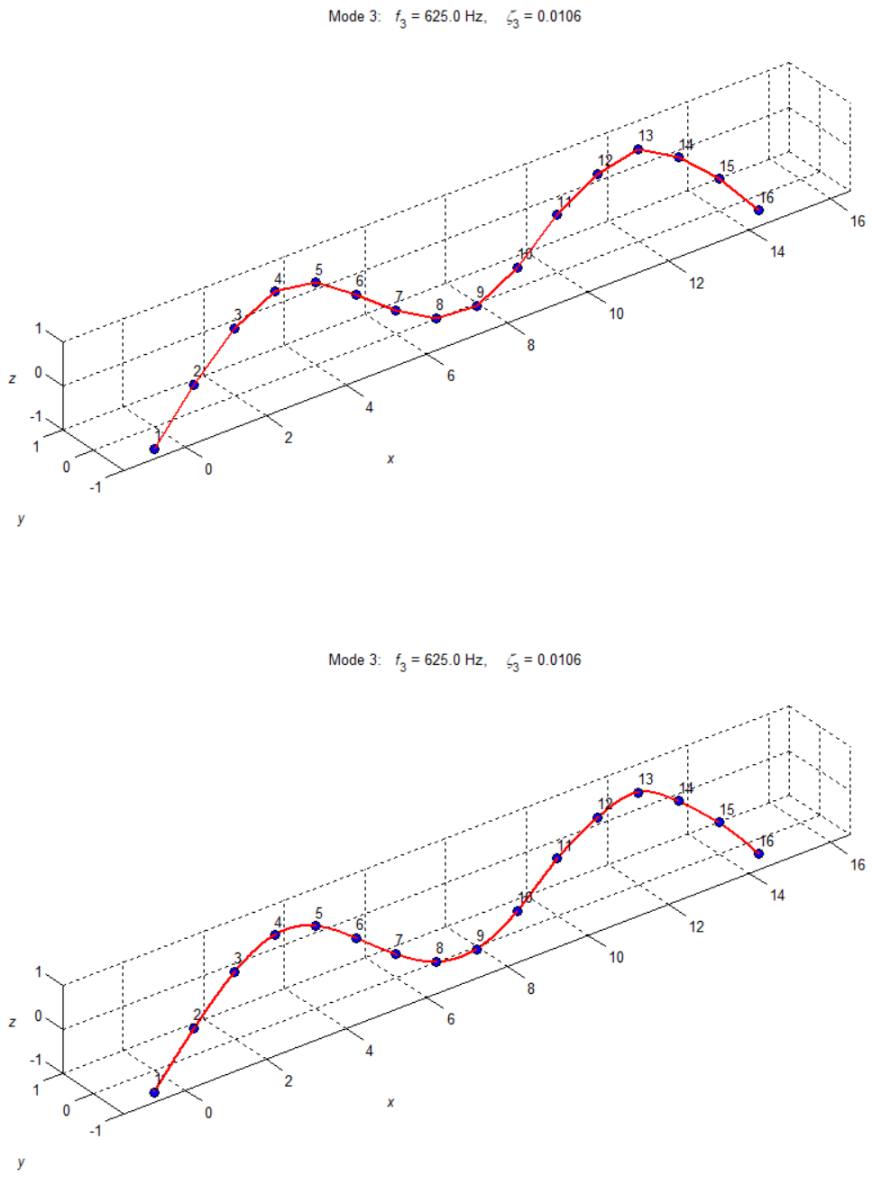


Figure 2.17: Spline curve fit to a beam, in this case a line of points.

of the polynomial coefficients can then be completed by Gaussian Elimination. This is detailed from previous class notes that are attached in Appendix A.5. This is an argument for realistic and natural aesthetics than for saving on computational power, although a complete analysis on the computational requirements is given in [4, pp. 284 - 294]. This code does not include animating

the mode shapes, but could be easily applied. This can be easily done with MATLAB's spline command, but two functions were created for EEP to generate the splines.

The first goal was to fit a line of points which is the general use of a spline. This was done with a beam shown in Figure 2.17. EEP figures out what points create a line, the function create_spline.m generates the spline data, and spline_points.m creates the spline to overlay on the plot of points. In general the points should be in numerical order, but if that is not the case the coding inside EEP is able to find the line as long as the actual position metrics data are correct.

The second goal was to take a mesh of points and fit splines to each individual line. Again this is done by looking at position data to find where the points are dimensionally. As can be seen in Figure 2.18, the results are visually stunning. The plate in the figure is the same used in the example in Section 2.2.5. The plate's mode shape is visually enhanced by the curvature resolution of the spline compared to straight lines. This also makes its mode shape look natural.

The third and final goal was to fit the splines to objects that have features in more than one plane. This originally did not work. The first iteration of code used for the beam and plate followed multiple point lines that were in numerical order. For the bracket shown in Figure 2.19 this code only created splines across the top and side faces of the bracket. Splines with two points create straight lines. Once the code was changed to look at how the x, y, and z data differed in each plane, it was able to find the continuous lines in the bracket and apply splines to them.

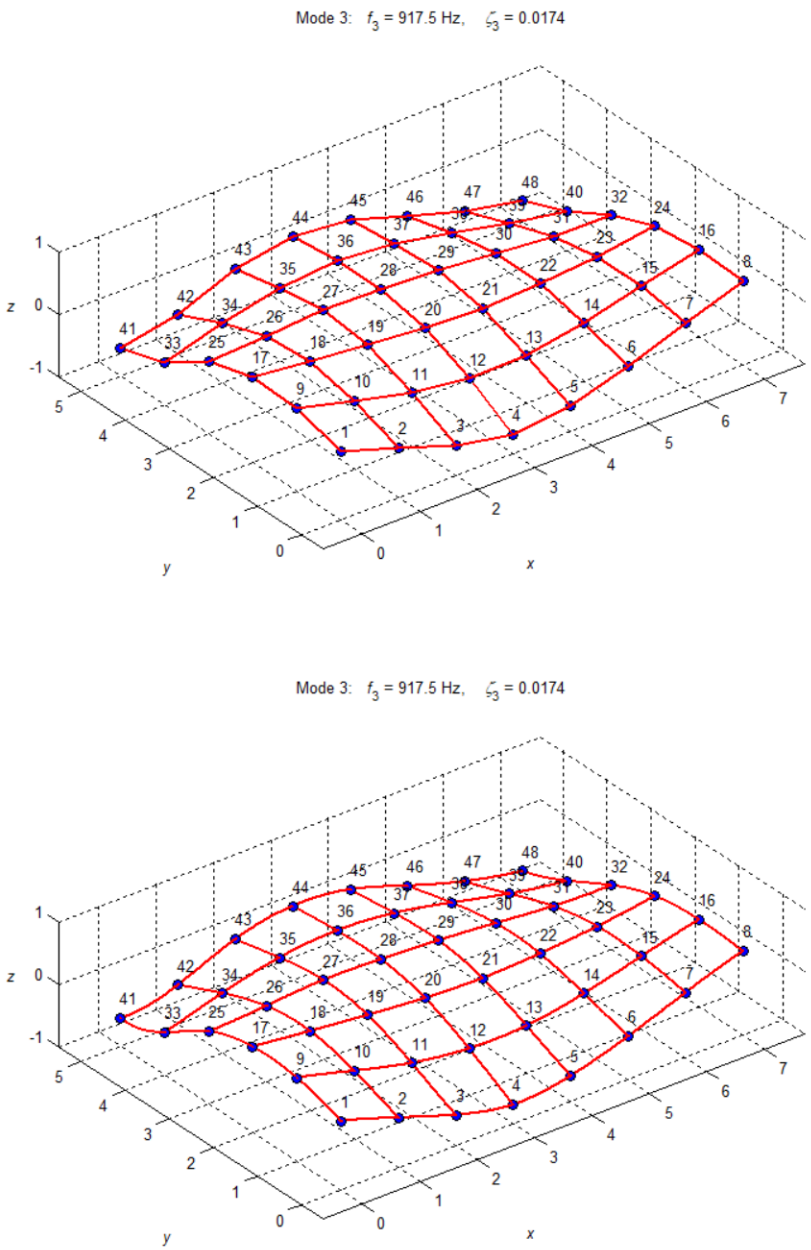


Figure 2.18: Spline curve fit to a plate, a grid of points.

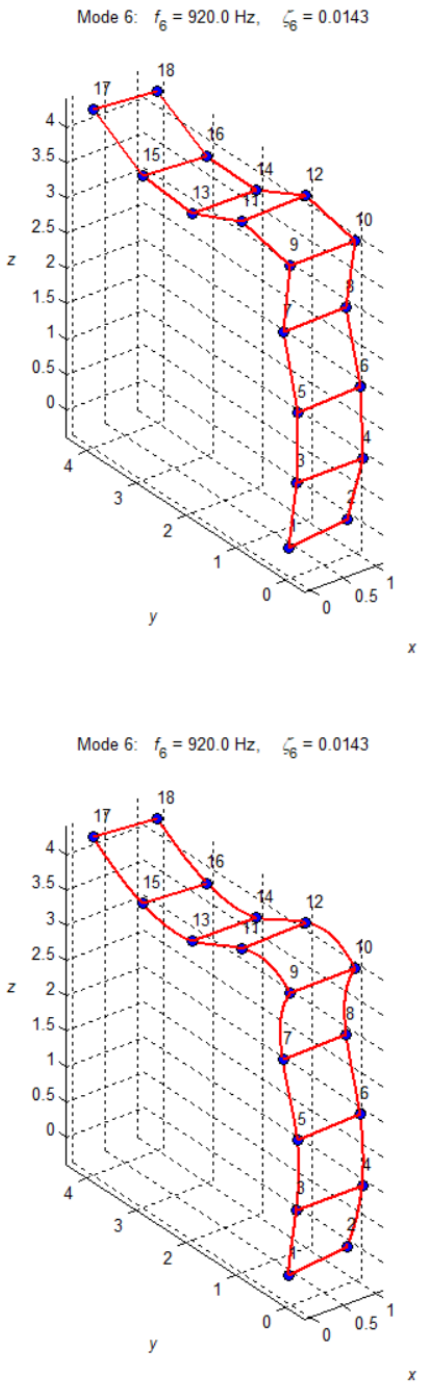


Figure 2.19: Spline curve fit of an L-bracket.
(Two grids of points with one grid 90° to the other.)

3 Building on a Shake Table

The first lab designed for furthering students' application of theory is the vibratory response of a structure undergoing base motion. A mock two-story building is subjected to displacement at its base by a shaker table. A plate keeps two moving carts solidly fixed together so that they move synchronously. Within the physical limits of the motor and mechanical system, the base can input any displacement profile desired. The lab comes in several parts that are designed to reflect different applications of studying vibration response. The first part is to visually observe the building going through a sine-sweep to get a general idea of where the first natural frequency is. The second natural frequency is also observed, but for the tallest building only. The second part is done by inputting distinct frequency sine waves into the building and finding the frequency that produces the highest acceleration amplitude to give the experimental natural frequency. The final aspect compares the experimental data with an analytical model.

3.1 Purpose

To visually study the vibration of the structure the known displacement profile has to start at a frequency lower than the natural frequency and run up to a frequency higher than the natural frequency. The building was designed with three different leg heights: 18", 24", and 30". For the purposes of this study the desired natural frequency to observe was the first. To run a sine sweep from a starting frequency to an ending frequency over an equal time gradient, the chirp signal generator was used in MATLAB Simulink. The computers that are serially connected with the ECP modules use Simulink and QUARC concurrently to build files that the control module

can understand and then act upon the building. Figure 3.1 shows how the computer with Simulink interacts with QUARC and the ECP module. An analytical model is created to be compared to the experimental results.

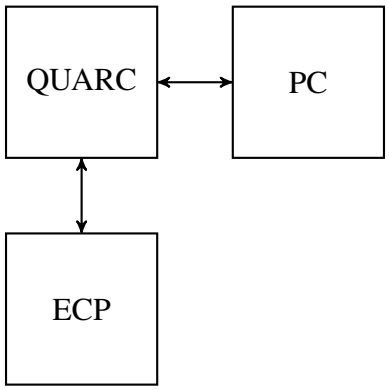


Figure 3.1: Functional diagram for software and hardware interactions.

3.2 Model

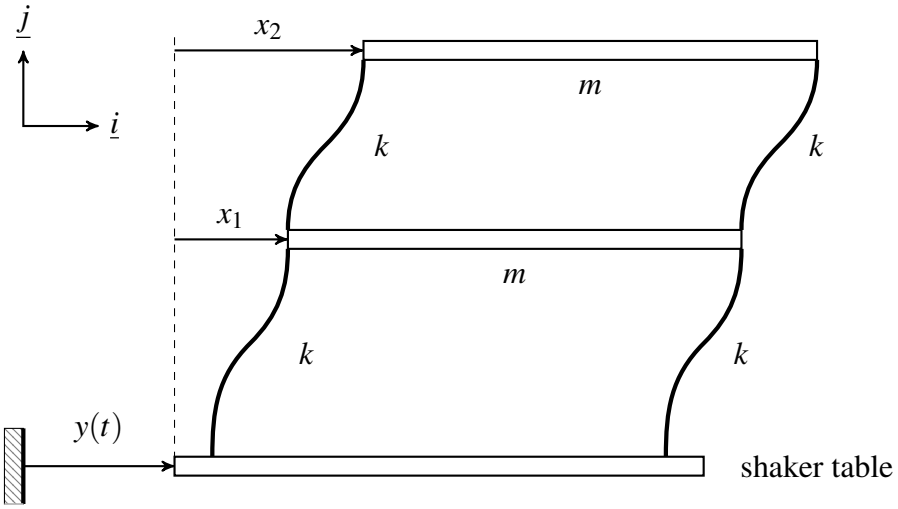


Figure 3.2: Two-story building model for shaker testing analytical correlation and validation.

The analytical model for the two-story building is shown in Figure 3.2. It is modeled with a

mass for the floors, vertical legs acting as springs, and the shaker table providing base motion into the building. Defining the variables of the building, m is the mass of the floor, k is the leg stiffness acting on the floor, x_1 is the position of the first floor relative to the shaker table, x_2 is the position of the second floor relative to the shaker table, and y is the base input position. The kinetic diagram (KD) and free-body diagram (FBD) for the first floor are given in Figure 3.3.

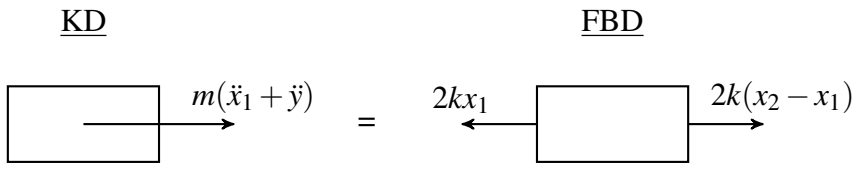


Figure 3.3: Kinetic and free body diagram for first floor of building.

Applying the rate form of linear momentum conservation [2, p. 557] gives the equation of motion for the first floor, Equation (3.1):

$$m\ddot{x}_1 + 4kx_1 - 2kx_2 = -m\ddot{y} \tag{3.1}$$

where each superposed dot denotes a derivative with respect to time. The relative acceleration of the first floor is $\ddot{x}_1$, and $\ddot{y}$ is the acceleration of the base input. The KD and FBD for the second floor are given in Figure 3.4.

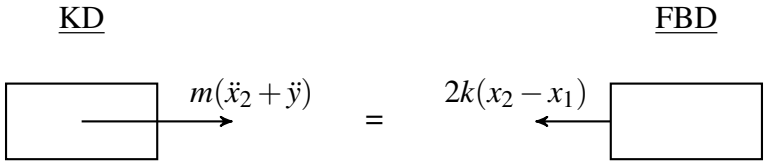


Figure 3.4: Kinetic and free body diagram for second floor of building.

Again applying the rate form of linear momentum conservation gives the second floor's equation of motion, Equation (3.2):

$$m\ddot{x}_2 - 2kx_1 + 2kx_2 = -m\ddot{y} \tag{3.2}$$

Placing Equation (3.1) and Equation (3.2) into matrix-vector form of Equation (3.3):

$$\underline{\mathbf{M}}\ \underline{\ddot{\mathbf{x}}} + \underline{\mathbf{K}}\ \underline{\mathbf{x}} = \underline{\mathbf{f}}(t) \tag{3.3}$$

yields the system's mass matrix $\underline{\mathbf{M}}$ and stiffness matrix $\underline{\mathbf{K}}$:

$$\underline{\mathbf{M}} = \begin{bmatrix} m & 0 \\ 0 & m \end{bmatrix}$$

$$\underline{\mathbf{K}} = \begin{bmatrix} 4k & -2k \\ -2k & 2k \end{bmatrix}$$

3.3 Experimental Procedure

After creating the analytical model, the physical model of the building was created. The building legs are removable so that the height can be changed with different sized legs. The base plate joins the two moving carts together. Masses used for other vibrations and controls labs was used on the carts to sit on top of the base plate. The building sits in front of the carts. The masses weigh the plate down while nuts are used to secure the plate to the carts. The part drawings for the building are included in Appendix B.1. All main building parts were cut to length on a band saw and machined on a manual Bridgeport mill in the ME Machine Shop. The base plate was waterjet cut at Rose-Hulman Ventures. The actual building is shown in Figure 3.5.

The experimental lab setup is shown in Figure 3.6. On the left is a laptop running RT Pro and

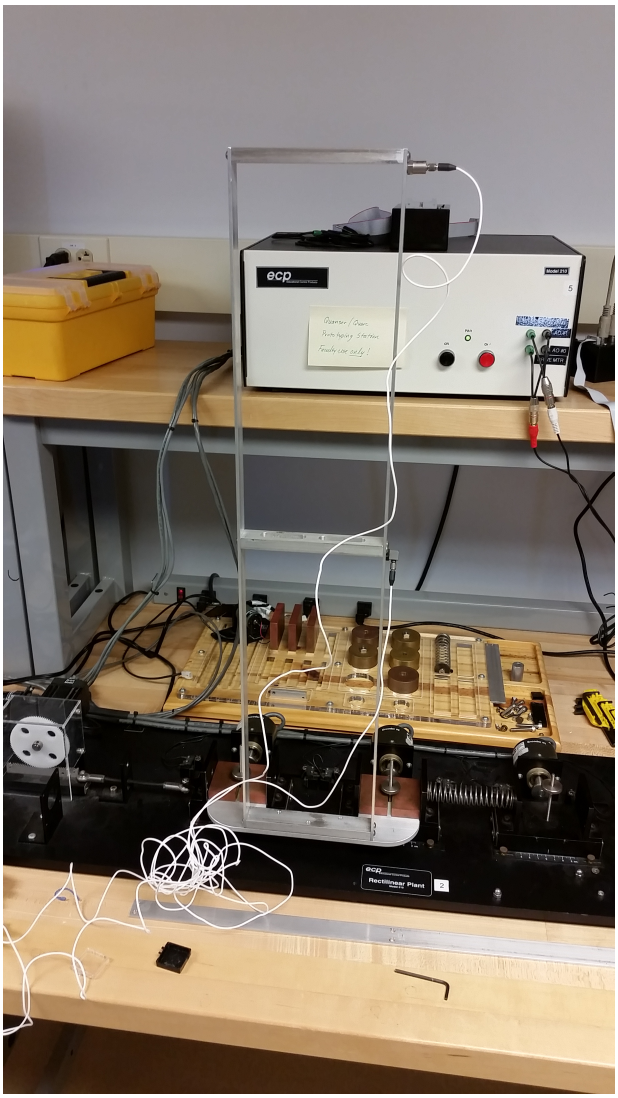


Figure 3.5: Mock two-story building.

connected to a Photon II unit. Connected to the Photon II are two accelerometers collecting data from the motion of the building. The accelerometers are attached at the sides of the building as close to the center of each floor as possible. These can be seen better in Figure 3.5. The two bolts that mount the legs of the buildings to the floors keep them from being in the exact center. On the right is the lab desktop computer that runs the QUARC I/O board that controls the ECP system. Between the two computers is the ECP control module with the building being rigidly fixed on the two moving carts. This desktop stays in the lab and uses Simulink to create block models with user

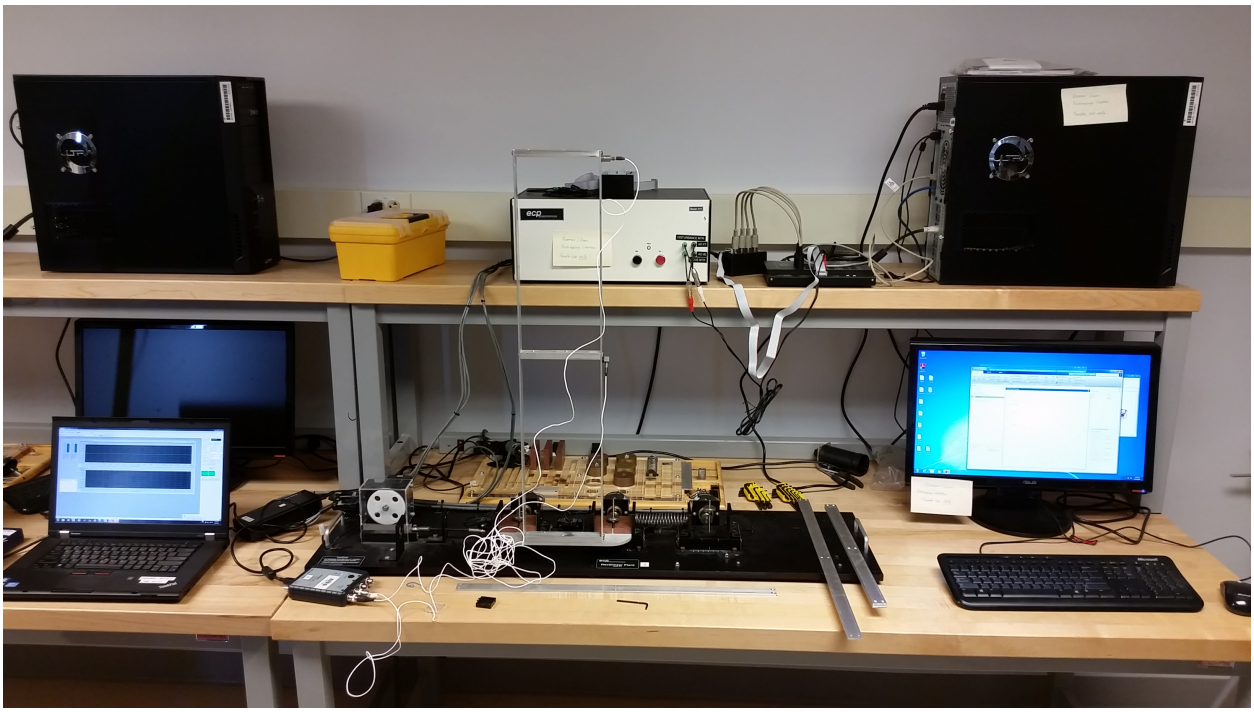


Figure 3.6: Experimental test setup of mock two-story building.

specified input parameters that output to the QUARC module. QUARC uses the Simulink data and converts those bytes into electrical signals that runs the ECP system DC motors.

3.3.1 Sine Sweep Testing

Students start with using the desktop computer only. Using Simulink, a sine sweep is input into the shaker rig to shake the building. The Simulink block diagram for this is shown in Figure 3.7. During testing, finding the natural frequency is completed visually. While the sine sweep is running, it is easy to identify a range in which the natural frequency is seen. Students input a time to run the sweep and a range of frequency to start and end with. For all experiments, 40 seconds was used as the total run time and the frequency range used was 0.1 to 40 Hz. These parameters are shown input into the sine sweep or chirp signal in Figure 3.8. As the student watches the test, they then need to record the time which they identify as the natural frequency. This occurs when

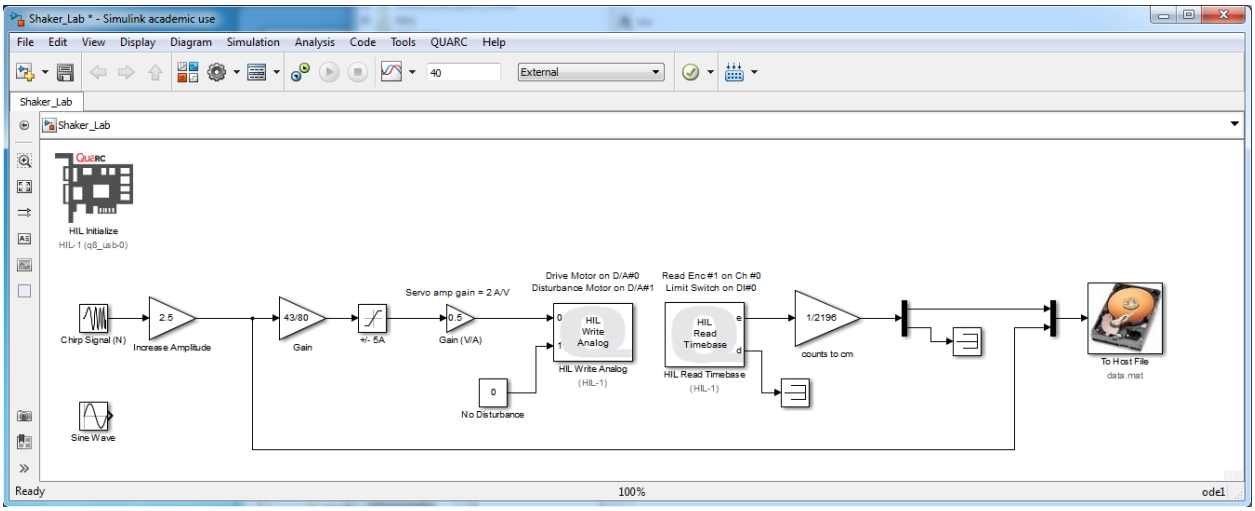


Figure 3.7: Simulink diagram for sine sweep input on building.

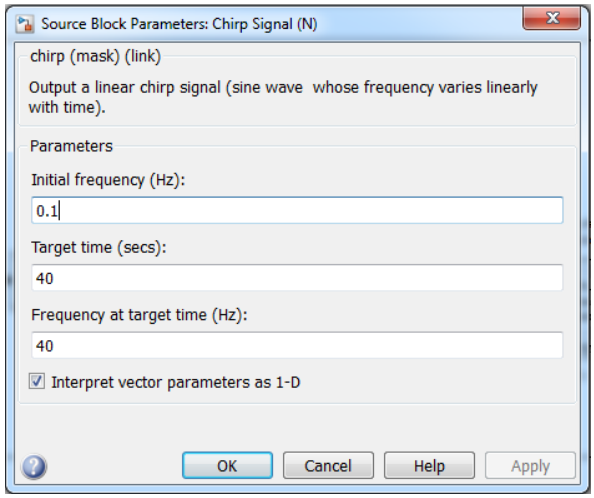


Figure 3.8: Simulink sine sweep input block parameters.

the amplitude of floor movement ceases to increase and starts to decrease.

The timing is important because each second correlates to a 1 Hz change in frequency. For example, if it is noticed that the natural frequency occurs around 24 seconds into the sweep, the natural frequency is estimated to be and then recorded at 24 Hz. The next step is to start running an actual sine wave on the building. The sine wave testing involves using both computers simultaneously.

After finishing each sine sweep with the building at different heights, the observed first natu-

ral frequencies for each and the second natural frequencies for the 30”-tall building were combined in Table 3.1. The second natural frequency was included for the 30”-tall building as it was noticed

Table 3.1: Visualized ω_n During Sine Sweeps.

Height [in]	Observed ω_n [Hz]
18	25.5
24	16.0
30 (ω_{n_1})	11.0
30 (ω_{n_2})	29.0

within the 40 second sine sweep test. If the total sine sweep time is increased, the second natural frequencies for the 18”-tall and 24”-tall building can be seen. They were not included as to not change the original testing sequence for this lab.

3.3.2 Sine Wave Input Testing

Figure 3.9 shows the block diagram for running the sine wave into the shake table. It is exactly the same as used in the sine sweep, but using a sine wave as the input instead of the chirp signal. Both diagrams include both input blocks, but only one is linked into the system at a time. The sine wave input runs for 40 seconds or whatever total time is desired. This value is input at the top of the Simulink window. The sine wave input parameters are shown in Figure 3.10. Once the sine wave is running on the ECP shaker, the RT Pro software then needs to be run. In RT Pro, acceleration data is recorded in time spans of 0.8 second. RT Pro can be run as many times as necessary to obtain the desired signal capture while the building is still being shaken. Once the RT Pro measurement is acceptable and saved, the sine wave program can be stopped. The RT Pro data

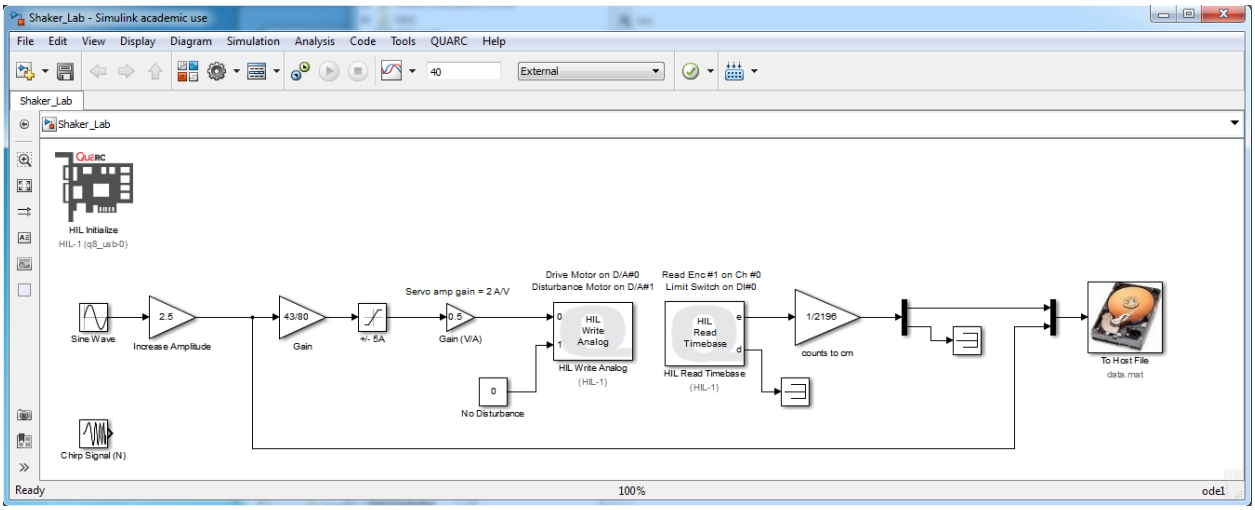


Figure 3.9: Simulink diagram for Sine Wave input on building.

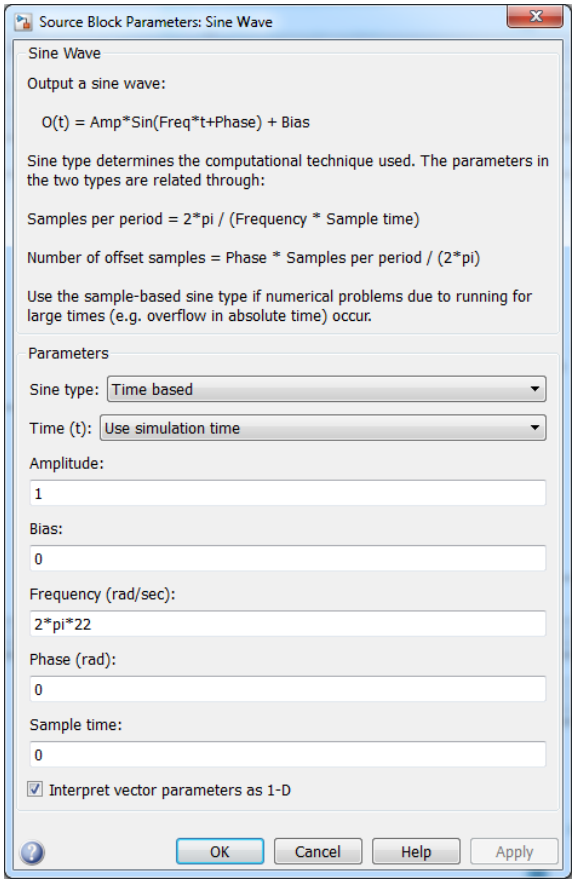


Figure 3.10: Simulink sine wave input block parameters.

is exported and saved as an Excel file.

During the sine wave input testing, the input frequency will be changed with every new run.

It is critical to know what sine wave frequency is being input into the system to save the correct data. It is easiest to input frequency in Hz. Therefore, it must be multiplied by a factor of 2π to be used in the sine wave block as it is expecting units of rad/sec instead of Hz.

3.4 Data Processing

From visual inspection and referencing Table 3.1, the natural frequency decreases as the height of the structure increases. Considering just one floor by itself, as shown in Figure 3.11, can help to understand what is happening. The mass of the floor does not change, only the height of the

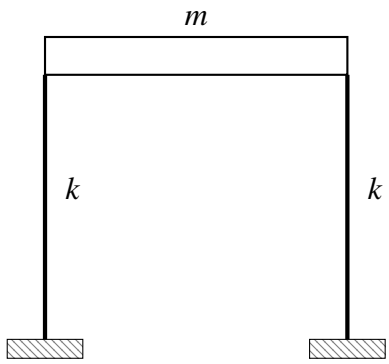


Figure 3.11: Single story building frame model.

legs of the building frame. The mass of the legs is also neglected. The equation for the undamped natural frequency of the single story building model is [2, p. 136]:

$$\omega_n = \frac{1}{2\pi}\sqrt{\frac{2k}{m}}$$

where k is the stiffness of the leg and m is the mass of the floor. For the single story model, each change in leg height is 3" or half of the total leg height of the two-story building. With the mass of the floor unchanged, ideally only the stiffness would cause the change in natural frequency.

Shorter beams have a higher stiffness than taller ones. Overall, any change in natural frequency for the buildings will only come with changes in height of the legs of the building. The mass of the taller legs do increase, but this is discussed in the next section.

The first analysis reviews the RT Pro data sets from the sine wave input testing. Figure 3.12 plots the maximum acceleration amplitude responses of all four building cases that were tested. For all cases the center data point is the observed natural frequency recorded in Table 3.1. Then two data points above and two data points below that frequency were taken with a frequency difference of 0.5 Hz.

Absolute Acceleration Amplitude Data from Sine Wave Input Testing

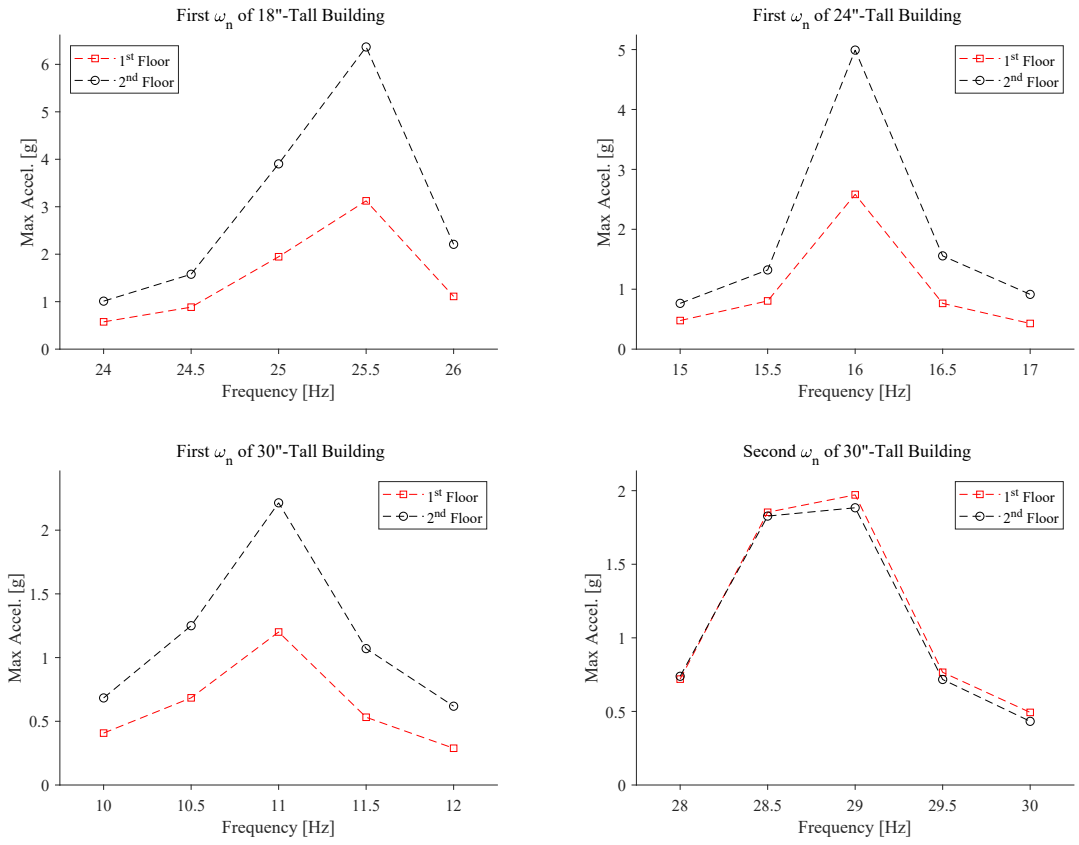


Figure 3.12: Maximum acceleration amplitude response from sine wave input testing for all building heights at frequency ranges around the observed natural frequency.

The 18”-tall (top left plot) building’s max amplitude at its first natural frequency was at 25.5 Hz, 0.5 Hz higher than observed. The observed first natural frequency for the 24”-tall (top right plot) and 30”-tall (bottom left plot) building are the same as experimentally tested at 16 Hz and 11 Hz, respectively. The max amplitude for the 30”-tall building at its second natural frequency (bottom right plot) was at 29 Hz, again the same as observed. Its amplitude at 28.5 Hz is slightly lower, but the max was taken at 29 Hz. Extra data points between 28.5 and 29 Hz could have been taken, but it was preferred to keep the procedure the same between all tests. Overall, three of four tests showed experimental results correlating exactly with observations. These observations could not be precisely quantified as it was assumed to be a rough estimation. It does show however that the normal observer could guess the location of the natural frequency within $\pm$ 0.5 Hz given the change in time and change in frequency are equivalent.

3.4.1 Sine Unrectifier

An issue that can occur when using the time capture feature in RT Pro is that the signal comes out rectified. The movement of the building under excitation is sinusoidal. When a sinusoidal signal is rectified, all of the data is passed through as an absolute value. The signal is visualized like a continuation of half-sine waves all above zero. A second GUI program was created to assist with this problem. For just finding the amplitude of motion, this is not needed, but is used later to plot comparison of mode shapes and how the base input acceleration relates to the building floor acceleration.

Figure 3.13 is the initialized window for the Shaker Data Analyzer. At the top left corner is a drop-down menu that opens a pop-up menu for the user to select the file that needs to be unrectified. The file should be one that was output from RT Pro in the form of an Excel .xls file.

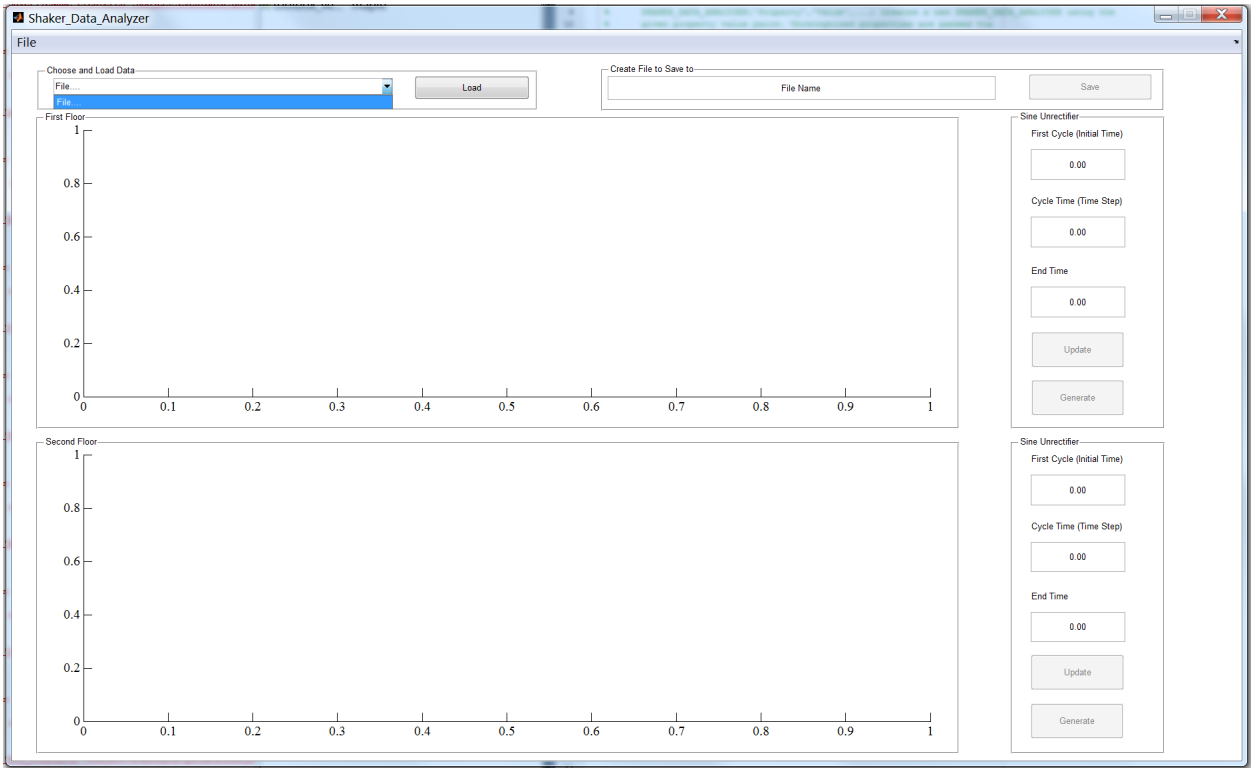


Figure 3.13: Shaker Data Analyzer MATLAB GUI program. Drop-down list in top left for users to select data file.

The program filters out all other file extension types to aid the user in the correct file selection. Once the file is selected, its filename will populate in the text of the drop-down button.

Once the 'Load' button is clicked the actual data is graphed into two separate plots as shown in Figure 3.14. The top plot is for the acceleration of the first floor of the building and the bottom is for the second floor of the building. The plots of the unrectified sine wave can then be visualized. The total time of the signal is 0.8 seconds which is the entire data signal captured from RT Pro during the building shaking. The input boxes to the right of each graph is then used to help in unrectifying the signal. The program requires user input and does not unrectify the data automatically.

There are three data input boxes to aid in unrectifying the sine waves of each floor. The boxes are the same for each plot. The input values can be the same, but will most likely be different as

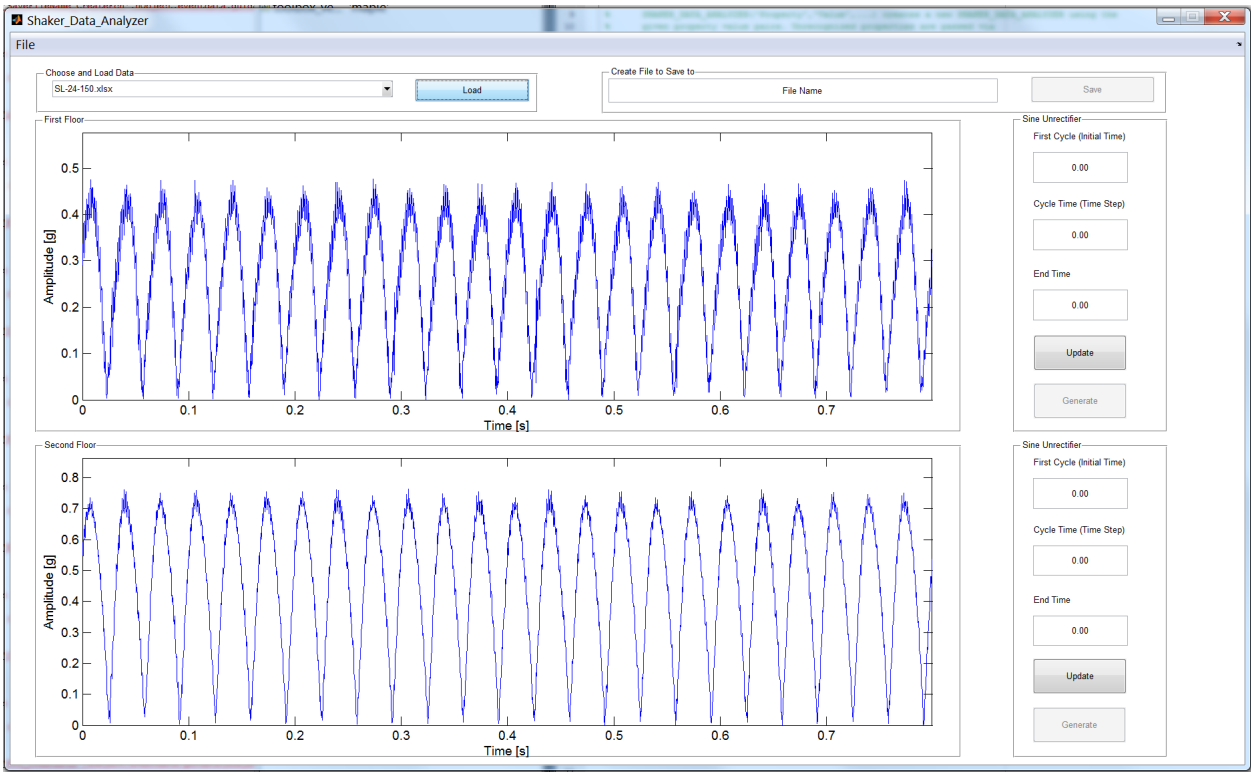


Figure 3.14: Rectified sine waves plotted for each floor in GUI.

the acceleration amplitude for each floor is different. The first input is for the first cycle or initial starting time. This number should be set as close to the first peak in the sine wave as possible. This also creates the first vertical dashed line in Figure 3.15. The second input is for cycle time or time step. This value will generate the remainder of the vertical dashed lines. Each line will be the same time step away until the end of the time axis. The third input is for the end time of the signal. This should only be as large as 0.8 second, but is recommended to be smaller. Otherwise, the plot will be blank past the time of 0.8 second. A smaller time axis makes the sine wave easier to visualize with the lower density of data. The start and end time set the bounds of the data to be saved.

Once the variables are adjusted for finding the period of the sinusoid or at least making sure all vertical dashed lines are within the half-sine waves, the data can be unrectified. The data will only be unrectified with what is shown on the axis and is done by pressing the ‘Generate’ button

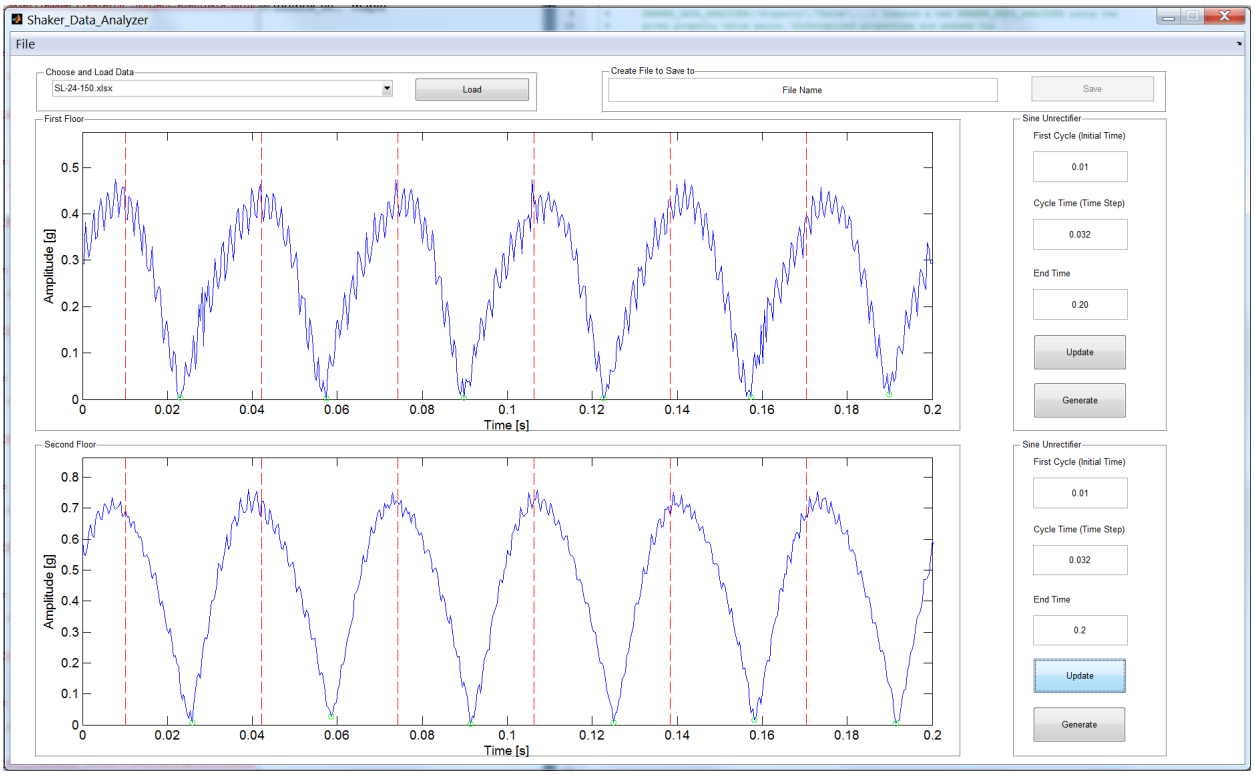


Figure 3.15: Periodic lines plotted to match peaks in sine data.

as shown in Figure 3.16. After the 'Generate' button is pressed, the changes cannot be undone. The process would need to be started over without saving what had been previously done to the data up to that point. The data has a small amount of hysteresis [2, pp. 46-47]. Hysteresis in a system's future response is dependent on its prior response. The sine waves starts to lag or lead after the positive and negative peaks. The floor accelerations lag behind the base acceleration as it changes direction. It is most likely due to a loss of mechanical energy in the legs under cyclical loading because of a small amount of damping from internal friction in the aluminum. After the data sets for both floors has been unrectified, it can be saved as an Excel file with a user definable filename. At the top left of the window the desired filename can be input into the textbox and the 'Save' button creates a pop-up menu for the user to save the file in the folder of choice. This data can now be used for analytical comparison.

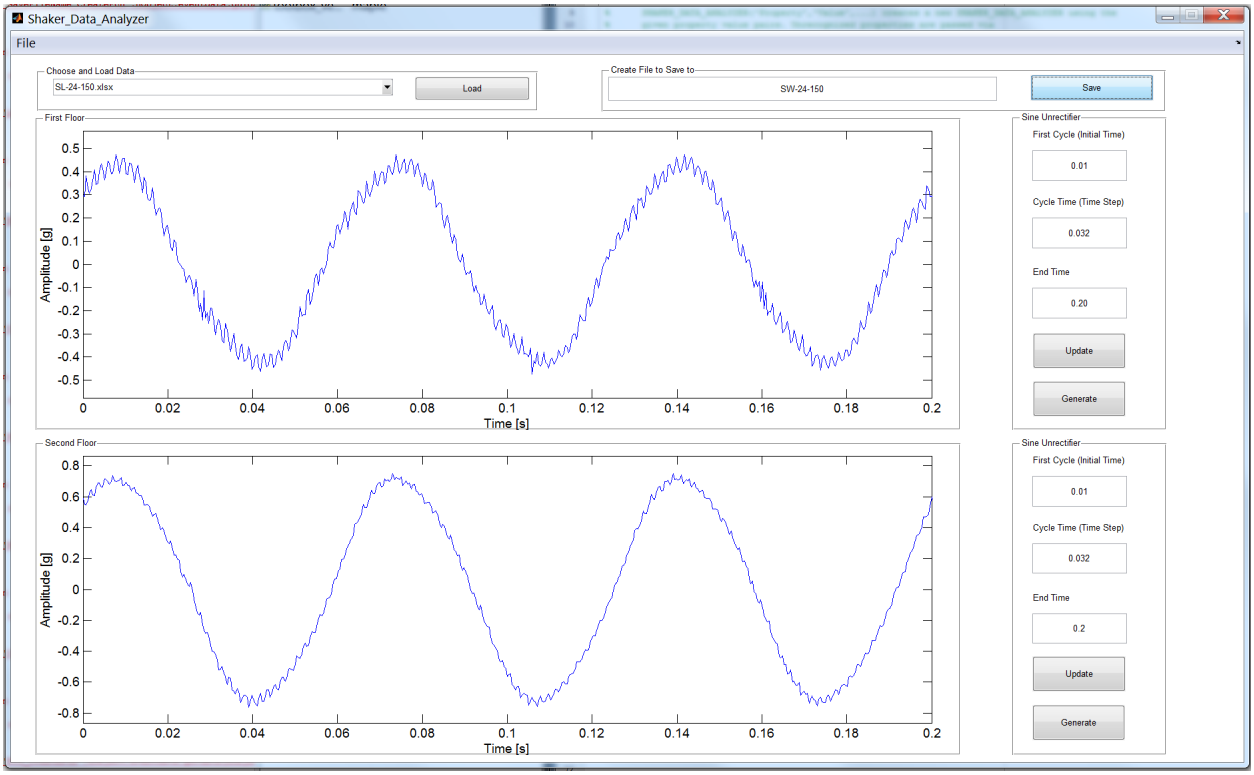


Figure 3.16: Unrectified sine waves ready to be saved and used for better data analysis.

3.5 Comparison with Analytical Results

3.5.1 Experimental Data Compared with Analytical Model

The analytical building model uses a lumped-mass model. This is determined by adding the equivalent mass of the beams, m_{beam}, lumped at their ends with the mass of the floor, m_{floor}, as shown in Equation (3.4):

$$m = m_{eq} = m_{floor} + 2\left(\frac{13}{35}m_{beam}\right) \tag{3.4}$$

The equivalent beam mass is for half of the total leg of the building. The total equivalent mass, m_{eq}, is for a single-story of the building. The stiffness of the beams or legs of the building were estimated as a beam under horizontal loading and is given in Equation (3.5):

$$k = k_{eq} = \frac{12EI}{L^3} \tag{3.5}$$

The derivation for both equations are given in Appendix B.2. The weight of each part of the building are shown in Table 3.2. The weight of each leg was included as it needed to be changed for each test case in the code of `nat_freq_shaker.m`, from Appendix B.3. If the data is not changed for each building condition, the analytical results will not be correct.

Table 3.2: Weight of Building Parts.

Part	Weight [oz]	Beam Weight $\left(\frac{Leg\ \ Weight}{2}\right)$ [oz]
Floor	3.15	-
Leg (18")	3.50	1.75
Leg (24")	4.70	2.35
Leg (30")	6.00	3.00

The code of `nat_freq_shaker.m` is the analytical model completed in MATLAB. The following commands, in that script, are used to find the natural frequencies, which are square roots of the eigenvalues of the matrices. The natural frequency is calculated in radians per second and needs to be converted to Hz for comparison with the sine wave input testing results.

```
[V,Om2] = eig(K,M);
Om = sqrt(Om2);      % Natural Frequency in rad/s
freq = Om/(2*pi);    % Natural Frequency in Hz
```

Table 3.3 shows the observed and calculated natural frequencies as well as the percent difference between the two. The 24" and 30" tests percent difference was less than 5% and only a fraction higher than 5% for the 18" test. Overall, the minimal difference in results shows that

the analytical model gives a good approximation of the actual building natural frequencies. All calculated results were higher than the experimental natural frequencies. It is either that the actual stiffness is lower than what was calculated or the equivalent mass is higher than calculated, but this is not an major issue for this lab.

Table 3.3: Difference of Observed and Calculated ω_n.

Height [in]	Experimental ω_n [Hz]	Calculated ω_n [Hz]	Difference
18	25.5	26.84	5.25%
24	16.0	16.62	3.82%
30 (ω_{n_1})	11.0	11.34	3.14%
30 (ω_{n_2})	29.0	29.70	2.42%

3.5.2 Mode Shape Comparison

The analytical model only accounts for the relative acceleration of each floor. During shaker testing the accelerometers are recording the absolute acceleration of each floor. The relative accelerations' amplitudes of both the experimental and analytical models will be compared. To get the relative acceleration, the base motion acceleration needs to be subtracted from the absolute acceleration. Assuming the damping in the building frame is slight, the phase difference between the base input and floor outputs will be minimal. Displacement data was collected from all four experimental natural frequencies. The amplitude for each was used to construct a sine wave for the base acceleration that was in phase with the absolute accelerometer data. The constructed base accelerations were subtracted from the unrectified absolute accelerations to get the relative accelerations of the tested data. The amplitudes from the relative accelerations are then compared with the mode shapes calculated from the model. The modes shapes are determined from the eigenvectors

of the matrices. The eigenvectors are the column vectors of the matrix `v` from the prior MATLAB code. The first column is the eigenvector for the first natural frequency and the second column is for the second natural frequency. The first row corresponds to the first floor of the building and the second row to the second floor.

The base input is the same sine wave detailed in Section 3.3.2 and expressed in Equation (3.6) [2, pp. 55-60].

$$y = y_0 \sin(\omega t + \phi) \tag{3.6}$$

where y is the sine wave input position, y_0 is the amplitude, ω is the frequency, t is time, and ϕ is the phase offset of the sine wave. Differentiating Equation 3.6 twice expresses the acceleration of the base input, $\ddot{y}$, as shown in Equation (3.7):

$$\ddot{y} = -\omega^2 y_0 \sin(\omega t + \phi) = -\omega^2 y \tag{3.7}$$

The amplitude is given by the recorded encoder data from the ECP cart and is in units of centimeters. The only change is the constants added in front of the sine. Multiplying the positional data by $-\omega^2$ and converting from centimeters to meters gives the acceleration input in m/s^2. The accelerometer data is saved in units of g's from the Photon II. This data must be multiplied by 9.81 m/s^2 per g to convert it to the same units as the base input acceleration.

Normalization is done for both the experimental data and the calculated mode shapes from the model. The data is put into vectors to imitate the calculated mode shapes. It is also easier to normalize a vector in MATLAB using the `norm()` command than calculating it by hand. The data and calculated mode shapes are normalized the same way, in that the sum of the squared vector values equal 1. The normalized data and calculated mode shapes are shown in Figure 3.17.

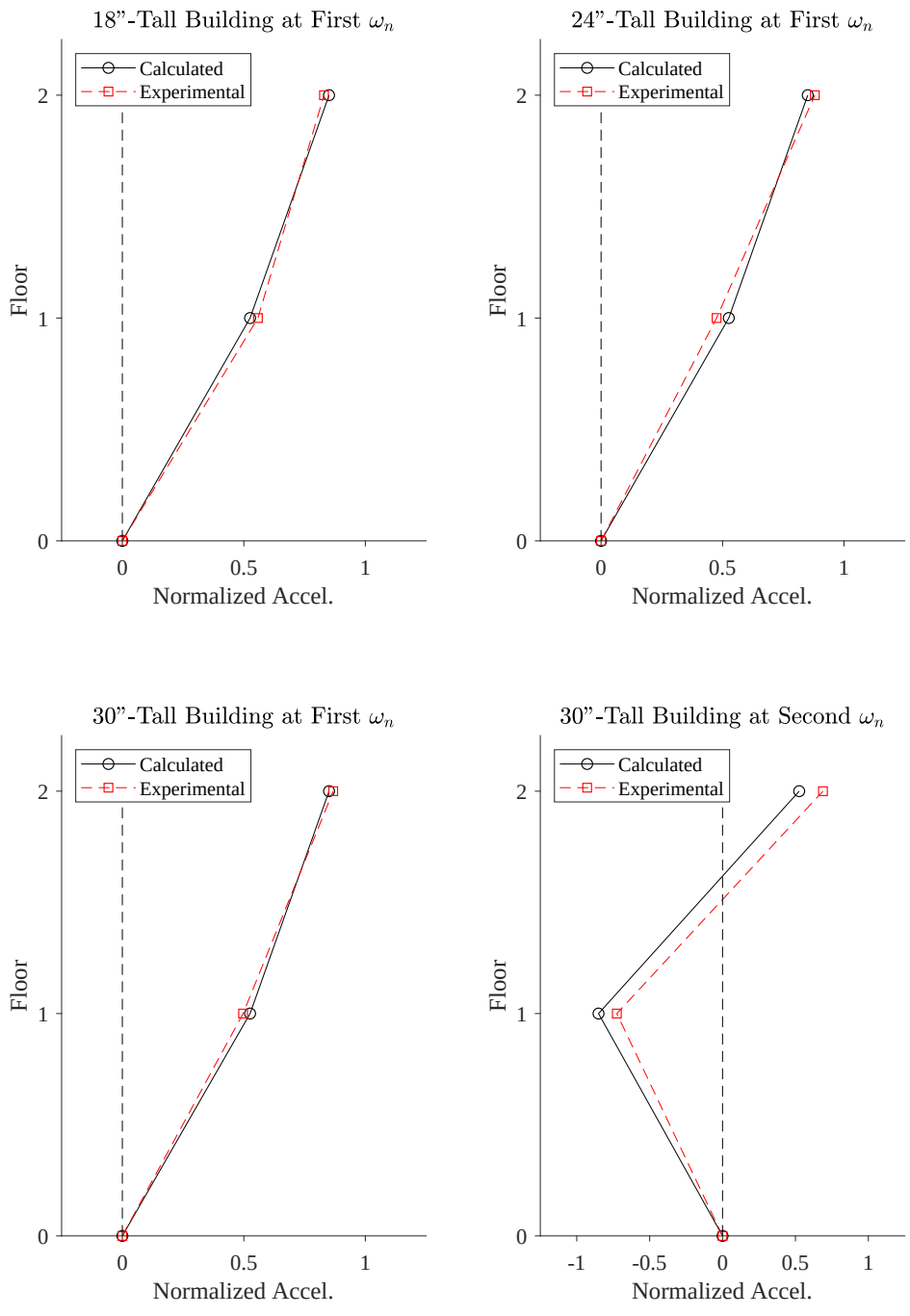


Figure 3.17: Mode shape comparison of experimental and calculated normalized accelerometer amplitudes.

The experimental data and calculated mode shapes for the 18”-, 24”-, and the first natural frequency of the 30”-tall building are in good agreement with each other. There is a similar gap between floors on the second natural frequency of the 30”-tall building. The second natural frequency of the 30”-tall building has similar gaps between the first and second floors. The difference between how close the tests were at the first natural frequency compared to the second natural frequency is not clear. Overall, it does show the calculated and experimental mode shapes are very similar. Therefore, looking at the results in Figure 3.17, because there is good agreement, the light damping approximation is supported.

In the future, a simpler option for doing the mode shape comparison could be to add a third accelerometer to the base of the building. RT Pro would record the base acceleration and be able to directly compare with the floor accelerations. Simulink would then just be used to move the base and not record the position data. This approach would save the step of generating the base input acceleration from the Simulink base position data. This test method could be used to determine the actual amount of damping in the building. If a third accelerometer is used to measure the base acceleration; the phase difference between the floor accelerations and base input acceleration can be used to calculate the modal damping ratio. The modal damping ratio could also be experimentally determined by displacing the floors, allowing them to freely vibrate, and then looking at the decay in the time-domain response.

4 Tuned-Mass Damper

4.1 Purpose

The extension of the building shaker lab involves designing and implementing a tuned-mass damper (TMD) for the building. TMDs are common in sky scrapers where they are installed to help reduce movement caused by wind forces or ground vibrations. Their function is to remove excessive vibration and prevents the structure from moving outside expected limits. The goal is to reduce the movement around a building's natural frequencies. This is not an issue for the model building in the previous lab as it was tested so that the natural frequencies could be visualized. The goal of this section was to design the tuned-mass damper housing and make it capable of quickly changing its mass. Three different metals with different densities were used to adjust the weight within the same used volume. The reduction in absolute acceleration amplitude for each weight is compared to the previous building model not having the TMD. The spring length for the TMD had to be adjusted to fit in its housing. The analysis in reducing the floors amplitude was used to identify the TMD's actual spring stiffness. The frequency responses of the previous and new building models are compared to better understand the changes on the frequency spectrum.

4.2 Model

The previous building model is used, but with the TMD added to the top as shown in Figure 4.1. The relative position of the TMD is described by x_d, the stiffness of the spring on both sides of the TMD is k_d, and its mass is m_d. The TMD mass is the combined mass of the metal weight and

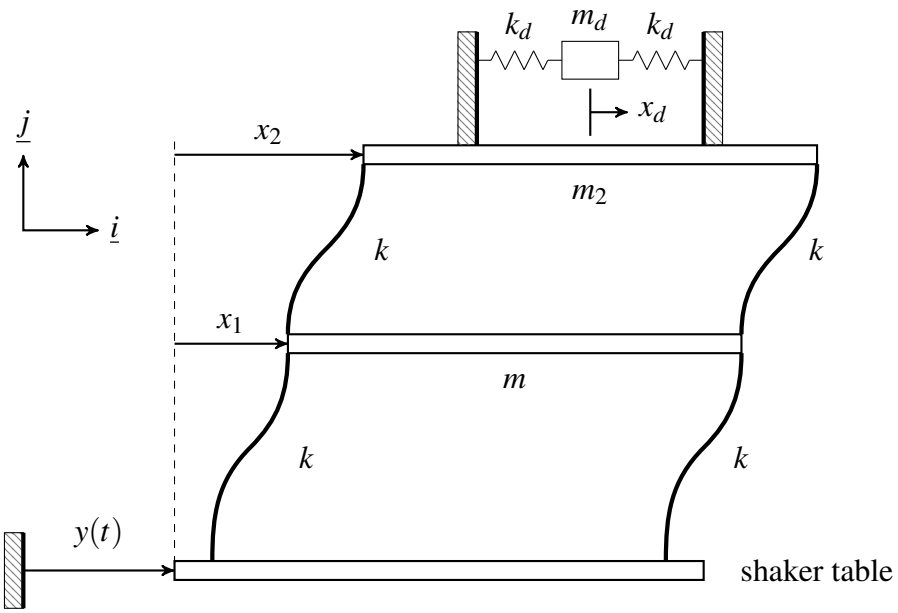


Figure 4.1: Two-story building model with tuned-mass damper.

the moving cart it sits in. The first floor remains the same and the only change in the mathematical model comes from the addition of the tuned-mass damper on the second floor. The equivalent mass of the second story plus the mass of the TMD housing is given by m_2. The KD and FBD for the first floor are given in Figure 4.2.

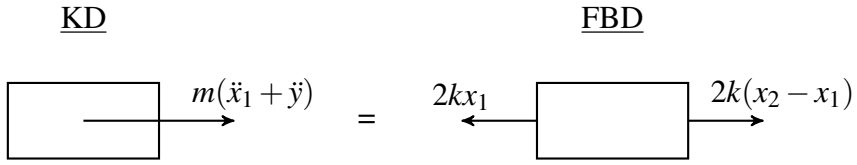


Figure 4.2: Kinetic and free body diagram for first floor of building, the same as is used in the previous two-story building model.

Applying the rate form of linear momentum conservation gives Equation (4.1), the equation of motion for the first floor:

$$m\ddot{x}_1 + 4kx_1 - 2kx_2 = -m\ddot{y} \tag{4.1}$$

The KD and FBD for the second floor are given in Figure 4.3.

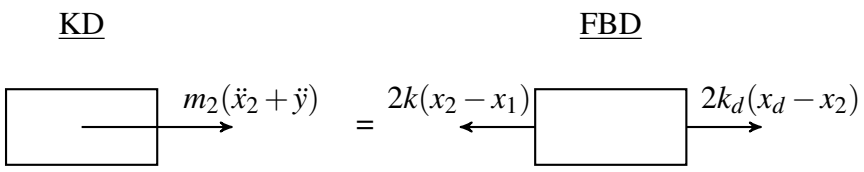


Figure 4.3: Updated kinetic and free body diagram for second floor of building.

Again applying the rate form of linear momentum conservation gives the second floor's equation of motion, Equation (4.2):

$$m_2\ddot{x}_2 + 2(k+k_d)x_2 - 2kx_1 - 2k_dx_d = -m_2\ddot{y} \tag{4.2}$$

The KD and FBD for the tuned-mass damper are given in Figure 4.4.

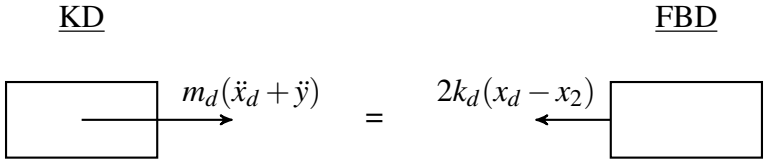


Figure 4.4: Kinetic and free body diagram for the tuned-mass damper.

Once more applying the rate form of linear momentum conservation gives Equation 4.3, the TMD's equation of motion:

$$m_d\ddot{x}_d + 2k_dx_d - 2k_dx_2 = -m_d\ddot{y} \tag{4.3}$$

where $\ddot{x}_d$ is the relative acceleration of the damper mass. Expressing these equations of motion in matrix-vector form gives the mass and stiffness matrices for the building with tuned-mass damper.

$$\underline{\mathbf{M}} = \begin{bmatrix} m & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_d \end{bmatrix}$$

$$\underline{\mathbf{K}} = \begin{bmatrix} 4k & -2k & 0 \\ -2k & 2(k+k_d) & -2k_d \\ 0 & -2k_d & 2k_d \end{bmatrix}$$

4.3 Experimental Procedure

4.3.1 Physical Model

The TMD is shown in both Figures 4.5 and 4.6. Figure 4.5 gives a top view of how it works.

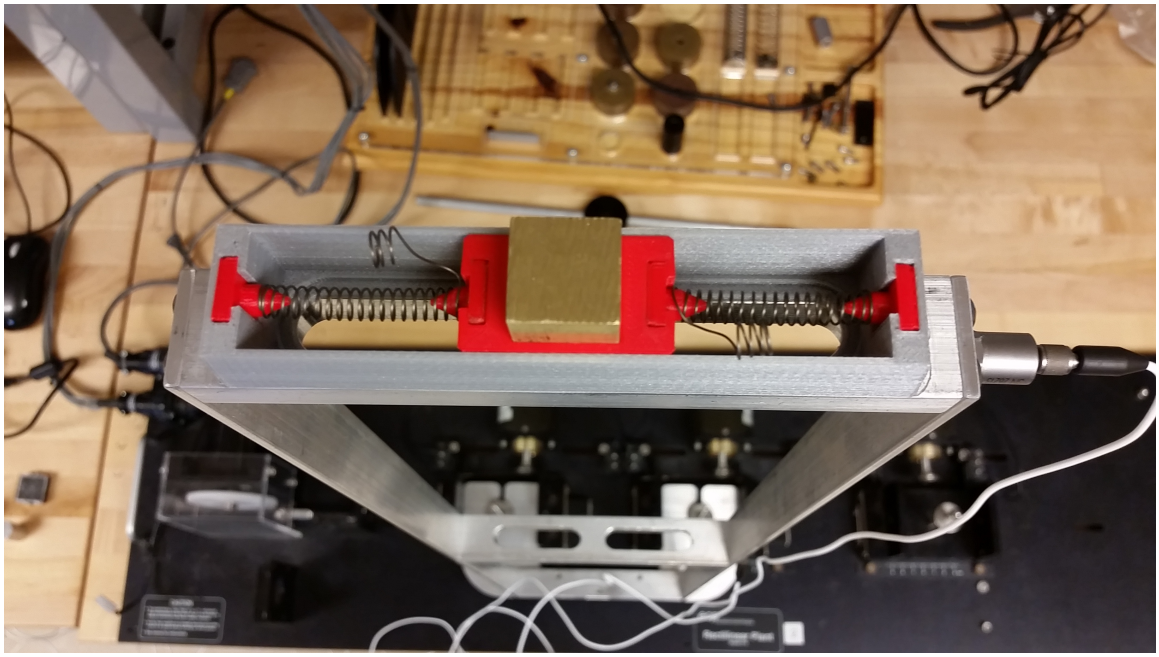


Figure 4.5: Top view of tuned-mass damper on two-story building.

A small red cart sits inside of the gray housing channel. Gravity keeps the weight and cart sitting in the channel. The cart contains a 3/4" square pocket to hold the pieces of metal or any other weight that can fit in that cart. The channel makes sure the cart and weight can only move in the same axis of motion of the building floors. The channel is the same width as the floors and is attached to

the top floor by superglue. A couple drops of superglue keep the channel rigidly fixed to the floor. A utility knife blade can fit between the bottom of the channel and the floor and cut through the superglue to remove it. Two springs, one on each side, work to restore the weight and cart to the center of the channel. Slots in both the channel and the cart were for the red cone-shaped spring holders. The cone ends keep the spring coil radially centered on the holders. Figure 4.6 shows how the springs had to be shortened for the TMD to work.

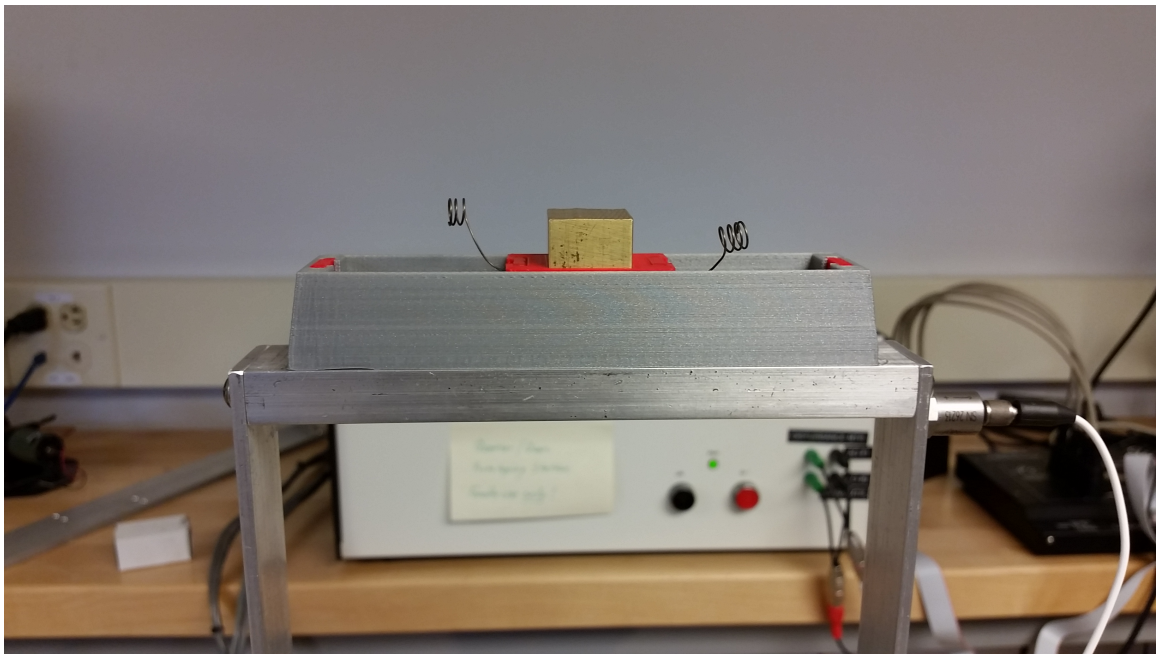


Figure 4.6: Front view of tuned-mass damper.

A few coils of each spring were stretched out to reduce the length of the spring. The spring's length as shown was at 1.375" length with a spring rate at 0.9 lb/in, with the spring's original length at 1.75". If the spring is too stiff or has to compress too much to fit in the channel the TMD's effect is diminished because its movement relative to the motion of the building is hindered. A shaft through the center of a stiffer spring would help in the current design, as the springs would buckle and fly out of the channel. If it is too soft the TMD will move too much and could end up being in

phase with the floor and act like the floor and the weight are a larger combined mass. The weight of the grey outer housing is 0.038 lbs and the red TMD cart is 0.019 lbs. Table 4.1 gives the weights for the TMD masses used.

Table 4.1: TMD Mass Weights

Material	Weight [lb]	Metal + Cart Weight (m_d) [lb]
Aluminum	0.048	0.067
Titanium	0.078	0.097
Brass	0.165	0.184

4.3.2 Setup

This lab uses similar parts to the previous lab. The same sine input Simulink model is used. The absolute acceleration data was then recorded with the accelerometers through the Photon data acquisition unit and saved with RT Pro. The tallest building size was used as it would give, in theory, the largest visible change in amplitude for the observer to notice. This test is applicable to all building heights.

4.4 Data Processing

The amplitudes of the absolute acceleration data are found from the generated Excel files from RT Pro. This process is the same as from Section 3.4.

4.5 Comparison with Analytical Results

For the TMD experiment three different weights were used. The metals used, in order of increasing density, were aluminum, titanium, and brass, respectively. The aluminum and titanium were 3/4” round bar stock and the brass was 3/4” square bar stock, all 1” long. The shaker oscillated at the first natural frequency of the 30”-tall building, 11 Hz. Figure 4.7 plots the absolute acceleration amplitudes for each TMD weight and the building with no TMD. The weights are the combined material and TMD cart weight. Overall, the TMD reduces the amplitude by over 50%

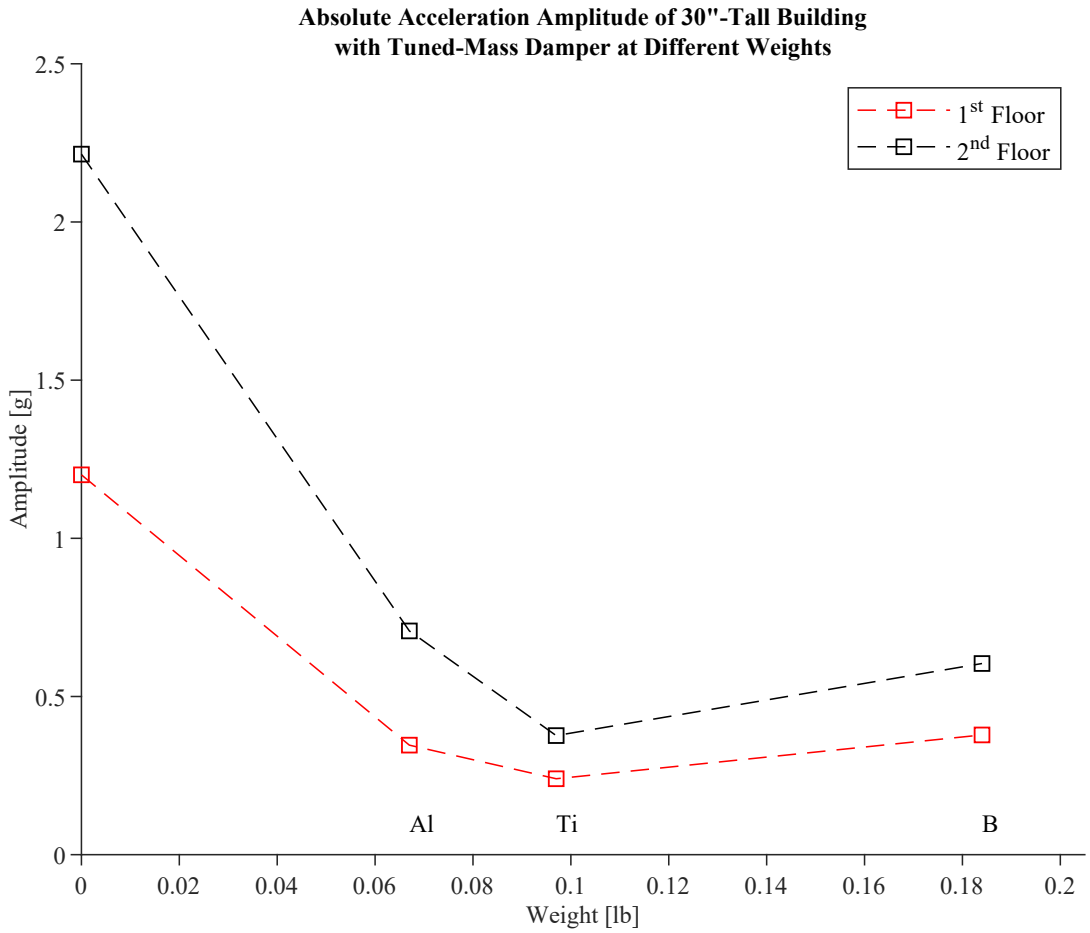


Figure 4.7: Maximum acceleration response range around the first natural frequency for 30”-tall mock building with tuned-mass damper.

in all cases. The titanium weight had the best reduction in amplitude at over 75%. It becomes evident that just increasing weight is not the optimal solution, but overall having a TMD makes a very considerable difference in reducing the amplitude of the building's response.

4.5.1 TMD Stiffness Identification

The springs were reduced in length to fit into the TMD housing and the spring rate was most likely also affected. The spring's compressed length was 0.67". This caused the total allowable movement to be reduced from 1.08" to 0.71". An estimate for the actual spring rate could be calculated after testing the different weights. The TMD is used optimally when its natural frequency is equal to the input frequency. If the TMD is analyzed alone, its natural frequency is given by Equation (4.4):

$$\omega_n = \frac{1}{2\pi}\sqrt{\frac{2k_d}{m_d}} \tag{4.4}$$

This calculation removes the influence of the base input and second floor of the building. Using the original spring stiffness of 0.9 lb/in (10.8 lb/ft), the natural frequency of the TMD for each mass was calculated and reported in the second column of Table 4.2. Based off of that calculation the

Table 4.2: TMD Calculated Frequencies

Material	TMD ω_n [Hz]	TMD ω_n with k_d = 0.6 lb/in [Hz]
Aluminum	16.22	13.24
Titanium	13.48	11.00
Brass	9.78	7.99

brass weight should have had the greatest amplitude reduction with its calculated natural frequency closest to that of the actual building. Although the brass weight should theoretically be more

effective, the titanium weight actually provided the greatest reduction in amplitude. Using the experimental natural frequency for the 30”-tall building at its first natural frequency in Table 3.1, a stiffness was back calculated with the titanium weight that came out to be 7.2 lb/ft or 0.6 lb/in. While the calculated stiffness is an estimate it is assumed to be close to the actual stiffness knowing that the titanium weight was the most effective. The third column reflects the calculated natural frequencies at the stiffness estimated from the data.

After shortening the spring the stiffness became smaller. Theoretically, it should have increased as a shorter spring is stiffer than a longer one. Most of this has to do with how the spring was shortened and then stilled used in the TMD. The TMD was shortened to avoid buckling and this can still be an issue if the TMD is pushed by hand more than 1/4” to either side of center. Also, as the spring moves farther than 1/16” to either side of center, the spring increasing in length comes off of one of the spring rests and is then not providing any force to the cart. It is not known how much the cart is moving at the natural frequency, but if only one spring is working at a time the spring rate would be twice as much as calculated. This would then match the manufacturers spring rate, 1.2 lb/in, for a spring length of 1.375”.

4.5.2 Changes in Frequency Response

After investigating the stiffness of the TMD, it is useful to look at the changes in the building’s frequency response with and without the TMD. To calculate the frequency response functions (FRF) for the building model it is assumed that the building is at steady-state response due to base motion. This model, just like in Section 3.2, does not include damping. Looking back at Equation (3.3), $\underline{\mathbf{f}}(t) = -\underline{\mathbf{m}}\,\underline{\ddot{\mathbf{y}}}(t) = \underline{\mathbf{m}}\omega^2 y_0 \sin\omega t$ and the steady-state response is $\underline{\mathbf{x}}_{ss} = \underline{\mathbf{u}} \sin\omega t$ where ω is the input frequency.

The base motion input for the two-story building is

$$\underline{\mathbf{f}}(t) = \begin{bmatrix} m \\ \\ m \end{bmatrix} \omega^2 y_0 \sin \omega t$$

Plugging $\underline{\mathbf{x}}_{ss}$ into Equation 3.3 gives

$$\underline{\mathbf{M}}(-\omega^2 \underline{\mathbf{u}} \sin \omega t) + \underline{\mathbf{K}}(\underline{\mathbf{u}} \sin \omega t) = \underline{\mathbf{m}} \omega^2 y_0 \sin \omega t$$

Dividing by $\sin \omega t$ simplifies the equation to

$$(\underline{\mathbf{K}} - \omega^2 \underline{\mathbf{M}})\underline{\mathbf{u}} = \underline{\mathbf{m}} \omega^2 \mathbf{y}_0 = \begin{bmatrix} m \\ \\ m \end{bmatrix} \omega^2 y_0$$

Solving for a scaled version of $\underline{\mathbf{u}}$ gives the position related frequency response:

$$\frac{\underline{\mathbf{u}}}{m\omega^2 y_0} = (\underline{\mathbf{K}} - \omega^2 \underline{\mathbf{M}})^{-1} \begin{bmatrix} 1 \\ \\ 1 \end{bmatrix}$$

The acceleration related frequency response for the two-story building without a TMD can be found by multiplying the position related response by $-\omega^2$:

$$\frac{\underline{\ddot{\mathbf{u}}}}{m\omega^2 y_0} = -\omega^2 (\underline{\mathbf{K}} - \omega^2 \underline{\mathbf{M}})^{-1} \begin{bmatrix} 1 \\ \\ 1 \end{bmatrix}$$

The base motion input for the two-story building with TMD has one more term:

$$\underline{\mathbf{f}}(t) = \begin{bmatrix} m \\ \\ m_2 \\ \\ m_d \end{bmatrix} \omega^2 y_0 \sin \omega t$$

Determining the acceleration related frequency response for the building with the TMD is done by the same process as described above, but there is now one additional term:

$$\frac{\ddot{\underline{\mathbf{u}}}}{m\omega^2 y_0} = -\omega^2(\underline{\mathbf{K}} - \omega^2\underline{\mathbf{M}})^{-1}\begin{bmatrix} 1 \\ \frac{m_2}{m} \\ \frac{m_d}{m} \end{bmatrix}$$

The frequency responses for the two building models are plotted in Figure 4.8. The original building's natural frequencies are at the calculated frequencies, 11.3 Hz and 29.7 Hz. When the TMD is added to the model, the third natural frequency is in nearly the same place as the second for the original building. On the lower end of the frequency response the original first natural frequency is now split into the first and second natural frequencies of the building with the TMD. Those frequencies are near 8.9 Hz and 13.9 Hz. Two anti-resonances for the first and second floor, respectively, are approximately 11 Hz. The model confirms that the addition of the TMD reduces the acceleration amplitude when oscillating the building at the first natural frequency.

The housing weight was not added to frequency calculation of the regular two-story building model. There is concern about the TMD mass loading the building model. The natural frequencies of the first natural frequency of the original two-story building, adding in the TMD housing and then the TMD housing, TMD cart, and titanium weight, are 10.9 Hz and 9.9 Hz, respectively. The addition of the housing reduces the calculated natural frequency by 0.4 Hz and should give a similar reduction experimentally. Visually, during shaking, the TMD was moving out of phase with the second floor. It seems to show it working as it should be, moving against the movement of the floors. If it were mass loading the second floor, the TMD would have been moving in phase with the second floor. Again, visually the TMD looked to be working as intended, but better verification

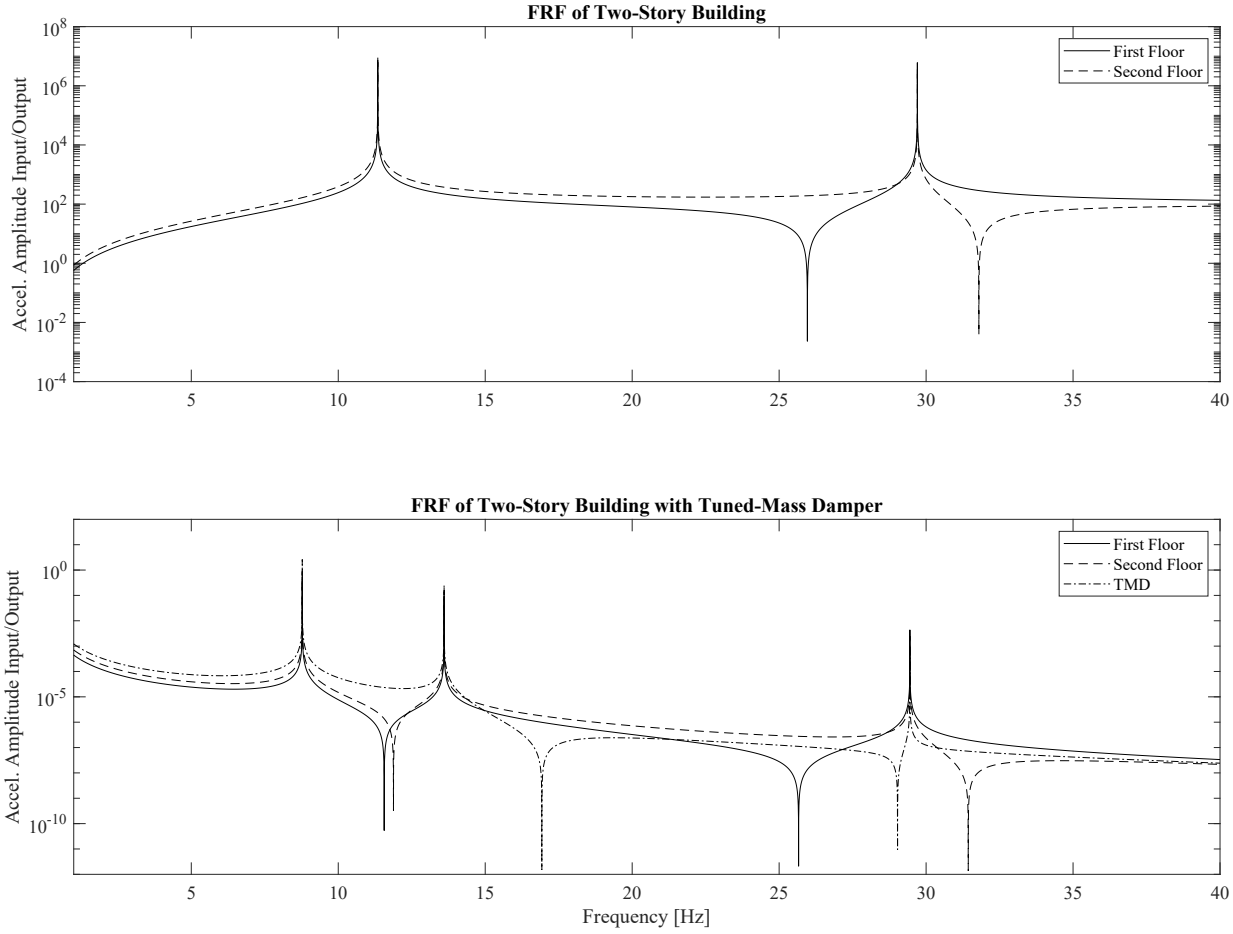


Figure 4.8: Frequency responses of floors for two-story building and two-story building with TMD.

is needed.

Future work would include testing different spring rates along with the different weights. The optimal stiffness to weight ratio is dependent on the allowable amplitude of movement [2, pp. 832-836]. Using springs that do not need adjusted would help to fix the problems encountered when trying to identify the actual spring rate. Also, the frequency response could include the damping coefficient found from the future work in the previous chapter. The TMD could also be redesigned to be more easily removable from the floors. It would be helpful to run a sine-sweep on

the building to verify the TMD is working at the first natural frequency and not mass loading the second floor. The building with TMD could be shaken at its new natural frequencies, around 8.8 Hz and 13.6 Hz, to compare if the maximum acceleration amplitudes are lower, higher, or similar to the original building's at 11 Hz.

5 Other Lab Extensions

5.1 Rotating Unbalance

5.1.1 Purpose

The second lab design was focused on vibration caused by rotating objects, specifically a rotating unbalance. Rotating unbalance is another type of forcing like base motion described in the previous chapters. This issue or design problem is commonly found in engines, washing machines, tires, or anything else that uses rotary motion to create power or do work. This lab is scaled down from most real world applications and uses small video game controller vibration motors. These motors have weights attached to the motor shaft and are on only one half of the shaft. By having the weight offset on the shaft, vibration is induced once the motor is rotating. The motors come in two sizes as shown in Figure 5.1. The motors and housings are set in the ECP carts and are forced by the induced vibration. The motion of the carts should be visibly seen while position data is collected with the encoder attached to the ECP cart. A calculated amplitude of vibration can be found from known variables and compared to the experimental positional amplitude.

5.1.2 Model

The motors are modeled as drawn in Figure 5.2. The actual electric motors are the same, and only the masses are different. They move at the same angular position, θ, and run at the same angular velocity, ω. The radius, r, of the respective eccentric masses, m_e, are also shown for reference, but not to scale. The centroid of the eccentric mass is detailed in Figure 5.3.

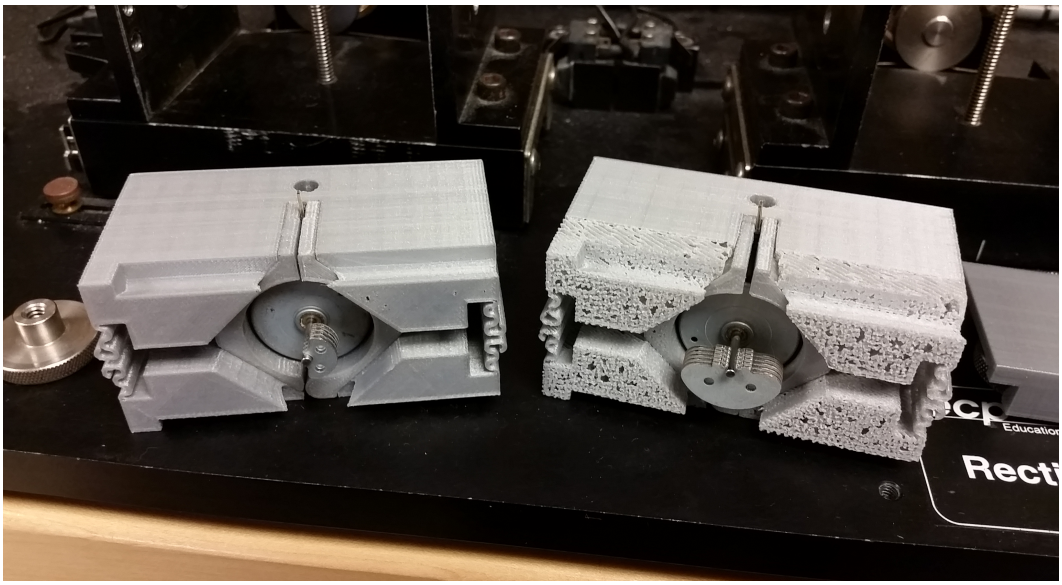


Figure 5.1: Small and large unbalanced mass motors in their protective housings.

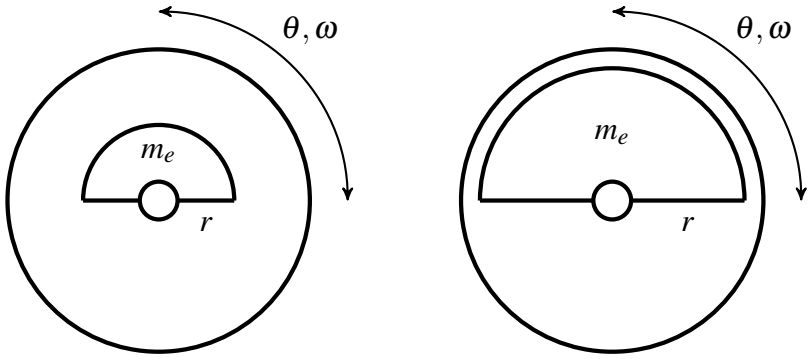


Figure 5.2: Small and large vibration motor diagrams.

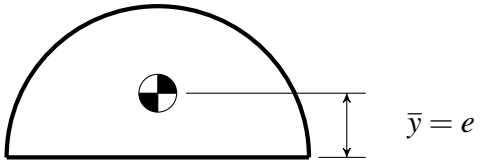


Figure 5.3: Centroid (eccentricity) of a semi-circle.

The masses are assumed to be ideal two-dimensional semi-circles. In this test the centroid, $\bar{y}$, and eccentricity, e, of the semi-circle are the same values. This model does not take into account the thickness of the actual masses. This is neglected because the measured movement is only in the direction perpendicular to the motor shaft. The carts of the ECP plant keep the motion in a single

axis. From Statics and Mechanics of Materials [5, pg. 216], the centroid of a semicircular area is given in Equation (5.1):

$$\bar{y} = e = \frac{4r}{3\pi} \tag{5.1}$$

The ECP cart, motor, and housing are considered a single system. The only potential force that can act upon the system is friction in the linear rail the system moves along. This is considered negligible and there is assumed to be no damping in the rail. Considering this, Equation (5.2) [2, p. 289] formulates the amplitude, X, of the rotating unbalance:

$$X = \frac{m_e e}{m_{eq}} \frac{r^2}{\sqrt{(1-r^2)^2 + (2\zeta r)^2}} \approx \frac{m_e e}{m_{eq}} = \frac{4}{3\pi} \frac{m_e r}{m_{eq}} \tag{5.2}$$

because

$$r >> 1 \;\; (X \to \frac{m_e e}{m_{eq}} \;\text{ as }\; r \to \infty)$$

where the ratio r of the input frequency, ω, to the natural frequency, ω_n, is very large because the input frequency is assumed to be very large compared to the natural frequency of the system. The eccentric (offset rotating) mass is defined by, m_e, and the total mass of the system is m_{eq}.

5.1.3 Experimental Procedure

The experimental setup is shown in Figure 5.4. A controller box on the left has an Arduino inside to control the speed. Two rotary potentiometers can adjust the speed of both motors at the same time through pulse-width modulation (PWM). In this experiment only one motor was used at a time, although two motors can be controlled at one time. The system is isolated from any input and is able to move unconstrained in the lateral direction (side-to-side on the rails). This allows the

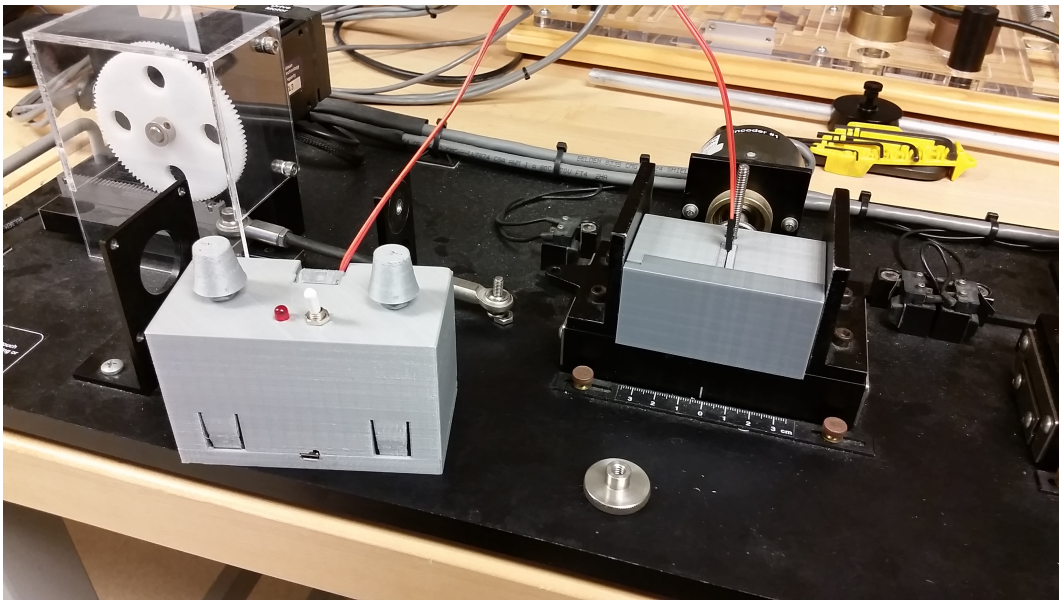


Figure 5.4: Experimental test setup vibration motor testing.

motor have complete control on all of the motion of the system. Initially, different speed settings with the potentiometers were set to be tested: 33%, 66%, and 100% of full speed. The final testing only used the motor at full speed and is discussed more in the next section.

Figure 5.5 is the Simulink block model for recording the motion of the cart due to the rotating unbalance. The input block for the disturbance motor is set to zero so that it does not run. The connecting rod joining the disturbance motor and the cart is also removed just in case the disturbance motor does run. The positional encoder records the force response from the vibration motor on the cart.

5.1.4 Data Processing

Initially, both motors were run with the speed at 100%. Visual inspection showed no movement of the cart on the base and indicated a very small amplitude of vibration. Only the 100% speed run was conducted after seeing no visible movement. Despite this, the data recorded on the

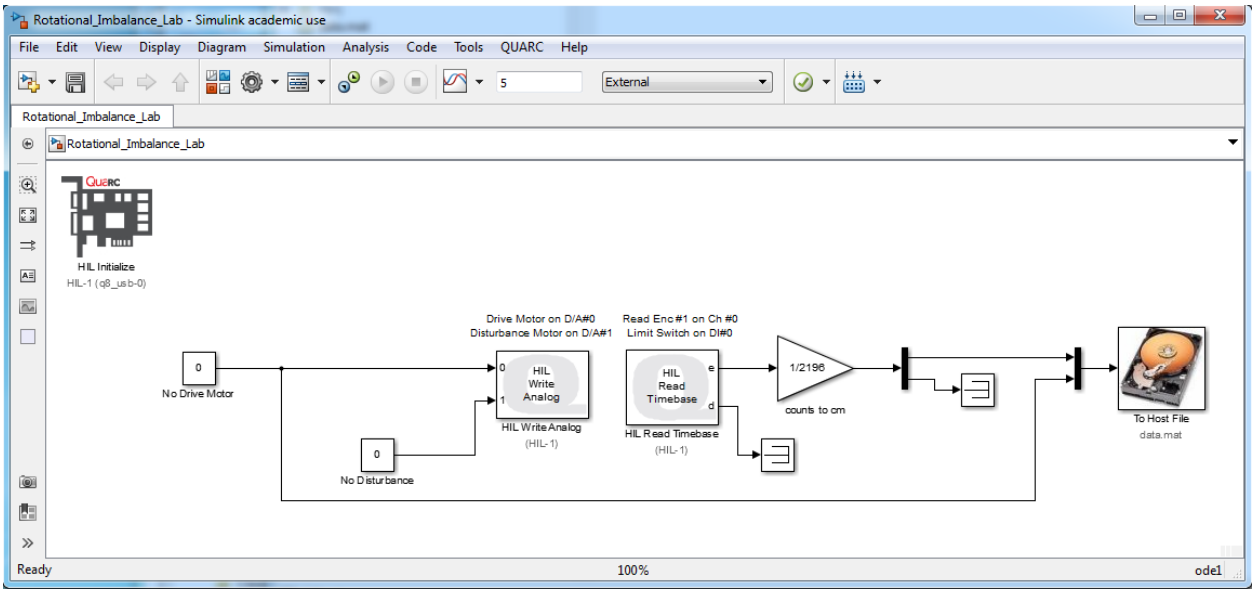


Figure 5.5: Simulink diagram of rotating unbalance.

computer was able to show the movement generated from the motors. The measured parameters of the motors are given in Table 5.1. The mass of the ECP cart is 500 grams. The large mass motor

Table 5.1: Motor Parameters

	Offset Mass [g]	Housing + Motor Mass [g]	Offset Mass Radius [cm]
Large Mass Motor	8	61	1.03
Small Mass Motor	2	65	0.58

housing ended up being lighter than the small mass motor because of an issue during 3D printing. Figures 5.6 and 5.7 are the sinusoidal response of the system's motion with the large offset mass and small offset mass, respectively. The data for both has been adjusted to be centered about zero displacement. The initial start-up of the motor was recorded offset and was moved in the plots to visualize the amplitude of the steady-state response.

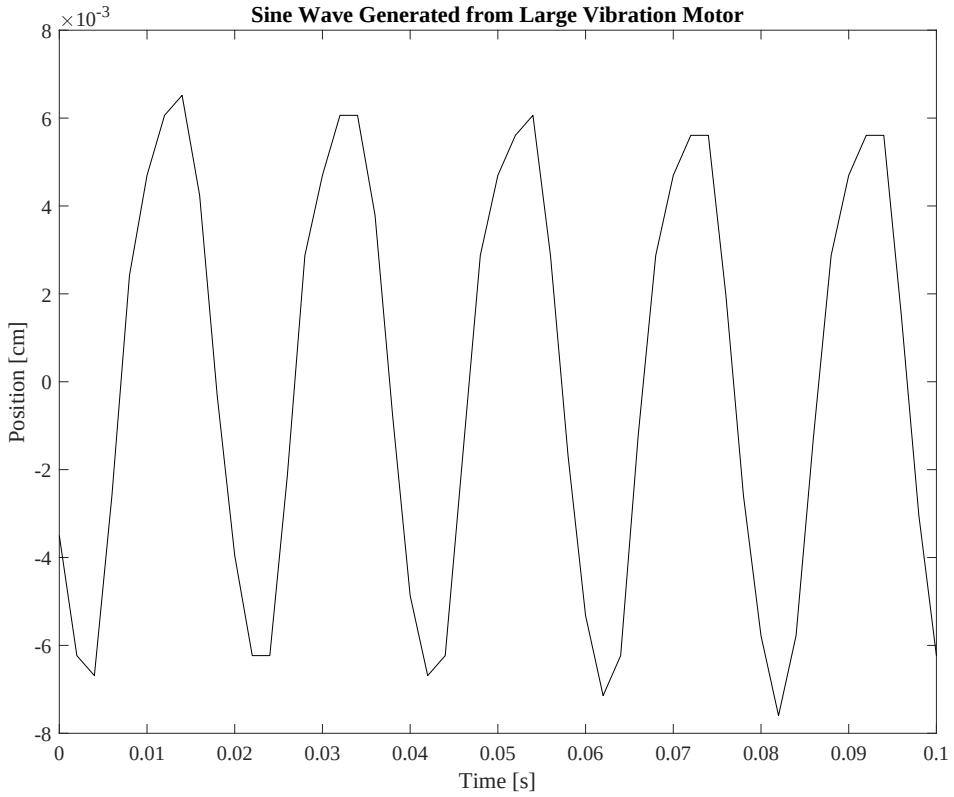


Figure 5.6: Sinusoidal position response of system with large offset mass.

5.1.5 Analysis of Results

The amplitude was much lower than expected. Table 5.2 gives the experimental and calculated values for the cart movement amplitude from the vibration motor. The amplitude for the large and small motor are 6.6×10^{-3} cm and 1.6×10^{-3} cm, respectively. It was assumed to be small

Table 5.2: Vibration Motor Amplitude

	Calculated Amplitude [cm]	Experimental Amplitude [cm]	Difference [%]
Large Mass Motor	6.2×10^{-3}	6.6×10^{-3}	6.1
Small Mass Motor	8.7×10^{-4}	1.6×10^{-3}	45.6

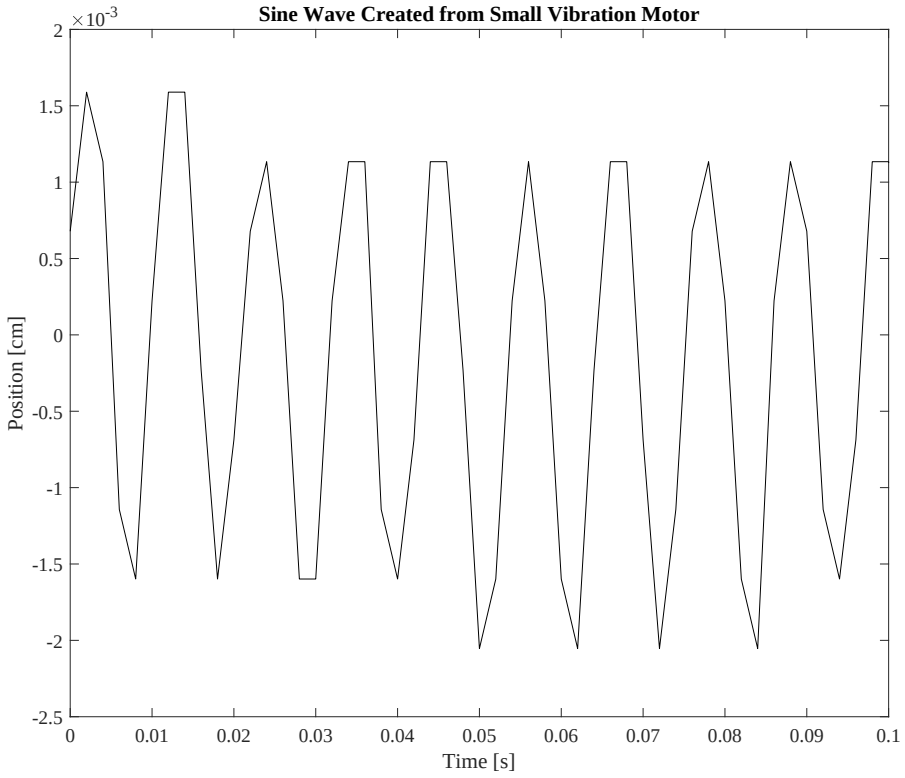


Figure 5.7: Sinusoidal position response of system with small offset mass.

as during the visual inspection the cart could not be seen to move. The large motor amplitude had less than 7% difference whereas the small motor amplitude was over 45%. Both data samples have plateaus at some of the peaks. The data for both tend to drift lower as time increases. The large motor amplitude calculation is close to experimental results. The small motor is not and it may have to do partly with the rotating masses eccentricity. The smaller rotating mass has flats on each end and is not as close to an actual semi-circle as the larger mass is. If the eccentricity was at least 1mm farther away from the shaft, the calculated amplitude would be 1×10^{-3} cm. This would make the percent difference 37.5%, a reduction of 8.1%. While the total percent difference is still high, it may not be best to assume the mass as an ideal semi-circle.

Also, the accuracy of the encoders is 2196 counts per centimeter which yields around $4 \times$

10^{-4} cm per count. This equates to 26 encoder counts, for the large motor, and 4 encoder counts, for the small motor. The number of counts for the motors are not significant compared to 2196 count resolution for a complete encoder revolution. It would be preferential to displace the cart more and use more of the measurement range of the encoders for cleaner graphs.

Although the large motor results were decent, a considerably larger vibration motor would be preferred. Future work needs to include finding a larger vibration motor that students could visibly make the cart move. The comparison of results would be better with a larger eccentric mass and with the exact eccentricity of the offset mass. Finding the exact eccentricity could make a considerable difference in the comparison. The motors used in this lab were used to first see if they could even move the carts. They were able to, but barely and therefore no stiffness was added to system. The amplitude of movement needs to be large enough to move with an attached spring to simulate an elastic foundation. Some form of damping can be added if large amplitudes are encountered near the system's natural frequency. Connecting a spring to the system would also eliminate the drift in the sinusoidal data.

5.2 Structural Health Monitoring

5.2.1 Purpose

The third designed lab was based on the field of structural health monitoring (SHM). The idea of this technology is for structures or systems to be continuously monitored with very few to thousands of sensors. Each sensor sends feedback to a computer to look for changes in behaviour that are outside of the optimal range of operation. With the usage of many sensors it is possible to more accurately pin point the abnormal responses source and follow through with appropriate

corrective action. SHM is a preventative damage and identification of potential or actual damage tool to help with the structure being observed [6]. Most structures are not observable from all angles at all times. The idea of the lab was to have a set structure with hidden internals, designated as a black box. The internals are interchangeable and can be configured in many ways. The small black box is tap tested on one end with a single accelerometer on the other. The measured frequency response for several known configurations are compared. Comparison is done with the data only and no analytical model.

5.2.2 Experimental Procedure

In Figure 5.8, the test setup for the lab is shown. The test done here is a static hammer test,

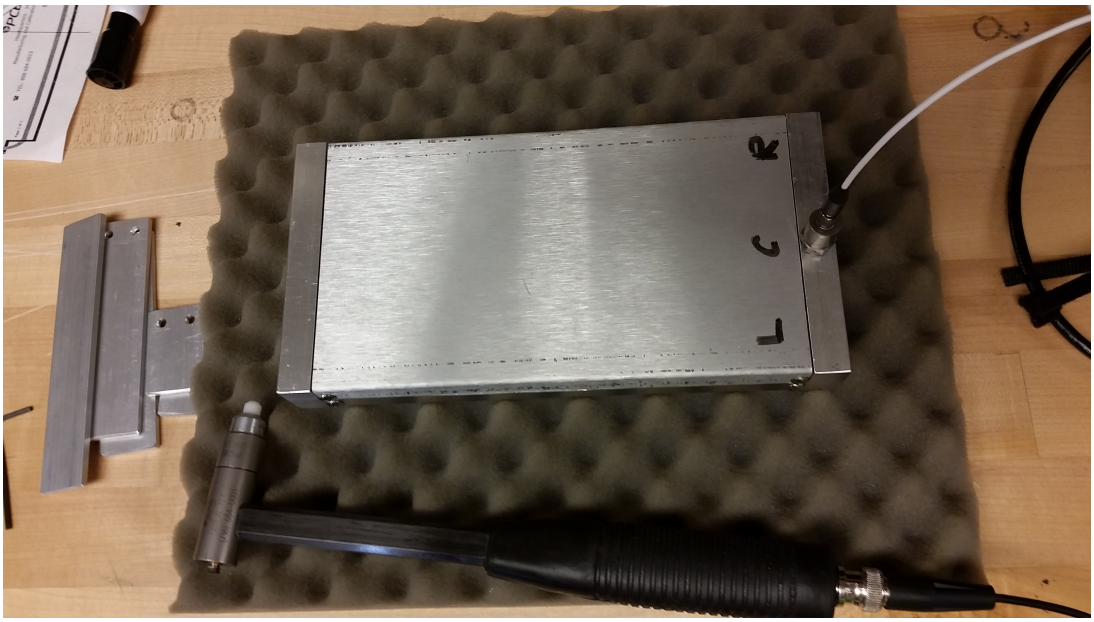


Figure 5.8: Experimental test of black box.

where the hammer hits the same spot and the accelerometer is roving. L,C, and R are the positions for the accelerometer to sit during a hammer test. The three positions become more clear in Figure 5.9. Initially there are three bars that run along the inside of the box and are bolted to each end

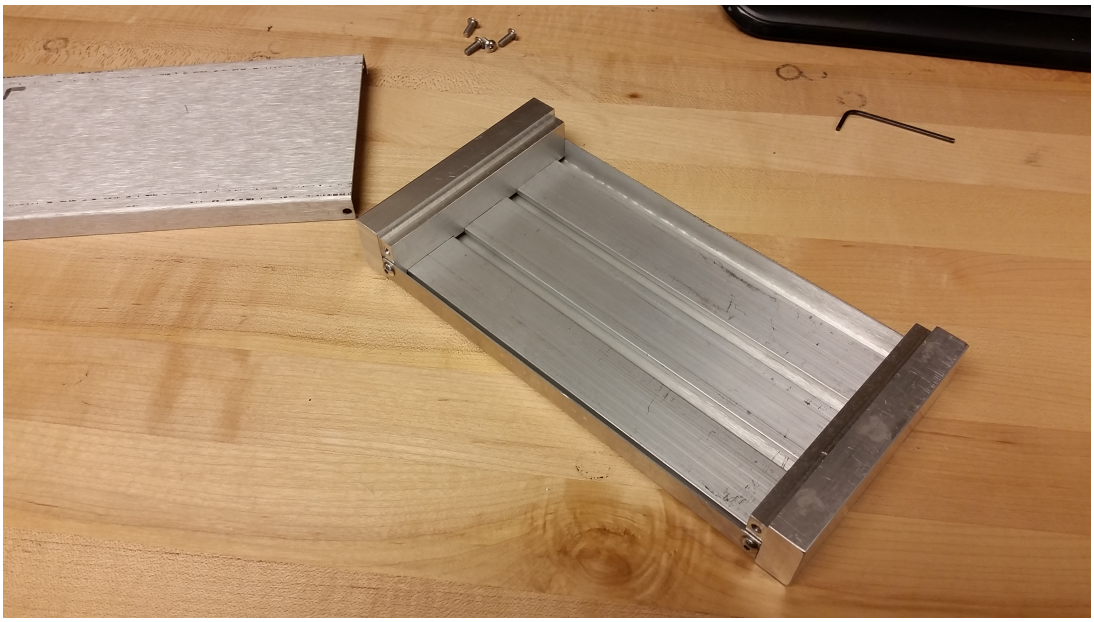


Figure 5.9: Black box with top cover removed.

block. The goal is to get an accelerometer reading from each bar position of left (L), center (C), and right (R). Figure 5.10 shows how the bars are bolted to the end blocks. The box is easy to

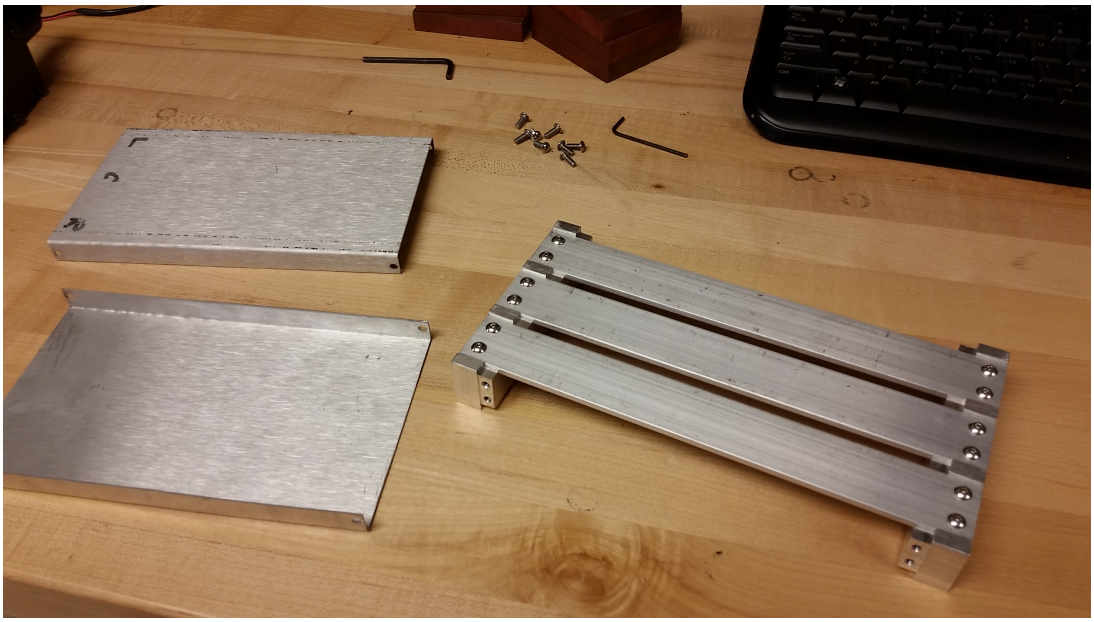


Figure 5.10: Both covers removed.

assemble and disassemble. The configurability of the box is shown in Figure 5.11. As can be seen

there are a few bars of different lengths to change the characteristics of the structure. Another possibility was to just loosen a bolt that secures a bar and see if that would make a change in modal data properties.

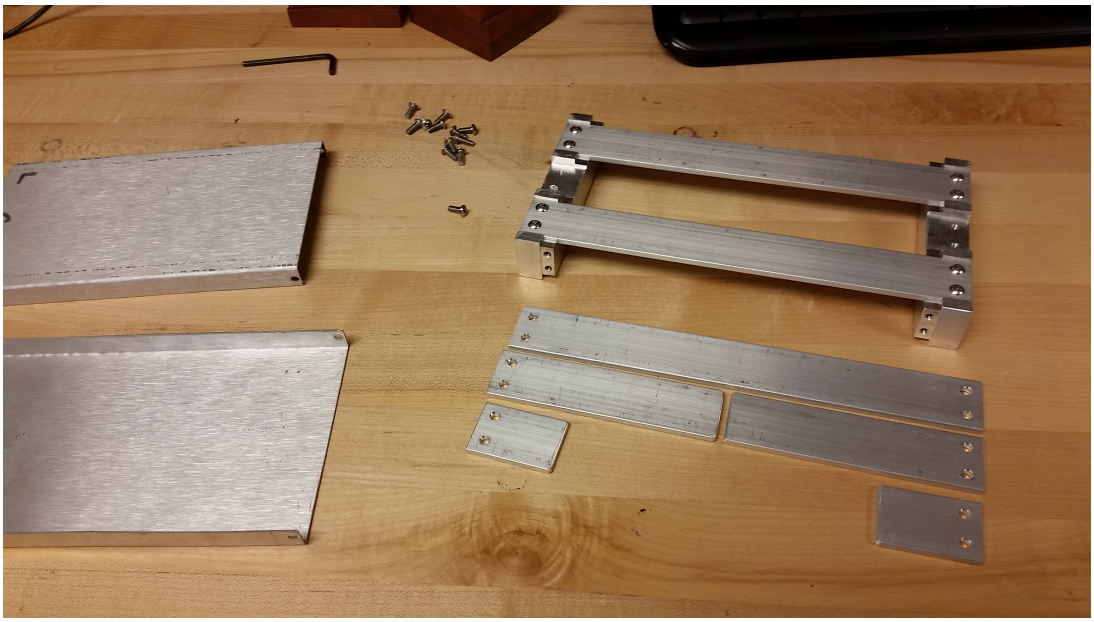


Figure 5.11: Configurable options of bars in black box.

5.2.3 Analysis of Results

Four box configurations' frequency responses are plotted together in Figure 5.12. The first plot is of the default configuration with three normal bars. The second has the left bar with electrical tape that works as a solid layer damper. The third has the center bar removed. The fourth has all bars removed.

There are two main changes throughout the four plots. The first is the peak created from adding the solid layer damping tape after 800 Hz, although nothing can be hypothesized from it. The second is that the main peak at 550 Hz is closer 500 Hz and is an order of magnitude lower when all bars are removed. The peak at 550 Hz may be around the natural frequency of the beams.

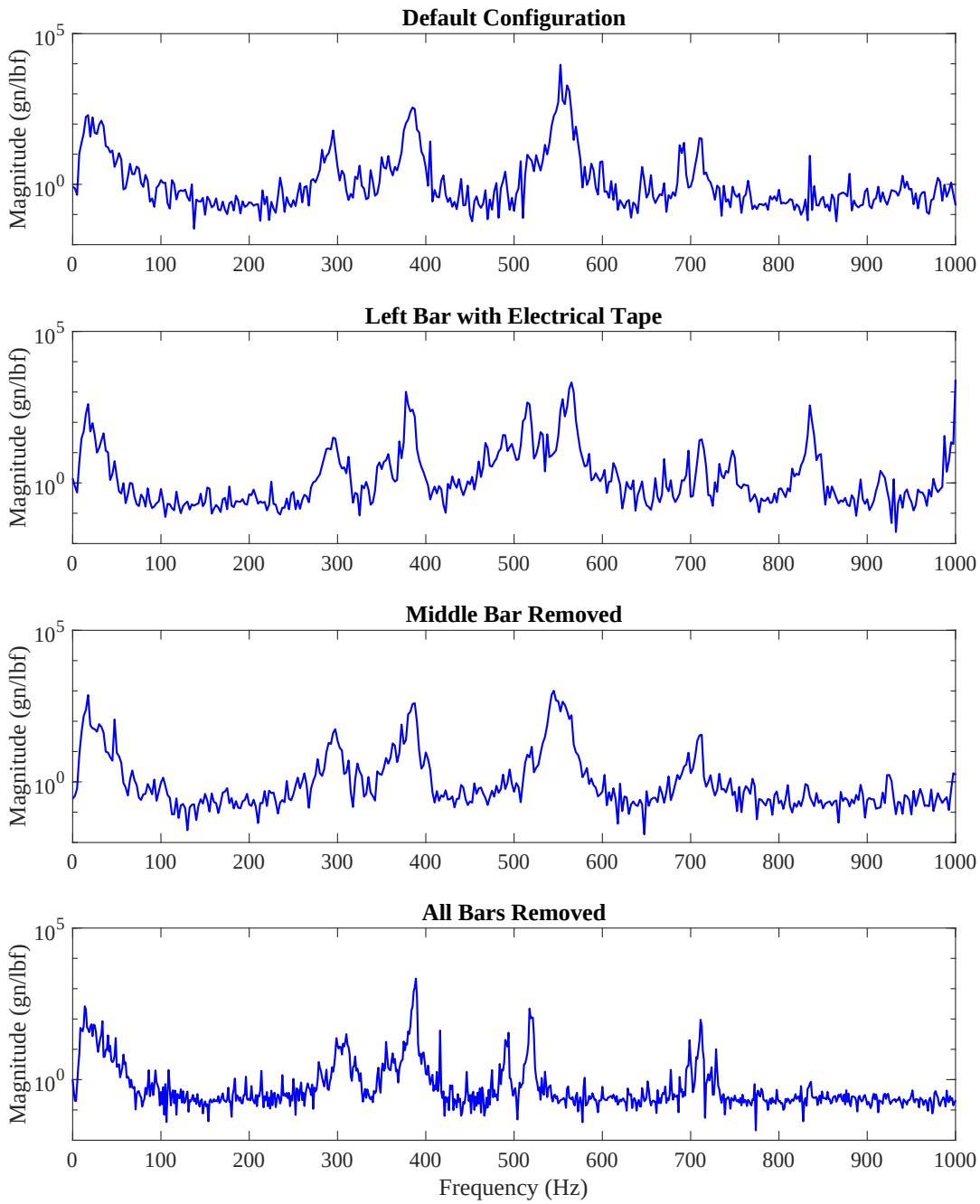


Figure 5.12: Frequency responses of four box configurations for comparison.

Many factors affected how the lab progressed; time and data were the biggest factors. There was not enough time that could properly be devoted to this lab. One sample set of data between a dozen configurations gave no indication of major changes happening during its excitation. Ob-

viously, one sample set is not enough to draw conclusions from, but minimal frequency response changes between the sets halted the focus on this lab. The coherence should also have been checked with each configuration. If the coherence is close to 1 the system has nearly linear response between the hammer input and accelerometer output, but this was not verified. Clean data from the structure was more difficult to gather because of the thin sheet metal covers. The thin aluminum does not dissipate the hammer vibrations as quickly as the end blocks and hidden bars. The simple design was used a quick and easy way to manufacture the box, but more stiff covers would be more beneficial.

Several changes are recommended for further development of this lab. First, the changes made to the structure should be minimized and there should be a small amount of them. Second, a baseline cannot be set when not enough tests of the same configuration are used. Third, the structure should be made simpler to hopefully better correlate changes in frequency response data with changes in the structure. Lastly, the coherence in the system should be closely analyzed along with the frequency response.

6 Conclusion

Using the designed program EEP, in Chapter 2, students are able to better understand the FRF data taken during experimental modal analysis. Several methods were discussed for calculating the modal damping ratio and how those methods were visualized. Distinguishing between good and poor quality FRF data sets is also made possible through EEP. Natural cubic splines were presented as an alternative to normal point-to-point plots for structural mode shapes. They provide a more realistic representation of a structure's mode shape.

In Chapter 3, a two-story model building was designed and shaker tested to visually determine the range to test around the building's natural frequencies at different heights. Accelerometer data for each floor is collected during sine wave input shaker testing. After testing, the range of acceleration amplitudes are plotted together. The peak determined the experimental natural frequency of the building at that height. Another application was designed to unrectify the accelerometer data for use later with the building's mode shapes. An analytical model successfully represented the behavior of the building. The experimental data and calculated mode shapes were close and the results justify that the building has slight damping. For future work, a third accelerometer could be used to directly measure the base input acceleration instead of calculating it from the base input position data. If desired, two options are given for experimentally determining the damping in the building.

In Chapter 4, a tuned-mass damper was attached to the top of the two-story model building. The springs for the TMD had to be shortened and therefore changed the stiffness of the system. Three different metal materials were used for the weights in the TMD. The building is shaker

tested and the amplitudes were compared. The stiffness can be back calculated from the weight that causes the greatest reduction in amplitude. Using the calculated stiffness for the TMD, the frequency responses for both building models were compared. In the future the springs should be changed so that the stiffness would be known and then possibly use different mass-spring combinations to find the set with the greatest reduction in building amplitude.

In Chapter 5, the first lab uses a motor with an offset mass run at full speed. The offset mass induces vibration on a moving cart and the position is recorded by an encoder. A calculated amplitude is compared to the experimental amplitude by knowing the mass of the system, the offset mass's mass, and the center of mass of the offset mass. Two offset mass sizes were tested. The smaller had almost no induced movement and the larger had just enough to make a decent comparison. The lab works but a motor with more power and a larger offset mass is needed for a good analytical to experimental comparison. The second lab uses a designed black box to be tested with experimental modal analysis for the intended purpose of structural health monitoring. The box is 'flawed' so that testing with different box configurations will produce different frequency responses. The original intent was to leave the configuration unknown during testing, but it was difficult to determine the differences between the configurations from modal analysis alone. It would be possible to do so if many tests were averaged and individual tests by students could be compared to those configurations averages. Otherwise simplifying the structure or number of configurations is desired.

Overall, the two main goals of the thesis were accomplished. The first goal of expanding the functionality of EMAP was completed with the designed program EEP and is ready for implementation alongside EMAP. The calculation and visualization of different methods for modal damping ratio estimation are useful and the code can be changed for other methods if desired in

the future. Mode shapes plotted using splines increase aesthetics over regular point-to-point plots. The second goal, based on the design of four new labs, was completed, despite a few of the labs needing further development. Three of the four labs gave good results; however, only the first, the base motion lab, correlated analytical models to experimental data well enough to be used directly in a new lab assignment. The tuned-mass damper and rotational unbalance labs gave good experimental results, but need recommended design changes. The design changes need to be tested and if verified would then be ready to be used in new lab assignments. The structural health monitoring lab needs a considerable more amount of work and time to get to a point where it could be considered for use as a lab.

List of References

[1] Easy Modal Analysis Program (EMAP). Dr. Daniel Kawano. Rose-Hulman Institute of Technology, IN, US. kawano@rose-hulman.edu. 27/09/2014.

[2] Rao, Singiresu S. *Mechanical Vibrations, Fifth Edition.* Upper Saddle River: Prentice Hall, 2011. Print.

[3] Richardson, Mark H. & Formenti, David L. *Parameter Estimation From Frequency Response Measurements Using Rational Fraction Polynomials.* Nov. 1982, Orlando, Fl. n.p. Print.

[4] Leader, Jeffery J. *Numerical Analysis and Scientific Computation.* Pearson Education, 2004. Print.

[5] Riley, William F., Sturges, Leroy D., Morris, Don H. *Statics and Mechanics of Materials: An Integrated Approach, Second Edition.* New York: John Wiley and Sons, Inc., 2002. Print.

[6] Farrar, Charles R. & Worden, Keith. "An introduction to structural health monitoring." *Philosophical Transactions of the Royal Society A 365* (2007): 303-315. Web. 16 Mar. 2016.

[7] P. Cermelj, 'UFF File Reading and Writing', 2004. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/6395-uff-file-reading-and-writing. [Accessed: 12- Apr- 2017]

Appendices

A EEP Files Appendix

A.1 EEP GUI MATLAB Code

This section includes the GUIDE layout of EEP and accompanying MATLAB code created for the user interface and plotting the CMIF.

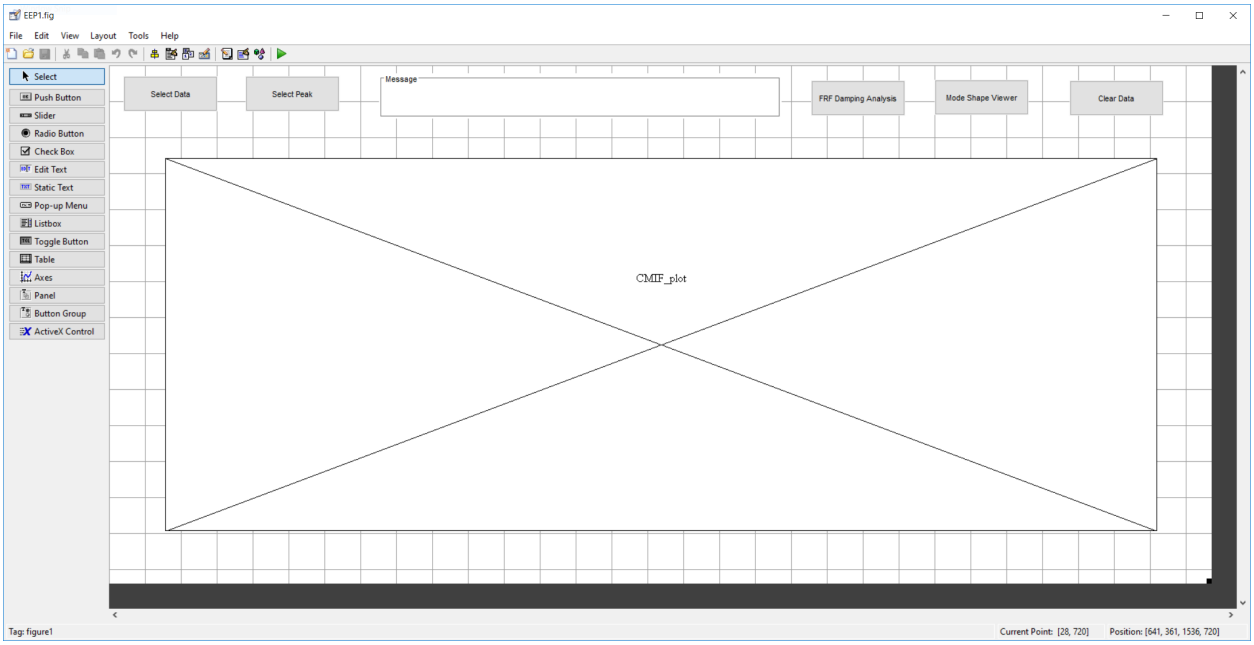


Figure A.1: GUIDE layout design of EEP.

```
function varargout = EEP1(varargin)
% EEP1 MATLAB code for EEP1.fig
%      EEP1, by itself, creates a new EEP1 or raises the existing
%      singleton*.
%
%      H = EEP1 returns the handle to a new EEP1 or the handle to
%      the existing singleton*.
%
%      EEP1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in EEP1.M with the given input arguments.
%
```

```
%      EEP1('Property','Value',...) creates a new EEP1 or raises the
%      existing singleton*.  Starting from the left, property value pairs
%      are applied to the GUI before EEP1_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
%      application stop.  All inputs are passed to EEP1_OpeningFcn via
%      varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help EEP1

% Last Modified by GUIDE v2.5 26-Sep-2016 16:50:50

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @EEP1_OpeningFcn, ...
                   'gui_OutputFcn',  @EEP1_OutputFcn, ...
                   'gui_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
% End initialization code - DO NOT EDIT


% --- Executes just before EEP1 is made visible.
function EEP1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to EEP1 (see VARARGIN)

% Choose default command line output for EEP1
handles.output = hObject;

% Disable all push buttons off
set(handles.Select_Peak,'Enable','off')
set(handles.FRF,'Enable','off')
set(handles.Mode_Shapes,'Enable','off')
set(handles.Clear_Data,'Enable','off')

% Clear plot area
```

```
axes(handles.CMIF_plot)
cla('reset')

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes EEP1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = EEP1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in Select_Data.
function Select_Data_Callback(hObject, eventdata, handles)
% hObject    handle to Select_Data (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Creates pop-up box interface for user to get .mat file from EMAP
[modesFileName,modesPathName] = uigetfile({'*.mat', ...
                                           'MAT-files (*.mat)'}, ...
                                           ['Select the .mat file ', ...
                                            'containing the mode ', ...
                                            'shapes.']);

data = load([modesPathName,modesFileName]); % data becomes all variables
                                            % from data structure in user
                                            % selected file

% loop finds the natural frequencies of the measured structure
% and indicates where they are in the total freq. array
for kk = 1:length(data.NatFreq)
   for ii = 1:length(data.freq)
      if data.freq(ii) == data.NatFreq(kk)
         ind(kk) = ii;
      else
      end
   end
end

f_ind = data.freq(ind);

axes(handles.CMIF_plot)
cla('reset')
```

```
% Plot the CMIF
semilogy(data.freq(ind),data.CMIF(ind),'or',data.freq,...
    data.CMIF, '-b', 'linewidth', 1);
set(gcf,'color','w')
xlim([0 max(data.freq)*1.02])
xlabel('Frequency (Hz)')
ylabel('Magnitude (gn/lbf)')
title('CMIF')
axis auto

% Adjust uicontrol handles
set(handles.Select_Peak,'Enable','on')
% set(handles.FRF,'Enable','off')
% set(handles.Mode_Shapes,'Enable','off')
set(handles.Clear_Data,'Enable','on')

% Create first message
set(handles.Message_Box,'String',...
    'Select the bounds of the natural frequency peak of interest.')

data.ind   = ind;
data.f_ind = f_ind;
handles.data = data; % add data to global gui variable

guidata(hObject, handles); % update gui variables

% --- Executes on button press in Select_Peak.
function Select_Peak_Callback(hObject, eventdata, handles)
% hObject    handle to Select_Peak (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Select the frequency band to analyze

data = handles.data; % Load in gui data

% Re-plot the CMIF_plot
axes(handles.CMIF_plot)
cla('reset')

semilogy(data.freq(data.ind),data.CMIF(data.ind),'or',data.freq,...
    data.CMIF, '-b', 'linewidth', 1);
set(gcf,'color','w')
xlim([0 max(data.freq)*1.02])
xlabel('Frequency (Hz)')
ylabel('Magnitude (gn/lbf)')
title('CMIF')

f_ind = handles.data.f_ind;  % Recall frequency indices

% Let user select points of interest to
% generate frequency band limit lines
[x,y,key] = ginput(2); % Only x axis data used
x1 = [x(1) x(1)];      % First frequency bound
```

```
x2 = [x(2) x(2)];       % Second freguency bound
y_lim = [10E-5 10E5];
hold on
axis(axis)

% plot vertical dashed lines where user selected
l1 = semilogy(x1,y_lim,'--k');
l2 = semilogy(x2,y_lim,'--k');

x1 = min(x);             % Make x1 the lower frequency bound
x2 = max(x);             % Make x2 the upper frequency bound

f_up = (f_ind > x1);     % All frequency lower than x1 equals zero
f_dn = (f_ind < x2);     % All frequency higher than x2 equals zero

search = f_up.*f_dn;     % All frequency outside of the range between
                         % x1 and x2 equal zero

NF = find(search); % Natural Frequency user selected

data.x1 = x1; % add variables to data array structure
data.x2 = x2;
data.NF = NF;
handles.data = data;  % add data to global gui variable

assignin('base','data',data); % Pass data through to workspace

% Adjust uicontrol handles
set(handles.FRF,'Enable','on')
set(handles.Mode_Shapes,'Enable','on')

% Create first message
set(handles.Message_Box,'String',...
    'Select the bounds of the natural frequency peak of interest.')

guidata(hObject, handles); % update gui variables


% --- Executes on button press in FRF.
function FRF_Callback(hObject, eventdata, handles)
% hObject    handle to FRF (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

FRF_DA

% --- Executes on button press in Mode_Shapes.
function Mode_Shapes_Callback(hObject, eventdata, handles)
% hObject    handle to Mode_Shapes (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

data = handles.data;
```

```
Mode_Shapes(data);

% --- Executes on button press in Clear_Data.
function Clear_Data_Callback(hObject, eventdata, handles)
% hObject    handle to Clear_Data (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

close all
EEP1
```

A.2 FRF GUI MATLAB Code

This section includes the GUIDE layout of the FRF damping analysis window and accompanying MATLAB code created for the user interface and plotting all FRF data graphics. `FRF_DA.m` is the code generated after creating the user interface and as was needed. The code for `peak_pick.m` was used to find all peaks in the FRF plots. The code for `ls_circle_fit.m` and `circle_info.m` created the least squares circle fit data to be plotted.

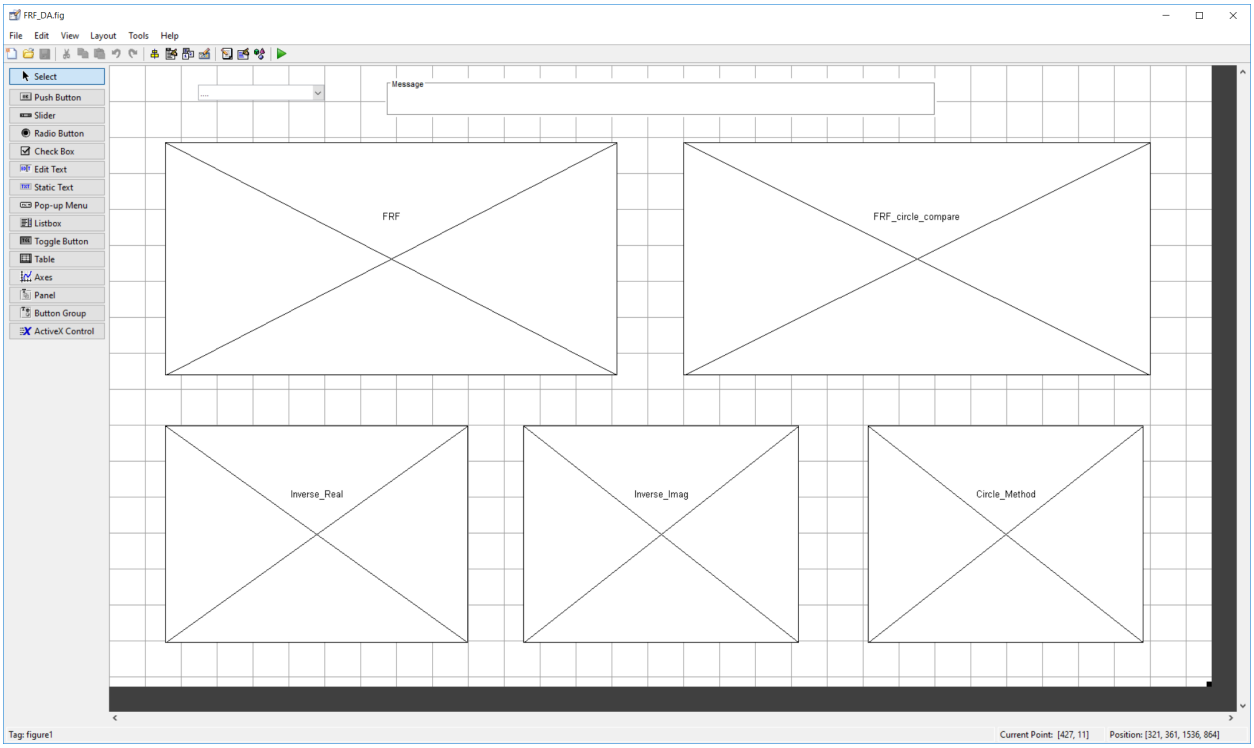


Figure A.2: GUIDE layout design of FRF_DA.

```
function varargout = FRF_DA(varargin)
% FRF_DA MATLAB code for FRF_DA.fig
%      FRF_DA, by itself, creates a new FRF_DA or raises the existing
%      singleton*.
%
%      H = FRF_DA returns the handle to a new FRF_DA or the handle to
```

```
%      the existing singleton*.
%
%      FRF_DA('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in FRF_DA.M with the given input arguments.
%
%      FRF_DA('Property','Value',...) creates a new FRF_DA or raises the
%      existing singleton*.  Starting from the left, property value pairs
%      are applied to the GUI before FRF_DA_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
%      application stop.  All inputs are passed to FRF_DA_OpeningFcn via
%      varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help FRF_DA

% Last Modified by GUIDE v2.5 27-Sep-2016 10:52:26

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @FRF_DA_OpeningFcn, ...
                   'gui_OutputFcn',  @FRF_DA_OutputFcn, ...
                   'gui_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
% End initialization code - DO NOT EDIT

% --- Executes just before FRF_DA is made visible.
function FRF_DA_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to FRF_DA (see VARARGIN)

% Choose default command line output for FRF_DA
handles.output = hObject;

data = evalin('base','data'); % Load data from EEP1.m

% Create the list for popupmenu
```

```
% Find frequency higher than lower bound
freq_low  = (data.freq > data.x1(1));
% Find frequency lower than higher bound
freq_high = (data.freq < data.x2(1));
% All data between lower and higher should be a 1 (T) all other 0 (F)
freq_lim = freq_low.*freq_high;
% multiply by actual frequency values within bounds
new_freq = freq_lim.*data.freq;

    limits = new_freq;
    limits(limits == 0) = [];   % remove all data equal to zero

    [a,b] = size(data.FRF);

    FRF_lim = zeros([a b]);
    FRF_useful = zeros([length(limits) b]);

    % Make all data outside of the FRF range selected equal to zero
    for ii = 1:b

        FRF_lim(:,ii) = data.FRF(:,ii).*freq_lim;
        FRF_use = FRF_lim(:,ii);

        FRF_use(FRF_use == 0) = [];
        FRF_useful(:,ii) = FRF_use;

    end

    % Generate FRF strings for each FRF data set
    FRF_str{1} = '....';
    peak_count(1) = 0;
    for ii = 2:b+1

        [peaks,index] = peak_pick(abs(FRF_useful(:,ii-1)'));
        peak_count(ii) = length(peaks);

        if length(peaks) == 1

            num = num2str(ii-1);
            str = ['FRF_',num];
            FRF_str{ii} = str;

        % Quarantine data sets with more than one peak found
        elseif length(peaks) > 1

            num = num2str(ii-1);
            str = ['FRF_',num,'_Q'];
            FRF_str{ii} = str;

        end
    end

    set(handles.FRF_menu,'Enable','on','String',FRF_str)
```

```
115
    set(handles.Message_Box,'String', ...
        'Select the FRF (tap point) of interest.');

    data.limits = limits;
120 data.b = b;
    data.FRF_useful = FRF_useful;
    handles.data = data;

    assignin('base', 'limits', limits)
125 assignin('base', 'FRF_data', FRF_useful)

% Update handles structure
guidata(hObject, handles);

130 % UIWAIT makes FRF_DA wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
135 function varargout = FRF_DA_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
140
% Get default command line output from handles structure
varargout{1} = handles.output;


145 % --- Executes on selection change in FRF_menu.
function FRF_menu_Callback(hObject, eventdata, handles)
% hObject    handle to FRF_menu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
150
% Hints: contents = cellstr(get(hObject,'String')) returns FRF_menu
%           contents as cell array
%        contents{get(hObject,'Value')} returns selected item from FRF_menu

155 val = get(handles.FRF_menu,'Value');

cla(handles.FRF,'reset')
cla(handles.Inverse_Real,'reset')
cla(handles.Inverse_Imag,'reset')
160 cla(handles.Circle_Method,'reset')
cla(handles.FRF_circle_compare,'reset')

% data = handles.data;

165 if val == 1

else
```

```
    val = val - 1;

    % FRF data range selected
    FRF_useful = handles.data.FRF_useful(:,val);
    % user selected upper and lower frequency
    limits = handles.data.limits;

    % Find all peaks in FRF data
    [peaks,index] = peak_pick(abs(FRF_useful));

    % FRF magnitude data
    mag = abs(FRF_useful);
    % natural frequency peak magnitude and frequency it occurs at
    [maxmag,mi] = max(mag);
    % half-band power magnitude
    hbp = maxmag/sqrt(2);
    % lower half-band power frequency
    lf = interp1(abs(FRF_useful(1:mi)),limits(1:mi),hbp);
    % upper half-band power frequency
    uf = interp1(abs(FRF_useful(mi:end)),limits(mi:end),hbp);
    % modal damping ratio
    dr = (uf - lf)/(2*limits(mi));

    % Plot original FRF with half-band power points
    axes(handles.FRF)
    semilogy(limits, abs(FRF_useful), '-b', 'linewidth', 1)
    hold on
    semilogy(limits(index),peaks,'or');
    semilogy([lf uf],[hbp hbp],'r+');
    set(gcf, 'color', 'w')
    xlabel('Frequency (Hz)')
    ylabel('Magnitude (gn/lbf)')
    title(sprintf('FRF Magnitude  |  \\omega_n = %4.1f  |  \\zeta = %5.4f',
        limits(mi),dr))
    ylim([.9*min(abs(FRF_useful)) 1.2*maxmag])

    % Inverse Method / CoQuad plot
    [mxr,mxri] = max(real(FRF_useful));
    [mnr,mnri] = min(real(FRF_useful));
    [mxi,mxii] = max(abs(imag(FRF_useful)));
    peak_dir = sign(imag(FRF_useful(mxii)));
    imag_peak = mxi*peak_dir;
    drc = abs(limits(mxri) - limits(mnri))/(2*mxi);

    % Plot Co-Quad Real
    axes(handles.Inverse_Real)
    plot(limits,real(FRF_useful),'-b', 'linewidth', 1)
    hold on
    plot([limits(1),limits(end)],[0,0],'--k')
    plot([limits(mxri),limits(mnri)],[mxr,mnr],'or')
    set(gcf, 'color', 'w')
    xlabel('Frequency (Hz)')
    xlim([limits(1)-5,limits(end)+5])
    ylabel('Real (gn/lbf)')
```

```
    ylim([mnr*1.1,mxr*1.1])
    title(sprintf('CoQuad Plot - Real Part  |  \\zeta = %5.4f',drc))

    % Plot Co-Quad Imag
    axes(handles.Inverse_Imag)
    plot(limits,imag(FRF_useful),'-b', 'linewidth', 1)
    set(gcf, 'color', 'w')
    xlabel('Frequency (Hz)')
    xlim([limits(1)-5,limits(end)+5])
    ylabel('Imaginary (gn/lbf)')
    ylim([min(imag(FRF_useful))*1.1-10,max(imag(FRF_useful))*1.1+10])
    title(sprintf('CoQuad Plot - Imaginary Part  |  \\omega_n = %4.1f',...
        limits(mxii)))
    hold on
    plot(limits(mxii),imag_peak,'or')

    % Circle fit visualization
    [h,k,r] = ls_circle_fit(FRF_useful);
    [FRF_gen,theta] = circle_info(FRF_useful,h,k,r);

    axes(handles.Circle_Method)
    plot(real(FRF_useful),imag(FRF_useful),'*','Linewidth',1);
    hold on ;
    t = 0:0.01:2*pi ;
    plot(r*cos(t)+h,r*sin(t)+k,'color','r','Linewidth',1);
    hold on ;
    plot([h,r*cos(theta(mi)*pi/180)+h],[k,r*sin(theta(mi)*pi/180)+k],...
        'color','g','Linewidth',1);
    axis equal
    xlabel('FRF Real Part (gn/lbf)')
    ylabel('FRF Imag Part (gn/lbf)')
    ylim([min(imag(FRF_useful))*1.1-10,max(imag(FRF_useful))*1.1+10])
    title('FRF Nyquist Circle w/ Least Squares Circle Fit')

    % Circle fit generated FRF comparison
    [peaks,index] = peak_pick(abs(FRF_gen));
    mag = abs(FRF_gen);
    [maxmag,mi] = max(mag);
    hbp = maxmag/sqrt(2);
    lf = interp1(abs(FRF_gen(1:mi)),limits(1:mi),hbp);
    uf = interp1(abs(FRF_gen(mi:end)),limits(mi:end),hbp);
    dr = (uf - lf)/(2*limits(mi));

    axes(handles.FRF_circle_compare)
    semilogy(limits, abs(FRF_useful), '-b', 'linewidth', 1)
    hold on
    semilogy(limits, abs(FRF_gen), '-r', 'linewidth', 1)
    semilogy(limits(index),peaks,'or');
    semilogy([lf uf],[hbp hbp],'r+');
    set(gcf, 'color', 'w')
    xlabel('Frequency (Hz)')
    ylabel('Magnitude (gn/lbf)')
    title(sprintf('Circle Fit FRF Magnitude  |  \\omega_n = %4.1f  |  \\zeta =
        %5.4f',limits(mi),dr))
```

```
    ylim([.9*min(abs(FRF_useful)) 1.2*maxmag])
    legend('FRF_{meas}','FRF_{gen}','Location','Northeast')

end

% --- Executes during object creation, after setting all properties.
function FRF_menu_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FRF_menu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function [peaks,index] = peak_pick(data)

%--------------------------------------------------------------------------
%
%  DESCRIPTION:
%  Once the frequency limits are created find all peaks of the CMIF or FRF
%  within those limits.
%
%  INPUTS:      ** CMIF or FRF data of interest
%
%  OUTPUTS:     ** Every peak within the bounds given based on looking at
%               a point to point change in height before and after data
%               data point.
%
%  NOTE:    This is very adaptable to any set of data assumed to be
%           continuous.
%
% Ethan Cating
% 11/12/2016
%--------------------------------------------------------------------------

k = 1;

for ii = 1:length(data)-2

    f1 = data(ii);                  % First data point

    f2 = data(ii+1);                % Middle data point

    f3 = data(ii+2);                % Last data point

    if ((f2 > f1) && (f3 < f2))

        peaks(k) = f2;              % Middle point higher than
```

```
                                    % first & last (peak)

        index(k) = ii+1;            % Where peak is in vector

        k = k + 1;                  % Index peak vector

    end

end
```

```
function [h,k,r] = ls_circle_fit(FRF_useful)

%-------------------------------------------------------------------------
%
% DESCRIPTION:
% This function takes calculates a least squares best fit circle fit to
% the input FRF data base on its real and imaginary parts.
%
% Based on the equation (x - h)^2 + (y - k)^2 = r^2.
%
% INPUTS:
%
%        ** FRF data of interest
%
% OUTPUTS:
%
%        h - circle offset on x-axis
%        k - circle offset on y-axis
%        r - radius of best fit circle
%
% Ethan Cating
% 11/12/2016
%-------------------------------------------------------------------------

    x = real(FRF_useful);   % x-data
    y = imag(FRF_useful);   % y-data

    n = length(x);          % number of data points

    % Least Squares Information
    xy = x.*y;

    x2 = x.^2;
    y2 = y.^2;

    Sx  = sum(x);
    Sy  = sum(y);
    Sxy = sum(xy);
    Sxx = sum(x2);
    Syy = sum(y2);

    Sxxx = sum(x2.*x);
```

```
    Syyy = sum(y2.*y);
    Sxxy = sum(x2.*y);
    Sxyy = sum(x.*y2);

    M = [Sxx Sxy Sx;      % Mx = B
         Sxy Syy Sy;
          Sx  Sy  n];

    B = [-(Sxxx + Sxyy);
         -(Sxxy + Syyy);
         -(Sxx  + Syy)];

    coeff = M\B;      % Solve for A, B, & C coefficients in derivation

    h = -coeff(1)/2;
    k = -coeff(2)/2;
    r = sqrt(coeff(1)^2/4 + coeff(2)^2/4 - coeff(3));
```

```
function [FRF_gen,theta] = circle_info(FRF_useful,h,k,r)

%-------------------------------------------------------------------------
%
% DESCRIPTION:
%        This function takes the least squares circle fit outputs and
%        generates an FRF magnitude plot of the circle fit data.
%
% INPUTS:
%
%        FRF_useful - FRF data selected
%        h - x offset
%        k - y offset
%        r - circle radius
%
% OUTPUTS:
%
%        FRF_gen - FRF generated from circle fit coefficients
%        theta - Angle of the circle where the peak of the original FRF is
%
%
% Ethan Cating
% 11/12/2016
%
%-------------------------------------------------------------------------

theta = atan2d(imag(FRF_useful)-k,real(FRF_useful)-h);

FRF_gen_x = r*cos(theta*pi/180) + h;
FRF_gen_y = r*sin(theta*pi/180) + k;

FRF_gen = FRF_gen_x + FRF_gen_y*1i;
```

A.3 Mode Shape Figure MATLAB Code

This section includes all MATLAB code files for generating the spline mode shape plot. `Mode_Shapes.m` creates the MATLAB plot window with both the normal and spline plotted mode shapes. The code `spline_points.m` organizes the mode shape data points into the sets of lines that `create_spline.m` can then generate the cubic polynomial splines for each line.

```
function [] = Mode_Shapes(data)

%--------------------------------------------------------------------------
%
% DESCRIPTION:
%        This function takes the slopes found from create_spline to create
%        the vector of x and y points for the spline.
%
% INPUTS:
%
%        data - natural frequency peak, EMAP mode shape data
%
% OUTPUTS:
%
%        mode shape plot of model with splines instead of standard point to
%        point lines
%
%
% Ethan Cating
% 11/12/2016
%
%--------------------------------------------------------------------------

    % EMAP and EEP data
    nodeNum = data.nodeNum;
    X = data.X;
    Y = data.Y;
    Z = data.Z;
    NF = data.NF;
    NatFreq = data.NatFreq;
    Zeta = data.Zeta;
    traceNum = data.traceNum;
    startNode = data.startNode;
    endNode = data.endNode;
    xAxisLim = data.xAxisLim;
    yAxisLim = data.yAxisLim;
    zAxisLim = data.zAxisLim;
```

```
    x0 = data.x0;
    y0 = data.y0;
    z0 = data.z0;

    %  Plot the nodes: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    figure('Name','Mode Shape Comparison')
    set(gcf,'Color','w')
    subplot(2,1,1)

    plot3(0, 0, 0)
    hold on
    xlabel('\itx')
    ylabel('\ity')
    zlabel('\itz    ', 'rotation', 0)

    for j = 1:length(nodeNum)

        nodes(j,1) = line('xdata',  X(j,NF), 'ydata',  Y(j,NF), ...
                          'zdata', Z(j,NF), 'marker', 'o', ...
                          'markerfacecolor', 'b');
        nodeNums(j,1) = text(X(j,NF), Y(j,NF), Z(j,NF)+0.3, ....
                             int2str(nodeNum(j)));
    end

    title({['Mode ', int2str(NF), ':   ', '\itf_{\rm', ...
          int2str(NF), '}\rm = ', ...
          num2str(NatFreq(NF), '%6.1f'), ' Hz,    ', ...
          '\it\zeta_{\rm', int2str(NF), '}\rm = ', ...
          num2str(Zeta(NF), '%6.4f')]; []})
    xlabel('\itx')
    ylabel('\ity')
    zlabel('\itz    ', 'rotation', 0)
    grid on
    hold on

    %  Plot the tracelines: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    for k = 1:traceNum

       plot3([X(startNode(k),NF), ...
              X(endNode(k),NF)], ...
             [Y(startNode(k),NF), ...
              Y(endNode(k),NF)], ...
             [Z(startNode(k),NF), ...
              Z(endNode(k),NF)], '-r', 'linewidth', 2);

    end
    clear j
    clear k
    hold off
    axis equal
    xlim(xAxisLim)
    ylim(yAxisLim)
```

```
    zlim(zAxisLim)

    %% Spline Plots
    subplot(2,1,2)
    nn = length(nodeNum);

    % Find x plane lines %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    xu = unique(x0);
    xd(:,1) = nodeNum;
    xd(:,2) = X(:,NF) - x0;
    xd(any(xd==0,2),:)=[];

    if isempty(xd)
        xpoints = length(xu);
        x_l_tot = nn/length(xu);
        if isempty(xu)
            xlines = 0;
        else
        for ii = 1:xpoints

           xlines(ii,:) = find(x0 == xu(ii));

        end
        xs = x0(xlines(:,:));
        end
    else
        xpoints = length(xd(:,1));
        x_l_tot = xpoints/length(xu);
        xlines = 0;
    end

    % Find y plane lines %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    yu = unique(y0);
    yd(:,1) = nodeNum;
    yd(:,2) = Y(:,NF) - y0;
    yd(any(yd==0,2),:)=[];

    if isempty(yd)
        ypoints = length(yu);
        y_l_tot = nn/length(yu);
        if isempty(yu)
            ylines = 0;
        else
        for ii = 1:ypoints

           ylines(ii,:) = find(y0 == yu(ii));

        end
        ys = y0(ylines(:,:));
        end
    else
        ypoints = length(yd(:,1));
        y_l_tot = ypoints/length(yu);
        ylines = 0;
```

```
    end

    % Find Z plane lines %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    zu = unique(z0);
150 zd(:,1) = nodeNum;
    zd(:,2) = Z(:,NF) - z0;
    zd(any(zd==0,2),:)=[];

    if isempty(zd)
155     zpoints = length(zu);
        z_l_tot = nn/length(zu);
        if isempty(zu)
            zlines = 0;
        else
160     for ii = 1:zpoints

           zlines(ii,:) = find(z0 == zu(ii));

        end
165     zs = z0(zlines(:,:));
        end
    else
        zpoints = length(zd(:,1));
        z_l_tot = zpoints/length(zu);
170     zlines = 0;
    end

    if size(ylines,1) == 1 && size(ylines,2) == 1 ...
            && size(zlines,1) == 1 && size(zlines,2) == 1
175
        pt = 1;

        ylx = ismember(xlines,yd(:,1));
        ylx = ylx.*xlines;
180     ylx(:,any(ylx==0,1)) = [];

        zlx = ismember(xlines,zd(:,1));
        zlx = zlx.*xlines;
        zlx(:,any(zlx==0,1)) = [];
185
    else

        pt = 2;

190 end

    % Plot splines %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %  Plot the nodes:
195
        plot3(0, 0, 0)
        hold on
        xlabel('\itx')
        ylabel('\ity')
```

```
        zlabel('\itz    ', 'rotation', 0)

        for j = 1:length(nodeNum)

            nodes(j,1) = line('xdata',  X(j,NF), 'ydata',  Y(j,NF), ...
                              'zdata', Z(j,NF), 'marker', 'o', ...
                              'markerfacecolor', 'b');
            nodeNums(j,1) = text(X(j,NF), Y(j,NF), Z(j,NF)+0.3, ....
                                 int2str(nodeNum(j)));
        end

        title({['Mode ', int2str(NF), ':   ', '\itf_{\rm', ...
              int2str(NF), '}\rm = ', ...
              num2str(NatFreq(NF), '%6.1f'), ' Hz,    ', ...
              '\it\zeta_{\rm', int2str(NF), '}\rm = ', ...
              num2str(Zeta(NF), '%6.4f')]; []})
        xlabel('\itx')
        ylabel('\ity')
        zlabel('\itz    ', 'rotation', 0)
        grid on
        hold on
        axis equal

    % if line or plane of points

    if pt == 2

    % plot x lines
        if size(ylines,1) == 1 % one point only, no line to plot

        elseif size(ylines,1) == 2 % two points, plot straight line

        else % three or more points, plot spline

            for ii = 1:size(xlines,1)

                sx = Y(xlines(ii,:),NF)';
                sy = Z(xlines(ii,:),NF)';
                [A,B,C,D,H,F,h] = create_spline(sx,sy,[0,0],'knot');
                [xs,zs] = spline_points(sx,A,B,C,D);
                ys = linspace(X(xlines(ii,1),NF),X(xlines(ii,end),NF),...
                    length(zs));

                plot3(ys,xs,zs, 'color', 'r', 'linewidth', 2);

            end

        end

        % plot y lines
        if size(xlines,1) == 1 % one point only, no line to plot

        elseif size(xlines,1) == 2 % two points, plot straight line
```

```
        else % three or more points, plot spline

            for ii = 1:size(xlines,2)

                sx = X(xlines(:,ii),NF)';
                sy = Z(xlines(:,ii),NF)';
                [A,B,C,D,H,F,h] = create_spline(sx,sy,[0,0],'knot');
                [xs,zs] = spline_points(sx,A,B,C,D);
                ys = linspace(Y(xlines(1,ii),NF),Y(xlines(end,ii),NF),...
                    length(zs));

                plot3(xs,ys,zs, 'color', 'r', 'linewidth', 2);

            end

        end

    elseif pt == 1

        % plot xy lines
        if size(ylx,1) == 1

        elseif size(ylx,1) == 2

            for ii = 1:size(ylx,2)

                xs = X(ylx(:,ii),NF);
                ys = Y(ylx(:,ii),NF);
                zs = Z(ylx(:,ii),NF);

                plot3(xs,ys,zs, 'color', 'r', 'linewidth', 2);

            end

        else

            for ii = 1:size(ylx,2)

                sx = X(ylx(:,ii),NF);
                sy = Y(ylx(:,ii),NF);
                [A,B,C,D,H,F,h] = create_spline(sx,sy,[0,0],'knot');
                [xs,zs] = spline_points(sx,A,B,C,D);
                ys = linspace(Z(ylx(1,ii),NF),Z(ylx(end,ii),NF),...
                    size(zs,2));

                plot3(xs,ys,zs, 'color', 'r', 'linewidth', 2);

            end

        end

        % plot xz lines

        if size(zlx,1) == 1
```

```
        elseif size(zlx,1) == 2

            for ii = 1:size(zlx,2)

                xs = X(zlx(:,ii),NF);
                ys = Y(zlx(:,ii),NF);
                zs = Z(zlx(:,ii),NF);

                plot3(xs,ys,zs, 'color', 'r', 'linewidth', 2);

            end

        else

            for ii = 1:size(zlx,2)

                sx = X(zlx(:,ii),NF);
                sy = Z(zlx(:,ii),NF);
                [A,B,C,D,H,F,h] = create_spline(sx,sy,[0,0],'knot');
                [xs,zs] = spline_points(sx,A,B,C,D);
                ys = linspace(Y(zlx(1,ii),NF),Y(zlx(end,ii),NF),...
                    size(zs,2));

                plot3(xs,ys,zs, 'color', 'r', 'linewidth', 2);

            end

        end

        % plot yx lines
        for ii = 1:size(ylx,1)

            sx = Z(ylx(ii,:),NF);
            sy = Y(ylx(ii,:),NF);
            [A,B,C,D,H,F,h] = create_spline(sx,sy,[0,0],'knot');
            [xs,zs] = spline_points(sx,A,B,C,D);
            ys = linspace(X(ylx(ii,1),NF),X(ylx(ii,end),NF),size(zs,2));

            plot3(ys,zs,xs, 'color', 'r', 'linewidth', 2);

        end

        % plot zx lines
        for ii = 1:size(zlx,1)

            sx = Y(zlx(ii,:),NF);
            sy = Z(zlx(ii,:),NF);
            [A,B,C,D,H,F,h] = create_spline(sx,sy,[0,0],'knot');
            [xs,zs] = spline_points(sx,A,B,C,D);
            ys = linspace(X(xlines(ii,1),NF),X(xlines(ii,end),NF),...
                size(zs,2));

            plot3(ys,xs,zs, 'color', 'r', 'linewidth', 2);
```

```
        end
    end
    grid on
    axis equal
    xlim(xAxisLim)
    ylim(yAxisLim)
    zlim(zAxisLim)
```

```
function [xs,ys] = spline_points(x,A,B,C,D)

%-------------------------------------------------------------------------
%
% DESCRIPTION:
%       This function takes the slopes found from create_spline to create
%       the vector of x and y points for the spline.
%
% INPUTS:
%
%       x - linear range that corresponds to desired spline f(x) points
%       A,B,C,D - third order variables for calculating spline
%
% OUTPUTS:
%
%       xs - base data from original linear range
%       ys - f(xs) spline data
%
%
% Ethan Cating
% 11/12/2016
%
%-------------------------------------------------------------------------

xvec = zeros(length(x)-1,100);            % generates x array of zeros
line = zeros(length(x)-1,100);            % generates y array of zeros

for ii = 1:length(x)-1

    xvec(ii,:) = linspace(x(ii),x(ii+1),100);   % Fill in x array with
                                                % structure point position
                                                % data

    line(ii,:) = A(ii) + B(ii).*(xvec(ii,:)-x(ii)) +...
                 C(ii).*((xvec(ii,:)-x(ii)).^2) +...
                 D(ii).*((xvec(ii,:)-x(ii)).^3);

end

xs = xvec;
ys = line;
```

```
function [A,B,C,D,H,F,h] = create_spline(x,y,fp,endcond)

%-------------------------------------------------------------------------
%
% DESCRIPTION:
% This function finds the slopes for each part of the cubic spline function
% fitted to the points given. Where A, B, C, and D are found from the S
% vector in H*S = F;
%
% INPUTS:
%
%        x - x-axis data array
%        y - y-axis data array
%        fp - slope at start and end of spline
%        endcond - clamped, natural, or knot possible spline end conditions
%
% OUTPUTS:
%
%        A - y-intercept
%        B - first order polynomial coefficient
%        C - second order polynomial coefficient
%        D - third order polynomial coefficient
%        H - tridiagonal matrix
%        F - function intercept vector
%        h - vector of point to point deltas
%
%
% Ethan Cating
% 11/12/2016
%
%-------------------------------------------------------------------------

% Find the delta between each x point
for aa = 1:length(x)-1

    h(aa) = x(aa+1) - x(aa);

end

% Allocate memory for the H matrix and F vector
H = zeros(length(x));

F = zeros([length(x),1]);

% Figure out the end conditions
if strcmp(endcond,'clamped') == 1               %

    H(1,:)   = [1, zeros(1,length(x)-1)];
    H(end,:) = [zeros(1,length(x)-1), 1];

    F(1)   = fp(1);
    F(end) = fp(2);
```

```
elseif strcmp(endcond,'natural') == 1

    H(1,:)   = [2, 1, zeros(1,length(x)-2)];
    H(end,:) = [zeros(1,length(x)-2), 1, 2];

    F(1)   = 3*((y(2) - y(1))/h(1));
    F(end) = 3*((y(end) - y(end-1))/h(end));

elseif strcmp(endcond,'knot') == 1

    H(1,:)   = [h(2), h(2) + h(1), zeros(1,length(x)-2)];
    H(end,:) = [zeros(1,length(x)-2), h(end) + h(end-1), h(end-1)];

    F(1)   = (((y(3) - y(2))/h(2))*(h(1)^2) + ((y(2) -...
               y(1))/h(1))*h(2)*(2*(x(3)-x(1)) + h(1)))/(x(3)-x(1));
    F(end) = (((y(end-1) - y(end-2))/h(end-1))*(h(end)^2) +...
              ((y(end) - y(end-1))/h(end))*h(end-1)*...
              (2*(x(end)-x(end-2)) + h(end)))/(x(end)-x(end-2));

else

    error('Please select and endpoint condition for the spline fit.')

end

% Fill the H Matrix (tridiagonal)
for ii = 2:length(x)-1

    H(ii,ii-1) = h(ii);
    H(ii,ii)   = 2*(h(ii) + h(ii-1));
    H(ii,ii+1) = h(ii-1);

end

% Find the F vector
for jj = 2:length(x)-1

    F(jj) = 3*(((y(jj) - y(jj-1))/(x(jj)-x(jj-1)))*h(jj) +...
            h(jj-1)*((y(jj+1) - y(jj))/(x(jj+1)-x(jj))));

end

% Solve for S
S = H\F;

% Solve for A, B, C, and D
A = y(1:end-1);

B = S(1:end-1);

for kk = 1:length(x)-1

    D(kk) = (S(kk+1) + S(kk) -...
            2*((y(kk+1) - y(kk))/(x(kk+1)-x(kk))))/((x(kk+1)-x(kk))^2);
```

```
    C(kk) = (((y(kk+1) - y(kk))/(x(kk+1)-x(kk))) - S(kk))...
110             /(x(kk+1)-x(kk)) - D(kk)*(x(kk+1)-x(kk));

end
```

A.4 Circle Fit Method Calculation

The equation for a circle is given again where x is the real part of FRF data, y is the imaginary part of FRF data, h is the x-axis offset of the circle's center, k is the y-axis offset of the circle's center, and r is the radius of the fitted circle:

$$(x-h)^2+(y-k)^2=r^2$$

The next step brings all variables to the left side of the equation:

$$(x-h)^2+(y-k)^2-r^2=0$$

The polynomial is completely expanded to see all terms:

$$x^2+y^2-2hx-2ky-r^2+h^2+k^2=0$$

New constants A, B, and C are equated to the circle constants, h, k, and r:

$$A=-2h$$

$$B=-2k$$

$$C=r^2-h^2-k^2$$

The constants are substituted back into the expanded polynomial equation where A, B, and C will be easily solvable through linear algebra:

$$x^2+y^2+Ax+By+C=0$$

The squared terms are moved to the right side so that the equation can be made into the matrix form of $Mx = b$:

$$Ax + By + C = -(x^2 + y^2)$$

For n number of data points, all are input into the linear equations from $i = 1$ to n:

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix} \begin{pmatrix} A \\ B \\ C \end{pmatrix} = - \begin{pmatrix} {x_1}^2 + {y_1}^2 \\ {x_2}^2 + {y_2}^2 \\ \vdots \\ {x_n}^2 + {y_n}^2 \end{pmatrix}$$

The least squares solution is defined as the solution to the linear system [4, pp. 508-509]:

$$M^T M x = M^T b$$

The solution to the system minimizes the residual or the sum of squares of the differences between the recorded FRF data and the generated circle-fit data:

$$r = ||b - Mx||$$

Applying the matrix transpose to each side gives

$$\begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \\ 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix} \begin{pmatrix} A \\ B \\ C \end{pmatrix} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \\ 1 & 1 & \dots & 1 \end{bmatrix} (- \begin{pmatrix} x_1^2 + y_1^2 \\ x_2^2 + y_2^2 \\ \vdots \\ x_n^2 + y_n^2 \end{pmatrix})$$

Matrix multiplying through all data points creates summations for the different multiplication combinations of x and y data points:

$$\begin{bmatrix} \sum_{i=1}^{n} {x_n}^2 & \sum_{i=1}^{n} x_n y_n & \sum_{i=1}^{n} x_n \\ \sum_{i=1}^{n} x_n y_n & \sum_{i=1}^{n} {y_n}^2 & \sum_{i=1}^{n} y_n \\ \sum_{i=1}^{n} x_n & \sum_{i=1}^{n} y_n & \sum_{i=1}^{n} 1 \end{bmatrix} \begin{pmatrix} A \\ B \\ C \end{pmatrix} = - \begin{pmatrix} \sum_{i=1}^{n} ({x_n}^3 + x_n {y_n}^2) \\ \sum_{i=1}^{n} ({x_n}^2 y_n + {y_n}^3) \\ \sum_{i=1}^{n} ({x_n}^2 + {y_n}^2) \end{pmatrix}$$

Simplifying the summations into the variables the MATLAB code uses gives the final matrix system. The subscripts define the multiplication combinations or if it is just the sum of all x or y values:

$$\begin{bmatrix} S_{xx} & S_{xy} & S_x \\ S_{xy} & S_{yy} & S_y \\ S_x & S_y & 1 \end{bmatrix} \begin{pmatrix} A \\ B \\ C \end{pmatrix} = - \begin{pmatrix} (S_{xxx} + S_{xyy}) \\ (S_{xxy} + S_{yyy}) \\ (S_{xx} + S_{yy}) \end{pmatrix}$$

Then using MATLAB with the file `ls_circle_fit.m` the backslash command solves the system of equations to solve for A, B, and C. The constants h, k, and r can be solved for and the least squares circle-fit equation for the set of FRF data points can be generated.

```
coeff = M\B;      % Solve for A, B, & C coefficients in derivation
```

A.5 Dr. Elton Graves' Cubic Spline Class Notes

The section appends the document used to create the spline functions in MATLAB for the mode shapes viewer window in EEP. The document was a hand out in Dr. Graves Linear Algebra class at the Rose-Hulman Institute of Technology.

that derivative information will be given at data points of a contour map, we will improve upon the idea of using piecewise cubic polynomials to draw our contour maps.

Recall the four conditions we needed to construct a piecewise cubic polynomial were:

(2.8) (i) $P(x_i) = f(x_i)$ for $i = 1, 2, \ldots, n+1$,

(ii) $P_i(x_i) = f(x_i)$ and

$P_i(x_{i+1}) = f(x_{i+1})$ for $i = 1, 2, \ldots, n$,

(iii) $P_{i-1}(x_i) = P_i(x_i)$ for $i = 2, 3, \ldots, n$,

(iv) $P'_{i-1}(x_i) = P'_i(x_i)$ for $i = 2, 3, \ldots, n$.

Instead of imposing the condition $P'(x_i) = f'(x_i)$ for $i = 1, 2, \ldots, n+1$, as we did earlier, let's add the condition

(2.9) $P''_{i-1}(x_i) = P''_i(x_i)$ for $i = 2, 3, \ldots, n$.

Condition (2.9) requires $P(x)$ to be twice differentiable at each of the knots $x_1, x_2, \ldots, x_{n+1}$, adding smoothness to $P(x)$ and no longer requiring knowledge of $f'(x)$ at each knot.

As before we will require $P_i(x)$ to have the form

(2.10) $P_i(x) = A_i + B_i(x-x_i) + C_i(x-x_i)^2 + D_i(x-x_i)^3$.

We also let $s_i = P'_i(x_i)$ and $s_{i+1} = P'_i(x_{i+1}) = P'_{i+1}(x_{i+1})$ as before. By repeating the calculations used in the Hermite case we again arrive at (2.7) which we write as

(2.11) $A_i = f(x_i)$ for $i = 1, 2, \ldots, n$,

$B_i = s_i$ for $i = 1, 2, \ldots, n$,

$$D_i = \frac{s_{i+1} + s_i - 2f[x_i, x_{i+1}]}{(x_{i+1}-x_i)^2} \quad \text{for } i = 1, 2, \ldots, n,$$

$$C_i = \frac{f[x_i, x_{i+1}] - s_i}{x_{i+1}-x_i} - D_i(x_{i+1}-x_i) \quad \text{for } i = 1, 2, \ldots, n.$$

We do not yet know the values of s_i for $i = 1, 2, \ldots, n+1$. We can set up a system of equations to find the s_i's using the following procedure. If we differentiate $P_i(x)$ in (2.10) twice we find

(2.12) $P''_{i-1}(x) = 2C_{i-1} + 6D_{i-1}(x-x_{i-1})$ and

$P''_i(x) = 2C_i + 6D_i(x-x_i)$.

Now use (2.9), setting $P''_{i-1}(x_i) = P''_i(x_i)$, to get

$2C_{i-1} + 6D_{i-1}(x_i-x_{i-1}) = 2C_i + 6D_i(x_i-x_i)$, or

(2.13) $2C_{i-1} + 6D_{i-1}(x_i-x_{i-1}) = 2C_i$ for $i = 2, 3, \ldots, n$.

By setting $h_{i-1} = (x_i-x_{i-1})$ and $h_i = (x_{i+1}-x_i)$ (2.11) yields

$$(2.14) \quad C_{i-1} = \frac{f[x_{i-1}, x_i] - s_{i-1}}{h_{i-1}} - D_{i-1}h_{i-1}$$

and

$$(2.15) \quad C_i = \frac{f[x_i, x_{i+1}] - s_i}{h_i} - D_i h_i.$$

We now substitute (2.14) and (2.15) into (2.13) to get

$$2\left[\frac{f[x_{i-1}, x_i] - s_{i-1}}{h_{i-1}} - D_{i-1}h_{i-1}\right] + 6D_{i-1}h_{i-1} = 2\left[\frac{f[x_i, x_{i+1}]-s_i}{h_i} - D_i h_i\right]$$

or

$$(2.16)\quad 2\left[\frac{f[x_{i-1},x_i]-s_{i-1}}{h_{i-1}}\right]+4D_{i-1}h_{i-1}=2\left[\frac{f[x_i,x_{i+1}]-s_i}{h_i}\right]-D_ih_i.$$

Substituting $\dfrac{s_i+s_{i-1}-2f[x_{i-1},x_i]}{(h_{i-1})^2}$ for D_{i-1} and

$\dfrac{s_{i+1}+s_i-2f[x_i,x_{i+1}]}{(h_i)^2}$ for D_i in (2.16) we have

$$\frac{2f[x_{i-1},x_i]}{h_{i-1}}-\frac{2s_{i-1}}{h_{i-1}}+4h_{i-1}\left[\frac{s_i+s_{i-1}-2f[x_{i-1},x_i]}{(h_{i-1})^2}\right]$$

$$=\frac{2f[x_i,x_{i+1}]}{h_i}-\frac{2s_i}{h_i}-2h_i\left[\frac{s_{i+1}+s_i-2f[x_i,x_{i+1}]}{(h_i)^2}\right]$$

or

$$-\frac{2s_{i-1}}{h_{i-1}}+\frac{4s_i}{h_{i-1}}+\frac{4s_{i-1}}{h_{i-1}}-\frac{6f[x_{i-1},x_i]}{h_{i-1}}$$

$$=-\frac{2s_i}{h_i}-\frac{2s_i}{h_i}-\frac{2s_{i+1}}{h_i}+\frac{6f[x_i,x_{i+1}]}{h_i}.$$

Moving all s_{i-1}, s_i, and s_{i+1} terms to the left hand side of the equation and multiplying both sides of the equation by $\dfrac{(h_{i-1})(h_i)}{2}$ we get

$$(2.17)\quad h_is_{i-1}+2(h_{i-1}+h_i)s_i+h_{i-1}s_{i+1}=3(f[x_{i-1},x_i]h_i+f[x_i,x_{i+1}]h_{i-1})$$

for $i = 2, 3, \ldots, n$.

Notice that equation (2.17) is a system of n-1 equations in n+1 unknowns. Thus we need two more equations to uniquely solve for the s_i's.

There are many different ways to construct the extra two equations needed to solve the system in (2.17). We will mention three.

Case I.

The easiest method of adding two equations to (2.17) is to assume $f'(x_1)$ and $f'(x_{n+1})$ are known. For then

(2.18) $s_1 = f'(x_1)$ and $s_{n+1} = f'(x_{n+1})$,

and we arrive at the system

(2.19) $\overline{H}\overline{S} = \overline{F}$ where $\overline{H}$ is given by

$$\overline{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ h_2 & 2(h_2+h_1) & h_1 & 0 & & \vdots \\ 0 & h_3 & 2(h_3+h_2) & h_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & h_n & 2(h_n+h_{n-1}) & h_{n-1} \\ 0 & \cdots & 0 & 0 & 0 & 1 \end{bmatrix},$$

$\overline{S}$ is given by:

$$\overline{S} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \\ s_{n+1} \end{bmatrix},$$

and $\overline{F}$ is given by

$$\overline{F} = \begin{bmatrix} f'(x_1) \\ 3(f[x_1,x_2]h_2 + f[x_2,x_3]h_1) \\ 3(f[x_2,x_3]h_3 + f[x_3,x_4]h_2) \\ \vdots \\ 3(f[x_{n-1},x_n]h_n + f[x_n,x_{n+1}]h_{n-1}) \\ f'(x_{n+1}) \end{bmatrix}.$$

Notice that system (2.19) is a tridiagonal diagonally dominant system of n+1 equations in n+1 unknowns (see Appendix I). Being a tridiagonal system makes solving the system by Gaussian elimination easy for both humans and computers (see Appendix II). Diagonal dominance ensures there is a unique solution for the s_i's and also eliminates the need for pivoting when solving the system by Gaussian elimination.

Once the s_i's are determined, by solving system (2.19), we use the equations in (2.11) to solve for A_i, B_i, C_i and D_i for $i = 1, 2, \ldots, n$.

When $f'(x_1)$ and $f'(x_{n+1})$ are given P(x) is called a "clamped" or "complete" cubic spline [2], [6].

Case II.

Suppose $f''(x_1)$ and $f''(x_{n+1})$ are given. We now assume $P_1''(x_1) = f''(x_1)$ and $P_n''(x_{n+1}) = f''(x_{n+1})$. Since $P_1(x) = A_1 + B_1(x-x_1) + C_1(x-x_1)^2 + D_1(x-x_1)^3$, $P_1''(x) = 2C_1 + 6D_1(x-x_1)$. Thus $P_1''(x_1) = 2C_1 = f''(x_1)$. Substituting in the values for C_1 from (2.11) we have

$$2\left[\frac{f[x_1,x_2]-s_1}{h_1} - \frac{s_2 + s_1 - 2f[x_1,x_2]}{h_1}\right] = f''(x_1)$$

or

$$\frac{2f[x_2,x_1]}{h_1} - \frac{2s_1}{h_1} - \frac{2s_2}{h_1} - \frac{2s_1}{h_1} + \frac{4f[x_1,x_2]}{h_1} = f''(x_1).$$

Collecting s_1 and s_2 terms and multiplying by $-\frac{h_1}{2}$ we have

$$(2.20)\quad 2s_1 + s_2 = 3f[x_1,x_2] - \frac{f''(x_1)h_1}{2}.$$

Similarly, by setting $P_n''(x_{n+1}) = f''(x_{n+1})$ we have

$$(2.21)\quad s_n + 2s_{n+1} = 3f[x_n,x_{n+1}] + \frac{f''(x_{n+1})h_n}{2}\ .$$

By letter (2.20) and (2.21) be the first and last equations of system (2.17) we arrive at system

$$(2.22)\quad \overline{H}\,\overline{S} = \overline{F} \text{ where}$$

$$\overline{H} = \begin{bmatrix} 2 & 1 & 0 & 0 & 0 \\ h_2 & 2(h_2+h_1) & h_1 & 0 & \cdot \\ 0 & h_3 & 2(h_3+h_2) & h_2 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & 0 \\ 0 & \cdot & \cdot & h_n & 2(h_n+h_{n-1}) \quad h_{n-1} \\ 0 & \cdot & \cdot & 0 & 1 \quad 2 \end{bmatrix},$$

$$\overline{S} = \begin{bmatrix} s_1 \\ s_2 \\ \cdot \\ \cdot \\ \cdot \\ s_n \\ s_{n+1} \end{bmatrix} \quad \text{and}$$

$$\overline{F} = \begin{bmatrix} 3f[x_1,x_2] - \dfrac{f''(x_1)h_1}{2} \\ 3(f[x_1,x_2]h_2 + f[x_2,x_3]h_1) \\ \vdots \\ 3(f[x_{n-1},x_n]h_n + f[x_n,x_{n+1}]h_{n-1}) \\ 3f[x_n,x_{n+1}] + \dfrac{f''(x_{n+1})h_n}{2} \end{bmatrix} .$$

System (2.22) is again a tridiagonal diagonally dominate system.

In the special case when $f''(x_1) = f''(x_{n+1}) = 0$ we have what is usually called the "natural" cubic spline [2], [6].

Case III.

When no derivative information is known we can use what is called the "not-a-knot" cubic spline [2]. In this case we choose s_1 so that $P_1=P_2$ and choose s_n so that $P_{n-1}=P_n$. Since $P_1=P_2$ and $P_{n-1}=P_n$, this means that the points x_2 and x_n do not act as break points between two different polynomials as do the knots $x_3, x_4, \ldots, x_{n-1}$. In this case x_2 and x_n are said to be "not active," thus the term "not-a-knot."

To determine the first equation needed to complete the system (2.17) we notice that $P_1(x) = P_2(x)$ which means $P_1'''(x_2) = P_2'''(x_2)$.

36

$$\bar{H} = \begin{bmatrix} h_2 & (h_2+h_1) & 0 & 0 & \cdots & 0 \\ h_2 & 2(h_2+h_1) & h_1 & 0 & & \vdots \\ 0 & h_3 & 2(h_3+h_2) & h_2 & & \vdots \\ 0 & 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & h_n & 2(h_n+h_{n-1}) & h_{n-1} \\ 0 & \cdots & 0 & 0 & (h_n+h_{n-1}) & h_{n-1} \end{bmatrix},$$

$$\bar{S} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \\ s_{n+1} \end{bmatrix} \quad \text{and}$$

$$\bar{F} = \begin{bmatrix} \dfrac{f[x_2,x_3](h_1)^2 + f[x_1,x_2](h_2)[2(x_3-x_1) + h_1]}{x_3-x_1} \\ 3(f[x_1,x_2]h_2 + f[x_2,x_3](h_1)) \\ \vdots \\ 3(f[x_{n-1},x_n]h_n + f[x_n,x_{n+1}]h_{n-1}) \\ \dfrac{f[x_{n-1},x_n](h_n)^2 + f[x_n,x_{n+1}](h_{n-1})[2(x_{n+1}-x_{n-1}) + h_n]}{x_{n+1}-x_{n-1}} \end{bmatrix}.$$

We notice that the system in (2.30) is tridiagonal but is not diagonally dominant, (since (x_3-x_1) can be larger than $h_2 = x_2-x_1$, and $(x_{n+1}-x_{n-1})$ can be larger than $h_{n-1} = x_n-x_{n-1}$.) However the lack of diagonal dominance does not affect the use of Gaussian elimination.

Example 3: Write down the formulas for the "clamped," the "natural," and the "not-a-knot" cubic splines which interpolate the function $f(x)=1/(1+100x^2)$ at the data points given in Example 1.

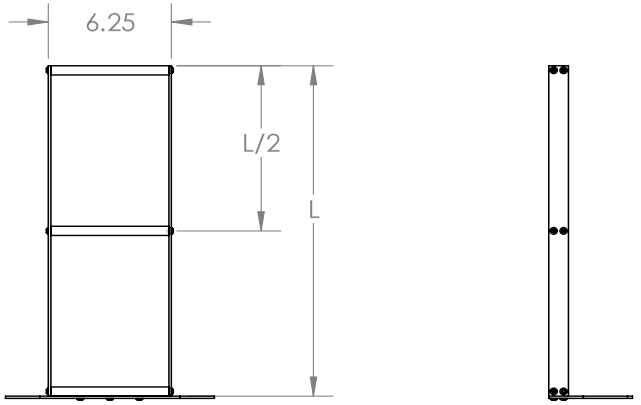
The clamped cubic spline, which interpolates $f(x)=1/(1+100x^2)$ at the nine given data points, can be written as:

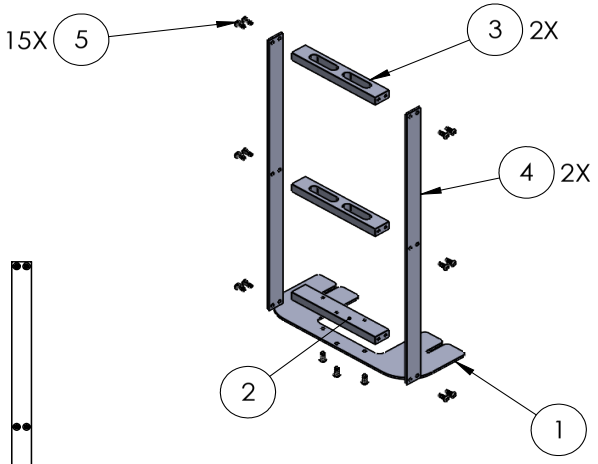
B Shaker Lab Appendix

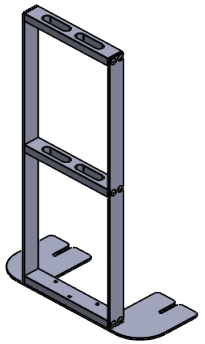
B.1 Shaker Lab SolidWorks Prints

Included in this appendix section are all of the drawing prints to create the two-story scale building. The assembly consists of two floor stiffeners, one base stiffener to connect the building legs to the base plate. All parts are bolted together with #10-32 button head cap screws, 1/2" in length. The legs and stiffener bars were machined on a manual mill in the Mechanical Engineering Machine Shop. The base plate was waterjetted at Rose-Hulman Ventures.

ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	SL-003	Base Plate	1
2	SL-001	Bottom Stiffener	1
3	SL-002	Floor Stiffener	2
4	SL-004	Side Plates	2
5	98164A178	McMaster-Carr Type 316 SS BHCS #10-32 X 1/2"	15







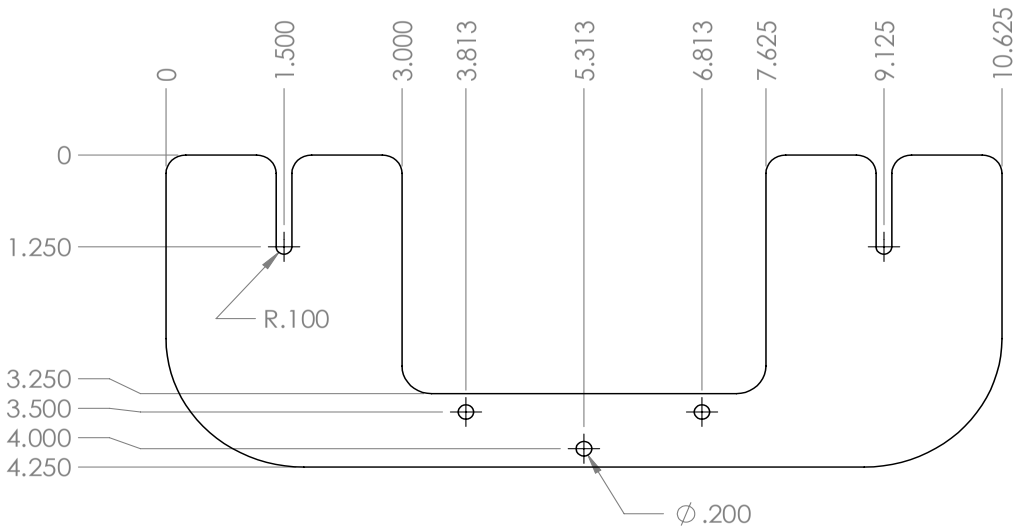
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005	DRAWN	EC	3/12/16
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

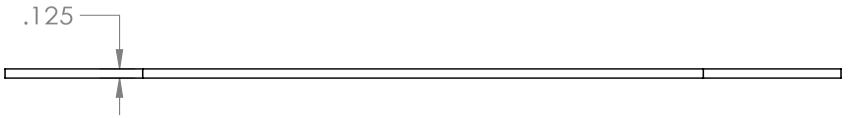
TITLE: Two-Story Scale Building

SIZE	DWG. NO.	REV
A	Shaker_lab	A

SCALE: 1:8 | WEIGHT: | SHEET 1 OF 1

5 4 3 2 1



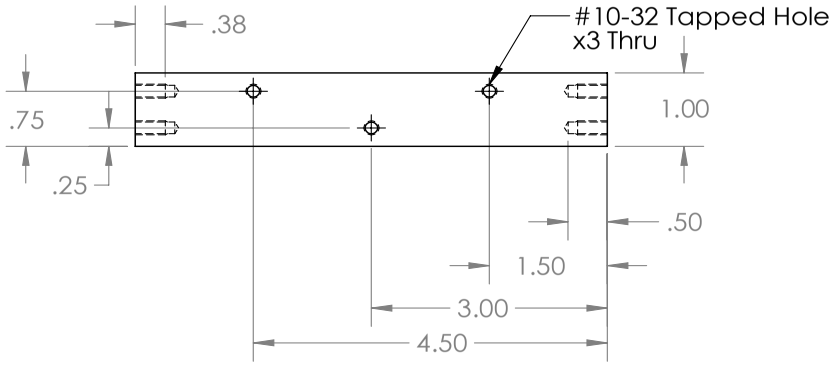


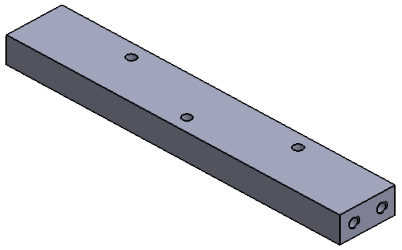
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005	DRAWN	EC	3/27/15
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL 6061 Aluminum	COMMENTS: - Waterjetted - Break all edges		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

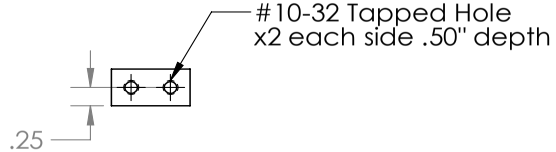
TITLE:
Base Plate
Shaker lab

SIZE	DWG. NO.	REV
A	SL-003	A
SCALE: 1:4	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1





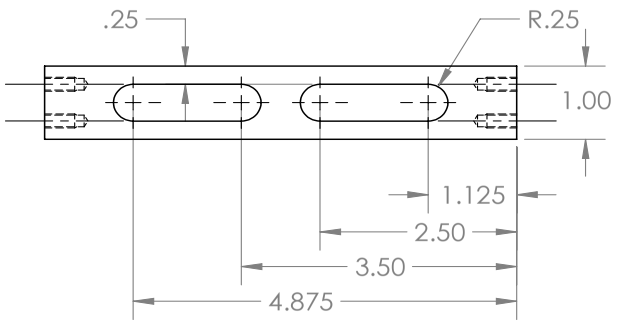


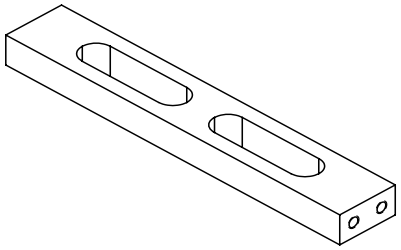
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005	DRAWN	EC	3/27/15
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL 6061 Aluminum	COMMENTS: Break all edges		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

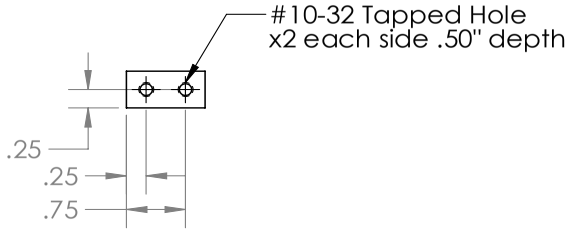
TITLE:
Bottom Stiffener
Shaker lab

SIZE	DWG. NO.	REV
A	SL-001	A
SCALE: 1:2	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1





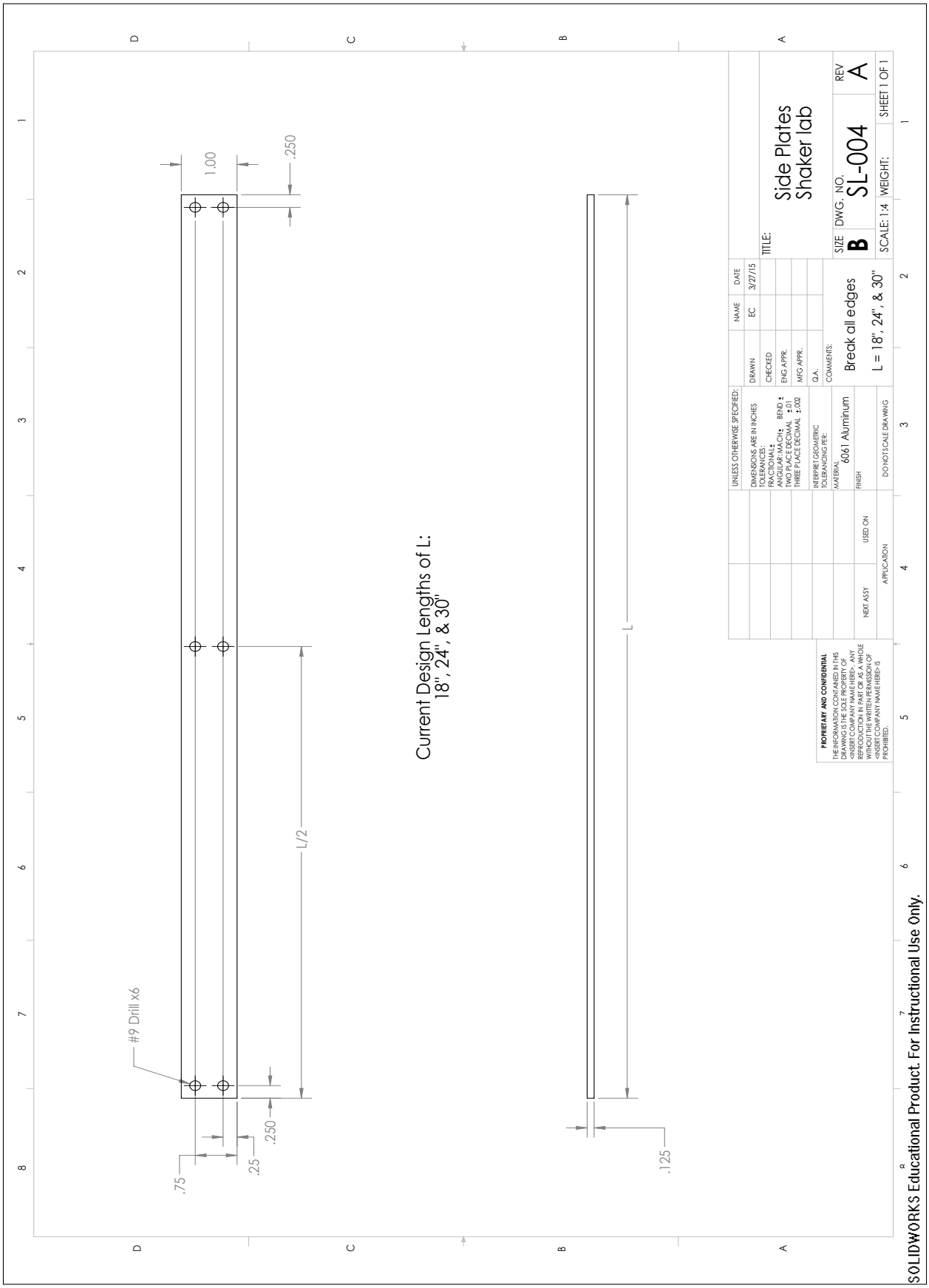


		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005	DRAWN	EC	3/27/15
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL 6061 Aluminum	COMMENTS: Break all edges		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:
Floor Stiffener Shaker lab

SIZE	DWG. NO.	REV
A	SL-002	A
SCALE: 1:2	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1


#9 Drill x6
1.00
.250
.75
.25
.250
L/2
L
.125
Current Design Lengths of L:
18", 24", & 30"
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±.01
THREE PLACE DECIMAL ±.002
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
6061 Aluminum
FINISH
DO NOT SCALE DRAWING
NEXT ASSY
USED ON
APPLICATION
DRAWN
EC
3/27/15
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
Break all edges
L = 18", 24", & 30"
TITLE:
Side Plates
Shaker lab
SIZE
B
DWG. NO.
SL-004
REV
A
SCALE: 1:4
WEIGHT:
SHEET 1 OF 1
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
SOLIDWORKS Educational Product. For Instructional Use Only.

B.2 Equivalent Mass and Stiffness Calculations

This section explains the derivation for finding the equivalent stiffness of the beams supporting the floors of the two-story building and the equivalent combined mass of the single-story floor and beam masses.

B.2.1 Beam Deflection

The first derivation is for the deflection of a single beam. Figure B.1 represents the column buckling of a slender vertical beam fixed at the bottom and attached to a mass at the top. The height of the beam is defined by length, L. The lateral movement is defined in the y position with y_{max} being the maximum y position the beam can move at the joint with the mass, m. The beam is rigidly attached to the base and the bottom of the mass. Therefore the is no slope at the ends of the beam.

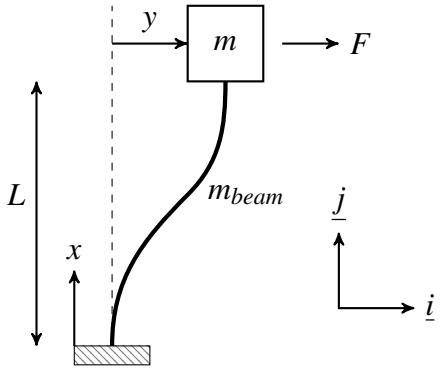


Figure B.1: Beam rigidly fixed at one end and a mass on the other.

The lateral position y is at its max when the vertical position x of the beam is at its max:

$$y = y_{max}$$

$$x = L$$

The lateral position y is half of its max when the vertical position x is halfway up the length of the beam:

$$y = \frac{y_{max}}{2}$$

$$x = \frac{L}{2}$$

Let the beam's lateral position as a function of the vertical position x be a cubic polynomial:

$$y(x) = A + Bx + Cx^2 + Dx^3$$

Starting with the previous assumption of no slope at its ends, the boundary conditions at the base of the beam, at $x = 0$, for lateral position and slope are both zero:

$$y(0) = 0$$

$$\left.\frac{dy}{dx}\right|_{x=0} = 0$$

Substituting $x = 0$ into the cubic polynomial makes A and B equal to zero:

$$y(0) = A = 0$$

$$\left.\frac{dy}{dx}\right|_{x=0} = B + 2C(0) + 3D(0)^2 = B = 0$$

The cubic polynomial equation then reduces to two terms:

$$y(x) = Cx^2 + Dx^3$$

The lateral position at the top of the beam, at $x = L$, is at its maximum and the slope of the beam at the top is also zero, so

$$y(L) = y_{max}$$

$$\left.\frac{dy}{dx}\right|_{x=L} = 0$$

Therefore,

$$y(L) = CL^2 + DL^3 = y_{max}$$

$$\left.\frac{dy}{dx}\right|_{x=L} = 2CL + 3DL^2 = 0$$

The equation for the slope at the top of the beam is used to solve for C in relation to D:

$$C = -\frac{3DL}{2}$$

Substituting for C in the position equation gives an equation for D in relation to y_{max}:

$$\left(-\tfrac{3DL}{2}\right)L^2 + DL^3 = -\tfrac{DL^3}{2} = y_{max}$$

$$D = -\tfrac{2y_{max}}{L^3}$$

C can then be related to y_{max}.

$$C = \frac{3y_{max}}{L^2}$$

The equation of the beam's lateral movement is then related to its maximum position, y_{max}, and the vertical position along the beam:

$$y(x) = \frac{3y_{max}}{L^2}x^2 - \frac{2y_{max}}{L^3}x^3$$

B.2.2 Beam Equivalent Stiffness

All beam movement is assumed to be in the elastic region. From Statics and Mechanics of Materials [5] the expression for shear force, F of a beam under elastic deformation is

$$F(x) = -EI\frac{d^3y}{dx^3}\bigg|_{x=L}$$

Young's Modulus, E, and the area moment of inertia, I, are known from the aluminum bar used for the beams in the building. The third derivative of the beam's lateral movemtn, y(x), is

$$\frac{d^3y}{dx^3} = -\frac{12y_{max}}{L^3}$$

Evaluating the integral at $x = L$ gives the equation for the shear force in the beam:

$$F = \frac{12EIy_{max}}{L^3}$$

Inserting the value of the shear force into Hooke's Law and solving for the equivalent spring rate, k_{eq}, at the maximum y position, y_{max}, gives

$$k_{eq} = \frac{F}{y_{max}} = \frac{12EI}{L^3}$$

B.2.3 Equivalent Mass

The expression for the equivalent mass can be obtained through the maximum kinetic energy of the beam, which comes from Rayleigh's energy method [2, pp. 153-158]:

$$KE_{max_{beam}} = \frac{1}{2}\int_0^L \frac{m_{beam}}{L}\ \dot{y}(x)^2\ dx$$

where m_{beam} is the total mass of the beam. The velocity variation, $\dot{y}(x)$, from the beam deflection equation is

$$\dot{y}(x) = \frac{\dot{y}_{max}}{L^2}\left(3x^2 - \frac{2x^3}{L}\right)$$

The definite integral is updated with the velocity variation expression:

$$KE_{max_{beam}} = \frac{1}{2}\int_0^L \frac{m_{beam}}{L}\ \left(\frac{\dot{y}_{max}}{L^2}\left(3x^2 - \frac{2x^3}{L}\right)\right)^2\ dx$$

Removing constants from the integral gives

$$KE_{max_{beam}} = \frac{1}{2}\frac{m_{beam}\dot{y}_{max}^2}{L^5}\int_0^L \left(\left(3x^2 - \frac{2x^3}{L}\right)\right)^2\ dx$$

Integrating yields

$$KE_{max_{beam}} = \frac{1}{2}\frac{m_{beam}\dot{y}_{max}^2}{L^5}\left(\frac{13}{35}L^5\right) = \frac{1}{2}\left(\frac{13}{35}m_{beam}\right)\dot{y}_{max}^2$$

The maximum kinetic energy expressed with the equivalent mass of the beam is

$$KE_{max_{beam}} = \frac{1}{2}m_{eq,beam}\dot{y}_{max}^2$$

The equivalent mass of the beam is therefore

$$m_{eq,beam} = \frac{13}{35} m_{beam}$$

The total equivalent mass of a single story of the building is

$$m_{eq} = m_{floor} + 2\left(\frac{13}{35} m_{beam}\right)$$

The lumped-mass model of a single-story building model is shown in Figure B.2.

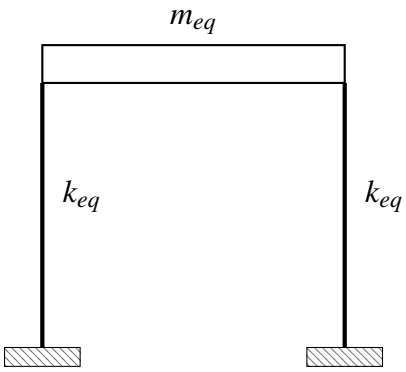


Figure B.2: Equivalent mass and stiffness of floor and beam system.

B.3 Sine Unrectifier MATLAB Code

This section includes the GUIDE layout of the Sine Unrectifier program, `Shaker_Data_Analyzer.m` and accompanying MATLAB code created for the user interface and necessary functions. `Excel_File_Output.m` and `Excel_File_Time_Gaps.m` are functions used to load the building accelerometer data files need to view, edit, and create the unrectified sine waves.

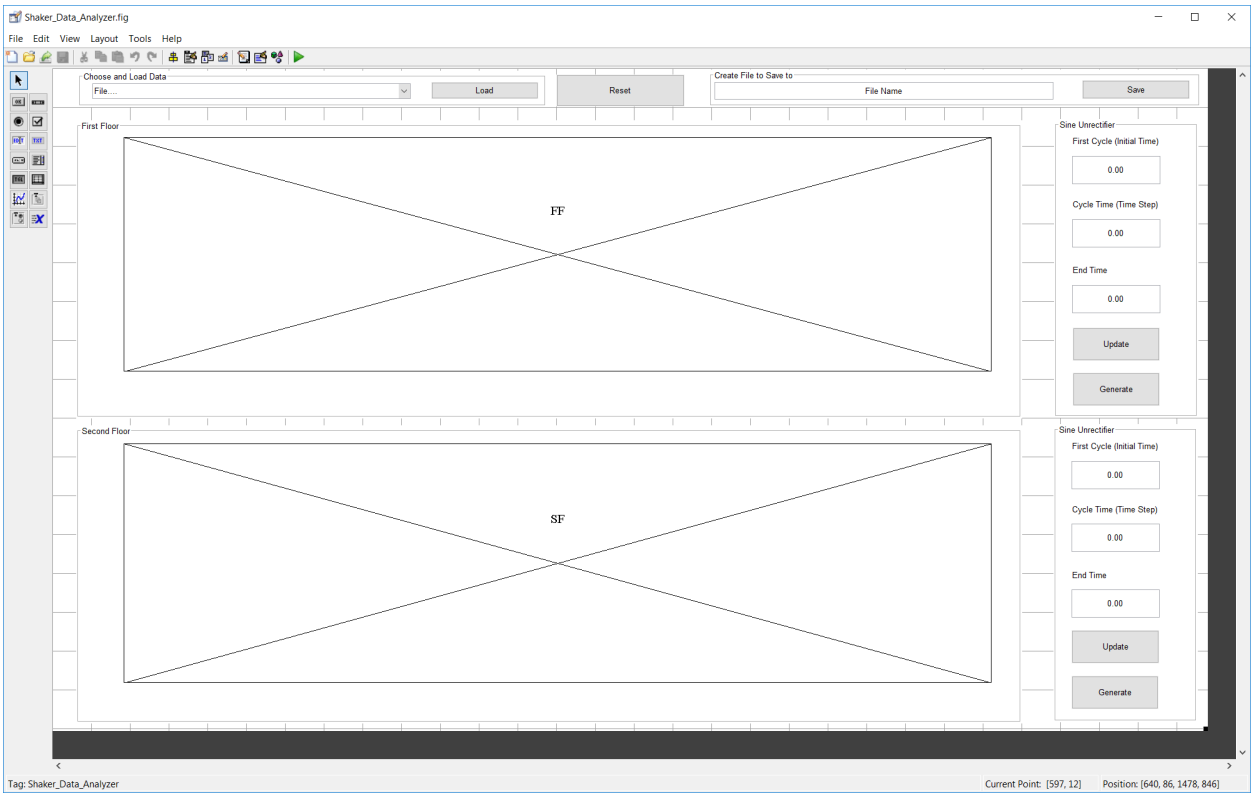


Figure B.3: GUIDE layout design of Sine Unrectifier program.

```
function varargout = Shaker_Data_Analyzer(varargin)
%SHAKER_DATA_ANALYZER M-file for Shaker_Data_Analyzer.fig
%      SHAKER_DATA_ANALYZER, by itself, creates a new SHAKER_DATA_ANALYZER or
    raises the existing
%      singleton*.
%
%      H = SHAKER_DATA_ANALYZER returns the handle to a new
```

```
    SHAKER_DATA_ANALYZER or the handle to
%       the existing singleton*.
%
%       SHAKER_DATA_ANALYZER('Property','Value',...) creates a new
    SHAKER_DATA_ANALYZER using the
%       given property value pairs. Unrecognized properties are passed via
%       varargin to Shaker_Data_Analyzer_OpeningFcn.  This calling syntax
    produces a
%       warning when there is an existing singleton*.
%
%       SHAKER_DATA_ANALYZER('CALLBACK') and SHAKER_DATA_ANALYZER('CALLBACK',
    hObject,...) call the
%       local function named CALLBACK in SHAKER_DATA_ANALYZER.M with the given
    input
%       arguments.
%
%       *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%       instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Shaker_Data_Analyzer

% Last Modified by GUIDE v2.5 17-Sep-2018 22:56:59

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Shaker_Data_Analyzer_OpeningFcn, ...
                   'gui_OutputFcn',  @Shaker_Data_Analyzer_OutputFcn, ...
                   'gui_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
% End initialization code - DO NOT EDIT


% --- Executes just before Shaker_Data_Analyzer is made visible.
function Shaker_Data_Analyzer_OpeningFcn(hObject, eventdata, handles, varargin
    )
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)
```

```
55
% Choose default command line output for Shaker_Data_Analyzer
handles.output = hObject;

set(handles.GenerateFF,'Enable','off')
60 set(handles.GenerateSF,'Enable','off')
set(handles.Save,'Enable','off')
set(handles.UpdateFF,'Enable','off')
set(handles.UpdateSF,'Enable','off')
cla
65
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Shaker_Data_Analyzer wait for user response (see UIRESUME)
70 % uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Shaker_Data_Analyzer_OutputFcn(hObject, eventdata,
    handles)
75 % varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

80 % Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in popup.
85 function popup_Callback(hObject, eventdata, handles)
% hObject    handle to popup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

90 % Hints: contents = cellstr(get(hObject,'String')) returns popup contents as
    cell array
%        contents{get(hObject,'Value')} returns selected item from popup

[FileName,PathName,FilterIndex] = uigetfile('*.*','Find the data file you want
    to process.');

95 set(handles.popup,'String',FileName);

file = strcat(PathName,FileName);

handles.file = file;
100
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function popup_CreateFcn(hObject, eventdata, handles)
105 % hObject    handle to popup (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in Save.
function Save_Callback(hObject, eventdata, handles)
% hObject    handle to Save (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

file = handles.file;

[ data ] = Excel_File_Output( file );

ti = str2double(get(handles.FC1,'String'));
ts = str2double(get(handles.CT1,'String'));
te = str2double(get(handles.ET1,'String'));
SaveFile = get(handles.SaveFileName,'String');

FF_ind = handles.FF_ind;
SF_ind = handles.SF_ind;

for aa = 1:2:length(FF_ind)-1

    data(FF_ind(aa):FF_ind(aa+1),2) = -data(FF_ind(aa):FF_ind(aa+1),2);

end

for bb = 1:2:length(SF_ind)-1

    data(SF_ind(bb):SF_ind(bb+1),3) = -data(SF_ind(bb):SF_ind(bb+1),3);

end

ind = find(data(:,1) >= ti & data(:,1) < (te+ts));

xlsdata = data(ind,:);

[FileName,PathName,FilterIndex] = uiputfile('*.xls','Select File to Write to',
    SaveFile);

newfile = strcat(PathName,FileName);

xlswrite(newfile,xlsdata);

% --- Executes on button press in Load.
function Load_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to Load (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

cla(handles.FF)
cla(handles.SF)

file = handles.file;

[ data ] = Excel_File_Output( file );

axes(handles.FF)

plot(data(:,1),data(:,2))
hold on
set(gcf,'Color','w')
xlabel('Time [s]')
ylabel('Amplitude [g]')
axis([0 max(data(:,1)) 0 max(data(:,2))+.1])

axes(handles.SF)

plot(data(:,1),data(:,3))
hold on
set(gcf,'Color','w')
xlabel('Time [s]')
ylabel('Amplitude [g]')
axis([0 max(data(:,1)) 0 max(data(:,3))+.1])

set(handles.UpdateFF,'Enable','on');
set(handles.UpdateSF,'Enable','on');

guidata(hObject,handles)

% --- Executes on button press in GenerateFF.
function GenerateFF_Callback(hObject, eventdata, handles)
% hObject    handle to GenerateFF (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

file = handles.file;

[ data ] = Excel_File_Output( file );

FF_ind = handles.FF_ind;
te = str2double(get(handles.ET1,'String'));

for aa = 1:2:length(FF_ind)-1

    data(FF_ind(aa):FF_ind(aa+1),2) = -data(FF_ind(aa):FF_ind(aa+1),2);

end

axes(handles.FF)
```

```
cla

plot(data(:,1),data(:,2))
hold on
set(gcf,'Color','w')
xlabel('Time [s]')
ylabel('Amplitude [g]')
axis([0 te -(max(data(:,2))+.1) max(data(:,2))+.1])

set(handles.SaveFileName,'Enable','on')
set(handles.Save,'Enable','on')

% --- Executes on button press in GenerateSF.
function GenerateSF_Callback(hObject, eventdata, handles)
% hObject    handle to GenerateSF (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

file = handles.file;

[ data ] = Excel_File_Output( file );

SF_ind = handles.SF_ind;
te = str2double(get(handles.ET2,'String'));

for bb = 1:2:length(SF_ind)-1

    data(SF_ind(bb):SF_ind(bb+1),3) = -data(SF_ind(bb):SF_ind(bb+1),3);

end

axes(handles.SF)

cla

plot(data(:,1),data(:,3))
hold on
set(gcf,'Color','w')
xlabel('Time [s]')
ylabel('Amplitude [g]')
axis([0 te -(max(data(:,3))+.1) max(data(:,3))+.1])

set(handles.SaveFileName,'Enable','on')
set(handles.Save,'Enable','on')

function FC2_Callback(hObject, eventdata, handles)
% hObject    handle to FC2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of FC2 as text
%        str2double(get(hObject,'String')) returns contents of FC2 as a double

```

```
% --- Executes during object creation, after setting all properties.
function FC2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FC2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function CT2_Callback(hObject, eventdata, handles)
% hObject    handle to CT2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of CT2 as text
%        str2double(get(hObject,'String')) returns contents of CT2 as a double


% --- Executes during object creation, after setting all properties.
function CT2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CT2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET2_Callback(hObject, eventdata, handles)
% hObject    handle to ET2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET2 as text
%        str2double(get(hObject,'String')) returns contents of ET2 as a double


% --- Executes during object creation, after setting all properties.
function ET2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in UpdateSF.
function UpdateSF_Callback(hObject, eventdata, handles)
% hObject    handle to UpdateSF (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

ti = str2double(get(handles.FC2,'String'));
ts = str2double(get(handles.CT2,'String'));
te = str2double(get(handles.ET2,'String'));

if te <= ti
    h1 = msgbox('End time must be greater than start time.','Error','error');
elseif ts <=0
    h2 = msgbox('Time step must be greater than zero.','Error','error');
elseif ti < 0
    h3 = msgbox('Start time must be greater than or equal to zero.','Error','
        error');
else

    axes(handles.SF)

    cla

    file = handles.file;

    [ data,split ] = Excel_File_Time_Gaps( file,ti,ts,te );

    plot(data(:,1),data(:,3))
    hold on
    set(gcf,'Color','w')
    xlabel('Time [s]')
    ylabel('Amplitude [g]')
    axis([0 te 0 max(data(:,3))+.1])

    for jj = 1:length(split)

        plot([data(split(jj),1),data(split(jj),1)],[0,max(data(:,3))+.1],'--r'
            )

    end

    for kk = 1:length(split)-1

        [trans(kk),min_ind(kk)] = min(data([split(kk):split(kk+1)],3));
```

```
    end

    min_ind = min_ind + split(1:end-1) - 1;

    for ll = 1:length(min_ind)

        plot(data(min_ind(ll),1),trans(ll),'go')

    end

    handles.SF_ind = min_ind;

    set(handles.GenerateSF,'Enable','on')

end

guidata(hObject,handles)

function FC1_Callback(hObject, eventdata, handles)
% hObject    handle to FC1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of FC1 as text
%        str2double(get(hObject,'String')) returns contents of FC1 as a double


% --- Executes during object creation, after setting all properties.
function FC1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FC1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function CT1_Callback(hObject, eventdata, handles)
% hObject    handle to CT1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of CT1 as text
%        str2double(get(hObject,'String')) returns contents of CT1 as a double


% --- Executes during object creation, after setting all properties.
function CT1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CT1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function ET1_Callback(hObject, eventdata, handles)
% hObject    handle to ET1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET1 as text
%        str2double(get(hObject,'String')) returns contents of ET1 as a double


% --- Executes during object creation, after setting all properties.
function ET1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in UpdateFF.
function UpdateFF_Callback(hObject, eventdata, handles)
% hObject    handle to UpdateFF (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get data from input boxes
ti = str2double(get(handles.FC1,'String'));
ts = str2double(get(handles.CT1,'String'));
te = str2double(get(handles.ET1,'String'));

% Error handle the input data
if te <= ti
    h1 = msgbox('End time must be greater than start time.','Error','error');
elseif ts <=0
    h2 = msgbox('Time step must be greater than zero.','Error','error');
elseif ti < 0
    h3 = msgbox('Start time must be greater than or equal to zero.','Error','
        error');
else
```

```
    axes(handles.FF)

    cla

    file = handles.file;

    [ data,split ] = Excel_File_Time_Gaps( file,ti,ts,te );

    plot(data(:,1),data(:,2))
    hold on
    set(gcf,'Color','w')
    xlabel('Time [s]')
    ylabel('Amplitude [g]')
    axis([0 te 0 max(data(:,2))+.1])

    for jj = 1:length(split)

        plot([data(split(jj),1),data(split(jj),1)],[0,max(data(:,2))+.1],'--r'
            )

    end

    for kk = 1:length(split)-1

        [trans(kk),min_ind(kk)] = min(data([split(kk):split(kk+1)],2));

    end

    min_ind = min_ind + split(1:end-1) - 1;

    for ll = 1:length(min_ind)

        plot(data(min_ind(ll),1),trans(ll),'go')

    end

    handles.FF_ind = min_ind;

    set(handles.GenerateFF,'Enable','on')

end

guidata(hObject,handles)

% --------------------------------------------------------------------
function FileMenu_Callback(hObject, eventdata, handles)
% hObject    handle to FileMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function OpenMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to OpenMenuItem (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function PrintMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to PrintMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function CloseMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to CloseMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



function SaveFileName_Callback(hObject, eventdata, handles)
% hObject    handle to SaveFileName (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of SaveFileName as text
%        str2double(get(hObject,'String')) returns contents of SaveFileName as
    a double

% set(handles.Save,'Enable','on')

% --- Executes during object creation, after setting all properties.
function SaveFileName_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SaveFileName (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in Reset.
function Reset_Callback(hObject, eventdata, handles)
% hObject    handle to Reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.GenerateFF,'Enable','off')
set(handles.GenerateSF,'Enable','off')
set(handles.Save,'Enable','off')
set(handles.UpdateFF,'Enable','off')
set(handles.UpdateSF,'Enable','off')
```

```
axes(handles.FF)
cla
axes(handles.SF)
cla

set(handles.FC1,'String','0.00');
set(handles.CT1,'String','0.00');
set(handles.ET1,'String','0.00');

set(handles.FC2,'String','0.00');
set(handles.CT2,'String','0.00');
set(handles.ET2,'String','0.00');

set(handles.SaveFileName,'Enable','on')
set(handles.SaveFileName,'String','File Name')
set(handles.popup,'String','File....')
```

```
function [ data ] = Excel_File_Output( file )

%-------------------------------------------------------------------------
% File opens rectified data from user selected excel file

% Inputs
% - file - user selected file of rectified sine data
%
% Outputs
% - data - data points from Excel file
%
% Ethan Cating
% 11/7/2016
%-------------------------------------------------------------------------

% Open Excel data to use and plot
[data, txt, raw] = xlsread(file);

end
```

```
function [ data,split ] = Excel_File_Time_Gaps( file,ti,ts,te )

%-------------------------------------------------------------------------
% Inputs
% - file - user selected file of rectified sine data
% - ti   - user input initial time for first sine peak
% - ts   - user input time step between peaks
% - te   - user input end time of data to save
%
% Outputs
% - data  - data points from Excel file
% - split - array of points at each time step starting from the initial
%           time to the end time
```

```
%
% Ethan Cating
% 11/7/2016
%-------------------------------------------------------------------------

% Open Excel data to use and plot
[data, txt, raw] = xlsread(file);

t = ti:ts:(te+ts); % time vector

split = zeros(1,length(t)); % initialize vector of zeros

% Find the times at those cycles
for ii = 1:length(t)

    split(ii) = find(data(:,1)>t(ii),1,'first');

end

end
```

B.4 Shaker Lab MATLAB Code

All remaining MATLAB files used for Chapter 2 are included here. `Natural_Frequency_Plots.m` code is used to generate Figure 3.12. Table 3.3 calculated values are made from `nat_freq_shaker.m`. The mode shapes plot, Figure 3.17 is generated from `Building_Shaker_Displacement.m`.

```
%--------------------------------------------------------------------------
%
% Natural_Frequency_Plots.m
%
% DESCRIPTION:
% Creates plots based on max acceleration from RT Pro data during shaker
% rig test to experimentally determine the natural frequency of the
% building at different leg heights and natural frequencies (30" only).
%
% INPUTS:
%
%        data18_1_1   - recorded Excel file from accelerometer data
%                       Leg Height_data point_floor number (18" & 24"
%                       Height)
%
%        data30_1_1_1 - Leg Height_Natural Frequency_data point_floor
%
% OUTPUTS:
%
%        Single figure window with four subplots:
%        18" Height @ 1st natural frequency
%        24" Height @ 1st natural frequency
%        30" Height @ 1st natural frequency
%        30" Height @ 2nd natural frequency
%
% Ethan Cating
% 4/12/2017
%--------------------------------------------------------------------------

clear all
clc
close all

% Setup for font and font size of figures
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 12)
set(0,'DefaultTextFontname', 'Times New Roman')
```

```
set(0,'DefaultTextFontSize', 12)

% Open Excel data to use and plot
data18_1 = xlsread('SL-18-240.xlsx');
data18_2 = xlsread('SL-18-245.xlsx');
data18_3 = xlsread('SL-18-250.xlsx');
data18_4 = xlsread('SL-18-255.xlsx');
data18_5 = xlsread('SL-18-260.xlsx');
freq_18 = [24, 24.5, 25, 25.5, 26];

data24_1 = xlsread('SL-24-150.xlsx');
data24_2 = xlsread('SL-24-155.xlsx');
data24_3 = xlsread('SL-24-160.xlsx');
data24_4 = xlsread('SL-24-165.xlsx');
data24_5 = xlsread('SL-24-170.xlsx');
freq_24 = [15, 15.5, 16, 16.5, 17];

data30_1_1 = xlsread('SL-30-100.xlsx');
data30_1_2 = xlsread('SL-30-105.xlsx');
data30_1_3 = xlsread('SL-30-110.xlsx');
data30_1_4 = xlsread('SL-30-115.xlsx');
data30_1_5 = xlsread('SL-30-120.xlsx');
freq_30_1 = [10, 10.5, 11, 11.5, 12];

data30_2_1 = xlsread('SL-30-280.xlsx');
data30_2_2 = xlsread('SL-30-285.xlsx');
data30_2_3 = xlsread('SL-30-290.xlsx');
data30_2_4 = xlsread('SL-30-295.xlsx');
data30_2_5 = xlsread('SL-30-300.xlsx');
freq_30_2 = [28, 28.5, 29, 29.5, 30];

% Find the maximum amplitudes of each data set for each "story"
max18_1_1 = max(data18_1(:,2));
max18_1_2 = max(data18_1(:,3));

max18_2_1 = max(data18_2(:,2));
max18_2_2 = max(data18_2(:,3));

max18_3_1 = max(data18_3(:,2));
max18_3_2 = max(data18_3(:,3));

max18_4_1 = max(data18_4(:,2));
max18_4_2 = max(data18_4(:,3));

max18_5_1 = max(data18_5(:,2));
max18_5_2 = max(data18_5(:,3));

% Max acceleration data points into one vector
max18_FF = [max18_1_1,max18_2_1,max18_3_1,max18_4_1,max18_5_1];
max18_SF = [max18_1_2,max18_2_2,max18_3_2,max18_4_2,max18_5_2];

max24_1_1 = max(data24_1(:,2));
max24_1_2 = max(data24_1(:,3));
```

```
max24_2_1 = max(data24_2(:,2));
max24_2_2 = max(data24_2(:,3));

max24_3_1 = max(data24_3(:,2));
max24_3_2 = max(data24_3(:,3));

max24_4_1 = max(data24_4(:,2));
max24_4_2 = max(data24_4(:,3));

max24_5_1 = max(data24_5(:,2));
max24_5_2 = max(data24_5(:,3));

% Max acceleration data points into one vector
max24_FF = [max24_1_1,max24_2_1,max24_3_1,max24_4_1,max24_5_1];
max24_SF = [max24_1_2,max24_2_2,max24_3_2,max24_4_2,max24_5_2];

max30_1_1_1 = max(data30_1_1(:,2));
max30_1_1_2 = max(data30_1_1(:,3));

max30_1_2_1 = max(data30_1_2(:,2));
max30_1_2_2 = max(data30_1_2(:,3));

max30_1_3_1 = max(data30_1_3(:,2));
max30_1_3_2 = max(data30_1_3(:,3));

max30_1_4_1 = max(data30_1_4(:,2));
max30_1_4_2 = max(data30_1_4(:,3));

max30_1_5_1 = max(data30_1_5(:,2));
max30_1_5_2 = max(data30_1_5(:,3));

% Max acceleration data points into one vector
max30_1_FF = [max30_1_1_1,max30_1_2_1,max30_1_3_1,max30_1_4_1,max30_1_5_1];
max30_1_SF = [max30_1_1_2,max30_1_2_2,max30_1_3_2,max30_1_4_2,max30_1_5_2];

max30_2_1_1 = max(data30_2_1(:,2));
max30_2_1_2 = max(data30_2_1(:,3));

max30_2_2_1 = max(data30_2_2(:,2));
max30_2_2_2 = max(data30_2_2(:,3));

max30_2_3_1 = max(data30_2_3(:,2));
max30_2_3_2 = max(data30_2_3(:,3));

max30_2_4_1 = max(data30_2_4(:,2));
max30_2_4_2 = max(data30_2_4(:,3));

max30_2_5_1 = max(data30_2_5(:,2));
max30_2_5_2 = max(data30_2_5(:,3));

% Max acceleration data points into one vector
max30_2_FF = [max30_2_1_1,max30_2_2_1,max30_2_3_1,max30_2_4_1,max30_2_5_1];
max30_2_SF = [max30_2_1_2,max30_2_2_2,max30_2_3_2,max30_2_4_2,max30_2_5_2];
```

```
% Plot for amplitudes at each frequency for first and second story
figure(1)
set(gcf,'Units','normalized')
set(gcf,'Position',[0.25 0.1875 0.5 0.625])

subplot(2,2,1)
plot(freq_18,max18_FF,'--sr')
hold on
plot(freq_18,max18_SF,'--ok')
set(gcf,'Color','w')
xlabel('Frequency [Hz]')
ylabel('Max Accel. [g]')
xlim([min(freq_18)-.25 max(freq_18)+.25])
ylim([0 max(max18_SF)+.25])
title('\rm First \omega_{n} of 18"-Tall Building');
set(gca,'TickDir','out')
legend('1^{st} Floor','2^{nd} Floor','Location','NorthWest',...
       'Interpreter','latex');
box off

subplot(2,2,2)
plot(freq_24,max24_FF,'--sr')
hold on
plot(freq_24,max24_SF,'--ok')
set(gcf,'Color','w')
xlabel('Frequency [Hz]')
ylabel('Max Accel. [g]')
xlim([min(freq_24)-.25 max(freq_24)+.25])
ylim([0 max(max24_SF)+.25])
title('\rm First \omega_{n} of 24"-Tall Building');
set(gca,'TickDir','out')
legend('1^{st} Floor','2^{nd} Floor','Location','NorthEast',...
       'Interpreter','latex');
box off

subplot(2,2,3)
plot(freq_30_1,max30_1_FF,'--sr')
hold on
plot(freq_30_1,max30_1_SF,'--ok')
set(gcf,'Color','w')
xlabel('Frequency [Hz]')
ylabel('Max Accel. [g]')
xlim([min(freq_30_1)-.25 max(freq_30_1)+.25])
ylim([0 max(max30_1_SF)+.25])
title('\rm First \omega_{n} of 30"-Tall Building');
set(gca,'TickDir','out')
legend('1^{st} Floor','2^{nd} Floor','Location','NorthEast',...
       'Interpreter','latex');
box off

subplot(2,2,4)
plot(freq_30_2,max30_2_FF,'--sr')
hold on
plot(freq_30_2,max30_2_SF,'--ok')
```

```
set(gcf,'Color','w')
xlabel('Frequency [Hz]')
ylabel('Max Accel. [g]')
xlim([min(freq_30_2)-.25 max(freq_30_2)+.25])
ylim([0 max(max30_2_SF)+.25])
title('\rm Second \omega_{n} of 30"-Tall Building');
set(gca,'TickDir','out')
legend('1^{st} Floor','2^{nd} Floor','Location','NorthEast',...
       'Interpreter','latex');
box off
```

```
%-------------------------------------------------------------------------
%
% nat_freq_shaker.m
%
% DESCRIPTION:
% Calculates natural frequency from floor and beam masses and beam lengths
% based on analytical model. Parts were made of 6061 Aluminum.
%
% INPUTS:
%
%        mfloor - mass of floor
%        mbeam  - mass of vertical legs
%        meq    - equivalent mass of floor and beams
%        E      - Modulus of Elasticity
%        L      - Height of beam
%        w      - width of leg
%        t      - thickness of leg
%        I      - Moment of inertia of leg
%        keq    - equivalent stiffness
%
% OUTPUTS:
%
%        V      - Eigenvectors of Mode Shapes
%        Om2    - Natural Frequencies squared (rad/sec)^2
%        F      - Natural Frequencies in Hz
%        error  - Percent difference from calculated and experimental
%                 natural frequencies
%
% Ethan Cating
% 4/12/2017
%-------------------------------------------------------------------------

clear all
close all
clc

mfloor = 3.15/16/32.2;           %  slug
mbeam = 3/16/32.2;               %  slug
meq = mfloor + 2*(13/35)*mbeam;  %  slug

E = 10000000;           %  psi
```

```
L = 15;                 %  in
w = 1;                  %  in
t = 1/8;                %  in
I = (1/12)*w*t^3;       %  in^4
keq = 12*E*I/(L^3)*12;  %  lb/ft

M = blkdiag(meq,meq);   %  Diagonal Mass Matrix
K = [4, -2; -2, 2]*keq; %  Stiffness Matrix

[V,Om2] = eig(K,M);     %  Finds eigenvalues (Om2) and
                        %  eigenvectors (V)

Om = sqrt(Om2);         %  rad/sec
F = diag(Om/(2*pi))     %  Hz

f1 = F(1);              %  Calculated Natural Frequency
f1_exp = 11;            %  Experimental Natural Frequency

error = abs(f1 - f1_exp)/f1_exp*100     % Percent error from calculated
                                        % to experimental frequency

Vn = V./norm(V);        % Normalized Mode Shapes
```

```
% Building_Shaker_Displacement.m

% Creates plots based on positional data from ECP during sine wave
% input testing

%--------------------------------------------------------------------------
%
% Building_Shaker_Displacement.m
%
% DESCRIPTION:
% Takes the positional data from ECP module during sine wave input testing,
% calculates base acceleration from position data and then compares input
% acceleration to floor accelerations.
%
% INPUTS:
%
%        ECP lab shaker input position data
%        RT Pro recorded building floors accelerometer data
%
% OUTPUTS:
%
%        Plots showing input acceleration to output acceleration
%        at natural frequencies of 30" tall building.
%
% Ethan Cating
% 4/12/2017
%--------------------------------------------------------------------------

% clear workspace
```

```
clear all
clc
close all

% Setup for font and font size of figures
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 12)
set(0,'DefaultTextFontname', 'Times New Roman')
set(0,'DefaultTextFontSize', 12)

%% 18" building @ 1st natural frequency
% Load base input data
data_base_18 = load('Data\Cating Thesis Quanser\Important_Data\
    Shaker_18_sine_NF1_5_28_15_3_59.mat');
tb_18 = data_base_18.data(1,:);       % Time data for ECP shaker
pb_18 = data_base_18.data(2,:)./100;  % Convert cm position data to m

% Create position data vector with time greater than 3 seconds and less
% than 3.5 seconds.
t_low_18 = (tb_18 > 3);            % Lower time limit
t_high_18 = (tb_18 < 3.5);         % Upper time limit
t_lim_18 = t_low_18.*t_high_18;     % Make all points zero outside limits
p_new_18 = t_lim_18.*pb_18;         % Multiply against shaker motor input
                                  % position data

A_18 = (max(p_new_18)-min(p_new_18))/2;  % peak-to-peak amplitude
offset_18 = max(p_new_18) - A_18; % offset of amplitudes from zero
p_new_18 = p_new_18 - offset_18;  % base input signal centered about 0

% Load unrectified sine wave data
data_accel_18 = xlsread('Data\Shaker Lab Data\Shaker_18_NF1_Unrectified.xls');
ta_18 = data_accel_18(1:end,1)+2.875; % Time data from RT Pro, not synced with
                                    % ECP shaker so time offset needed.

% Actual frequency input was 25.5 Hz, but 22 Hz made it in sync
om_18 = 22*2*pi;       % Second natural frequency in rad/sec to match base
                       % input data
phi_18 = 145*pi/180;   % Phase angle

% Generated sine wave signal
sine_18 = -max(p_new_18)*(om_18^2)*sin(om_18.*ta_18 + phi_18);

% Multiply floor abs. accel. data by gravity and subtract input accel
pf18_1 = data_accel_18(1:end,2).*9.81 - (sine_18);
pf18_2 = data_accel_18(1:end,3).*9.81 - (sine_18);

m18 = [max(pf18_1) max(pf18_2)]; % Find maximum mode shape values
m18 = m18./norm(m18);            % Normalize the mode shape

% Data comparison plot to
% figure(1)
% set(gcf,'Units','normalized')
% set(gcf,'Position',[0.25 0.1875 0.5 0.625])
%
```

```
% plot(ta_18,sine_18,'-og')
% hold on
% % plot(tb_18,-(p_new_18)*(om_18^2),'--k')
% plot(ta_18,pf18_1,'k')
% plot(ta_18,pf18_2,'r')
% xlim([3,3.5])
% set(gcf,'Color','w')
% xlabel('Time [s]')
% ylabel('Acceleration [m/sec^2]')
% title({'Shaker Base Input Accel. Signal Compared w/ Building Floor Accel.
   Data';...
%       'at 18" Building Height and at the 1^{st} Natural Frequency'})
% legend('Equation','1^{st} Floor','2^{nd} Floor','Location',...
%     'NorthEast');

%% 24" building @ 1st natural frequency
% Load base input data
data_base_24 = load('Data\Cating Thesis Quanser\Important_Data\
   Shaker_24_sine_NF1_5_28_15_4_20.mat');
tb_24 = data_base_24.data(1,:);       % Time data for ECP shaker
pb_24 = data_base_24.data(2,:)./100;  % Convert cm position data to m

% Create position data vector with time greater than 3 seconds and less
% than 3.5 seconds.
t_low_24 = (tb_24 > 3);           % Lower time limit
t_high_24 = (tb_24 < 3.5);        % Upper time limit
t_lim_24 = t_low_24.*t_high_18;    % Make all points zero outside limits
p_new_24 = t_lim_24.*pb_18;        % Multiply against shaker motor input
                                % position data

A_24 = (max(p_new_24)-min(p_new_24))/2;  % peak-to-peak amplitude
offset_24 = max(p_new_24) - A_24; % offset of amplitudes from zero
p_new_24 = p_new_24 - offset_24;  % base input signal centered about 0

% Load unrectified sine wave data
data_accel_24 = xlsread('Data\Shaker Lab Data\Shaker_24_NF1_Unrectified.xls');
ta_24 = data_accel_24(1:end,1)+2.893; % Time data from RT Pro, not synced with
                                   % ECP shaker so time offset needed.

% Actual frequency input was 16 Hz
om_24 = 16*2*pi;      % Second natural frequency in rad/sec to match base
                      % input data
phi_24 = 180*pi/180;  % Phase angle

% Generated sine wave signal
sine_24 = -max(p_new_24)*(om_24^2)*sin(om_24.*ta_24 + phi_24);

% Multiply floor abs. accel. data by gravity and subtract input accel
pf24_1 = data_accel_24(1:end,2).*9.81 - (sine_24);
pf24_2 = data_accel_24(1:end,3).*9.81 - (sine_24);

m24 = [max(pf24_1) max(pf24_2)]; % Find maximum mode shape values
m24 = m24./norm(m24);            % Normalize the mode shape
```

```
% Data comparison plot to
% figure(2)
% set(gcf,'Units','normalized')
% set(gcf,'Position',[0.25 0.1875 0.5 0.625])
%
% plot(ta_24,sine_24,'--g')
% hold on
% % plot(tb_24,-(p_new_24)*(om_24^2),'--k')
% plot(ta_24,pf24_1,'k')
% plot(ta_24,pf24_2,'r')
% xlim([3,3.5])
% set(gcf,'Color','w')
% xlabel('Time [s]')
% ylabel('Acceleration [m/sec^2]')
% title({'Shaker Base Input Accel. Signal Compared w/ Building Floor Accel.
   Data';...
%       'at 24" Building Height and at the 1^{st} Natural Frequency'})
% legend('Equation','1^{st} Floor','2^{nd} Floor','Location',...
%     'NorthEast');

%% 30" building @ 1st natural frequency
% Load base input data
data_base_1 = load('Data\Cating Thesis Quanser\Important_Data\
   Shaker_30_sine_NF1_5_28_15_4_38.mat');
tb_1 = data_base_1.data(1,:);       % Time data for ECP shaker
pb_1 = data_base_1.data(2,:)./100;  % Convert cm position data to m

% Create position data vector with time greater than 3 seconds and less
% than 3.5 seconds.
t_low_1 = (tb_1 > 3);           % Lower time limit
t_high_1 = (tb_1 < 3.5);        % Upper time limit
t_lim_1 = t_low_1.*t_high_1;    % Make all points zero outside limits
p_new_1 = t_lim_1.*pb_1;        % Multiply against shaker motor input
                                % position data

A_1 = (max(p_new_1)-min(p_new_1))/2;  % peak-to-peak amplitude
offset_1 = max(p_new_1) - A_1; % offset of amplitudes from zero
p_new_1 = p_new_1 - offset_1;  % base input signal centered about 0

% Load unrectified sine wave data
data_accel_1 = xlsread('Data\Shaker Lab Data\Shaker_30_NF1_Unrectified.xls');
ta_1 = data_accel_1(1:end,1)+2.875; % Time data from RT Pro, not synced with
                                    % ECP shaker so time offset needed.

% Actual frequency input was 11 Hz
om_1 = 10.5*2*pi;     % Second natural frequency in rad/sec to match base
                      % input data
phi_1 = 80*pi/180;  % Phase angle

% Generated sine wave signal
sine_30_1 = -max(p_new_1)*(om_1^2)*sin(om_1.*ta_1 + phi_1);

% Multiply floor abs. accel. data by gravity and subtract input accel
pf1_1 = data_accel_1(1:end,2).*9.81 - (sine_30_1);
```

```
pf1_2 = data_accel_1(1:end,3).*9.81 - (sine_30_1);

m1 = [max(pf1_1) max(pf1_2)]; % Find maximum mode shape values
m1 = m1./norm(m1);            % Normalize the mode shape

% Data comparison plot to
% figure(3)
% set(gcf,'Units','normalized')
% set(gcf,'Position',[0.25 0.1875 0.5 0.625])
%
% plot(ta_1,sine_30_1,'-og')
% hold on
% plot(tb_1,-(p_new_1)*(om_1^2),'--k')
% plot(ta_1,pf1_1,'k')
% plot(ta_1,pf1_2,'r')
% xlim([3,3.5])
% set(gcf,'Color','w')
% xlabel('Time [s]')
% ylabel('Acceleration [m/sec^2]')
% title({'Shaker Base Input Accel. Signal Compared w/ Building Floor Accel.
   Data';...
%       'at 30" Building Height and at the 1^{st} Natural Frequency'})
% legend('Equation','1^{st} Floor','2^{nd} Floor','Location',...
%     'NorthEast');

%% 30" building @ 2nd natural frequency
% Load base input data
data_base_2 = load('Data\Cating Thesis Quanser\Important_Data\
   Shaker_30_sine_NF2_5_28_15_4_46.mat');
tb_2 = data_base_2.data(1,:);       % Time data for ECP shaker
pb_2 = data_base_2.data(2,:)./100;  % Convert cm position data to m

% Create position data vector with time greater than 3 seconds and less
% than 3.5 seconds.
t_low_2 = (tb_2 > 3);           % Lower time limit
t_high_2 = (tb_2 < 3.5);        % Upper time limit
t_lim_2 = t_low_2.*t_high_2;    % Make all points zero outside limits
p_new_2 = t_lim_2.*pb_2;        % Multiply against shaker motor input
                                % position data

A_2 = (max(p_new_2)-min(p_new_2))/2;  % peak-to-peak amplitude
offset_2 = max(p_new_2) - A_2; % offset of amplitudes from zero
p_new_2 = p_new_2 - offset_2;  % base input signal centered about 0

% Load unrectified sine wave data
data_accel_2 = xlsread('Data\Shaker Lab Data\Shaker_30_NF2_Unrectified.xls');
ta_2 = data_accel_2(1:end,1)+2.875; % Time data from RT Pro, not synced with
                                    % ECP shaker so time offset needed.

% Actual frequency input was 29 Hz
om_2 = 28.5*2*pi;  % Second natural frequency in rad/sec to match base
                   % input data
phi_2 = -150*pi/180;  % Phase angle
```

```
% Generated sine wave signal
sine_30_2 = -max(p_new_2)*(om_2^2)*sin(om_2.*ta_2 + phi_2);

% Multiply floor abs. accel. data by gravity and subtract input accel
pf2_1 = data_accel_2(1:end,2).*9.81 - (sine_30_2);
pf2_2 = data_accel_2(1:end,3).*9.81 - (sine_30_2);

m2 = [-max(pf2_1) max(pf2_2)]; % Find maximum mode shape values
m2 = m2./norm(m2);             % Normalize the mode shape

% Data comparison plot to
% figure(4)
% set(gcf,'Units','normalized')
% set(gcf,'Position',[0.25 0.1875 0.5 0.625])
%
% plot(ta_2,sine_30_2,'-og')
% hold on
% % plot(tb_2,-(p_new_2)*(om_2^2),'--k')
% plot(ta_2,-pf2_1,'k')
% plot(ta_2,pf2_2,'r')
% xlim([3,3.5])
% set(gcf,'Color','w')
% xlabel('Time [s]')
% ylabel('Acceleration [m/sec^2]')
% title({'Shaker Base Input Accel. Signal Compared w/ Building Floor Accel.
   Data';...
%       'at 30" Building Height and at the 2^{nd} Natural Frequency'})
% legend('Equation','1^{st} Floor','2^{nd} Floor','Location',...
%     'NorthEast');

%% Mode Shape Comparison Plot

figure(5)
set(gcf,'Units','normalized')
set(gcf,'Position',[0.25 0.1875 0.3125 0.75])
set(gcf,'Color','w')

subplot(2,2,1)
plot([0 0.5257 0.8507],[0 1 2],'k-o') % Vn from nat_freq_shaker.m
hold on
plot([0 m18(1) m18(2)], [0 1 2],'r--s')
plot([0 0],[0 2],'k--')
set(gca,'TickDir','out')
box off
xlim([-0.25 1.25])
xlabel('Normalized Accel.')
ylim([0 2.25])
ylabel('Floor')
set(gca,'YTick', [0 1 2]);
title('18"-Tall Building at $\omega_{n}_{1}$','interpreter','latex')
legend('Calculated','Experimental','Location','Northwest')

subplot(2,2,2)
plot([0 0.5257 0.8507],[0 1 2],'k-o') % Vn from nat_freq_shaker.m
```

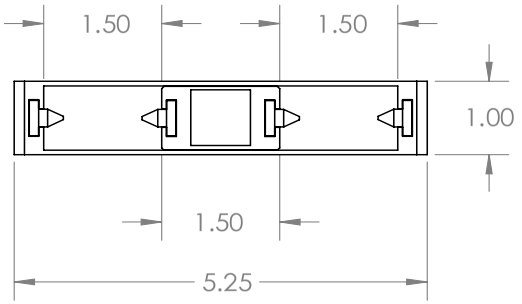
```
hold on
plot([0 m24(1) m24(2)], [0 1 2],'r--s')
plot([0 0],[0 2],'k--')
set(gca,'TickDir','out')
box off
xlim([-0.25 1.25])
xlabel('Normalized Accel.')
ylim([0 2.25])
ylabel('Floor')
set(gca,'YTick', [0 1 2]);
title('24"-Tall Building at $\omega_{n}_{1}$','interpreter','latex')
legend('Calculated','Experimental','Location','Northwest')

subplot(2,2,3)
plot([0 0.5257 0.8507],[0 1 2],'k-o') % Vn from nat_freq_shaker.m
hold on
plot([0 m1(1) m1(2)], [0 1 2],'r--s')
plot([0 0],[0 2],'k--')
set(gca,'TickDir','out')
box off
xlim([-0.25 1.25])
xlabel('Normalized Accel.')
ylim([0 2.25])
ylabel('Floor')
set(gca,'YTick', [0 1 2]);
title('30"-Tall Building at $\omega_{n}_{1}$','interpreter','latex')
legend('Calculated','Experimental','Location','Northwest')

subplot(2,2,4)
plot([0 -0.8507 0.5257],[0 1 2],'k-o') % Vn from nat_freq_shaker.m
hold on
plot([0 m2(1) m2(2)], [0 1 2],'r--s')
plot([0 0],[0 2],'k--')
set(gca,'TickDir','out')
box off
xlim([-1.25 1.25])
xlabel('Normalized Accel.')
ylim([0 2.25])
ylabel('Floor')
set(gca,'YTick', [0 1 2]);
title('30"-Tall Building at $\omega_{n}_{2}$','interpreter','latex')
legend('Calculated','Experimental','Location','Northwest')
```

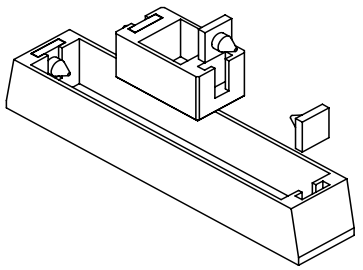
C Shaker Lab with Tuned-Mass Damper Appendix

C.1 TMD Lab SolidWorks Prints

Included in this appendix section are all of the drawing prints to create the tuned-mass damper that sits on the top of the second floor of the building. The assembly does not include the purchased parts of the tuned-mass damper mass or the springs that center the carriage. The parts of the prints attached were printed by 3D printers at Rose-Hulman.

ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	SL-005	Shaker lab TMD holder	1
2	SL-006	Shaker lab TMD weight carriage	1
3	SL-007	Shaker lab TMD spring holder	4





		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005	DRAWN	EC	4/15/15
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

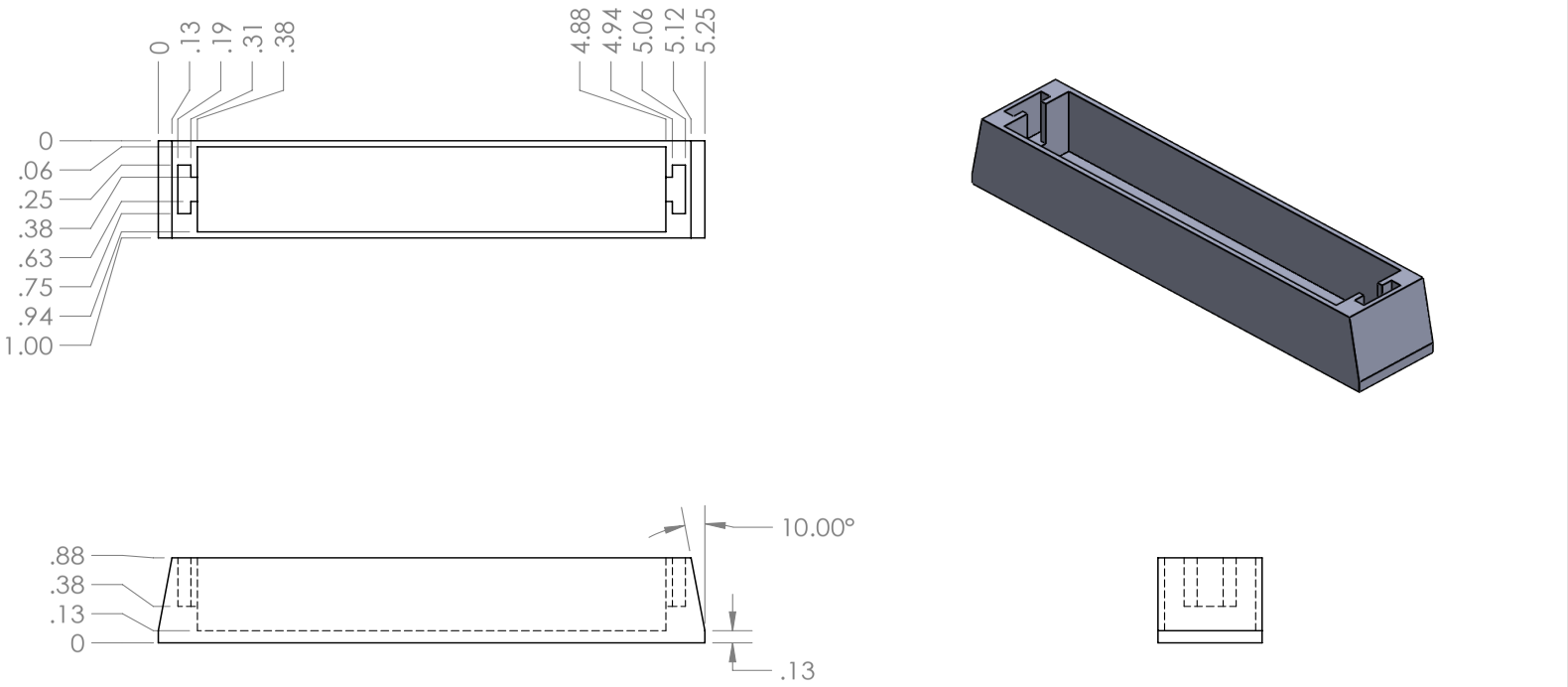
TITLE:

TMD Assembly

SIZE	DWG. NO.	REV
A	Shaker_TMD	

SCALE: 1:2 | SHEET 1 OF 1

5 4 3 2 1



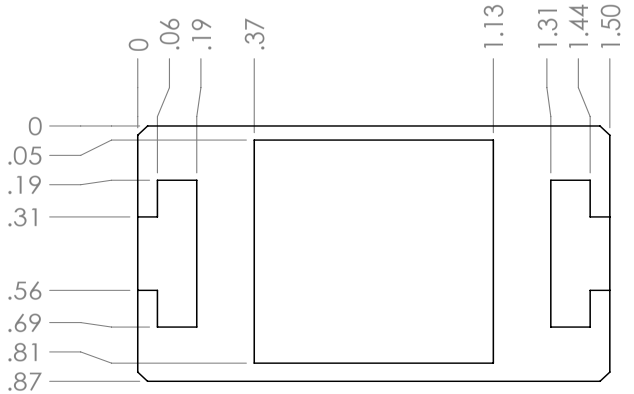
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005	DRAWN	EC	4/15/15
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL PLA Plastic	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

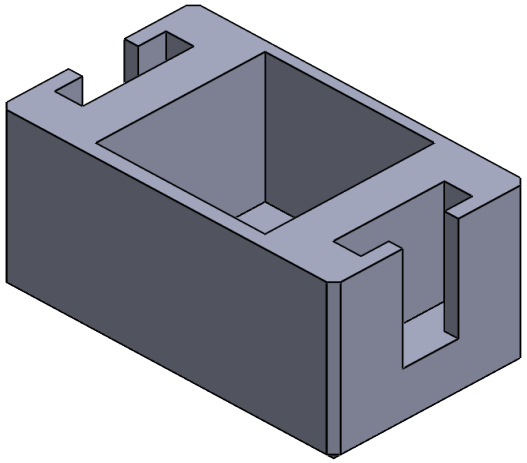
TITLE:

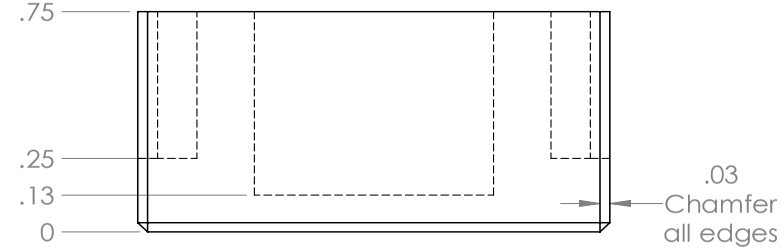
TMD holder

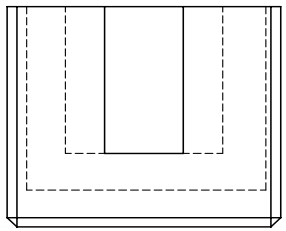
SIZE	DWG. NO.	REV
A	SL-005	
SCALE: 2:3	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1









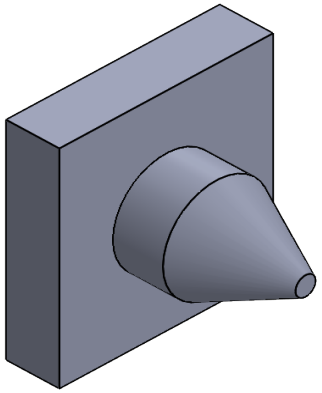
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005	DRAWN	EC	4/15/15
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL PLA Plastic	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

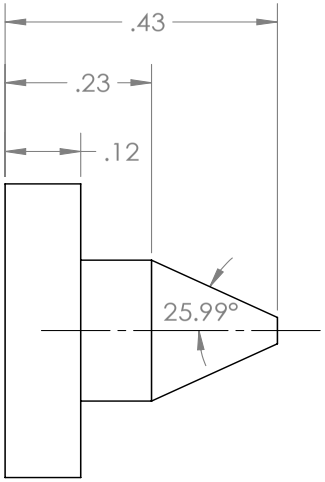
TITLE: TMD Weight Carriage

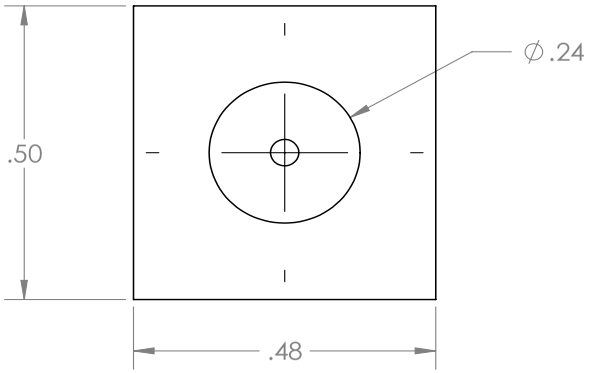
SIZE	DWG. NO.	REV
A	SL-006	

SCALE: 2:1 | WEIGHT: | SHEET 1 OF 1

5 4 3 2 1







		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005	DRAWN	EC	4/15/15
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL PLA Plastic	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

Tuned Mass Damper Spring Holder

SIZE	DWG. NO.	REV
A	SL-007	
SCALE: 4:1	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1

C.2 TMD Lab MATLAB Code

All MATLAB files used for Chapter 3 are included here. `Natural_Frequency_TMD_Plot.m` code is used to generate Figure 4.7. The plot showing the difference in frequency response between the models from Ch. 2 and Ch. 3, respectively, is made from `Building_Shaker_FRF_Analysis.m`.

```
%--------------------------------------------------------------------------
%
% Natural_Frequency_TMD_Plots.m
%
% DESCRIPTION:
% Creates plots based on max accelerations from RT Pro data during shaker
% rig test to experimentally determine the reduction in amplitude by use
% of a tuned-mass damper.
%
% INPUTS:
%
%        Excel files with acceleration data of normal building with sine
%        input compared to building with tuned-mass dampers of three
%        different materials.
%        TMD cart weighs 0.019 lb
%        Aluminum weighs 0.048 lb / with cart 0.067 lb
%        Titanium weighs 0.078 lb / with cart 0.097 lb
%        Brass weighs 0.165 lb    / with cart 0.184 lb
%
% OUTPUTS:
%
%        Plot showing decreased acceleration amplitude from using
%        tuned-mass dampers.
%
% Ethan Cating
% 4/12/2017
%--------------------------------------------------------------------------

clear all
clc
close all

% Setup for font and font size of figures
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 12)
set(0,'DefaultTextFontname', 'Times New Roman')
set(0,'DefaultTextFontSize', 12)

% Data sets
data1 = xlsread('SL-30-110.xlsx');
```

```
data2 = xlsread('SL-30-110-TMD-Al.xlsx');
data3 = xlsread('SL-30-110-TMD-SB.xlsx');
data4 = xlsread('SL-30-110-TMD-Ti.xlsx');
mass = [0, 0.067, 0.097, 0.184];

% Find the maximum amplitudes of each data set for each "story"
max1_1 = max(data1(:,2));
max1_2 = max(data1(:,3));

max2_1 = max(data2(:,2));
max2_2 = max(data2(:,3));

max3_1 = max(data3(:,2));
max3_2 = max(data3(:,3));

max4_1 = max(data4(:,2));
max4_2 = max(data4(:,3));

maxFF = [max1_1,max2_1,max3_1,max4_1];
maxSF = [max1_2,max2_2,max3_2,max4_2];

% Plot for amplitudes at each frequency for first and second story
figure(1)
set(gcf,'Units','normalized')
set(gcf,'Position',[0.25 0.1875 0.5 0.625])
set(gcf,'PaperPositionMode','manual')

plot(mass,maxFF,'--sr')
hold on
plot(mass,maxSF,'--sk')
set(gcf,'Color','w')
xlabel('Weight [lb]')
ylabel('Amplitude [g]')
xlim([0 0.205])
title({'Absolute Acceleration Amplitude of 30"-Tall Building',...
    'with Tuned-Mass Damper at Different Weights'})
set(gca,'TickDir','out')
leg2 = legend('1^{st} Floor','2^{nd} Floor','Location','NorthEast',...
    'Interpreter','latex');
tAl = text(0.067,0.1,'Al');
tTi = text(0.097,0.1,'Ti');
tSB = text(0.184,0.1,'B');
box off
```

```
%--------------------------------------------------------------------------
%
% Building_Shaker_FRF_Analysis.m
%
% DESCRIPTION:
% File calculates natural frequencies for both the two-story building and
% two-story building with a tuned-mass damper. Then generates the FRFs for
% each and plots them together into one figure. Changes in the FRFs are
```

```
% then easily identifiable.
%
% INPUTS:
%
%        Excel files with acceleration data of normal building with sine
%        input compared to building with tuned-mass dampers of three
%        different materials.
%
% OUTPUTS:
%
%        FRF plots for both the original two-story building and then the
%        two-story building with a tuned-mass damper.
%
% Ethan Cating
% 4/12/2017
%--------------------------------------------------------------------------

clear all
clc
close all

% Setup for font and font size of figures
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 12)
set(0,'DefaultTextFontname', 'Times New Roman')
set(0,'DefaultTextFontSize', 12)

% Knowns
E = 10000000;                           %  psi
I = (1)*(.125^3)/12;                    %  in^4
L = 15;                                 %  in

mfloor = 3.15/16/32.2;                  %  slug
mbeam = 3/16/32.2;                      %  slug
meq = mfloor + 2*(13/35)*mbeam;         %  slug

k = (12*E*I/(L^3))*12;                  %  lb/ft

mTMD_housing = .038/32.2;               %  slug
mTMD_cart = .019;                       %  lb
md = (0.078+mTMD_cart)/32.2;            %  slug
kd = 0.6*12;                            %  lb/ft

TMD_w = (1/(2*pi))*sqrt(2*kd/md)        % Tuned-mass damper frequency [Hz}

m1 = meq;                               % Equivalent first floor weight [slug]
m2 = meq + mTMD_housing;                % Equivalent second floor weight plus
                                        % TMD housing weight [slug]

M_1 = [meq   0;                          % Two-story Building Mass Matrix
       0   meq];

K_1 = [4*k -2*k;                        % Two-story Building Stiffness Matrix
      -2*k 2*k];
```

```
M_2 = [m1  0   0;                    % TMD Building Mass Matrix
       0   m2  0
       0   0  md];

K_2 = [4*k    -2*k      0;           % TMD Building Stiffness Matrix
      -2*k 2*(k+kd) -2*kd;
         0   -2*kd   2*kd];

[V_1,Om2_1] = eig(K_1,M_1);          % Eigenvalues and Eigenvectors

Om_1 = sqrt(Om2_1);                  % Two-story Building Natural Freqs. (rad/
   sec)

freq_1 = Om_1/(2*pi)                 % Two-story Building Natural Freqs. (Hz)

[V_2,Om2_2] = eig(K_2,M_2);          % Eigenvalues and Eigenvectors

Om_2 = sqrt(Om2_2);                  % TMD Building Natural Freqs. (rad/sec)

freq_2 = Om_2/(2*pi)                 % TMD Building Natural Freqs. (Hz)

ii = 1;                              % counter initialization

f_max = 40;                          % Maximum observed frequency (Hz)
w_max = f_max*2*pi;                  % Maximum observed frequency (rad/sec)

f_1 = [1; m2/m1];                    % Two-story Building forcing vector
f_2 = [1; m2/m1; md/m1];             % TMD Building forcing vector

for omega = 0.001:0.001:w_max        % For loop over range of forcing frequency

    % H_A & H_B is the output/input 3D vector

    % Generate 1st FRF
%     H_A = -((K_1 - (omega^2).*M_1)\(f_1)); % position frequency response
    H_A = -(omega^2)*((K_1 - (omega^2).*M_1)\(f_1)); % accel frequency
       response

    H1(1,ii) = H_A(1);                   % U_1 (X_1)
    H1(2,ii) = H_A(2);                   % U_2 (X_2)

    % Generate 2nd FRF w/ TMD
%     H_B = -(K_2 - (omega^2).*M_2)\(f_2); % position frequency response
    H_B = -(omega^2)*(K_2 - (omega^2).*M_2)\(f_2); % accel frequency response

    H2(1,ii) = H_B(1);                   % U_1 (X_1)
    H2(2,ii) = H_B(2);                   % U_2 (X_2)
    H2(3,ii) = H_B(3);                   % U_3 (X_3)

    ii = ii + 1;                         % counter

end
```

```
omega = 0.001:0.001:w_max;                % forcing frequency vector (Hz)

omega = omega/(2*pi);                     % forcing frequency vector (rad/sec)

figure(1)
set(gcf,'Units','normalized')
set(gcf,'Position',[0.25 0.1875 0.5 0.625])

subplot(2,1,1)
semilogy(omega,abs(H1(1,:)),'k')
hold on
semilogy(omega,abs(H1(2,:)),'--K')

xlim([1 f_max])
ylim([10^-4 10^8])
title('FRF of Two-Story Building')
ylabel('Accel. Amplitude Input/Output')
legend('First Floor','Second Floor','location','NorthEast')

subplot(2,1,2)
semilogy(omega,abs(H2(1,:)),'k')
hold on
semilogy(omega,abs(H2(2,:)),'--k')
semilogy(omega,abs(H2(3,:)),'-.k')

% Vertical lines for natural frequencies of base-two story building model
% semilogy([freq_1(1,1),freq_1(1,1)],[10^-12,100],'-.k')
% semilogy([freq_1(2,2),freq_1(2,2)],[10^-12,100],'-.k')

xlim([1 f_max])
ylim([10^-12 10^2])
title('FRF of Two-Story Building with Tuned-Mass Damper')
xlabel('Frequency [Hz]')
ylabel('Accel. Amplitude Input/Output')
legend('First Floor','Second Floor','TMD','location','NorthEast')
set(gcf,'color','w')
```

D Rotational Unbalance and SHM Appendix

D.1 Rotational Unbalance Arduino Code

Arduino code used to accept potentiometer input and output pulse width modulation (PWM) signal to run the motors. The switch at the top turns the entire system on with power coming from a micro USB power cable.

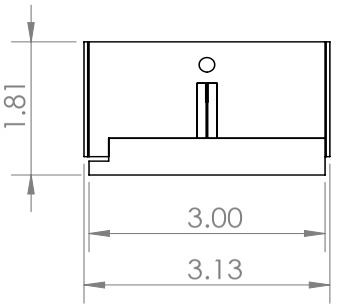
```
/**
*  Vibrational Imbalance Lab (Thesis Work)
*/

/*** Pin I/O ***/

#define PIN_ON_SWITCH 21

#define PIN_LIN_POT_LEFT 18
#define PIN_LIN_POT_RIGHT 19

#define MOTOR_PWM_1 5
#define MOTOR_PWM_2 6

float potReadingLeft = 0;
float potReadingRight = 0;
float leftMotorSpeed = 0;
float rightMotorSpeed = 0;

void setup() {

Serial.begin(9600);
pinMode(PIN_ON_SWITCH, INPUT);

}

void loop() {

potReadingLeft = analogRead(PIN_LIN_POT_LEFT);
potReadingRight = analogRead(PIN_LIN_POT_RIGHT);

leftMotorSpeed = round(potReadingLeft/1023*255);
rightMotorSpeed = round(potReadingRight/1023*255);

if( !digitalRead(PIN_ON_SWITCH) ) {
```

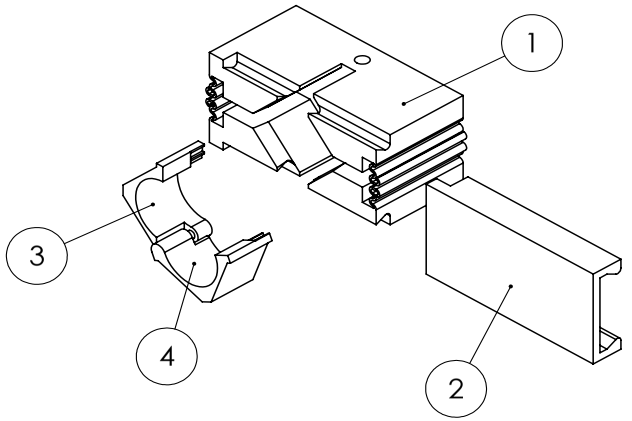
```
//  Serial.println("Left Pot Reading:");
//  Serial.println(potReadingLeft);
//  Serial.println("Right Pot Reading:");
//  Serial.println(potReadingRight);
//  Serial.println("Motor speeds:");
//  Serial.print(int(leftMotorSpeed));
//  Serial.print("   ");
//  Serial.println(int(rightMotorSpeed));

  analogWrite(MOTOR_PWM_1, int(leftMotorSpeed));
  analogWrite(MOTOR_PWM_2, int(rightMotorSpeed));

} else {

}

//delay(500);

}
```

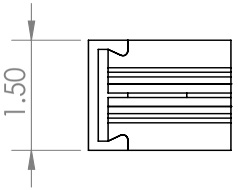
D.2 Rotational Unbalance SolidWorks Prints

Included in this appendix section are all of the drawing prints to create the motor housing parts and the Arduino control box parts for the unbalance lab.

ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	RI-010	Motor Housing	1
2	RI-011	Housing Cover	1
3	RI-013	Female Clamp Half	1
4	RI-012	Male Clamp Half	1







NEXT ASSY	USED ON	
APPLICATION		DO NOT SCALE DRAWING

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±.01
THREE PLACE DECIMAL ±.005

INTERPRET GEOMETRIC TOLERANCING PER:

MATERIAL
PLA Plastic

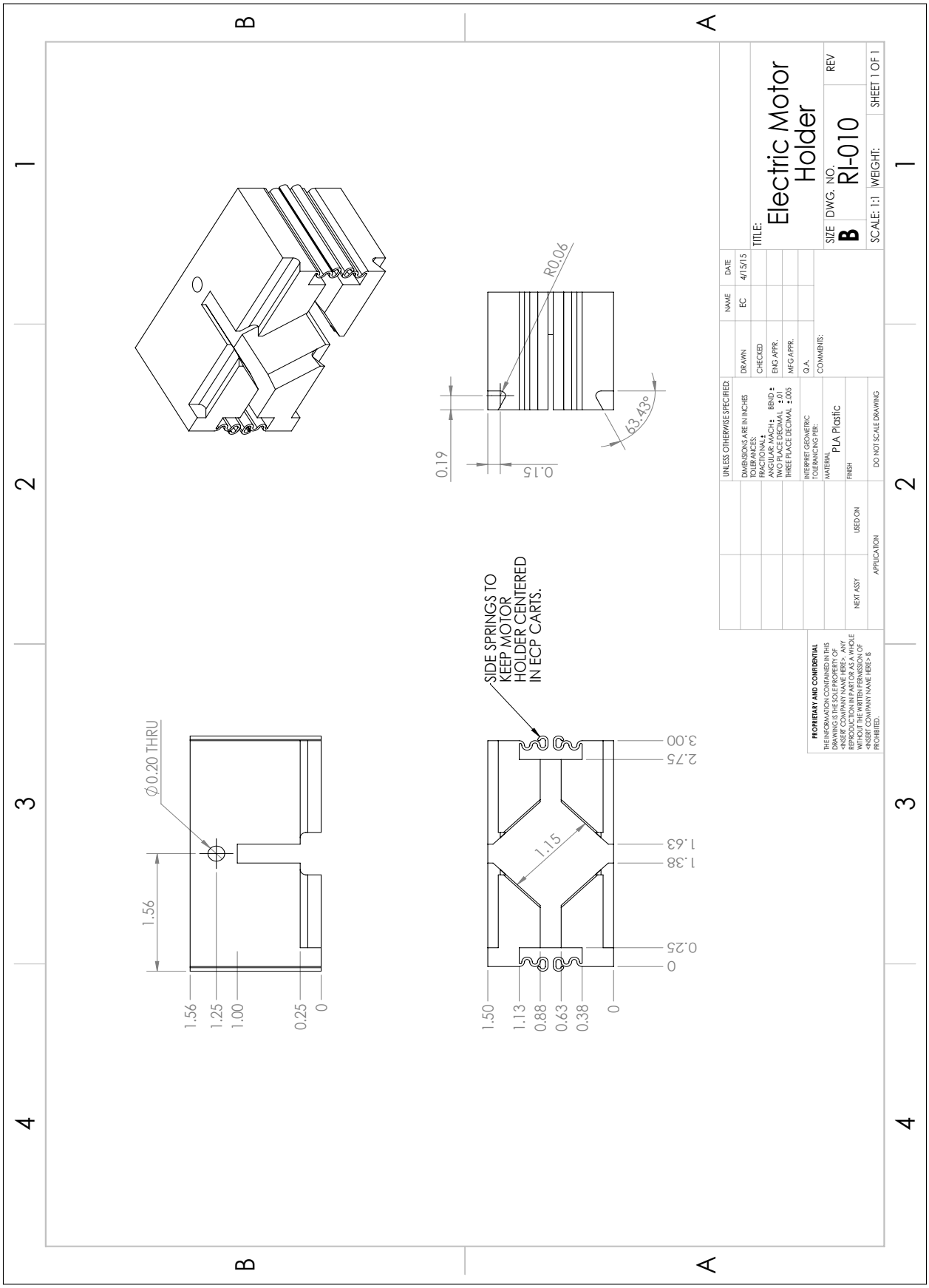
FINISH

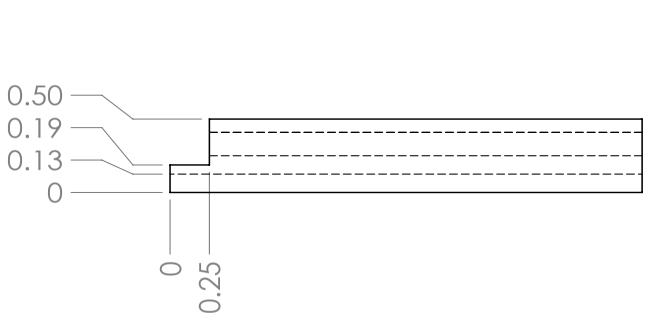
	NAME	DATE
DRAWN	EC	4/15/15
CHECKED		
ENG APPR.		
MFG APPR.		
Q.A.		

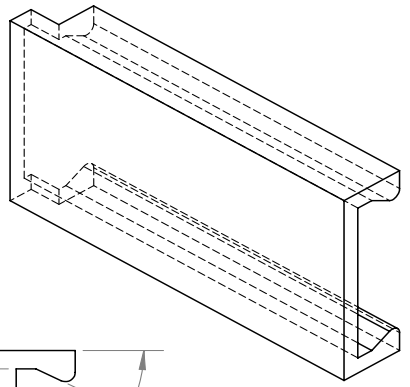
COMMENTS:

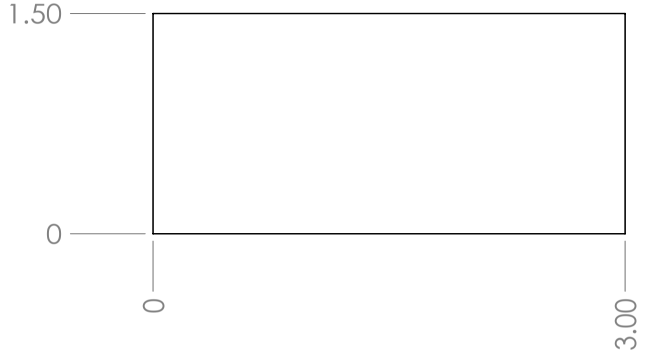
TITLE:
Motor Housing Assembly

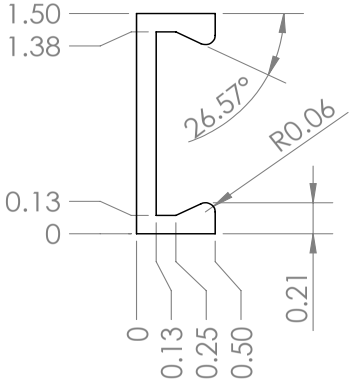
SIZE	DWG. NO.	REV
A	RI-001	
SCALE: 1:2	WEIGHT:	SHEET 1 OF 1


Electric Motor Holder
RI-010
SIZE B
SCALE: 1:1
WEIGHT:
SHEET 1 OF 1
REV
DWG. NO.
TITLE:
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
NAME
DATE
EC
4/15/15
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ± .01
THREE PLACE DECIMAL ± .005
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
PLA Plastic
FINISH
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
R0.06
0.19
0.15
63.43°
SIDE SPRINGS TO KEEP MOTOR HOLDER CENTERED IN ECP CARTS.
3.00
2.75
1.63
1.38
1.15
0.25
0
1.50
1.13
0.88
0.63
0.38
0
Ø0.20 THRU
1.56
1.56
1.25
1.00
0.25
0







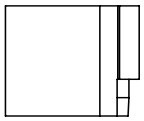


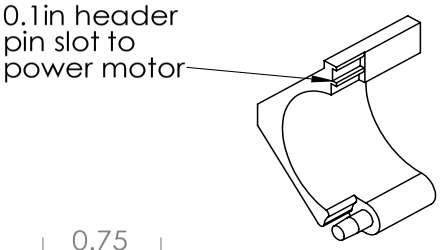
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005	DRAWN	EC	4/15/15
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL PLA Plastic	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

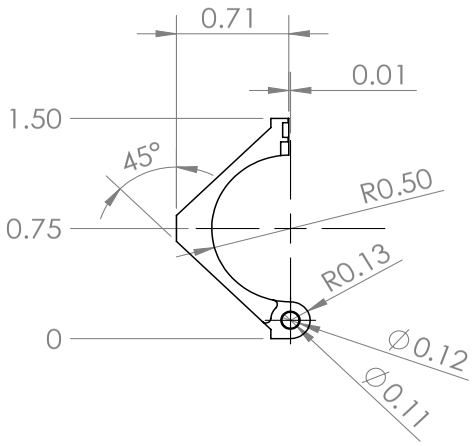
TITLE:

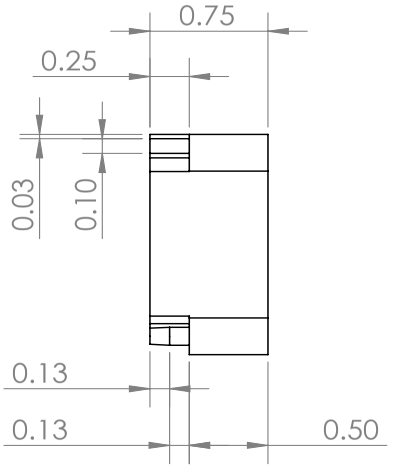
Motor Housing Cover

SIZE	DWG. NO.	REV
A	RI-011	
SCALE: 1:1	WEIGHT:	SHEET 1 OF 1









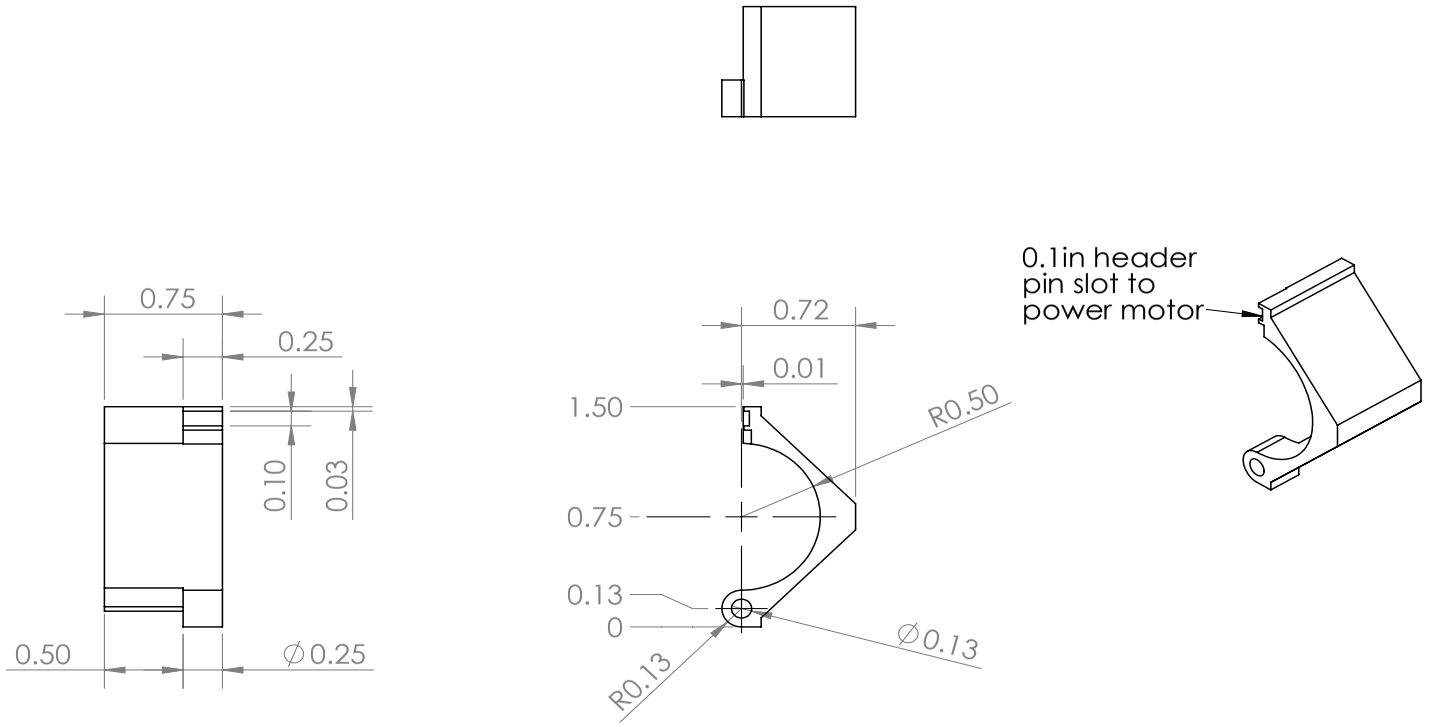
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005	DRAWN	EC	4/15/15
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL PLA Plastic	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

Motor Clamp with Pin

SIZE	DWG. NO.	REV
A	RI-012	

SCALE: 1:1	WEIGHT:	SHEET 1 OF 1



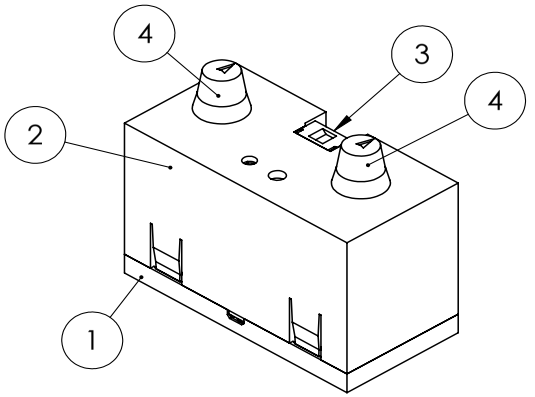
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005	DRAWN	EC	4/15/15
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL PLA Plastic	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

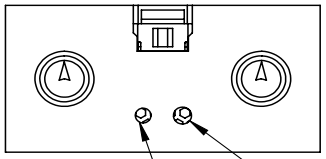
TITLE:

Motor Clamp with Hole

SIZE	DWG. NO.	REV
A	RI-013	
SCALE: 1:1	WEIGHT:	SHEET 1 OF 1

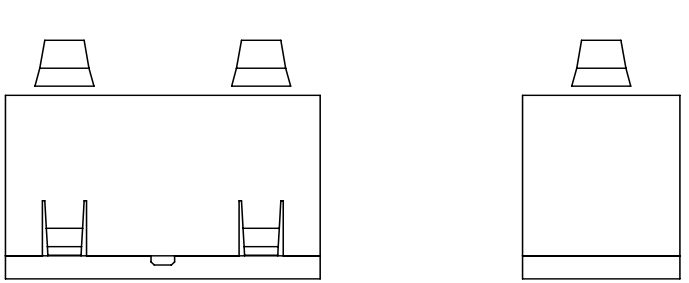
ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	RI-020	Bottom Cover	1
2	RI-021	Top Cover	1
3	RI-022	Wire Connector Holder	1
4	RI-023	Potientiometer Knobs	2





ON/OFF SWITCH HOLE

POWER INDICATOR LED

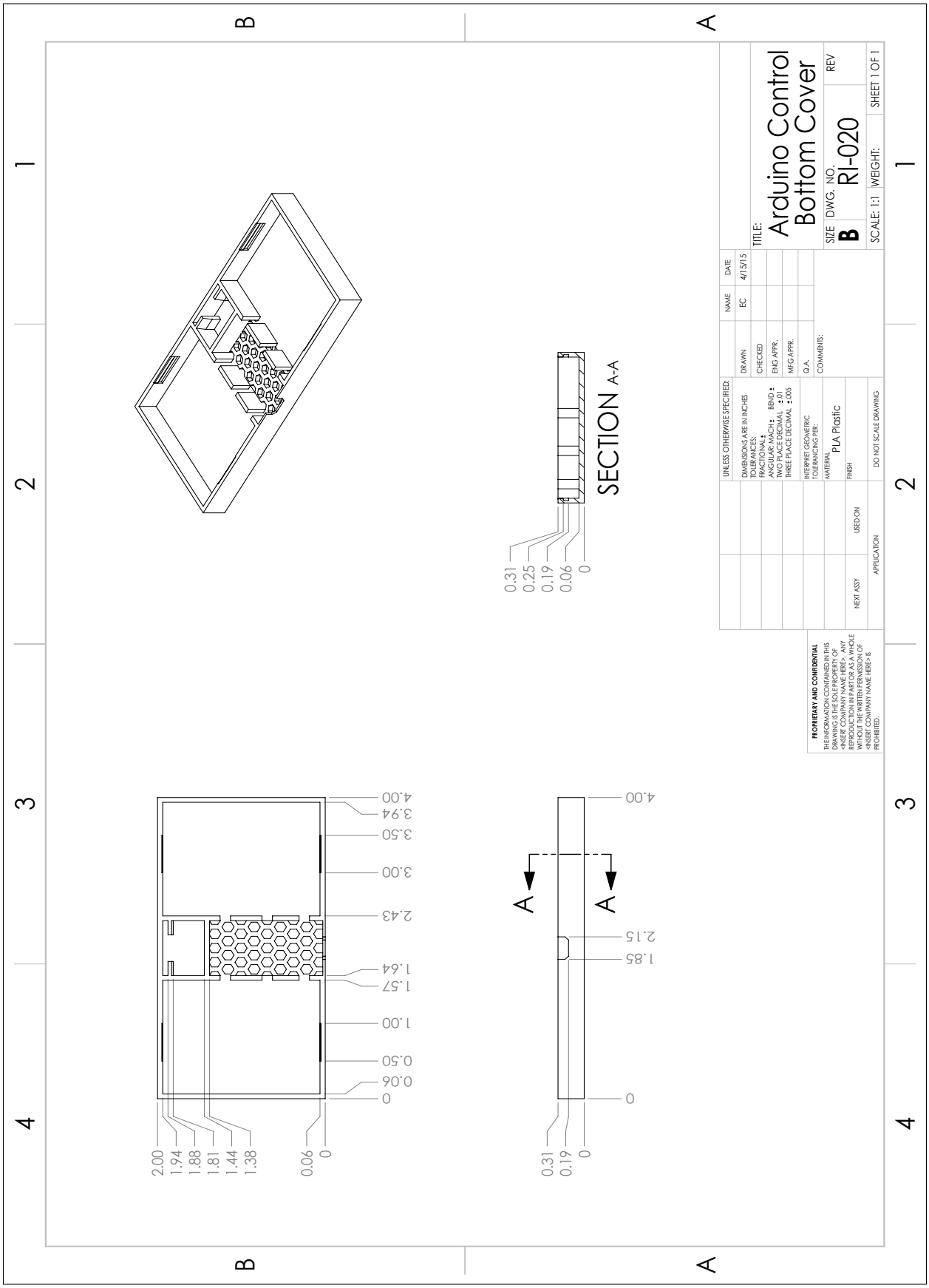


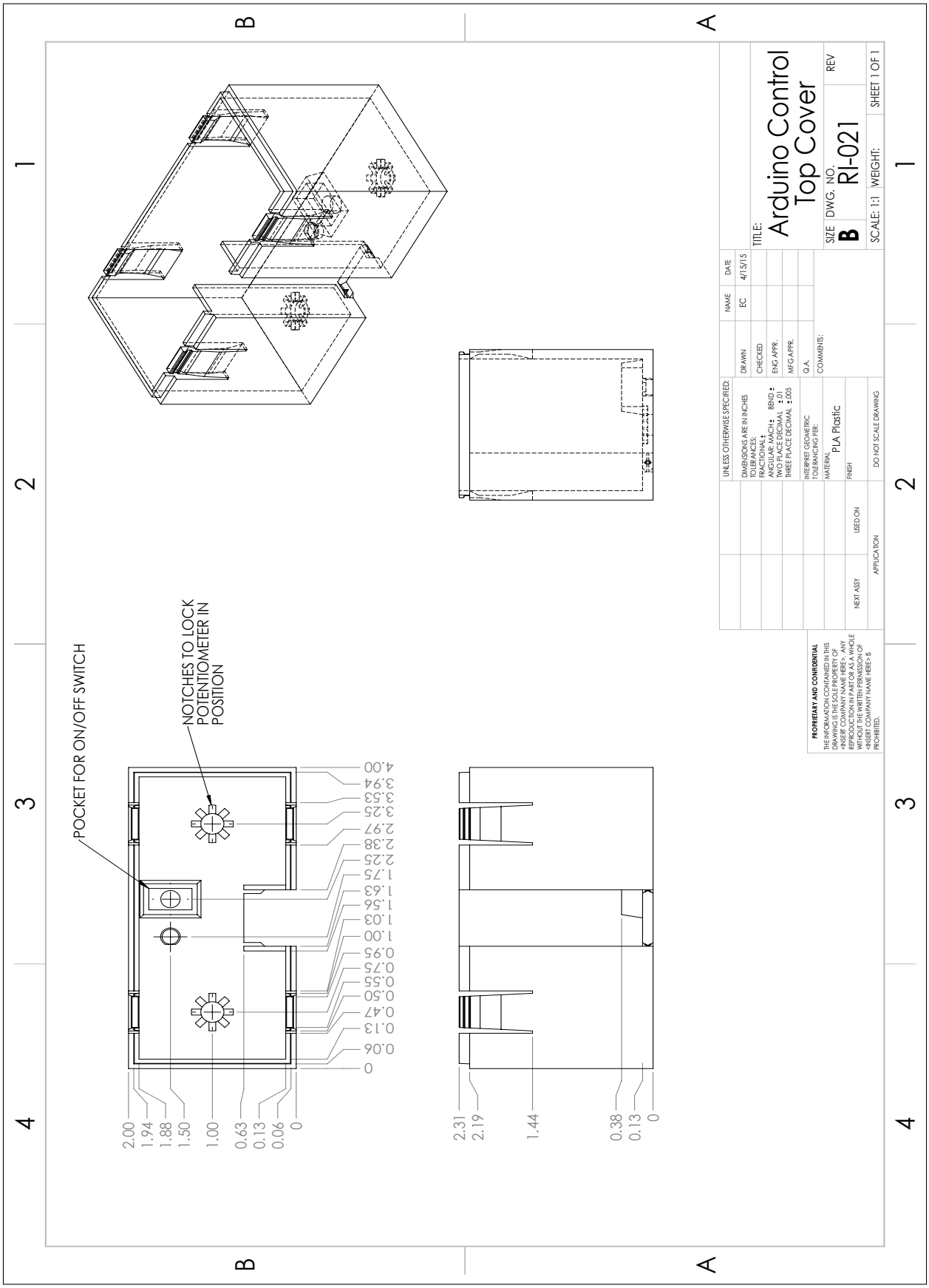
				NAME	DATE
		UNLESS OTHERWISE SPECIFIED:			
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005	DRAWN	EC	4/15/15
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL PLA Plastic	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

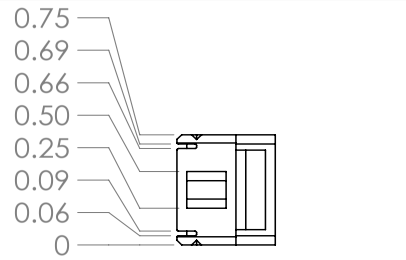
TITLE:

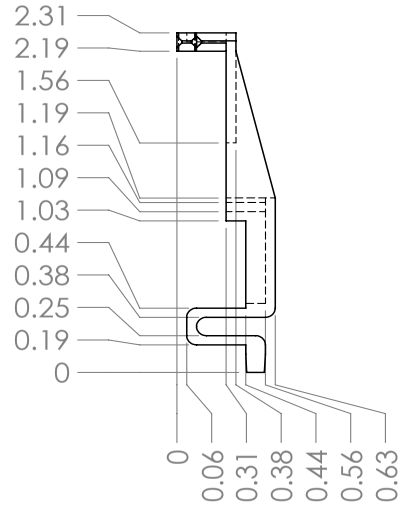
Arduino Housing & Motor Control Box

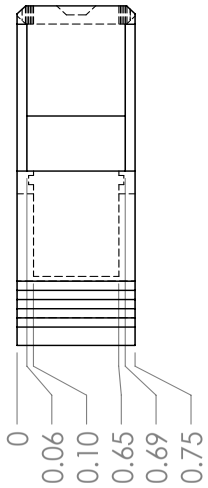
SIZE	DWG. NO.	REV
A	RI-002	
SCALE: 1:2	WEIGHT:	SHEET 1 OF 1

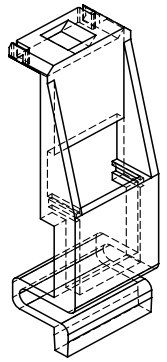

SECTION A-A
0.31
0.25
0.19
0.06
0
Arduino Control
Bottom Cover
TITLE:
SIZE B
DWG. NO. RI-020
REV
SCALE: 1:1
WEIGHT:
SHEET 1 OF 1
UNLESS OTHERWISE SPECIFIED:
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
NAME
DATE
EC
4/15/15
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±.01
THREE PLACE DECIMAL ±.005
INTERPRET GEOMETRIC
TOLERANCING PER:
MATERIAL
PLA Plastic
FINISH
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS
DRAWING IS THE SOLE PROPERTY OF
<INSERT COMPANY NAME HERE>. ANY
REPRODUCTION IN PART OR AS A WHOLE
WITHOUT THE WRITTEN PERMISSION OF
<INSERT COMPANY NAME HERE> IS
PROHIBITED.
A
B
1
2
3
4
2.00
1.94
1.88
1.81
1.44
1.38
0.06
0
4.00
3.94
3.50
3.00
2.43
1.64
1.57
1.00
0.50
0.06
0
0.31
0.19
0
2.15
1.85


POCKET FOR ON/OFF SWITCH
NOTCHES TO LOCK POTENTIOMETER IN POSITION
2.00
1.94
1.88
1.50
1.00
0.63
0.13
0.06
0
4.00
3.94
3.53
3.25
2.97
2.38
2.25
1.75
1.63
1.56
1.03
1.00
0.95
0.75
0.55
0.50
0.47
0.13
0.06
0
2.31
2.19
1.44
0.38
0.13
0
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±.01
THREE PLACE DECIMAL ±.005
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
PLA Plastic
FINISH
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
NAME
EC
DATE
4/15/15
TITLE:
Arduino Control Top Cover
SIZE
B
DWG. NO.
RI-021
REV
SCALE: 1:1
WEIGHT:
SHEET 1 OF 1
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.







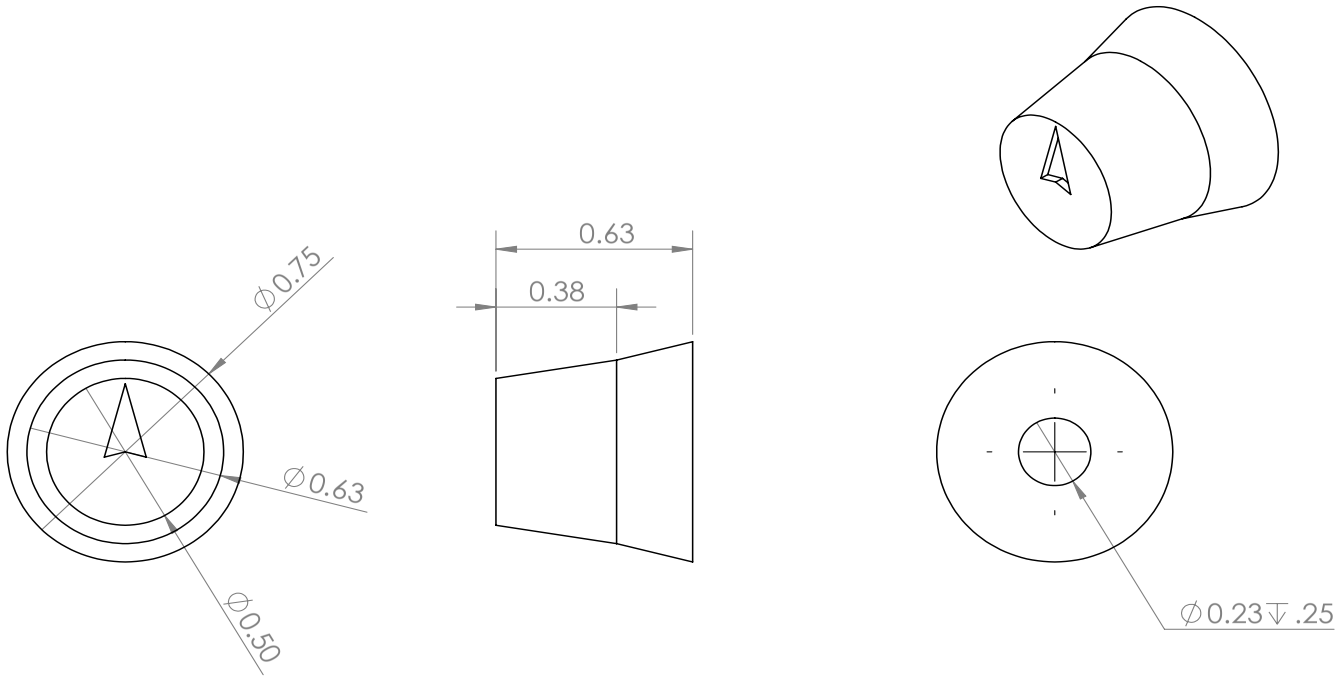


		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005	DRAWN	EC	4/15/15
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL PLA Plastic	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

Box to Motor Connector Clip

SIZE	DWG. NO.	REV
A	RI-022	
SCALE: 1:1	WEIGHT:	SHEET 1 OF 1



		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005	DRAWN	EC	4/15/15
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL PLA Plastic	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:
Potientiometer Covers

SIZE	DWG. NO.	REV
A	RI-021	
SCALE: 2:1	WEIGHT:	SHEET 1 OF 1

D.3 Rotational Unbalance Lab MATLAB Code

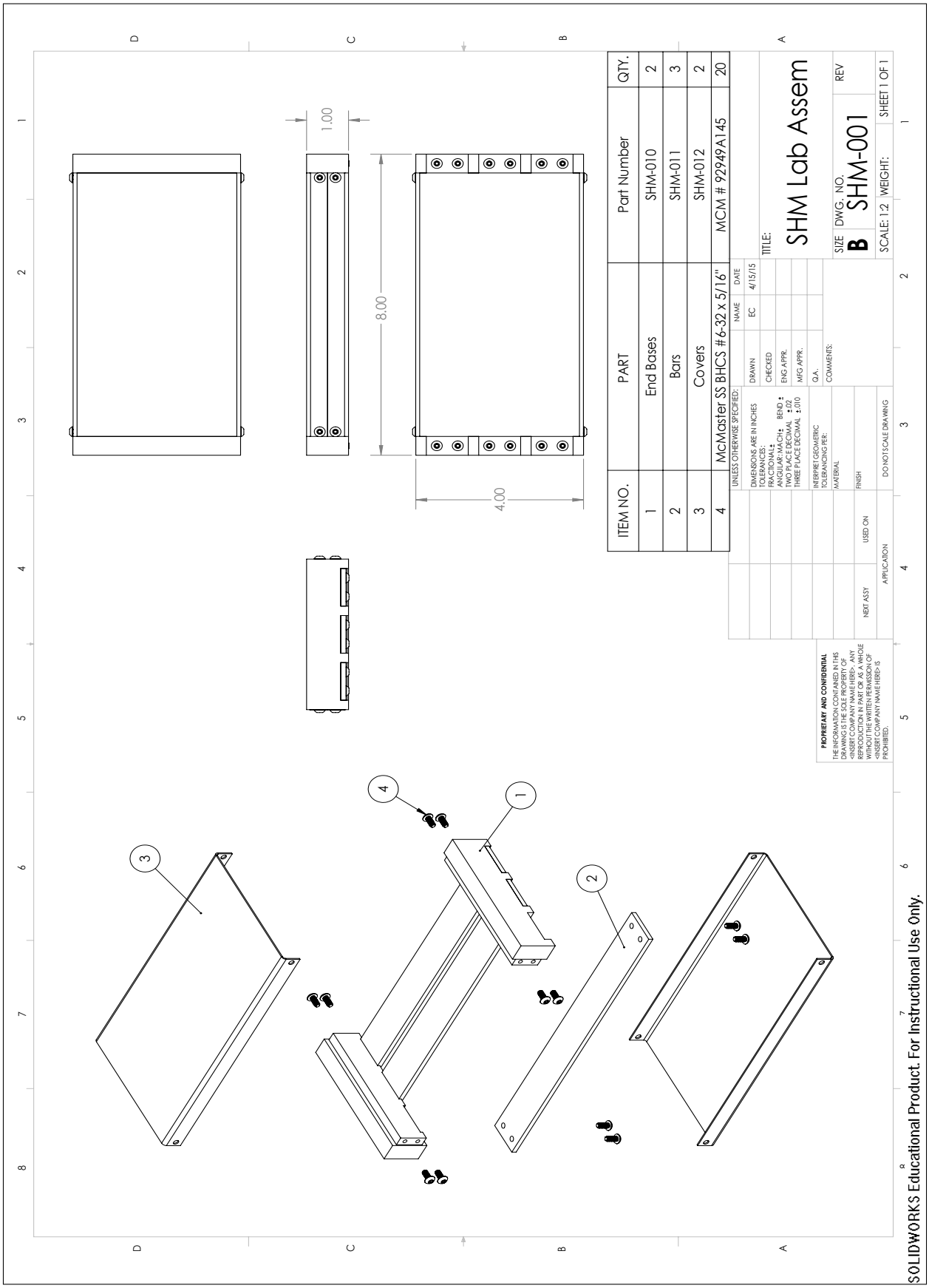
This section includes the only MATLAB file used for Section 5.1, Rotational Unbalance. `Rot_Imb.m` is used to generate Figures 5.6 & 5.7.

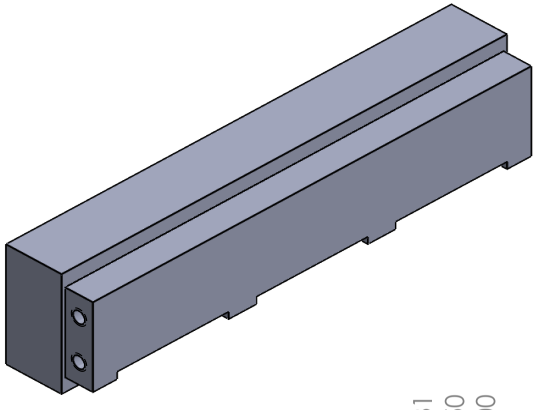
```
%--------------------------------------------------------------------------
%
% Rot_Imb.m
%
% DESCRIPTION:
% Plots positional data from vibration motor on ECP during free response
% run.
%
% INPUTS:
%
%       t - time vector
%       p - position data vector
%       s - calculated sine wave
%
% OUTPUTS:
%
%       Plot of experimental data
%       X_S - calculated max amplitude for small mass motor
%       X_L - calculated max amplitude for large mass motor
%
% Ethan Cating
% 4/12/2017
%--------------------------------------------------------------------------

% clear workspace
clear all
clc
close all

% Setup for font and font size of figures
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 12)
set(0,'DefaultTextFontname', 'Times New Roman')
set(0,'DefaultTextFontSize', 12)

% Mass of ECP rectilinear cart
cart = 500;       % grams

% Small mass motor data at full speed
data1 = load('Imbalance_Small_Motor_Fast_Speed.mat');

t1 = data1.data(1,:);             % time data [s]
```

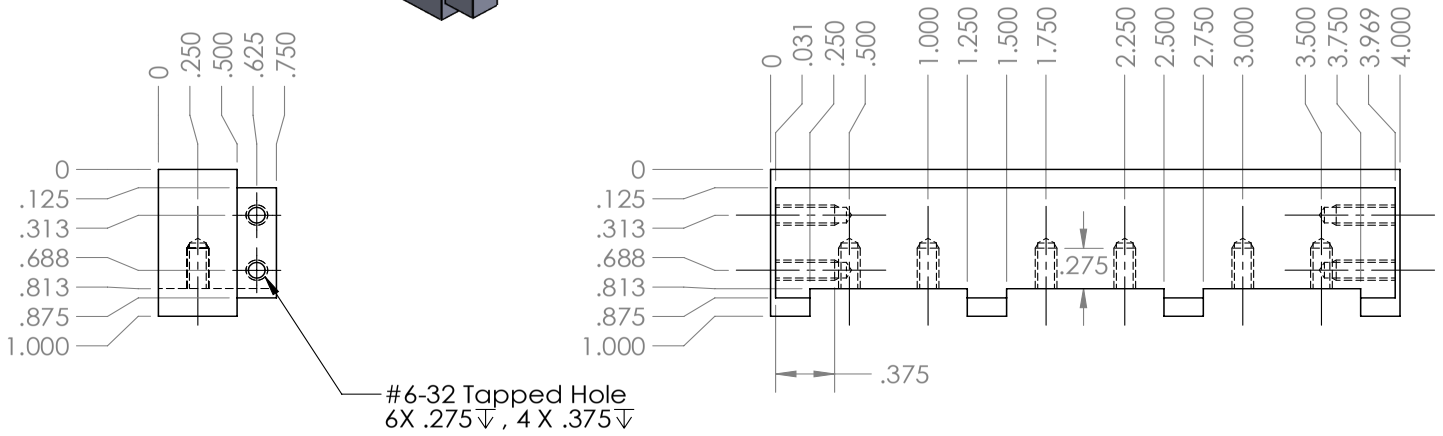
```
p1 = data1.data(2,:) + .0025;     % position data [cm]

s1 = 0.0018*sin(2*pi*95*t1 - 45*pi/180); % comparative sine wave

S_mass       = 2;           % small offset mass [g]
S_housing    = 65;          % mass of small mass motor and housing [g]
S_radius     = .2285*2.54;  % small mass radius [cm]

% Calculated max amplitude for small mass [cm]
X_S = ((4/(3*pi))*(S_mass*S_radius))/(cart + S_housing);

% Large mass motor data at full speed
data2 = load('Imbalance_Large_Motor_Fast_Speed.mat');

t2 = data2.data(1,:);               % time data [s]
p2 = data2.data(2,:) - .0035;       % position data [cm]

s2 = 0.0065*sin(2*pi*50*t2 - 120*pi/180); % comparative sine wave

L_mass = 8;                 % large offset mass [g]
L_housing = 61;             % mass of large mass motor and housing [g]
L_radius = .4065* 2.54;     % large mass radius [cm]

% Calculated max amplitude for small mass [cm]
X_L = ((4/(3*pi))*(L_mass*L_radius))/(cart + L_housing);

% Small mass motor vibration data plot
figure(1)
plot(t1,p1,'k')
hold on
% plot(t1,s1,'r')
xlim([0,0.1])
set(gcf,'Color','w')
xlabel('Time [s]')
ylabel('Position [cm]')
title('Sine Wave Created from Small Vibration Motor')

% Large mass motor vibration data plot
figure(2)
plot(t2,p2,'k')
hold on
% plot(t2,s2,'r')
xlim([0,0.1])
set(gcf,'Color','w')
% legend('Actual Data','Generated Sine Wave','Location','NorthEast')
xlabel('Time [s]')
ylabel('Position [cm]')
title('Sine Wave Created from Large Vibration Motor')
```

D.4 SHM Lab SolidWorks Prints

Included in this appendix section are all of the drawing prints to create the parts for the Structural Health Monitoring lab black box.


1.00
8.00
4.00
ITEM NO. | PART | Part Number | QTY.
1 | End Bases | SHM-010 | 2
2 | Bars | SHM-011 | 3
3 | Covers | SHM-012 | 2
4 | McMaster SS BHCS #6-32 x 5/16" | MCM # 92949A145 | 20
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±.02
THREE PLACE DECIMAL ±.010
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
DO NOT SCALE DRAWING
NAME | DATE
DRAWN | EC | 4/15/15
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
NEXT ASSY
USED ON
APPLICATION
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
TITLE:
SHM Lab Assem
SIZE B
DWG. NO. SHM-001
REV
SCALE: 1:2
WEIGHT:
SHEET 1 OF 1
SOLIDWORKS Educational Product. For Instructional Use Only.





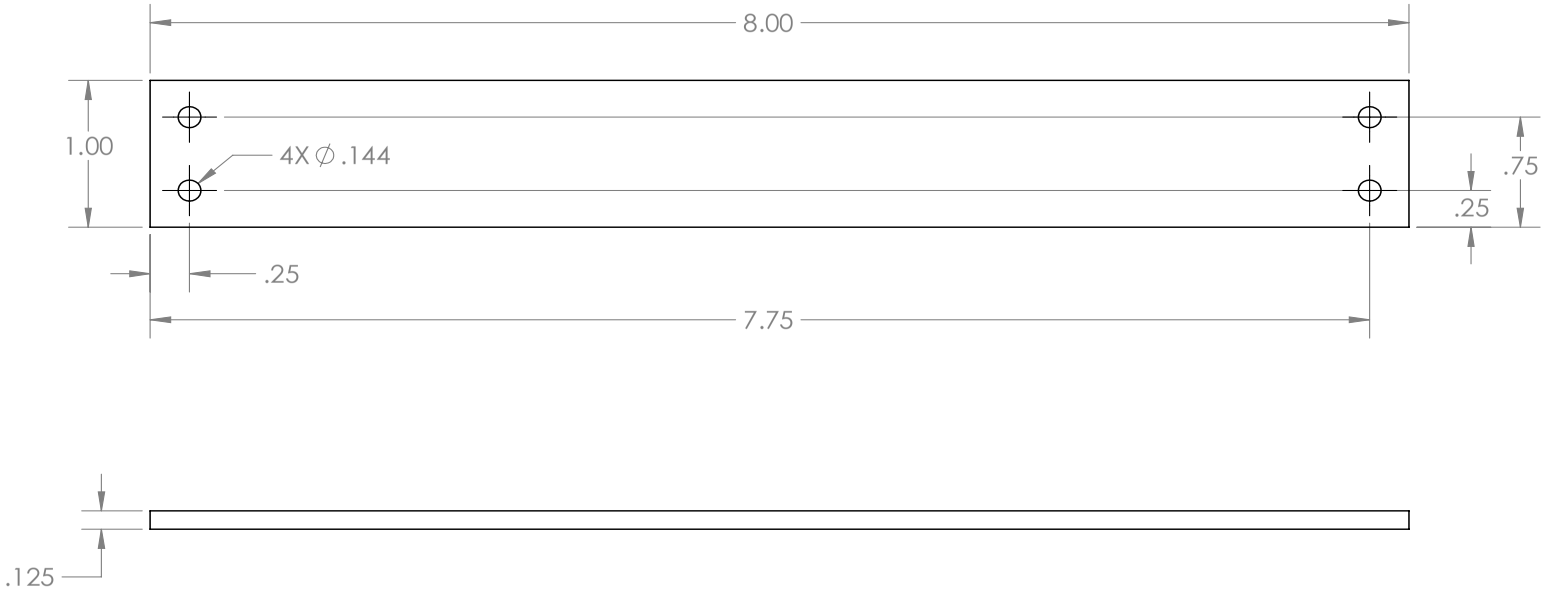
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005	DRAWN	EC	4/15/15
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL 6061 Aluminum	COMMENTS: Break all edges		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

End bases for SHM lab

SIZE	DWG. NO.	REV
A	SHM-010	B
SCALE: 1:1	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1



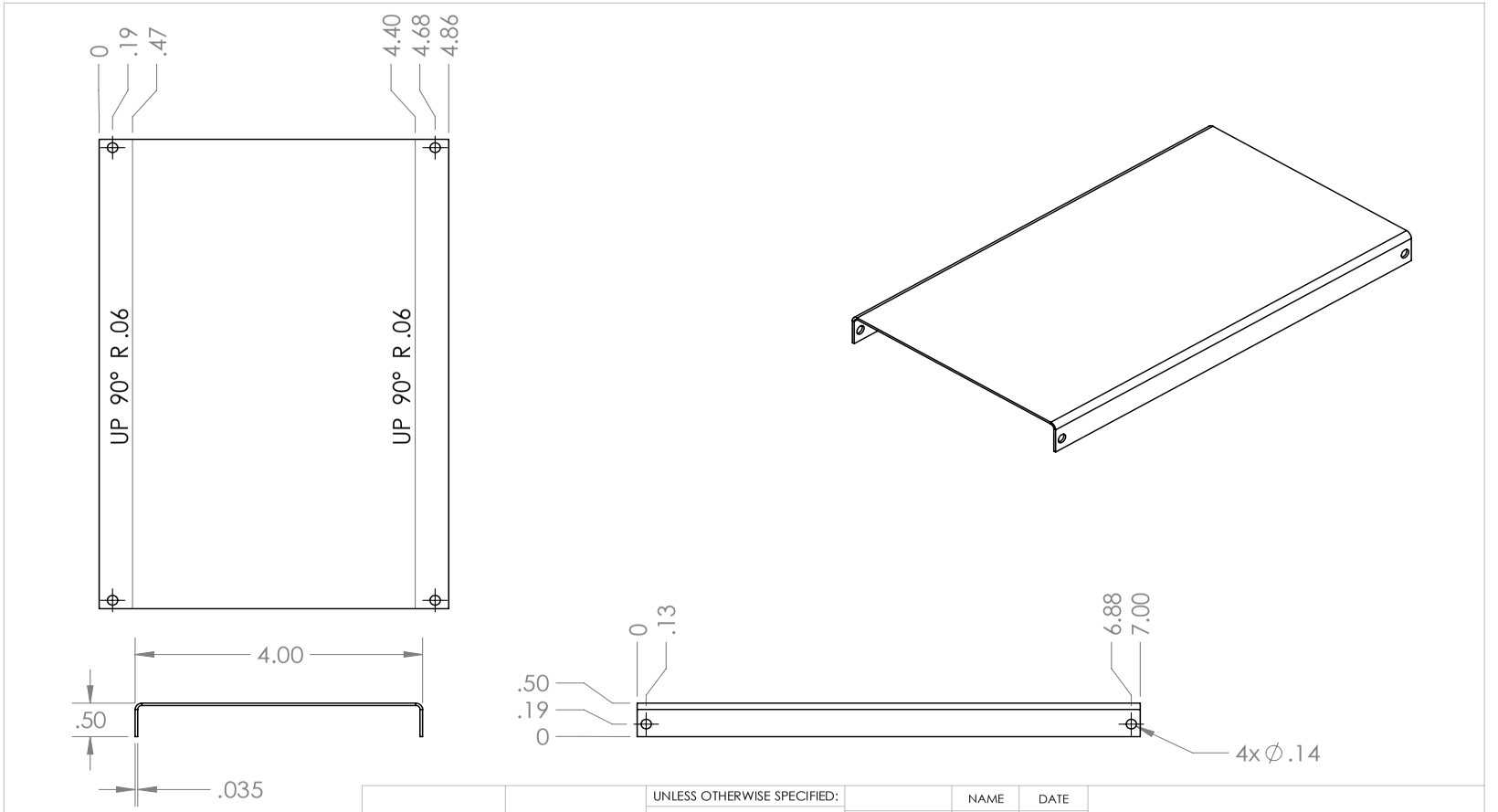
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005	DRAWN	EC	4/15/15
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL 6061 Aluminum	COMMENTS: Break all edges		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

Bar for SHM lab

SIZE	DWG. NO.	REV
A	SHM-011	B
SCALE: 1:1	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1



		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±.01 THREE PLACE DECIMAL ±.005	DRAWN	EC	4/15/15
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL 6061 Aluminum	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

Sheet Metal Covers

SIZE	DWG. NO.	REV
A	SHM-012	
SCALE: 1:2	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1

D.5 SHM Lab MATLAB Code

All MATLAB files used for Section 5.2, SHM, are included here. The code `save_uff_to_mat.m` is used to convert the data used for `plot_uff.m`. The conversion code is based on an online downloadable function file `read_uff.m` [7] that is able to open UFF (Universal File Format) text files saved from RT Pro.

```
% function save_uff_to_mat
% Use H2_2

%=========================================================================
%
%  Save the FRF data after importing the relevant information from the
%  corresponding UFF file.
%
%  INPUTS:      ** UFF file containing FRF and frequency data.
%
%  OUTPUTS:     ** FRF and frequency data saved to a .mat file.
%
%  NOTE:    The actual reading of the UFF file is done by the function
%           read_uff, which is a modified version of readuff.  With
%           read_uff, the UFF file does not need to be in the same
%           directory as the function.
%
%=========================================================================

% Clear workspace
clear all
clc
close all

% Identify current directory and save to curDirName variable:

curDirName = pwd;

%  Select a UFF file containing FRF data to plot:

[uffFileName,uffPathName] = uigetfile({'*.uff', 'UFF files (*.uff)'}, ...
                                      ['Select a UFF file with ', ...
                                       'FRF data.']);

%  Import the FRF and frequency data:

[UffDataSets,Info,errmsg] = read_uff(uffFileName, uffPathName, curDirName);
```

```
freq = (UffDataSets{1}.x)';
FRF = UffDataSets{1}.measData;

%  Save time and position data to .mat file to use in plot_uff
uisave({'FRF','freq'})

%==========================================================================
```

```
% function plot_uff

%==========================================================================
%
%  Plot the FRF magnitude after importing the relevant information from the
%  corresponding .mat file from original UFF file.
%
%  INPUTS:      ** .mat files from converted UFF files
%
%  OUTPUTS:     ** Plot of the FRF magnitude.
%
%==========================================================================

% clear workspace
clc
clear all
close all

% Setup for font and font size of figures
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 12)
set(0,'DefaultTextFontname', 'Times New Roman')
set(0,'DefaultTextFontSize', 12)

% Load in the .mat data created from UFF files

A = load('Normal_Center.mat');

B = load('Left_Damped_Left.mat');

C = load('Mid_Empty_Center.mat');

D = load('All_Empty_Center.mat');

%  Plot the FRF magnitude:

figure('name', 'FRF');

set(gcf, 'color', 'w')

subplot(4,1,1)
semilogy(A.freq, abs(A.FRF), '-b', 'linewidth', 1)
xlim([0 1000])
ylim([0.01 1*10^5])
```

```
ylabel('Magnitude (gn/lbf)')
title('Default Configuration')

subplot(4,1,2)
semilogy(B.freq, abs(B.FRF), '-b', 'linewidth', 1)
xlim([0 1000])
ylim([0.01 1*10^5])
ylabel('Magnitude (gn/lbf)')
title('Left Bar with Electrical Tape')

subplot(4,1,3)
semilogy(C.freq, abs(C.FRF), '-b', 'linewidth', 1)
xlim([0 1000])
ylim([0.01 1*10^5])
ylabel('Magnitude (gn/lbf)')
title('Middle Bar Removed')

subplot(4,1,4)
semilogy(D.freq, abs(D.FRF), '-b', 'linewidth', 1)
xlim([0 1000])
ylim([0.01 1*10^5])
ylabel('Magnitude (gn/lbf)')
title('All Bars Removed')
xlabel('Frequency (Hz)')

%=========================================================================
```