The University of Texas at Austin
Aerospace Engineering
and Engineering Mechanics
Cockrell School of Engineering

ASE 375 Electromechanical Systems
Section 14115

Monday: 3:00 - 6:00 pm

Report 6: Measuring Dynamic Response with Accelerometers

Andrew Doty, Andres Suniaga, Dennis Hom
Due Date: 03/25/2024

Contents

1 Introduction

This experiment explores the dynamic response of a built-up wing from measurements using (1) a Piezoelectric accelerometer and (2) a Micro-Electro-Mechanical system (MEMS) accelerometer. The goal of this lab experiment is to learn how the accelerometers work by performing a resonance assessment profile (rap or impulse) test on the make-shift wing.

The piezoelectric accelerometer cannot measure static quantities, meaning it does not take into the account the force exerted by gravity. On the other hand, the MEMS accelerometer does take into account the gravitational force in its measurements. This lab experiment provides a foundation for understanding the operational aspects of the accelerometers as well as a rap test use-case that show how these sensors are applied in testing of aircraft surfaces.

2 Equipment

Measurement devices and hardware used in this lab include:

- Built-Up Wing Model:

  In this lab we will use a scaled-down wing model for performing the rap test. This form of experiment is useful in application as rap tests are performed on control surfaces of aircraft.

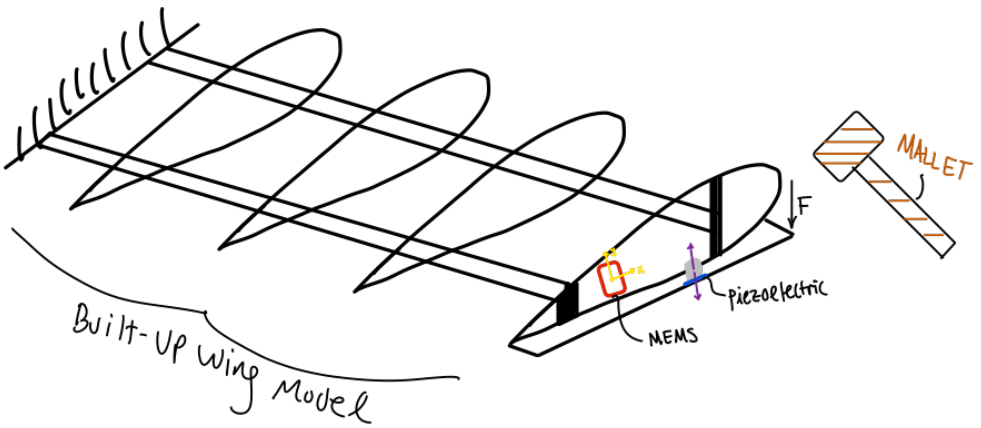

Figure 1: Sketch of Built-Up Wing Model setup

- Piezoelectric Accelerometer (IMI 660)

  An IMI Series 660 accelerometer is used in the experiment. This Piezoelectric accelerometer only measures acceleration in one direction, and is placed on the built-up wing model as shown in Figure 1.

  The operational principle of the Piezoelectric accelerometer is in it piezoelectric crystal, which generates charge from applied stress. It converts mechanical energy into electrical energy. The crystal acts like a capacitor (stores electrical energy). Operational Voltage = 5V

- MEMS Accelerometer (ADXL335)

  The ADXL335 is a small 3-Axis MEMS Accelerometer which operates based on differential capacitance. An applied force or acceleration will change the capacitance within the MEMS accelerometer. In this

experiment, the MEMS accelerometer is placed as shown in Figure 1, using only the X-and Y-axes. Operational Voltage = 3.3V

- Brass Mallet:

  A brass mallet is used to apply a tap force to the built-up wing model as shown in Figure 1.

- DAQ, NI-9215 Voltage Input Module, and LabVIEW:

  Data Acquisition System used to process sample measurements into digital data. NI-9215 is an analog input module used to measure the output voltage signals of sensors and send it through the DAQ system. LabVIEW used to model these output voltages read from the DAQ of the accelerometer measurements. We connect to the 3.3V and 5V ports of the DAQ for our experiment.

- Solderless Breadboard, Jumper Wires:

  Used to make connections to the input analog modules and to construct circuits. In this lab we connect the accelerometer inputs to the breadboard for power, ground, and signal to the measurement axes.

3 Procedure

Before beginning the experiment, we must build the block diagram which will be executed by LabVIEW to gather our accelerometer measurements.

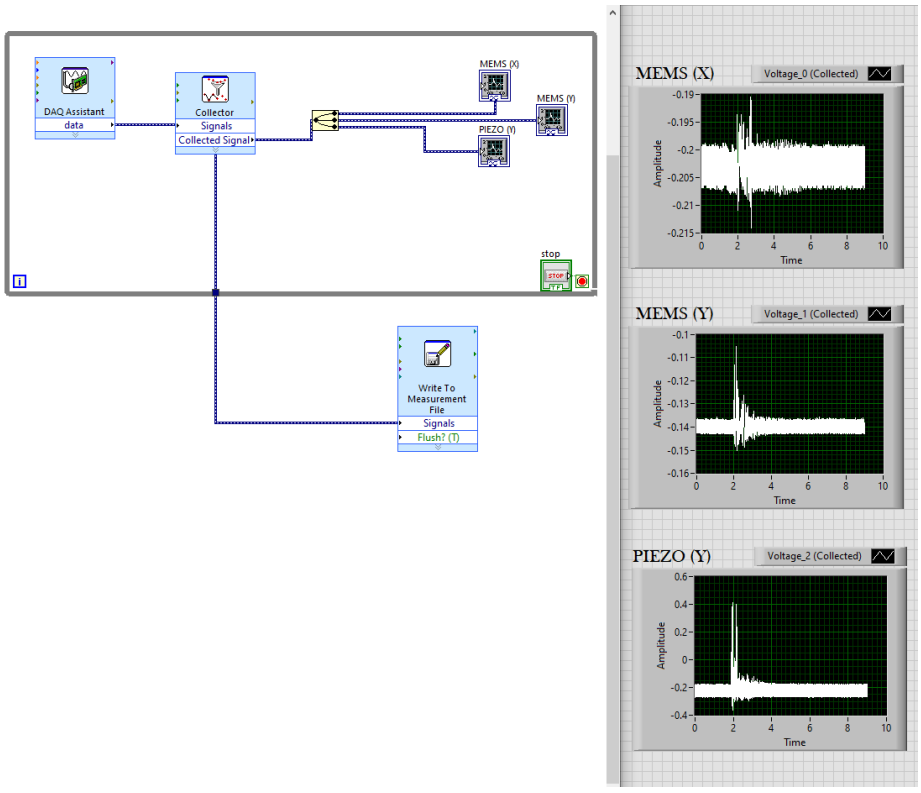

Figure 2: LabVIEW Model Setup for Accelerometers

3.1 Accelerometer Setup and Experiment

1. First, connect the MEMS and Piezoelectric accelerometers accordingly to power and ground, and connect the signal wires to corresponding NI-9215 ports.
2. Place the MEMS and Piezoelectric accelerometer on the wing model as shown in Figure 1.
3. Before hitting the wing with the brass mallet, the test that the MEMS and Piezoelectric accelerometers work by applying a force and observing the corresponding output in LabVIEW. Continue to the next step once working. If not working, check possible issues which may include:
   - Connections on breadboard
   - Faulty wires from NI-9215 Port
   - Faulty accelerometer sensor
4. Run the LabVIEW model as shown in Figure 2 and tap the wing with the brass mallet. Let it record the accelerometer data for 5 seconds. Save the data to a file and ensure it has been written correctly.
5. Repeat previous step (3) for 10 measurements.
6. Once complete, we are ready to plot the measured accelerations and evaluate the data. From this we can identify the natural frequencies of the wing.

4 Data Processing

4.1 Script Pseudocode

For the accelerometer portion of the lab, prior to the power spectrum analysis, all calculations were finished using this MATLAB script. The pseudocode is here, and the full MATLAB code will be detailed in the appendix 6.

```
read_data:
  file_path: "Lab Data/Lab6__1.xlsx"

processing_steps:
  - read_table:
      input: "Lab Data/Lab6__1.xlsx"
  - generate_time_sequence:
      start: 0.001
      end: 10
      step: 0.001
  - select_data_columns:
      columns: [2, 4, 6]
  - convert_to_array:
      input: data
  - calculate_averages:
      axis: 1
  - normalization:
      subtract: averages
      adjust_for_calibration:
        - mems_x: "scale by CaliMems"
        - mems_y: "scale by CaliMems, convert to degrees"
        - piezo_y: "scale by CaliPiezo, adjust scale factor"
  - fft_analysis:
      components: [X, mems_x, mems_y]
  - numerical_integration:
      calculate_velocity_and_displacement: true
```

```
      - plot_data:
          figures:
            - figure_1: "Acceleration and Velocity plots"
            - figure_2: "Displacement plots"
            - figure_3: "Frequency Response Analysis"

  calibration_constants:
    CaliMems: "300*3.3/3, converted to V/(m/s^2)"
    CaliPiezo: "10, converted to V/(m/s^2)"

  analysis:
    frequency_response:
      initial_analysis:
        period: "5 seconds"
      follow_up_analysis:
        period: "last second"
    derive_natural_frequencies:
      method: "find peak amplitude locations"
    compile_results:
      format: table
      contents: "Sensor, Frequency at 5 seconds, Frequency at Last Second"
```

For the power spectrum analysis, the following script was used in psuedocode and will be detailed in the appendix 6.

```
  file_processing:
    directory_path: "Lab Data/*.xlsx"
    get_files_and_folders: true
    file_check:
      is_file: true

  initializations:
    figure: 1
    natural_frequencies_5_seconds: []
    natural_frequencies_last_second: []

  processing_steps:
    for_each_file:
      if: "is_file"
      operations:
        - read_file:
            path: "Full path based on directory and file name"
            action: "Read table from Excel file"
        - preprocess_data:
            time_sequence: "0.001 to 10 in 0.001 increments"
            select_columns: [2, 4, 6]
            convert_to_array: true
            calculate_averages: true
            normalize_data: "Subtract averages, adjust for CaliPiezo"
        - fft_analysis:
            component: "data_norm(:,3)"
            frequency_range: "0 to 1000"
            calculate_amplitude: true
        - plot_frequency_response:
            figure: 1
```

```
            plot: "Frequency vs. Amplitude (5 seconds)"
            title: "Piezo (5 seconds)"
            add_legend: true
        - identify_natural_frequency:
            find_maximum_amplitude_location: true
            store_frequency: "NaturalFreqV0_5"
        - optional_section:
            commented_out: true
            description: "Analysis for the last second is prepared but not executed"

  second_loop:
    description: "Repeat steps for second analysis focusing on the last second"
    operations:
      - adjust_data_set: "Halve N, adjust frequency range"
      - fft_analysis:
          component: "data_norm(:,3) for last half"
          calculate_amplitude: true
      - plot_frequency_response:
          figure: 2
          plot: "Frequency vs. Amplitude (Last second)"
          title: "Piezo (Last second)"
          add_legend: true
      - identify_natural_frequency:
          find_maximum_amplitude_location: true
          store_frequency: "NaturalFreq_LastSec"

  final_steps:
    remove_initial_zeros: "NaturalFreq_LastSec adjustment"
    calculate_averages:
      - average_frequency_5_seconds: "Mean of NaturalFreqV0_5"
      - average_frequency_last_second: "Mean of NaturalFreq_LastSec"
```

The above power spectrum analysis script processes acceleration data from Excel files, normalizing and calibrating it before applying a Fast Fourier Transform (FFT) to shift from time domain to frequency domain. This reveals the power distribution across frequencies, allowing for identification of dominant frequencies or natural vibrations. It calculates the mean amplitudes over two distinct time intervals to detect any shifts in vibration characteristics, providing critical insights into the dynamic response of the system under observation.

5 Results and Analysis

5.1 Accelerometer Data

The piezoelectric data and MEMS accelerometer data were collected and analyzed in LabVIEW. The data was collected for 10 seconds in each iteration instead of 5 so that way the effects of the perturbation have more time to level out. The data collected is shown in Figure 3.

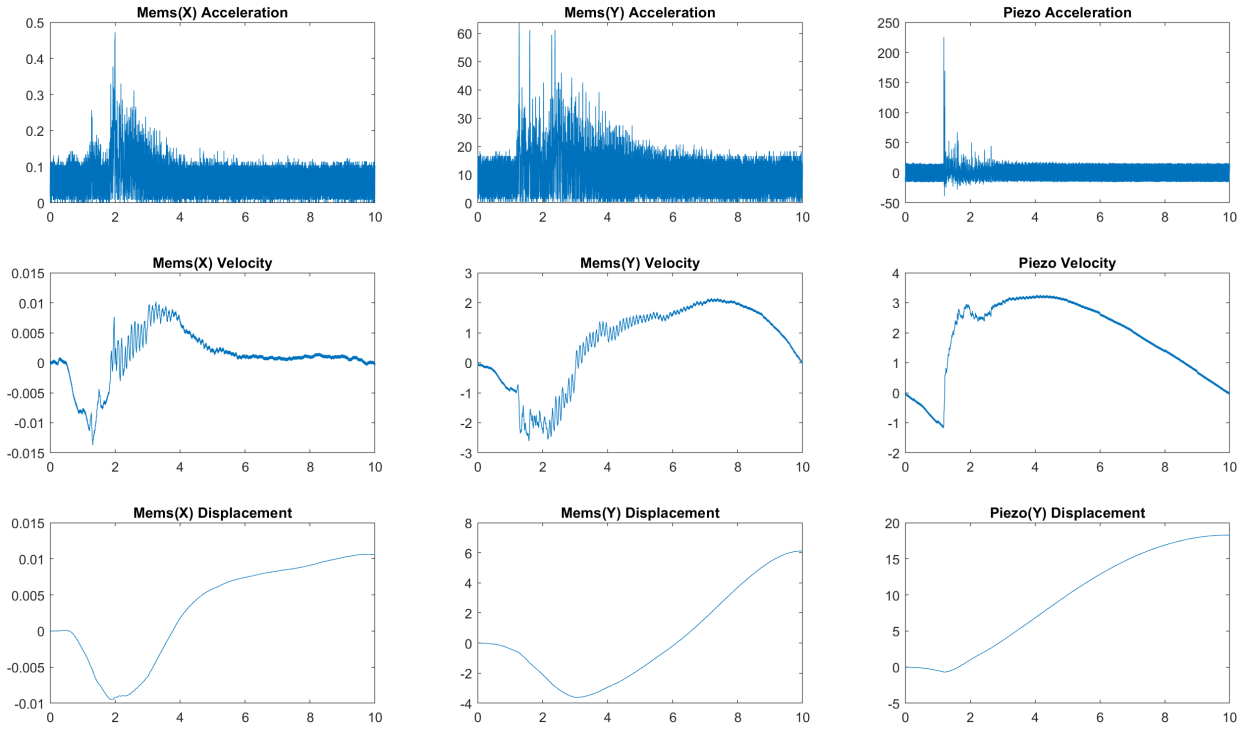

Figure 3: Acceleration, Velocity, and Displacement Data

In the graphs, we can clearly see the acceleration present from the initial disturbance, a velocity profile indicative of the acceleration, and a displacement profile indicative of the velocity. However, although these graphs are useful for understanding the data, there are errors inherent in the data that need to be addressed. The data is noisy and has a lot of high-frequency noise that needs to be filtered out, which can clearly be seen in the acceleration data. That band of noise is quite wide, for X acceleration the noise band is 0.1 with a peak acceleration near 0.5, and for Y acceleration the noise band is around 15 with a peak acceleration near 60. Lastly, the Piezo data shows a noise band of around 40 with a peak near 240. While the noise is also present in the velocity and displacement data, it is not as pronounced as in the acceleration data due to the integration process to get those results. However, that does not mean that the noise is not present in the velocity and displacement data, the integration is including it in the measurements and distorting the results. To have a higher power signal-to-noise ratio, we need to filter out the noise in the data. We accomplished that using a power spectrum analysis in MATLAB.

5.2 Power Spectrum Analysis

The power spectrum analysis should help filter out the noise in the data and identify the natural frequency. This was accomplished using the script whose pseudocode was detailed in the previous section. The results of the power spectrum analysis are shown in Figures 4 and 5.

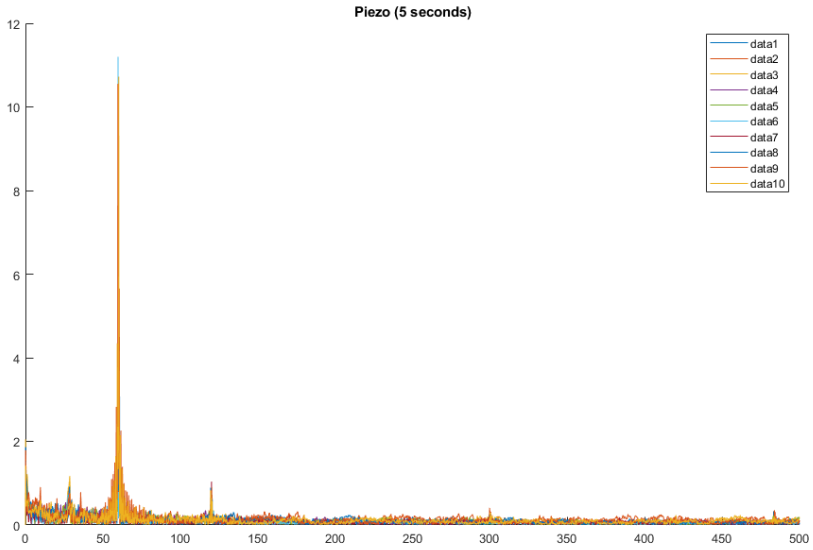

Figure 4: Frequency Analysis 5 Seconds

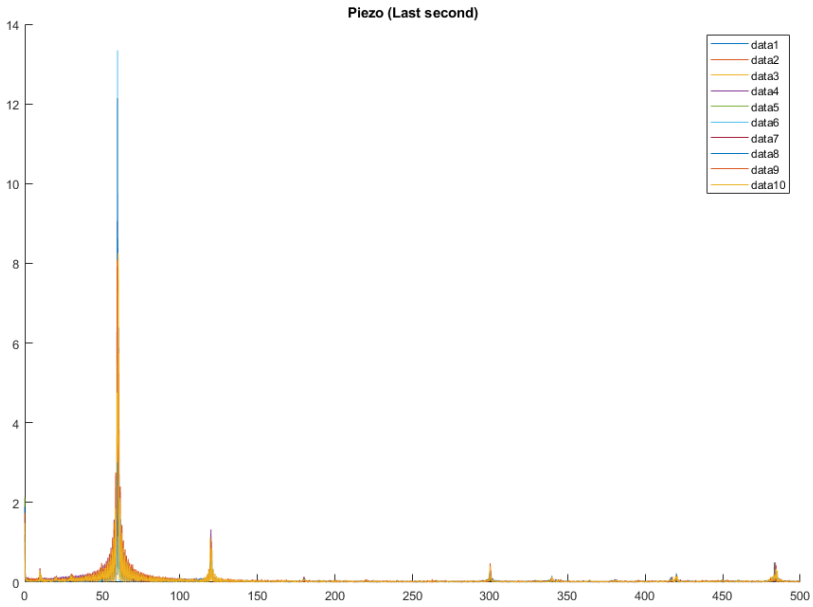

Figure 5: Frequency Analysis Last Second

This frequency response shows the noise present at higher frequencies in the oscillations, but a very dominant natural frequency appears at around 59.9 Hz in all cases, despite variations in the noise below the natural frequency and above the natural frequency. Computing the specific natural frequency for each case numerically, we get:

Natural Frequency 5 seconds =
[59.9060, 60.2289, 59.8060, 59.7060, 59.8060, 59.8060, 59.7060, 59.7844, 59.6733, 60.3060]

Natural Frequency Last Second =
[60.0120, 59.7911, 59.8120, 59.6119, 59.8120, 59.8120, 59.8120, 59.7911, 59.5688, 60.2120]

Average Natural Frequency over 5 Seconds = 59.8729

Average Natural Frequency over Last Second = 59.8235

The very close alignment between the natural frequencies highlights the accuracy of our approach. In addition, the average natural frequency over the 5-second interval is very close to the average natural frequency over the last second, indicating the stability of the system's dynamic response. This power spectrum analysis enables us to use various filters to remove noise now that we have identified the natural frequency of the system.

6 Conclusion

In this lab experiment we learned how the piezoelectric and MEMS accelerometers operate in order to measure acceleration and perform frequency analysis of an aircraft surface. These devices are very practical in analyzing aircraft control surfaces that experience lots of vibration as these sensors are very responsive to forces. Analysis of these surfaces provides critical information useful to the structural dynamics of an aircraft. Due to this, accelerometers play an important role in ensuring an aircraft is safe to fly, and power spectrum analysis allows for the identification of natural frequencies of the system. The results of the power spectrum analysis show that the natural frequency of the system is around 59.9 Hz, which is very close to the average natural frequency over the last second of the data. This indicates that the system is stable and has a consistent dynamic response.

Appendix

Accelerometer Analysis Matlab Script

```
T=readtable("Lab Data/Lab6__1.xlsx");

time=0.001:0.001:10;
data=[T(:,2) T(:,4) T(:,6)];
data=table2array(data);
averages=mean(data,1);
[N,M]=size(T);
% 1:mems (x)
% 2:mems (y)
% 3:piezo (y)
CaliMems=300*3.3/3; %mv/g
CaliMems=CaliMems*1/1000/9.81; %V/(m/s^2)
CaliPiezo=10; %mV/g
CaliPiezo=CaliPiezo*1/1000/9.81; %V/(m/s^2)
data_norm=bsxfun(@minus, data , averages);
data_norm(:,3)=data_norm(:,3)/CaliPiezo/2.815;
data_norm(:,1)=data_norm(:,1)/CaliMems; %mems x
data_norm(:,2)=data_norm(:,2)/CaliMems*180; %mems y
X=data_norm(:,3);


V0=fft(data_norm(:,1));
V1=fft(data_norm(:,2));
V2=fft(data_norm(:,3));
Frequency=linspace(0,1000,N);

%Numerical Integration
VelMemX=cumtrapz(time,data_norm(:,1));
VelMemX=VelMemX-VelMemX(1);
VelMemY=cumtrapz(time,data_norm(:,2));
VelMemY=VelMemY-VelMemY(1);
VelPiezoY=cumtrapz(time,X);
VelPiezoY=VelPiezoY-VelPiezoY(1);
disMemX=cumtrapz(time,VelMemX);
disMemX=disMemX-disMemX(1);
disMemY=cumtrapz(time,VelMemY);
disMemY=disMemY-disMemY(1);
disPiezoY=cumtrapz(time,VelPiezoY);
disPiezoY=disPiezoY-disPiezoY(1);

figure(1)
subplot(3,3,1)
plot(time,abs(data_norm(:,1)))
title('Mems(X) Acceleration')
subplot(3,3,2)
plot(time,abs(data_norm(:,2)))
title('Mems(Y) Acceleration')
subplot(3,3,3)
plot(time,X)
title('Piezo Acceleration')

subplot(3,3,4)
```

```
plot(time,VelMemX)
title('Mems(X) Velocity')
subplot(3,3,5)
plot(time,VelMemY)
title('Mems(Y) Velocity')
subplot(3,3,6)
plot(time,VelPiezoY)
title('Piezo Velocity')
subplot(3,3,7)
plot(time,disMemX)
title('Mems(X) Displacement')
subplot(3,3,8)
plot(time,disMemY)
title('Mems(Y) Displacement')
subplot(3,3,9)
plot(time,disPiezoY)
title('Piezo(Y) Displacement')


%Frequency Response

Amplitude1=2/N*real(abs(V0));
Amplitude2=2/N*real(abs(V1));
Amplitude3=2/N*real(abs(V2));
% 5 Seconds
figure(2);
subplot(3,2,1)
plot(Frequency(1:N/2),Amplitude1(1:N/2))
title('Mems(X) (5 seconds)')
subplot(3,2,3)
plot(Frequency(1:N/2),Amplitude2(1:N/2))
title('Mems(Y) (5 seconds)')
subplot(3,2,5)
plot(Frequency(1:N/2),Amplitude3(1:N/2))
title('Piezo (5 seconds)')

loc1=Amplitude1==max(Amplitude1(1:N/2));
loc2=Amplitude2==max(Amplitude2(1:N/2));
loc3=Amplitude3==max(Amplitude3(1:N/2));

NaturalFreqV0=Frequency(loc1);
NaturalFreqV1=Frequency(loc2);
NaturalFreqV2=Frequency(loc3);

NaturalFreqV0_5=NaturalFreqV0(1);
NaturalFreqV1_5=NaturalFreqV1(1);
NaturalFreqV2_5=NaturalFreqV2(1);
% Last second
N2=N2/2;
Frequency=linspace(0,1000,N2);
data_norm(1:N2,:)=[];
V0=fft(data_norm(:,1));
V1=fft(data_norm(:,2));
V2=fft(data_norm(:,3));
```

```
Amplitude1=2/N2*real(abs(V0));
Amplitude2=2/N2*real(abs(V1));
Amplitude3=2/N2*real(abs(V2));

subplot(3,2,2)
plot(Frequency(1:N2/2),Amplitude1(1:N2/2))
title('Mems(X) (Last second)')
subplot(3,2,4)
plot(Frequency(1:N2/2),Amplitude2(1:N2/2))
title('Mems(Y) (Last second)')
subplot(3,2,6)
plot(Frequency(1:N2/2),Amplitude3(1:N2/2))
title('Piezo (Last second)')

loc1=find(Amplitude1==max(Amplitude1(1:N2/2)));
loc2=find(Amplitude2==max(Amplitude2(1:N2/2)));
loc3=find(Amplitude3==max(Amplitude3(1:N2/2)));

NaturalFreqV0=Frequency(loc1);

NaturalFreqV1=Frequency(loc2);
NaturalFreqV2=Frequency(loc3);

NaturalFreqV0_Last=NaturalFreqV0(1);
NaturalFreqV1_Last=NaturalFreqV1(1);
NaturalFreqV2_Last=NaturalFreqV2(1);

Frequency_5s=[NaturalFreqV0_5; NaturalFreqV1_5; NaturalFreqV2_5];
Frequency_LastSec=[NaturalFreqV0_Last; NaturalFreqV1_Last; NaturalFreqV2_Last];
Sensor=["Mems(X)";"Mems(Y)";"Piezo(Y)"];
Table=table(Sensor, Frequency_5s, Frequency_LastSec)
```

Power Spectrum Analysis Matlab Script

```
% Get a list of all files and folders in the directory
Files = dir('Lab Data/*.xlsx');
i=1;
% Get a logical vector that tells which is a file
isFile = ~[Files.isdir];
figure()
% Loop only over the files
for iExcelSubject = 1:length(Files)
    if isFile(iExcelSubject)
        % Full path to file
        Report = fullfile('Lab Data', Files(iExcelSubject).name);
        % Read the table from the Excel file
        T = readtable(Report);
        % Do something with table T
        % ...
        time=0.001:0.001:10;
data=[T(:,2) T(:,4) T(:,6)];
data=table2array(data);
averages=mean(data,1);
```

```
    [N,M]=size(T);
    % 1:mems (x)
    % 2:mems (y)
    % 3:piezo (y)
    CaliPiezo=10; %mV/g
    CaliPiezo=CaliPiezo*1/1000/9.81; %V/(m/s^2)
    data_norm=bsxfun(@minus, data , averages);
    data_norm(:,3)=data_norm(:,3)/CaliPiezo/2.815;


    V0=fft(data_norm(:,3));
    Frequency=linspace(0,1000,N);
    Amplitude1=2/N*real(abs(V0));
    hold on;
    figure(1)
    plot(Frequency(1:N/2),Amplitude1(1:N/2))
    title('Piezo (5 seconds)')
    legend()

    loc1=Amplitude1==max(Amplitude1(1:N/2));
    NaturalFreqV0=Frequency(loc1);
    NaturalFreqV0_5(i)=NaturalFreqV0(1);

    i=i+1;
    % N2=N/2;
    % Frequency=linspace(0,1000,N2);
    % data_norm(1:N2,:)=[];
    % V0=fft(data_norm(:,1));
    % Amplitude1=2/N2*real(abs(V0));
    % hold on;
    % figure(2)
    % plot(Frequency(1:N2/2),Amplitude2(1:N2/2))
    % title('Piezo (Last second)')
    % legend()
        end
    end
    hold off;


    figure();
    hold on;
    % Loop only over the files
    for iExcelSubject = 1:length(Files)
        if isFile(iExcelSubject)
            % Full path to file
            Report = fullfile('Lab Data', Files(iExcelSubject).name);
            % Read the table from the Excel file
            T = readtable(Report);
            % Do something with table T
            % ...
            time=0.001:0.001:10;
    data=[T(:,2) T(:,4) T(:,6)];
    data=table2array(data);
    averages=mean(data,1);
```

```
    [N,M]=size(T);
    % 1:mems (x)
    % 2:mems (y)
    % 3:piezo (y)
    CaliPiezo=10; %mV/g
    CaliPiezo=CaliPiezo*1/1000/9.81; %V/(m/s^2)
    data_norm=bsxfun(@minus, data , averages);
    data_norm(:,3)=data_norm(:,3)/CaliPiezo/2.815;


    % V0=fft(data_norm(:,3));
    % Frequency=linspace(0,1000,N);
    % Amplitude1=2/N*real(abs(V0));
    % hold on;
    % figure(1)
    % plot(Frequency(1:N/2),Amplitude1(1:N/2))
    % title('Piezo (5 seconds)')
    % legend()

    N2=N/2;
    Frequency=linspace(0,1000,N2);
    data_norm(1:N2,:)=[];
    V0=fft(data_norm(:,3));
    Amplitude1=2/N2*real(abs(V0));
    hold on;
    figure(2)
    plot(Frequency(1:N2/2),Amplitude1(1:N2/2))
    title('Piezo (Last second)')
    legend()

    loc1=Amplitude1==max(Amplitude1(1:N2/2));
    NaturalFreqV0=Frequency(loc1);
    NaturalFreq_LastSec(i)=NaturalFreqV0(1);

    i=i+1;
        end
    end
    hold off;

    NaturalFreqV0_5
    NaturalFreq_LastSec;
    NaturalFreq_LastSec=NaturalFreq_LastSec(11:end)
    Avg5SecFreq=mean(NaturalFreqV0_5)
    AvgLastSecFreq=mean(NaturalFreq_LastSec)

```

NI-9215 Datasheet

https://www.amc-systeme.de/files/pdf/ni-9215-amc.pdf

IMI Series 660 Accelerometer Datasheet

https://pim-kft.hu/wp-content/uploads/2016/02/PCB_LowCost_Embeddable_Accelerometers.pdf

ADXL335 Accelerometer Datasheet

https://www.analog.com/media/en/technical-documentation/data-sheets/adxl335.pdf