Viscosity Dependence of Faraday Wave Formation Thresholds

A Senior Thesis

presented to

The Department of Physics

California Polytechnic State University, San Luis Obispo

Lisa Michelle Slaughter

Advised by Dr. Nilgun Sungar

December 14, 2013

I. Abstract

This experiment uses an electromagnetic shaker to produce standing wave patterns on the surface of a vertically oscillating sample of silicon liquid. These surface waves, known as Faraday waves, form shapes such as squares, lines, and hexagons. They are known to be dependent upon the frequency and amplitude of the forcing as well as on the viscosity and depth of the liquid in the dish. At a depth of 4mm and for various silicon liquids having kinematic viscosities of 10, 20, and 38 cSt, we determined the acceleration at which patterns form for frequencies between 10 and 60 Hz. For the 10 and 20 cSt cases, it was found that the acceleration at which these patterns form does not heavily depend upon the viscosity of their host liquid. This is in stark contrast with the current body of knowledge. The case for 38 cSt, however, varies from the other two cases in both value and shape.

II. Introduction and Theory

If you have ever set a wineglass to singing and noticed ripples on the surface of the liquid within, you have experienced the phenomenon of Faraday waves. Named for Michael Faraday, who first studied them in 1831[1], these standing waves can form stripes, squares, and even hexagons. Stripes occur when a single, continuous wave travels across the surface of the liquid, forming long crests. Ocean waves along the shore provide a decent visualization of these. A mathematical construct, called a k-vector, describes the direction in which the wave is travelling and points perpendicular to the stripes it forms. Squares, seen in Figure 1, are formed when the k-vectors of two waves cross perpendicular to each other. An example of hexagons is shown in Figure 2. These occur when three k-vectors cross with 120° between them. Patterns

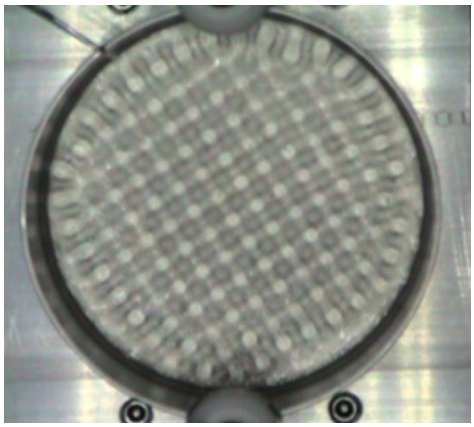

Figure 1. 20cSt, 52Hz. Pure standing squares. This shape is created by two sets of waves crossing at 90° of each other.

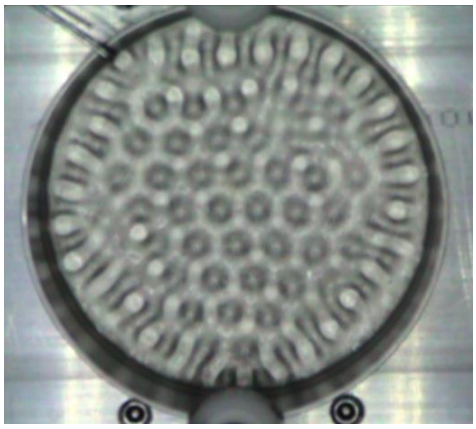

Figure 2. 10cSt, 44Hz. Pure standing hexagons. This shape is formed when three waves cross at 120° of each other.

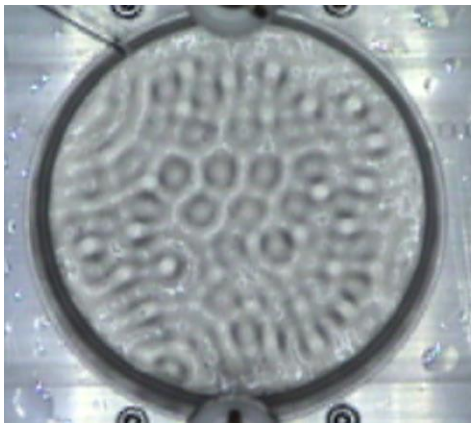

Figure 3. 20 cSt, 36 Hz. A screenshot of the chaotic, time-dependent combination of lines, squares, and hexagons.

Figure 4. 20cSt, 24 Hz. A second instance of chaotic, time-dependent combination under different circumstances.

do not, however, always form arrays that consist purely of these shapes. More often than not, a combination occurs, as shown in Figures 3 and 4. These are the cases of chaos, and are completely unpredictable. They are time-dependent, causing, for example, a square to appear where once there was a hexagon. The shapes are in competition for the same space, and they appear to melt into each other.

Faraday waves are known to be dependent upon the frequency and amplitude of the forcing waveform as well as on the viscosity and depth of the liquid in the dish. [2] This experiment aims to contribute to the body of knowledge on these waves by investigating the dependence of their formation threshold upon the viscosity of their host liquid. Using an electromagnetic shaker, patterns are produced on the surface of the vertically oscillating sample of silicon liquid. The viscosity of this liquid is varied and the threshold of pattern formation measured as a function of the frequency and maximum acceleration of the forcing.

III. Experimental Design

A block diagram of the experimental apparatus used in this experiment is shown in Figure 5. An ED-3 Electrodynamic Shaker was used to vertically oscillate a dish containing silicon oil of known viscosity at a depth of 4mm. It operates by using an electrified coil to move a magnet at a frequency and amplitude specified by the user through the Agilent 33220A Signal Generator. Proper operation of the shaker requires that a load of 2-5 pounds be applied. Since our sample and the dish containing it weigh only a few grams, a symmetrical aluminum block was machined and fastened to the plate of the shaker to achieve the needed load. To ensure that the oscillations were purely one-dimensional, the load was balanced through an optical

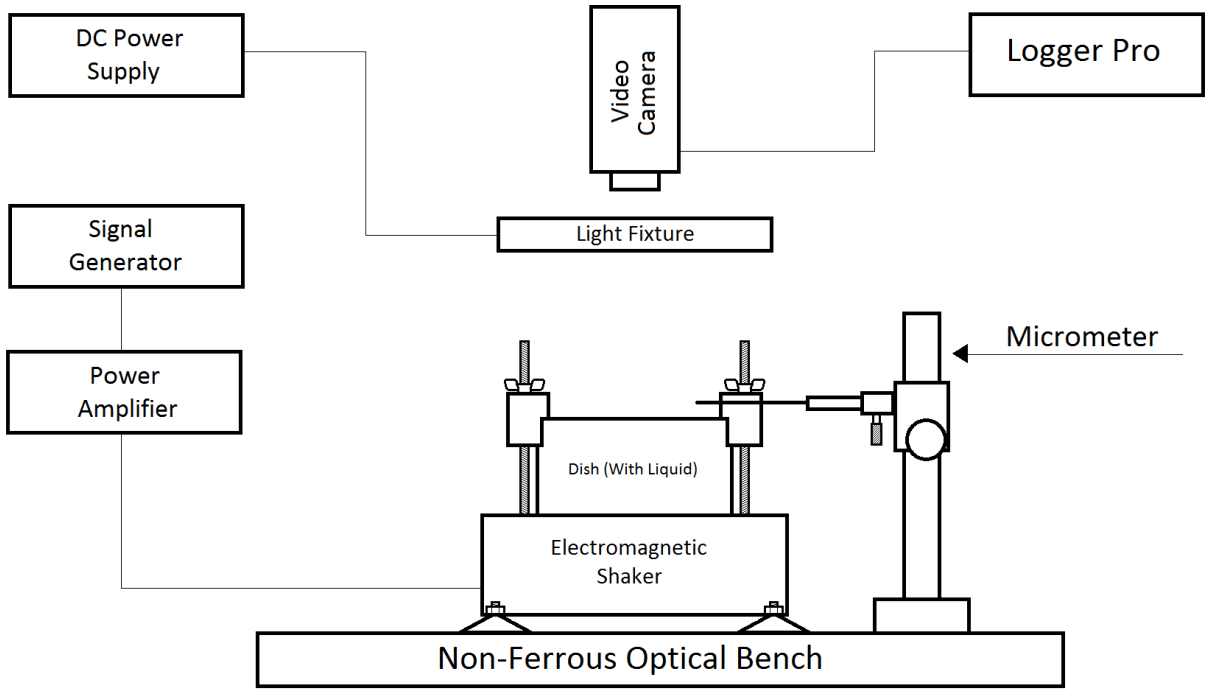

Figure 5. Diagram of experimental apparatus used to obtain data.

method. Two lasers, aligned as perfectly as possible, were reflected off the smooth surface of the custom-fabricated aluminum block onto a wall. The shaker was then sinusoidally driven and the motion of the laser points observed. If the laser point motion was not identical, the balance of the plate and shaker was painstakingly adjusted until the motion of the laser points coincided.

The motion of the shaker was defined using an Agilent 33220A Signal Generator, which offered control over the frequency, amplitude, and shape of the oscillations. Since the signal generator does not provide enough power to drive the shaker on its own, its output was fed through an intermediary power amplifier before reaching the shaker. This device was ordered along with the ED-3 and operates on factory-provided settings. An investigation of the shaker's response to changes in amplitude was performed and it was found that use of the amplifier's amplitude control knob resulted in a non-linear amplitude response in the shaker. Thus, it was necessary to maintain static amplifier settings for all measurements and instead use the signal generator controls to alter the original waveform amplitude.

The amplitude of the waveform is defined in terms of voltage by the signal generator. For these experiments, however, the physical amplitude of motion must be known in millimeters. To obtain this measurement, a micrometer was used. The zero mark on the micrometer was calibrated to correspond to the upper edge of the dish in equilibrium position. Then, while the signal was applied, the micrometer arm was lowered to the point that it could be heard barely

tapping against the dish edge. The number displayed was thus the amplitude of the upper half of the sinusoidal forcing oscillation. The micrometer and shaker were both fastened to a non-ferrous optical bench to provide stability and vibration isolation. It was important that no magnetic materials be used in the apparatus, as the motion of the shaker uses magnetism to operate and its sinusoidal nature could be affected.

A video camera was used to image the patterns forming on the liquid surface. Placed directly above the dish, its line of sight fell perpendicular to the surface of the liquid which ensured that no parallax occurred. Videos were acquired by Logger Pro, a popular data acquisition program. This program offers a feature that allows the user to define a physical scale against which distance measurements can be made. To this end, a ruler was placed next to the shaker so that such a scale is provided with each video. Still images can also be taken from video so that other analysis, such as a fast Fourier transform, can be performed.

To properly image the surface waves, they must be lit by bright, uniform lighting from directly above. Since the camera must also be mounted directly above the sample, a custom lighting system was constructed in which eighteen light-emitting diodes are arranged on a double-layer balsawood ring through which the camera could be aimed. These LEDs are incredibly bright and provide wonderfully uniform illumination. The higher the ring is placed above the system, the more uniform the lighting and the less apparent the reflection of the LEDs off the liquid's surface. For this experiment, the ring and camera were placed approximately two feet above the test surface.

The samples under investigation are silicon oils having manufacturer determined kinematic viscosities of 10, 20, and 38 cSt. For reference, water has a kinematic viscosity of 1 cSt. The depth of these liquids was maintained at 4mm for all experimentation.

IV. Improvements

After collecting the data presented herein, some improvements were made upon the apparatus in an attempt to remedy certain issues. These are shown in Figure 6. Unfortunately, there was not time enough to obtain new data utilizing said improvements.

During experimentation, inconsistent amplitude measurements were observed and the accuracy of our methodology for obtaining these measurements fell under scrutiny. In an attempt to remedy these inconsistencies, a more detailed method was adopted. Rather than obtaining a single measurement of the upper portion of the waveform, multiple measurements

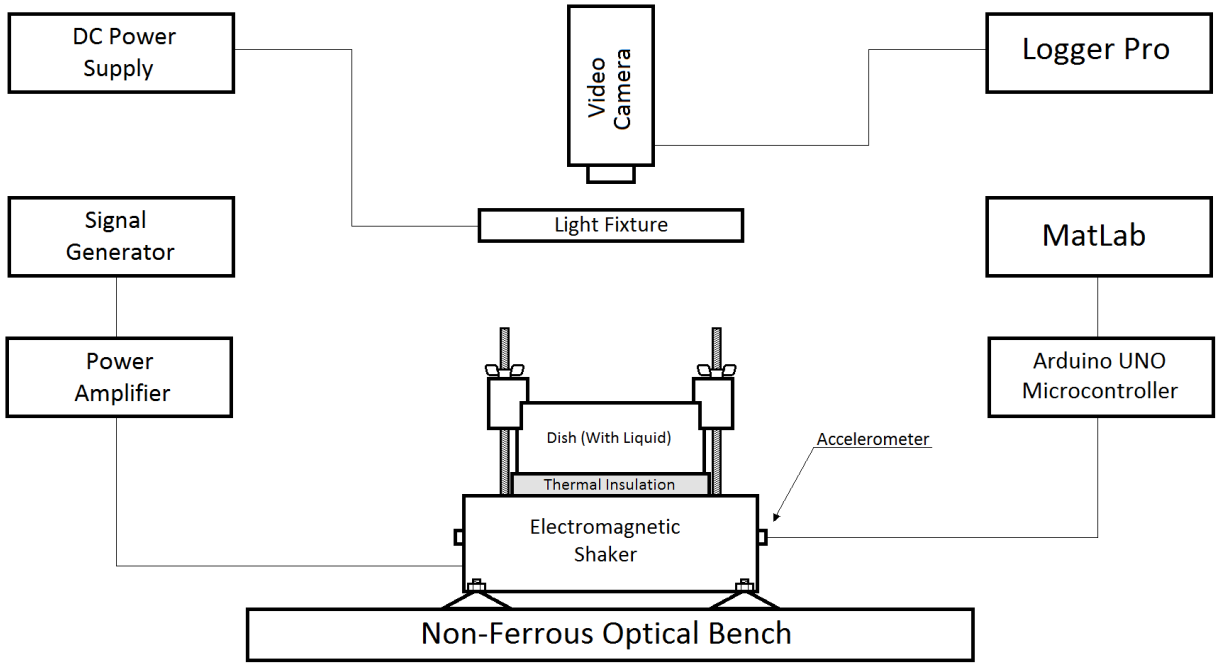

Figure 6. Experimental apparatus, with improvements. Thermal insulation was added between the shaker and sample as well as the accelerometers and their associated data acquisition system.

were taken for the upper half as well as the lower half and averaged, resulting in two amplitudes. A second average was then taken between these two values for a final result. Unfortunately, this did not solve the problem. It was decided that direct measurements of acceleration could be obtained digitally using accelerometers. By placing one of these sensors on each of the four corners of the shaker head, the phase of motion could also be compared and imbalances investigated. The analog accelerometer output was read by the Arduino UNO programmable microcontroller and converted to a 10 bit representation (a number between 0 and 1023). The Arduino then sent is information to the serial stream, where it was retrieved by a Matlab program written to convert it into acceleration units and perform some basic analysis. This code is provided in Appendix A and that for analyzing phase is in Appendix B.

A second improvement upon the apparatus came in the form of thermal insulation placed between the shaker head and the dish containing the silicon liquid. During operation, eddy currents would cause the aluminum load block to heat up. In the higher ranges of frequency and amplitude, this effect was strong enough to heat the block to scalding temperatures. It was observed that the additional heat energy was causing patterns to form at a lower threshold, which badly skewed the results obtained. Until a more suitable material could be found to use for this purpose, a flat piece of wood with uniform thickness kept wet with water

had to suffice. It is expected that ceramic would provide adequate temperature isolation without deforming in response to changes in acceleration.

V. Results

For silicon fluids having kinematic viscosities of 10, 20, and 38 cSt, we systematically increased the amplitude and frequency of the driving waveform. These values were recorded at the point where patterns began to form. The recorded amplitude was then used to calculate the maximum acceleration achieved by the sample during oscillations according to

$$\ddot{x} = A(2\pi f)^2$$

where A denotes amplitude, f frequency and $\ddot{x}$ acceleration in SI units. The acceleration was then converted to be represented in terms of earth's gravitational acceleration, g. For each of the three viscosities, these threshold values can be seen in Figure 7 as a function of frequency and acceleration. Interestingly, there appears to be no significant dependence of pattern formation threshold upon the viscosity of the liquid for 10 and 20 cSt viscosities. This is in stark opposition to the results found in a similar study by A. Kudrolli and J.P. Gollub [2], which showed that formation thresholds are heavily dependent upon viscosity. This discrepancy could be due to the fact that the data for each viscosity was taken in an environment that was not temperature controlled. Since each data set was taken at different times of the year, the temperature of the liquid may have differed by many degrees. As was previously mentioned, higher temperatures cause formation thresholds to occur at lower accelerations, thus skewing results. The threshold plot for 38 cSt varies from the other two in both shape and in value. Not only do the patterns

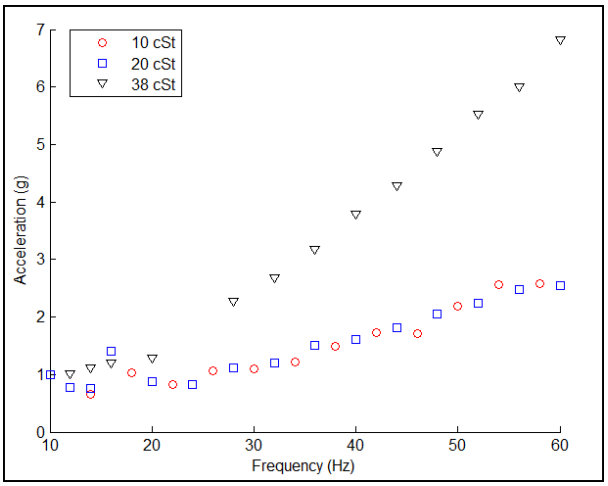

Figure 7. Pattern formation thresholds for 10, 20, and 38 cSt as a function of forcing frequency and acceleration. Note that the thresholds of 10 and 20 cSt are similar.

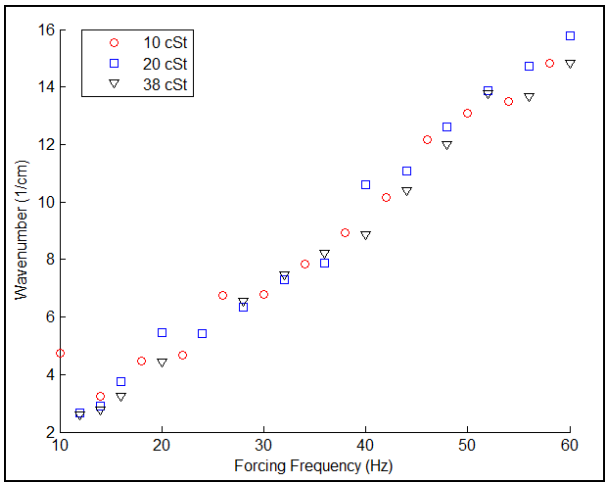

Figure 8. The relationship between pattern wavenumber and forcing frequency. As was expected from experimental observation and theory, this relationship is linear.

form at a greater acceleration overall than do the 10 and 20 cSt cases, but as frequency increases, the disparity in value also increases.

The size of the patterns was observed to decrease for increases in frequency. This is apparent from the comparison between the patterns pictured in Figure 3, taken at 36 Hz, and Figure 4, taken at 24 Hz. For waves in general, the k-vector magnitude, called the wavenumber, is inversely proportional to the wavelength. In the context of the patterns studied herein, wavelength can be found by measuring the separation between peaks. The wavenumber is thus proportional to the inverse of the peak separation. This value was obtained for the patterns formed at the investigated frequencies. The relationship between wavenumber and frequency, shown in Figure 8, was found to be inversely linear, as was expected from previous observation.

VI. Conclusions

In this experiment, silicon liquids having kinematic viscosities of 10, 20, and 38 cSt were oscillated in the vertical direction. It was found that the plots of threshold accelerations for the 10 and 20 cSt liquids were similar in shape and value, whereas the plot for the 38 cSt liquid differed in both ways. The result for 10 and 20 cSt viscosities opposes the findings of Kudrolli and Gollub by suggesting that, in some cases at least, the formation threshold does not depend upon viscosity. Further investigation of this system is expected, mainly to study the case for liquid having a 5 cSt viscosity. It was also found that the relationship between peak separation and frequency is inversely linear. This was expected due to previous qualitative observation.

In the future, we expect to investigate the phenomenon of stochastic resonance, where noise added to the signal affects formation thresholds. It is expected that the addition of purely random, non-periodic noise would lower the amplitude of formation. Additional future plans involve applying a complex waveform to the shaker, specifically the superposition of two sine waves with systematically applied differences in phase, wavelength, and amplitude. Other studies have shown that this regime provides for a richer variety of patterns than were found herein. Once this system is sufficiently well-understood, the test sample will graduate to something granular, such as sand, cornstarch, or steel shot.

References

1. T.B. Benjamin, F. Ursell. “The stability of the plane free surface of a liquid in vertical periodic motion.” *Proceedings of the Royal Society of London* 225 1954: 505-515.
2. A. Kudrolli, J.P. Gollub. “Patterns and spatiotemporal chaos in parametrically forced surface waves: a systematic survey at large aspect ratio.” *Physica D* 97 1996: 133-154.

Appendix A – Accelerometer Data Acquisition Code

MatLab code for accelerometer data acquisition:

```
% Accelerometer data acquisition file:
% By Lisa Michelle Slaughter, physics
% California Polytechnic State University, SLO
% Summer 2013
% Public Domain
% To be used with Arduino code 'AccelDataAcquisition.ino'
%
% **Ensure that the same COM port is being used here as the Arduino
% **Ensure that baud rates also match (max: 115200)
%
% This code pulls data from the serial port and arranges it into data
% vectors, then manipulates those vectors to provide an average
% acceleration for each sensor as well as an average acceleration for the
% whole sensor array.
% Output is in the form:
%   accel #1     accel #2
%   accel #4     accel #3
%   Average among the four
%
% This is in accordance with the way the accelerometers are arranged on the
% shaker. Accelerometer #1 is closest to the shaker's signal input.
%
% If you get an error saying that your COM port is unavailable, type
% "delete(instrfindall)" in the command line to free it up.

%% ------------------Data Acquisition------------------
clear all
close all
clc

cal1 = 489.9234;  % These points correspond to rest value for y-axis (-1g)
cal2 = 490.0310;  % To redetermine these values, run the very last cell of
cal3 = 489.0094;  % this script.
cal4 = 489.0868;

delete(instrfindall)   % open the com port
arduino = serial('COM4','BaudRate',115200);   % create serial object
fopen(arduino);   % open serial object
[temp,count] = fscanf(arduino); % get the value of numpts
if count ~= 0
    predata = textscan(temp,'%u'); % convert to cell
    numpts = cell2mat(predata);   % de-cell into vector.
else
    disp('No values read');
end

data = ones(numpts*4,2);

% Since instability in the serial stream is inherent, checks must be
% performed. Erroneous readings must be detected and rejected.
```

```
for i = 1:numpts*4
    [temp,count] = fscanf(arduino); % get the data
    if count ~= 0
        commas = 0;  % initialize comma count
        for j=1:length(temp)     % count the commas
            if temp(j) == ','
                commas = commas + 1;
            end
        end
    else
        disp('No values read');
    end
    if commas == 3
        predata = textscan(temp,',%u,%u,'); % convert to cell
        data(i,:) = cell2mat(predata);   % de-cell into vector.
    else
        continue;
    end
end
fclose(arduino);   % close serial object
delete(instrfindall);   % free up com port

% move data around
t1 = data(1:numpts,1);
t2 = data(numpts+1:numpts*2,1);
t3 = data(numpts*2+1:numpts*3,1);
t4 = data(numpts*3+1:end,1);

acc1 = data(1:numpts,2)+(510.3927-cal1);
acc2 = data(numpts+1:numpts*2,2)+(510.3927-cal2);
acc3 = data(numpts*2+1:numpts*3,2)+(510.3927-cal3);
acc4 = data(numpts*3+1:end,2)+(510.3927-cal4);
disp(['Sampling Rate = ' num2str(numpts*1000/max(t1)) ' Hz']);

% Conversion   (converts to g's. Comment out for re-calibration)
acc1 = (acc1-510.3927)/(510.3927-cal1);
acc2 = (acc2-510.3927)/(510.3927-cal2);
acc3 = (acc3-510.3927)/(510.3927-cal3);
acc4 = (acc4-510.3927)/(510.3927-cal4);

% Find peak acceleration (raw method)
[pks1,locs1] = findpeaks(acc1,'MINPEAKHEIGHT',0.15);
[pks2,locs2] = findpeaks(acc2,'MINPEAKHEIGHT',0.15);
[pks3,locs3] = findpeaks(acc3,'MINPEAKHEIGHT',0.15);
[pks4,locs4] = findpeaks(acc4,'MINPEAKHEIGHT',0.15);

% Take average of peaks and display
pkAcc1 = mean(pks1);
pkAcc2 = mean(pks2);
pkAcc3 = mean(pks3);
pkAcc4 = mean(pks4);
pkvec = [pkAcc1 pkAcc2 pkAcc3 pkAcc4];
FullAvg = mean(pkvec);
disp(['    ' num2str(pkAcc1) ' g      ' num2str(pkAcc2) ' g'])
disp(['    ' num2str(pkAcc4) ' g      ' num2str(pkAcc3) ' g'])
disp(['Average Peak Acceleration (raw): ' num2str(FullAvg) ' g '])
```

```
%%
% Plot accelerometer readouts in one tiled figure
figure(2)
subplot(2,2,1)
plot(t1,acc1,'k-');
title('Accelerometer #1');
ylabel('Acceleration (m/s^2)');
xlabel('Time (ms)');
%axis([0 5000 -3 3]);
subplot(2,2,2)
plot(t2,acc2,'k-');
title('Accelerometer #2');
ylabel('Acceleration (m/s^2)');
xlabel('Time (ms)');
%axis([0 5000 -3 3]);
subplot(2,2,4)
plot(t3,acc3,'k-');
title('Accelerometer #3');
ylabel('Acceleration (m/s^2)');
xlabel('Time (ms)');
%axis([0 5000 -3 3]);
subplot(2,2,3)
plot(t4,acc4,'k-');
title('Accelerometer #4');
ylabel('Acceleration (m/s^2)');
xlabel('Time (ms)');
%axis([0 5000 -3 3]);

%% Find peak acceleration (fit method)
sine = @(A,W,P,D,x) A*sin(2*pi*W*x+P)+D;
[fit1,gof1,info1]= fit(t1/1000,acc1,sine,'StartPoint',[4 70 0 .2]);
[fit2,gof2,info2]= fit(t2/1000,acc2,sine,'StartPoint',[4 70 0 .2]);
[fit3,gof3,info3]= fit(t3/1000,acc3,sine,'StartPoint',[4 70 0 .2]);
[fit4,gof4,info4]= fit(t4/1000,acc4,sine,'StartPoint',[4 70 0 .2]);

% get and display pertinent data
sigma1 = confint(fit1,.6827);
sigma2 = confint(fit2,.6827);
sigma3 = confint(fit3,.6827);
sigma4 = confint(fit4,.6827);
coeffs1 = coeffvalues(fit1);
coeffs2 = coeffvalues(fit2);
coeffs3 = coeffvalues(fit3);
coeffs4 = coeffvalues(fit4);
fitPkAcc1 = coeffs1(1);
fitPkAcc2 = coeffs2(1);
fitPkAcc3 = coeffs3(1);
fitPkAcc4 = coeffs4(1);
fitFullAvg = mean(abs([fitPkAcc1 fitPkAcc2 fitPkAcc3 fitPkAcc4]));
disp(['   ' num2str(abs(fitPkAcc1)) ' g      ' num2str(abs(fitPkAcc2)) ' g'])
disp(['   ' num2str(abs(fitPkAcc4)) ' g      ' num2str(abs(fitPkAcc3)) ' g'])
disp(['Average Peak Acceleration (fit): ' num2str(fitFullAvg) ' g '])

% Plot fit against data
figure(3)
hold on
```

```
plot(t1/1000,acc1,'b-');
plot(fit1,'k-');
hold off
figure(4)
hold on
plot(t2/1000,acc2,'b-');
plot(fit2,'k-');
hold off
figure(5)
hold on
plot(t3/1000,acc3,'b-');
plot(fit3,'k-');
hold off
figure(6)
hold on
plot(t4/1000,acc4,'b-');
plot(fit4,'k-');
hold off

%% To redetermine the cal1, cal2, cal3, and cal4 values, run this code!
clear all
close all
clc

delete(instrfindall)   % open the com port
arduino = serial('COM4','BaudRate',115200);   % create serial object
fopen(arduino);   % open serial object
[temp,count] = fscanf(arduino); % get the value of numpts
if count ~= 0
    predata = textscan(temp,'%u'); % convert to cell
    numpts = cell2mat(predata);   % de-cell into vector.
else
    disp('No values read');
end

data = ones(numpts*4,2);

% Since instability in the serial stream is inherent, checks must be
% performed. Erroneous readings must be detected and rejected.

for i = 1:numpts*4
    [temp,count] = fscanf(arduino); % get the data
    if count ~= 0
        commas = 0;  % initialize comma count
        for j=1:length(temp)     % count the commas
            if temp(j) == ','
                commas = commas + 1;
            end
        end
    else
        disp('No values read');
    end
    if commas == 3
        predata = textscan(temp,',%u,%u,'); % convert to cell
        data(i,:) = cell2mat(predata);   % de-cell into vector.
    else
        continue;
```

```
    end
end
fclose(arduino);    % close serial object
delete(instrfindall);    % free up com port

% move data around
acc1 = data(1:numpts,2);
acc2 = data(numpts+1:numpts*2,2);
acc3 = data(numpts*2+1:numpts*3,2);
acc4 = data(numpts*3+1:end,2);

% take average of each data vector
avg1 = mean(acc1);
avg2 = mean(acc2);
avg3 = mean(acc3);
avg4 = mean(acc4);

% display averages
disp(['cal1= ' num2str(avg1) '     cal2= ' num2str(avg2)]);
disp(['cal4= ' num2str(avg4) '     cal3= ' num2str(avg3)]);
disp('       ');
```

Arduino code for accelerometer data acquisition:

```
/* Accelerometer data acquisition file:
  By Lisa Michelle Slaughter, physics
  California Polytechnic State University, SLO
  Summer 2013
  Public domain
  For use with any voltage output accelerometer (0-5V)

  instructions:
  1) Select the appropriate COM port from tools -> Serial Port
  2) Ensure that 'AccelDataAcquisition.m' is also using this port
  3) Plug in Arduino board
  4) Click the 'upload' button above (round with an arrow)
  5) Run 'AccelDataAcquisition.m'
 **If you want more points of data, change 'numpts' below.
*/

int numpts = 3000;
int t = 0;
int dat1 = 0;
int dat2 = 0;
```

```
int dat3 = 0;
int dat4 = 0;
int time0 = 0;

void setup()
{
  Serial.begin(115200);
  Serial.println(numpts);
  time0 = millis();
  for(int i = 0; i < numpts; i++)
  {
    //dat1 = i;
    dat1 = analogRead(A1);
    t = millis()-time0;
    Serial.print(',');
    Serial.print(t);
    Serial.print(",");
    Serial.print(dat1);
    Serial.println(',');
  }
    time0 = millis();
    for(int i = 0; i < numpts; i++)
  {
    //dat2 = i;
    dat2 = analogRead(A2);
    t = millis()-time0;
    Serial.print(',');
    Serial.print(t);
    Serial.print(",");
    Serial.print(dat2);
    Serial.println(',');
  }
    time0 = millis();
    for(int i = 0; i < numpts; i++)
  {
    //dat3 = i;
    dat3 = analogRead(A3);
    t = millis()-time0;
```

```
    Serial.print(',');
    Serial.print(t);
    Serial.print(",");
    Serial.print(dat3);
    Serial.println(',');
  }
    time0 = millis();
    for(int i = 0; i < numpts; i++)
  {
    //dat4 = i;
    dat4 = analogRead(A4);
    t = millis()-time0;
    Serial.print(',');
    Serial.print(t);
    Serial.print(",");
    Serial.print(dat4);
    Serial.println(',');
  }
}

void loop()
{
}
```

Appendix B: Phase Analysis Code

MatLab code for phase analysis:

```
% Phase relationship analysis file:
% By Lisa Michelle Slaughter, physics
% California Polytechnic State University, SLO
% Summer 2013
% Public domain
% To be used with Arduino code 'PhaseRelationship.ino'
%
% **Ensure that the same COM port is being used here as the Arduino
% **Ensure that baud rates also match (max: 115200)
%
% Use this code to look at the phase relationship between each corner of
% the shaker.
%% ------------------Data Acquisition-----------------
clear all
close all
clc
numpts = 100;   % How many points would you like to capture?
cal1 = 489.9234;  % Taken from 'AccelerometerAcquisition.m'
cal2 = 490.031;
cal3 = 489.0094;
cal4 = 489.0868;

data = ones(numpts,5);
delete(instrfindall)   % open the com port
arduino = serial('COM4','BaudRate',115200);   % create serial object
fopen(arduino);   % open serial object
for i = 1:numpts
    [temp,count] = fscanf(arduino); % get the data
    if count ~= 0
        commas = 0;
        for j=1:length(temp)     % count the commas
            if temp(j) == ','
                commas = commas + 1;
            end
        end
    else
        disp('No values read');
    end
    if commas == 4
         predata = textscan(temp,'%u,%u,%u,%u,%u'); % convert to cell
         data(i,:) = cell2mat(predata);   % de-cell into vector.
    else
        continue;
    end
end
fclose(arduino);   % close serial object
delete(instrfindall);   % free up com port
t = data(:,1);
acc1 = data(:,2)+(510.3927-cal1);
acc2 = data(:,3)+(510.3927-cal2);
acc3 = data(:,4)+(510.3927-cal3);
```

```
acc4 = data(:,5)+(510.3927-cal4);
disp(['Sampling Rate = ' num2str(numpts*1000/max(t)) ' Hz']);

% Conversion
acc1 = (acc1-510.3927)/(510.3927-cal1);
acc2 = (acc2-510.3927)/(510.3927-cal2);
acc3 = (acc3-510.3927)/(510.3927-cal3);
acc4 = (acc4-510.3927)/(510.3927-cal4);

% Plot data on a single plane
figure(1)
hold on
plot(t1,acc1,'r-');
plot(t2,acc2,'g-');
plot(t3,acc3,'b-');
plot(t4,acc4,'k-');
hold off

%% Plot accelerometer readouts in one tiled figure
figure(2)
subplot(2,2,1)
plot(t,acc1,'r-');
title('Accelerometer #1');
ylabel('Acceleration (g)');
xlabel('Time (ms)');
%axis([0 5000 -3 3]);
subplot(2,2,2)
plot(t,acc2,'g-');
title('Accelerometer #2');
ylabel('Acceleration (g)');
xlabel('Time (ms)');
%axis([0 5000 -3 3]);
subplot(2,2,4)
plot(t,acc3,'b-');
title('Accelerometer #3');
ylabel('Acceleration (g)');
xlabel('Time (ms)');
%axis([0 5000 -3 3]);
subplot(2,2,3)
plot(t,acc4,'k-');
title('Accelerometer #4');
ylabel('Acceleration (g)');
xlabel('Time (ms)');
%axis([0 5000 -3 3]);
```

Arduino code for phase analysis:

/* Phase relationship analysis file:

By Lisa Michelle Slaughter, physics

California Polytechnic State University, SLO

Summer 2013

Public domain

```
   For use with any voltage-output accelerometer (0-5V)

   instructions:
   1) Select the appropriate COM port from tools -> Serial Port
   2) Ensure that 'PhaseRelationship.m' is also using this port
   3) Plug in Arduino board
   4) Click the 'upload' button above (round with an arrow)
   5) Run 'PhaseRelationship.m'
*/

int t = 0;
int dat1 = 0;
int dat2 = 0;
int dat3 = 0;
int dat4 = 0;

void setup()
{
  Serial.begin(115200);
}

void loop()
{
  dat1 = analogRead(A1);
  dat2 = analogRead(A2);
  dat3 = analogRead(A3);
  dat4 = analogRead(A4);
  t = millis();
  Serial.print(t);
  Serial.print(",");
  Serial.print(dat1);
  Serial.print(",");
  Serial.print(dat2);
  Serial.print(",");
  Serial.print(dat3);
  Serial.print(",");
  Serial.println(dat4);
}
```