ME 3870 Lab 5: Pressure Measurement

by Walker Blevins, Eshwar Pamula, and Hiromu Yamamoto

Executive Summary: The key objectives of this investigation were to apply Wheatstone bridge circuit and strain gauge theories to construct a lab-built pressure transducer capable of accurately, yet indirectly measuring voltage signals corresponding to strains experienced in the gauge diaphragm. Data for both static and dynamic calibration of each voltage and the measurement of each pressure were captured using appropriately tailored virtual instruments in LabView. The subsequent data analyses focus on the uncertainty of the overall measurement system based on its constituent components, the derivation of experimental data errors associated with least squares regression lines of varying order, the second-order system parameters associated with the transducer's dynamic responses, and the spectral data associated with these responses. The static analysis served to ensure the reliability and accuracy of the pressure transducer based on theoretical calculations, while the dynamic analysis allowed for characterization of the transducer's dynamic behavior and identification of potential aliasing misinterpretations. Ultimately, the experiment was conducted to gain practical experience in data sampling using an integrated pressure measurement system and to gauge its degree of correlation to theoretical predictions.

I. Introduction

The underlying motivation for building an accurate pressure transducer measurement system lies the necessity for precise and controlled pressure variation in a wide variety of industrial systems. Due to their high sensitivity and responsiveness, they are able to offer real-time feedback for fine adjustments within automated processes. This particular investigation utilized various physical theories in order to establish a basis off of which to compare experimental results. The benefit of constructing an integrated measurement system using a wheatstone bridge circuit with strain gauges on all legs (full-bridge configuration) lies in the circuit's ability to amplify small signals with minimal disturbance. Strain gauges are used to convert mechanical deformation into measurable electrical resistance changes. When a conductor is stretched or compressed, both its geometry and resistivity change. The total strain includes both axial and transverse components, with temperature effects requiring compensation. For this particular investigation, a full-bridge wheatstone bridge circuit was utilized because of its high sensitivity to changes in the diaphragm under pressure and its ability to compensate for various sources of noise. Because the strain gauges were placed in regions of maximum strain, the differential output voltage was maximized, allowing for highly-sensitive voltage fluctuations in response to relatively small deformations.

Strain Gauge Equations:

1. $\Delta R/R = GF \times \varepsilon$
2. $\varepsilon total = \varepsilon axial + \varepsilon transverse$

Wheatstone Bridge Equations:

1. $V_0 = Vex[(R_1R_4 - R_2R_3)/((R_1 + R_2)(R_3 + R_4))]$
2. $V_0 = Vex \times (R_3/(R_3+R_4) - R_1/(R_1+R_2))$
3. $V_0/Vex \approx (GF \times \varepsilon)/4$ (for quarter bridge, small strains)

The use of an amplifier in parallel with a reference pressure transducer results in increased complexity when considering the sensitivity of the measurement system in question. The total system sensitivity is a function of both the amplifier gain and the pressure transducer sensitivity.

In order to compare experimental data behavior with theoretical behavior based on underlying physical theory, it is necessary to establish lines of best fit based on predicted system behavior. The matrix method for least squares regression serves to accomplish this for any specified order of polynomial. The best-fit parameters are determined such that the squared residual errors are minimized, providing parameter estimates and associated error metrics.

1. $\hat{\beta} = (X^TX)^{-1}X^Ty$
2. $\sigma = \sqrt{(\Sigma(yi - \hat{y}i)^2/(n-p))}$
3. $SE(\hat{\beta}) = \sigma\sqrt{((X^TX)^{-1})}$
4. $R^2 = 1 - RSS/TSS$

Bandwidth and Aliasing refers to the concept that signal sampling must occur at sufficient frequency to prevent aliasing. Anti-aliasing filters must be implemented before digitization, with appropriate bandwidth considerations for the measurement system.

1. fs > 2fmax
2. Practical sampling rate: fs = (2.5 to 4) × fmax

II. Experimental Procedure

The experimental apparatus consisted of a pressure regulator and bourdon tube which took in air through an air supply and attached to the strain gauge diagram to provide measurable pressures to the system. The strain gauges were connected within a Wheatstone bridge circuit configuration, with a reference pressure transducer and signal amplifier connected in parallel and feeding into the DAQ connector box. Before utilizing the pressure transducer for data acquisition, a diaphragm schematic was sketched (view Schematic of Sensor). Finally, the appropriate input/output cables were connected so that the system worked properly and data could be collected. The following figure was provided for reference:

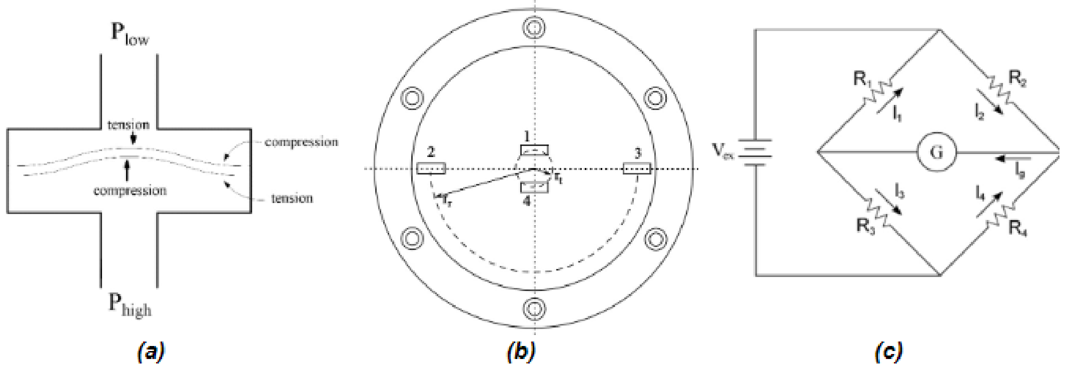
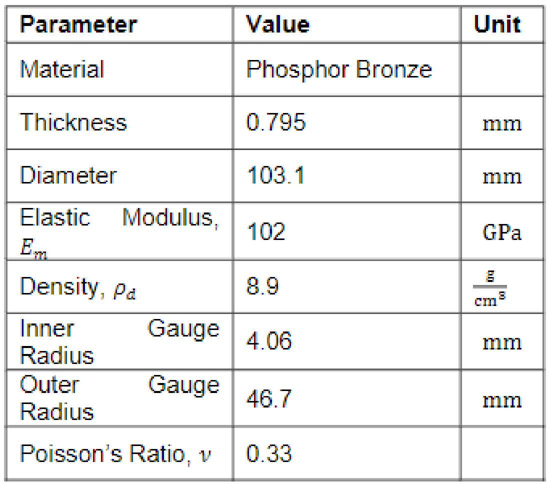


Figure A.1:Arrangement of the diaphragm in the pressure sensor (a), strain gauge placement (b), and Wheatstone bridge circuit used for pressure measurement ©

The excitation voltage was set to 10 volts in order to remain consistent with theoretical calculations. Next, the amplifier gain was set to the value of K_a – in our case, 1– so that we achieved amplified signals on the order of 5 volts for 1 bar input pressure. Depending on the following low and high bounding frequency conditions, an appropriate cut-off frequency was needed to be set. After all of these checks, we were able to turn on the signal conditioner. The pressure transducer used can be seen below, as well as the parameters of the diaphragm sensor.

Table A.1: Properties of the Diaphragm Sensor



Parameter	Value	Unit
Material	Phosphor Bronze	
Thickness	0.795	mm
Diameter	103.1	mm
Elastic Modulus, E_m	102	GPa
Density, ρ_d	8.9	$\frac{g}{cm^3}$
Inner Gauge Radius	4.06	mm
Outer Gauge Radius	46.7	mm
Poisson's Ratio, ν	0.33	

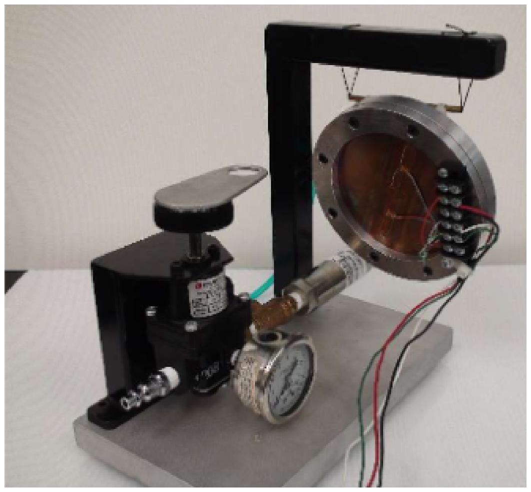


Figure A.2: Pressure transducer using phosphor bronze diaphragm with four strain gages

For our data to be collected, we utilized the application LabVIEW. To collect the data, we were given the file 'Pressure.vi' for our DAQ setup in LabVIEW. In the DAQ, we needed to set an appropriate voltage range for the data acquisition. To ensure our data was legible, an appropriate sampling frequency was needed varying based on the pressure and frequency we were testing at. The data is collected through the pressure transducer to the program on LabVIEW via BNC cable. Once all parameters are set for each respective frequency and pressure, we can press play on the front panel in LabVIEW. When collecting data, we supply air pressure into the system by opening the valve attached to the pressure transducer. When checking at a constant pressure, the valve can be turned before clicking play and ensuring it is at the desired pressure on the attached gauge.

To collect static calibration data, we recorded the behavior by slowly increasing from 0 psi to 10 psi, and then slowly descending back down to 0 psi. For dynamic calibration data, we set the pressure at a set psi level and then lightly hit the back of the diaphragm with the given ping hammer, also known as the "Impact test". This caused a spike in the data and signaled LabVIEW to collect the data. Depending on the pressure we were testing at, the trigger would need to be adjusted to capture the entire spike of the data. This was tested for 0 psi, 5 psi, and 10 psi. Remembering to re-zero the strain gauges would ensure proper data collection for each run.

Finally, to investigate behaviors under different frequencies, we ran the same impact test all at 5 psi, but at different pre-set frequencies. Those frequencies were 2 kHz, 2.3 kHz, 2.7 kHz, 3.1 kHz, and 10 kHz. This set of data was collected by following the same data for the first part of dynamic calibration data.

Once both static and dynamic calibration data were collected for all pressures and frequencies listed, we saved each test on our computer. These were saved as .lvm files which we converted to .txt files and transferred them into MATLAB for further data analysis. In MATLAB we derived plots and equations for our data points. This helped us find the minimum/maximum voltages and pressures produced in our runs. The plots helped us see the behavior of each run under their conditions.

III. Results and Discussion

The results of the initial analysis of the reference pressure transducer were mostly consistent with expectations and provided a general sense of the accuracy of the measurement system in question. The provided strain gauge data applied to Equations 11-19 allowed for the calculation of individual gain values for the amplifier circuit and the strain gauge pressure transducer, which in turn allowed for the derivation of a theoretical overall system sensitivity. This value provided a preliminary estimate of the system's theoretical sensitivity based on the Wheatstone bridge transfer function and resistance-strain relations associated with the particular strain gauge configuration in question. An initial analysis of the ADC provided prerequisite insight into the degree of accuracy of the measurement system constructed. Using information from the provided ADC datasheet and the theoretical system sensitivity, the ADC resolution was determined in both units of voltage and pressure. Using this theoretical sensitivity as a basis, the accuracy of the reference pressure transducer was determined, which notably contributed a considerably greater amount to the overall system uncertainty than the ADC resolution did. Results associated with these initial analyses are tabulated in Table*

Table A.2: Statistics for Initial Investigation of System Sensitivity and Accuracy

Derived Theoretical Sensitivity (V/psi)	0.3448
ADC Resolution (V)	0.000153
ADC Resolution (psi)	0.000229
Reference Pressure Transducer Accuracy (psi)	0.0725
Overall Uncertainty (psi)	0.072501

The analysis of static calibration data involved an initial comparison between the measured static sensitivity and the theoretical sensitivity derived prior to data collection. With the static calibration data loaded into MATLAB appropriately, the output voltage was plotted against the measured pressure for easy visualization. We then chose an approximately linear region (the first 0.5 psi) of the collected data to derive an experimental sensitivity as tabulated below in Table*.

Table B.1: Comparison of Measured Sensitivity to Theoretical (Ideal) Sensitivity

Measured Static Sensitivity (V/psi)	0.61574
Theoretical Static Sensitivity (V/psi)	0.3448
Difference (V/psi)	0.27094

It is important to note a considerable difference between the theoretical and experimental sensitivity values, and upon further consideration of the data's behavior, it was concluded that this variation arose based on the chosen region of data to base the calculations on. Since it was noted that the data appeared to follow a more nonlinear pattern, it was determined that the chosen data region resulted in the steepest possible experimental sensitivity value. It was therefore deemed necessary to perform linear regression analyses to get a better understanding of the system's behavior with theoretical expectations.

The first linear regression best-fit line was determined using the matrix method for total experimental data error minimization about a first-order polynomial fit. An overlay of the static calibration data and the derived linear regression line in Figure* indicates a fair degree of inaccuracy, suggesting that the lab-built pressure sensor exhibited more non-linear behavior during the experiment. This can be best explained by potential imperfections in the Wheatstone bridge circuit, which may not have converted the strain into measured voltage entirely linearly due to nonlinear mechanical responses of the diaphragm under stress. The errors associated with the fitting of the data about a linear regression line are tabulated in Table*. As was consistent with predictions, the hysteresis error contributed the most to the total error using this method, as the linear best-fit line was fairly inaccurate for higher-pressure domains.

Table B.2: Calculated Errors for Linear Least Squares Regression

Hysteresis Error (V)	1.869943
Total Error about Linear Fit (V)	1.880456

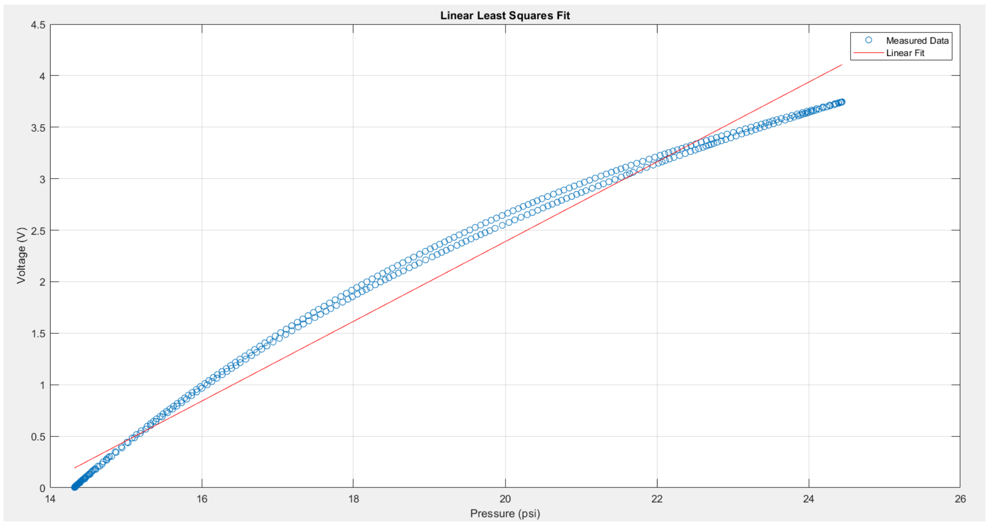


Figure B.1: Linear Regression Line for Static Calibration Data

To conclude with more certainty whether the pressure transducer was exhibiting nonlinear behavior, the second regression analysis was conducted for a non-linear best-fit curve. Utilizing the same error minimization theory in matrix form, the concepts were extended to a third-order polynomial, and thus the calculations returned four constraining coefficients for the regression line. This analysis proved to be more accurate for the experimental data, as suggested by the relatively higher correlation coefficient in comparison to the nonlinear regression method. A plot overlaying the nonlinear regression line and the experimental data is shown in Figure* and demonstrates this higher degree of correlation. A comprehensive comparison of the two regression methods is offered in Table*.

Table B.3: Comparison of Error for Linear/Nonlinear Least Squares Regression

Linear Least Squares Regression		**Nonlinear Least Squares Regression**	
Total Error (V)	1.880456	Total Error (V)	1.87109
Correlation Coeff.	0.978775	Correlation Coeff.	0.999276

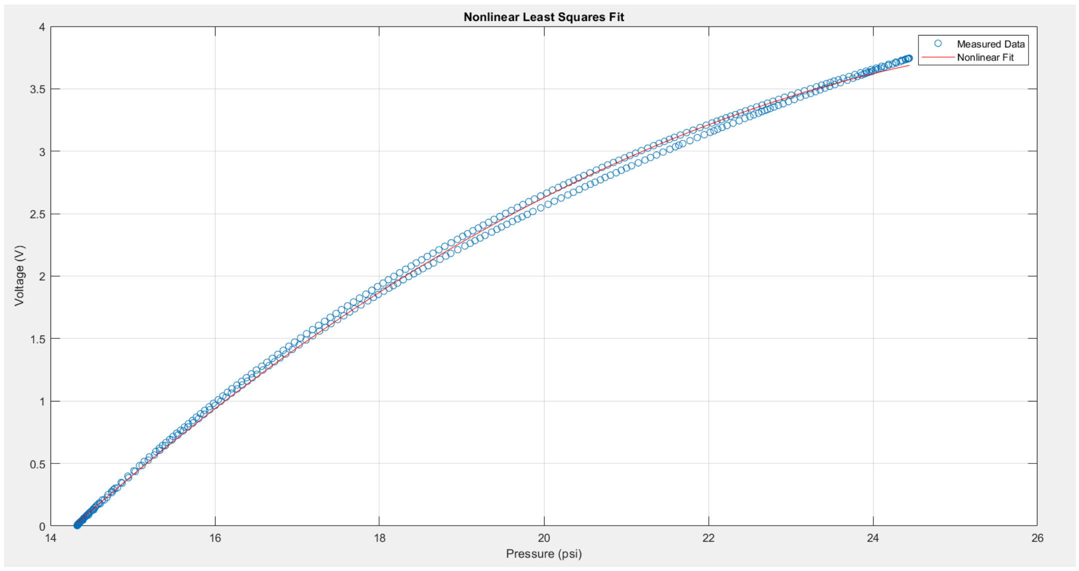


Figure B.2: Nonlinear Regression Curve for Static Calibration Data

For the Impulse Response investigation, we plot the transducer's response to an impulse at different pressures, we're observing how the diaphragm physically reacts to a sudden force. The resulting oscillations show us the transducer's dynamic behavior - imagine striking a drum and watching how it vibrates and eventually settles. These plots reveal important characteristics like how fast the diaphragm vibrates (frequency) and how quickly these vibrations die out (damping). By comparing responses at different pressures (0, 5, and 10 psi), we can see how pressure affects the transducer's dynamic behavior.

The natural frequency tells us how fast the diaphragm naturally wants to vibrate and we find this by measuring the time between peaks in the oscillations - a shorter time between peaks means a higher natural frequency. The damping ratio, on the other hand, tells us how quickly these oscillations die out, like how a door with a good damper closes smoothly without bouncing. We calculate this using the logarithmic decrement method, which looks at how each peak gets smaller than the previous one. A higher damping ratio means the oscillations die out faster, while a lower ratio means they persist longer.

Table C.1: Dynamic System Parameters at Various Pressures

Pressure	Natural Frequency	Damping Ratio
0 psi	557.21 Hz	0.006
5 psi	630.87 Hz	0.006
10 psi	602.87 Hz	0.003

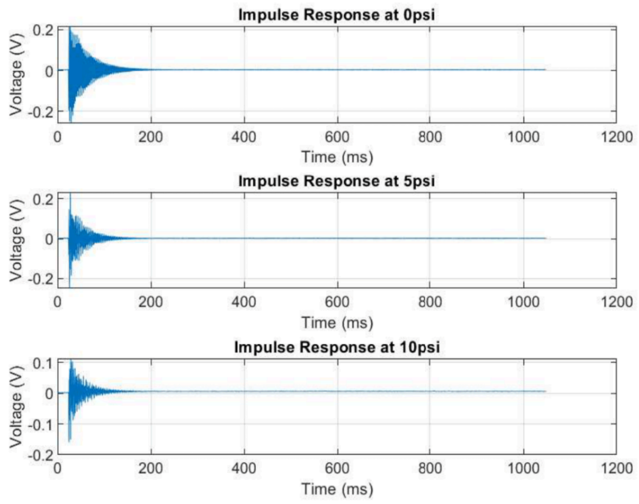


Figure C.1: Impulse Responses in Time for Dynamic Calibration Data

For FFT Analysis, we convert our time-domain signal into the frequency domain, essentially breaking down the complex vibration into its constituent frequencies. This is like taking a chord played on a piano and identifying each note being played. The FFT plots show us which frequencies are most prominent in the signal, with higher peaks indicating stronger frequency components. This analysis helps verify our natural frequency calculations and can reveal additional frequencies that might not be obvious in the time domain plot.

Table C.2: Locations of Largest Spectral Peaks for Dynamic Calibration Data

Pressure vs Frequency	Peak 1	Peak 2	Peak 3
0 psi	577.69 Hz	615.82 Hz	617.73 Hz
5 psi	737.85 Hz	251.67 Hz	695.90 Hz
10 psi	891.33 Hz	376.55 Hz	859.87 Hz

When we compare the natural frequencies found from the time domain (counting oscillations) versus the frequency domain (FFT peaks), we're essentially cross-checking our results using two different methods. Think of it like measuring a table's length with both a ruler and a measuring tape - they should give similar results. Any differences between these measurements might indicate measurement limitations or reveal interesting aspects of the system's behavior. This comparison helps validate our understanding of the transducer's dynamics.

Table C.3: Spectral Peak Location Variation between Time/Frequency Domains

Pressure	Time Domain Frequency	Frequency Domain Frequency	Percent Difference
0 psi	557.21 Hz	577.69 Hz	3.6753
5 psi	630.87 Hz	737.85 Hz	16.956
10 psi	602.87 Hz	0	100

Average Percent Difference = 40.21%

There are notable differences between time and frequency domain results.This could be due to noise in the signal, non-linear effects in the system, or sampling rate limitations

The bandwidth calculation tells us the range of frequencies over which our transducer can effectively measure pressure changes. We find this by identifying the frequency range where the signal strength remains above 70.7% (-3dB) of its maximum value. Think of bandwidth like the range of musical notes a microphone can accurately record - there's usually a limit at both low and high frequencies. This information is crucial because it tells us the fastest pressure changes the transducer can accurately measure, which is essential for choosing the right sensor for a particular application.

Table C.4: System Bandwidth and Frequency Content at Varying Pressures

Pressure	Peak Frequency	Lower Frequency	Upper Frequency	Bandwidth	Q Factor
0 psi	577.69 Hz	573.88 Hz	580.55 Hz	6.673	86.571
5 psi	737.85 Hz	733.08 Hz	740.71 Hz	7.6263	96.75
10 psi	0 Hz	0 Hz	0 Hz	0	NaN

Average Bandwidth: 4.77 ± 4.16 Hz

For Impulse Response plots, we examine the transducer's response at four different sampling frequencies (2.3 kHz, 2.7 kHz, 3.1 kHz, and 10 kHz). When we plot these, we're looking at how well we can capture the true behavior of the diaphragm at different sampling rates. We want to see if our sampling rate is fast enough to accurately record the diaphragm's vibrations.

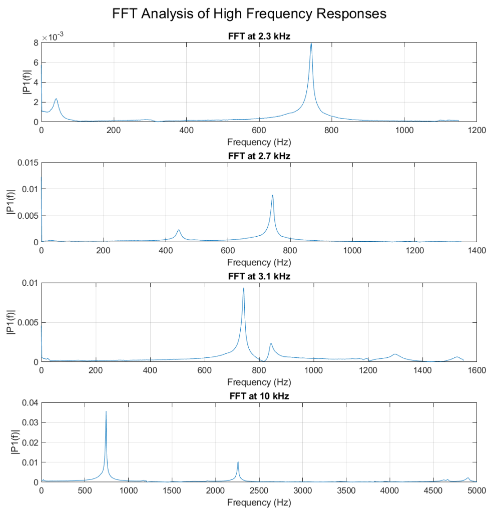


Figure C.2: Impulse Response Signal Spectra

The FFT plots for these different sampling rates reveal a crucial phenomenon called aliasing. When we sample a signal too slowly (below the Nyquist rate, which is twice the highest frequency in our signal), high-frequency components appear as false lower frequencies in our measurements. In our FFT plots, aliasing manifests as peaks appearing at frequencies that don't exist in the real signal. The 10 kHz sampling case serves as our reference since it's fast enough to capture the true frequencies.

We construct a detailed comparison of key frequencies and their aliases. The sample rate (fs) determines our Nyquist frequency (fN = fs/2), which is the highest frequency we can accurately measure. The actual second natural frequency of the diaphragm (f) appears as an aliased frequency (f1) when sampled too slowly. We can predict where these aliases will appear using the folding diagram or aliasing equations. By comparing our predicted aliased frequencies (f1) with what we observe in the FFT plots (f1,exp), we can verify our understanding of the aliasing phenomenon. The percent discrepancy between predicted and observed aliased frequencies tells us how well our theoretical understanding matches the experimental results.

Table C.5: Sample Rate, Nyquist Frequency, Second Natural Frequency, Aliased Frequency, Percent Discrepancy for Dynamic Calibration Data

Frequency	fs (Hz)	fN (Hz)	f (Hz)	f1 (Hz)	f1_exp (Hz)	Discrepancy
2.3 KHz	2300	1150	743.28	743.28	40.563	94.54277
2.7 KHz	2700	1350	742.56	742.56	441.42	40.55459
3.1 KHz	3100	1550	743.23	743.23	843.71	13.51889
10 KHz	10,000	5000	743.57	743.57	2259.3	203.8462

This aliasing investigation is crucial because it helps us understand the limitations of our measurement system. If we don't sample fast enough, we might think our diaphragm is vibrating at one frequency when it's actually vibrating at a higher frequency. This could lead to incorrect conclusions about the system's behavior and potentially catastrophic failures in real-world applications. The investigation also helps us determine the minimum sampling rate needed to accurately capture the diaphragm's dynamics, which is essential for proper sensor design and operation. A key insight from this analysis is that the second natural frequency of the diaphragm is particularly susceptible to aliasing because it's higher than the first natural frequency.

IV. Conclusion

This investigation resulted in the construction and methodic validation of a full-bridge wheatstone bridge configuration for a strain gauge-based pressure transducer. The application of linear regression models allowed for the verification of the system’s accuracy in converting various static pressure values to physical strains, which were in turn converted to voltage values. Similarly, the spectral analyses of the system response under impulsive inputs served to validate the system’s responsiveness while also identifying potential aliasing errors. Ultimately, this investigation provided an opportunity to create an integrated measurement system while drawing from a variety of theoretical sources.

Appendix A: Tables, Figures, and Equations for Initial Investigation

Figure A.1: Arrangement of the diaphragm in the pressure sensor (a), strain gauge placement (b), and Wheatstone bridge circuit used for pressure measurement (c)

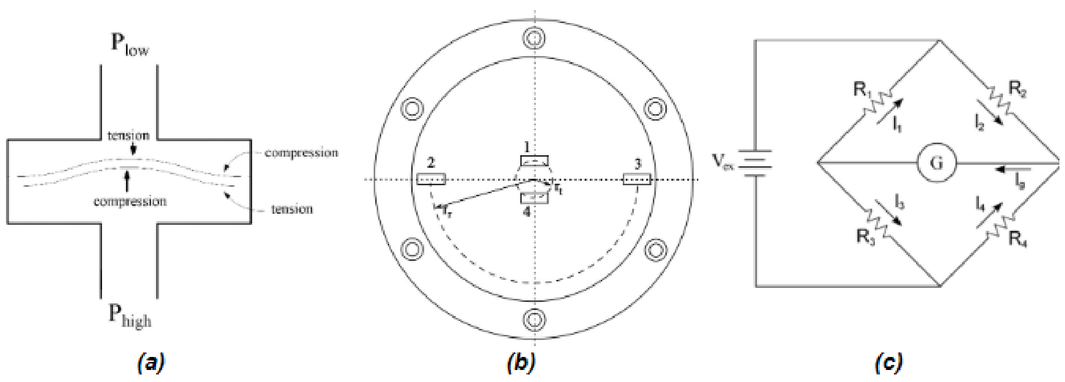


Figure A.2: Pressure transducer using phosphor bronze diaphragm with four strain gages

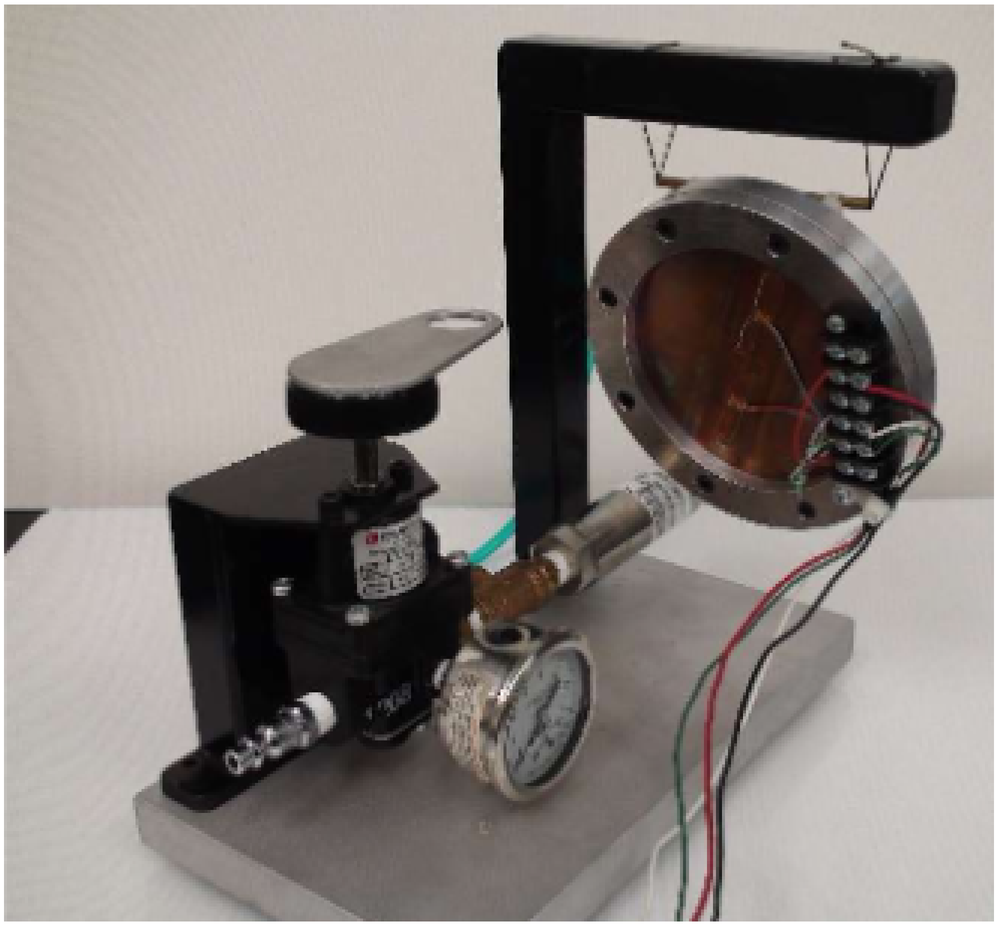


Figure A.3: Sensor Schematic Sketch

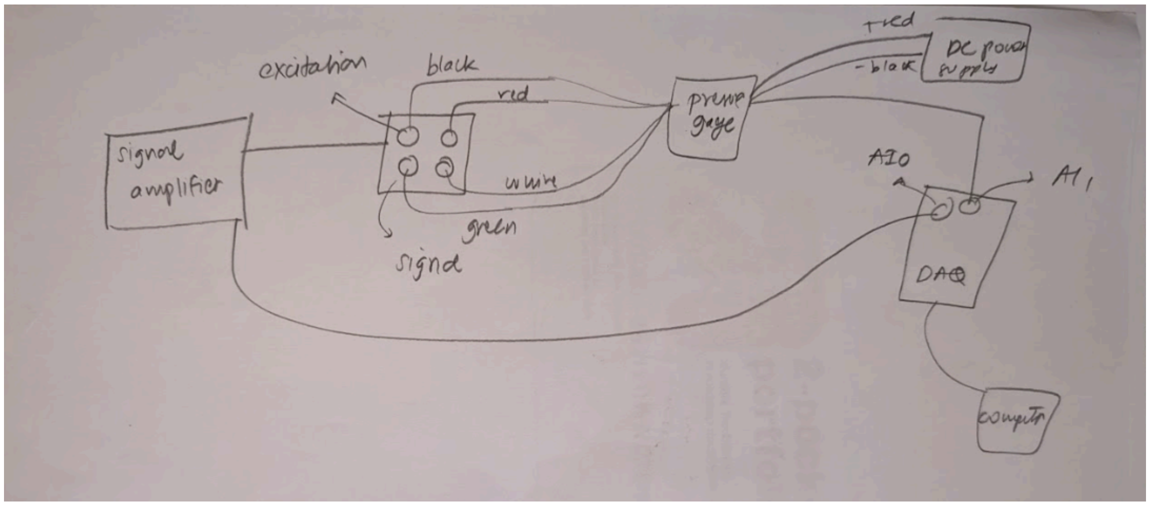


Table A.1: Properties of the Diaphragm Sensor

Parameter	Value	Unit
Material	Phosphor Bronze	
Thickness	0.795	mm
Diameter	103.1	mm
Elastic Modulus, E_m	102	GPa
Density, ρ_d	8.9	$\frac{g}{cm^3}$
Inner Gauge Radius	4.06	mm
Outer Gauge Radius	46.7	mm
Poisson's Ratio, ν	0.33	

Table A.2: Statistics for Initial Investigation of System Sensitivity and Accuracy

Derived Theoretical Sensitivity (V/psi)	0.3448
ADC Resolution (V)	0.000153
ADC Resolution (psi)	0.000229
Reference Pressure Transducer Accuracy (psi)	0.0725
Overall Uncertainty (psi)	0.072501

Appendix B: Tables, Figures, Equations for Static Calibration and Analysis

Figure B.1: Linear Regression Line for Static Calibration Data

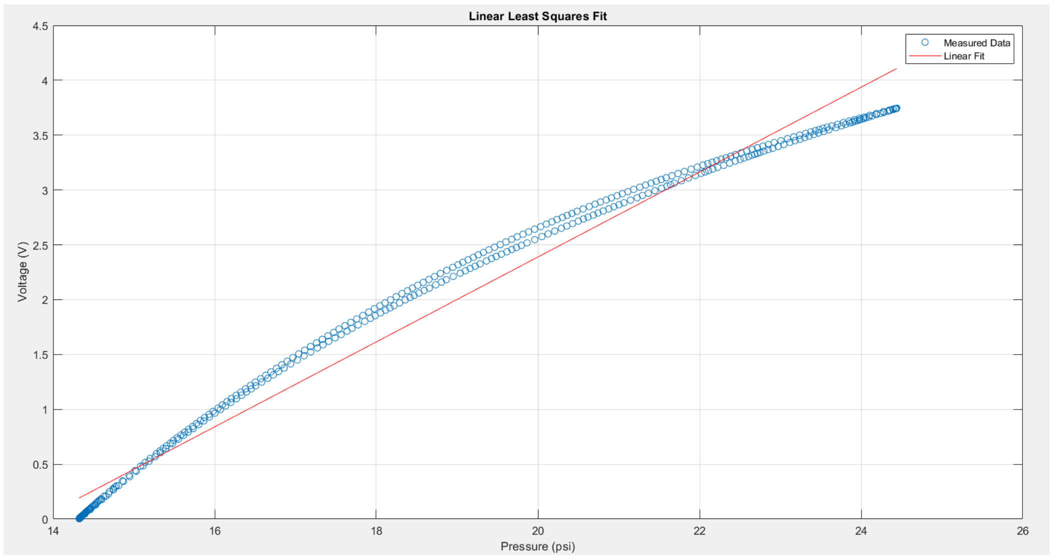


Figure B.2: Nonlinear Regression Curve for Static Calibration Data

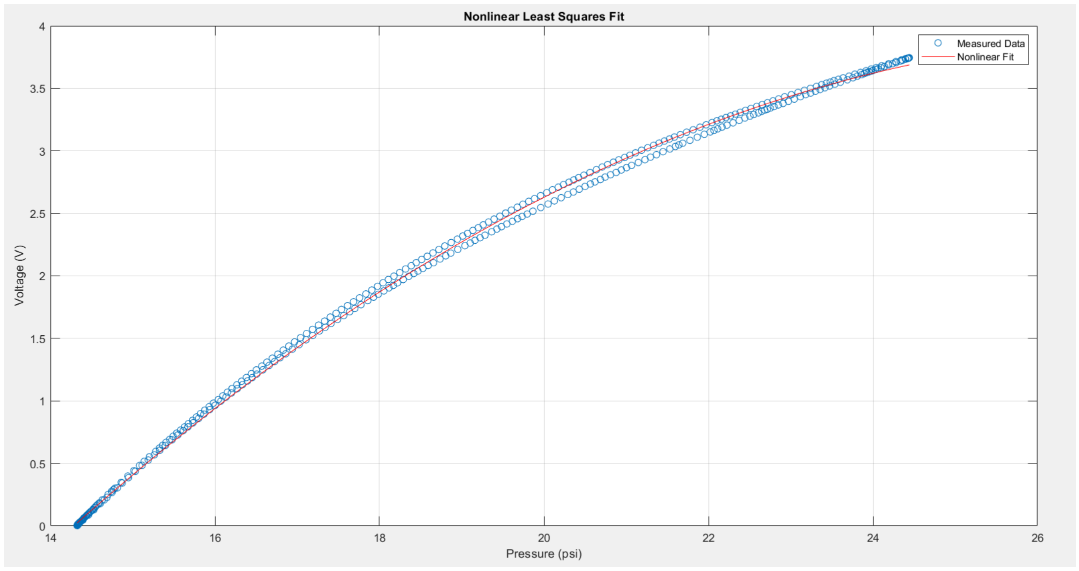


Figure B.3: Static Calibration Data Plot

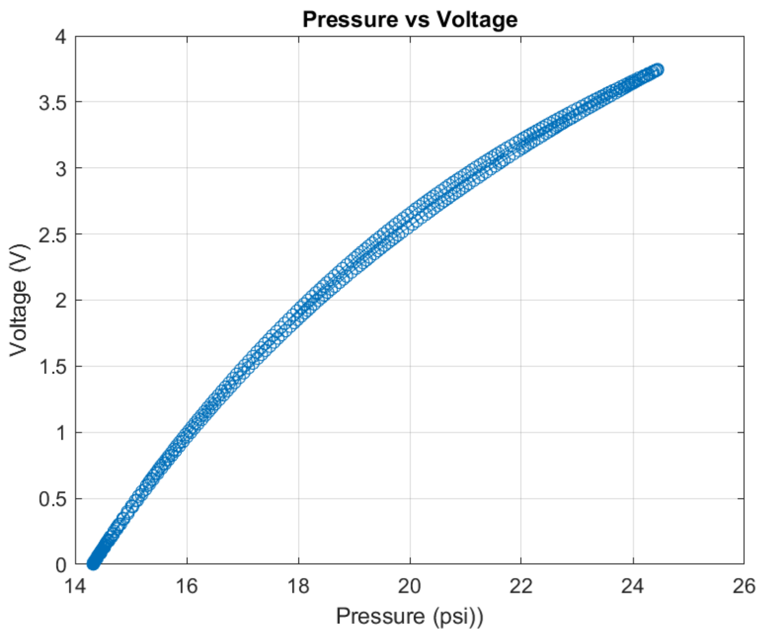


Table B.1: Comparison of Measured Sensitivity to Theoretical (Ideal) Sensitivity

Measured Static Sensitivity (V/psi)	0.61574
Theoretical Static Sensitivity (V/psi)	0.3448
Difference (V/psi)	0.27094

Table B.2: Calculated Errors for Linear Least Squares Regression

Hysteresis Error (V)	1.869943
Total Error about Linear Fit (V)	1.880456

Table B.3: Comparison of Error for Linear/Nonlinear Least Squares Regression

Linear Least Squares Regression		**Nonlinear Least Squares Regression**	
Total Error (V)	1.880456	Total Error (V)	1.87109
Correlation Coeff.	0.978775	Correlation Coeff.	0.999276

<u>Appendix C: Tables, Figures, Equations, for Static Calibration and Analysis</u>

Figure C.1: Impulse Responses in Time for Dynamic Calibration Data

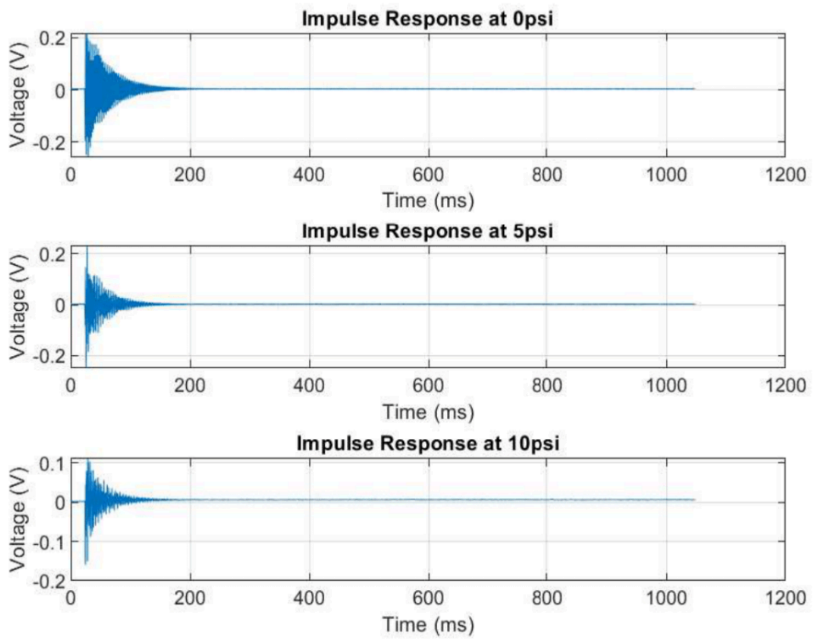


Figure C.2: Impulse Response Signal Spectra

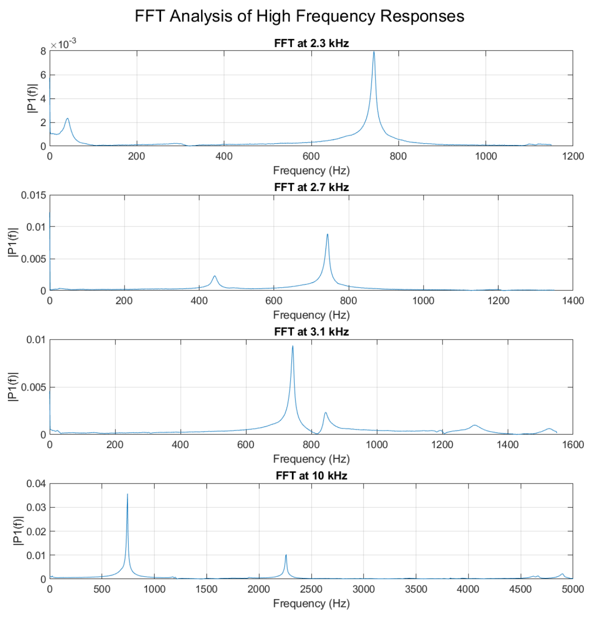


Table C.1: Dynamic System Parameters at Various Pressures

Pressure	Natural Frequency	Damping Ratio
0 psi	557.21 Hz	0.006
5 psi	630.87 Hz	0.006
10 psi	602.87 Hz	0.003

Table C.2: Locations of Largest Spectral Peaks for Dynamic Calibration Data

Pressure vs Frequency	Peak 1	Peak 2	Peak 3
0 psi	577.69 Hz	615.82 Hz	617.73 Hz
5 psi	737.85 Hz	251.67 Hz	695.90 Hz
10 psi	891.33 Hz	376.55 Hz	859.87 Hz

Table C.3: Spectral Peak Location Variation between Time/Frequency Domains

Pressure	Time Domain Frequency	Frequency Domain Frequency	Percent Difference
0 psi	557.21 Hz	577.69 Hz	3.6753
5 psi	630.87 Hz	737.85 Hz	16.956
10 psi	602.87 Hz	0	100

Table C.4: System Bandwidth and Frequency Content at Varying Pressures

Pressure	Peak Frequency	Lower Frequency	Upper Frequency	Bandwidth	Q Factor
0 psi	577.69 Hz	573.88 Hz	580.55 Hz	6.673	86.571
5 psi	737.85 Hz	733.08 Hz	740.71 Hz	7.6263	96.75
10 psi	0 Hz	0 Hz	0 Hz	0	NaN

Table C.5: Sample Rate, Nyquist Frequency, Second Natural Frequency, Aliased Frequency, Percent Discrepancy for Dynamic Calibration Data

Frequency	fs (Hz)	fN (Hz)	f (Hz)	f1 (Hz)	f1_exp (Hz)	Discrepancy
2.3 KHz	2300	1150	743.28	743.28	40.563	94.54277
2.7 KHz	2700	1350	742.56	742.56	441.42	40.55459
3.1 KHz	3100	1550	743.23	743.23	843.71	13.51889
10 KHz	10,000	5000	743.57	743.57	2259.3	203.8462

Pressure range:

Min: 0.01780916 kpa (0.002583 psi)

Max: 25.823624 kpa (3.7454 psi)

Excitation: 10 volts

Output range:

Min: 14.323 volts

Max: 24.4412 volts

Static sensitivity: 1.6958 V/kpa

Bandwidth: 4.77 ± 4.16 Hz

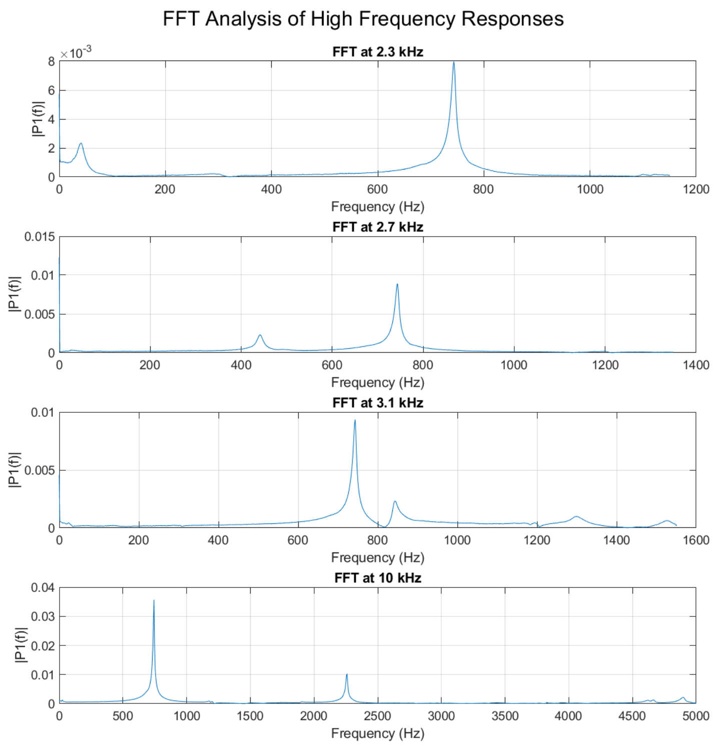
Minimum sampling rate: 2300.01 Hz

Schematic of sensor:

Static calibration curve:

```
function [f, P1] = get_fft_data(time, voltage)
```

Frequency Response:



Frequency Response

Errors:

Straight line fit

Equation:

Curve fit error: 13.7895 kpa

Hysteresis error: 5.0566 kpa

Total error: 5.0820 kpa

Cubic curve fit

Equation:

Curve fit error: 13.7895 kpa

Hysteresis error: 5.0566 kpa

Total error: 5.057894 kpa

Dynamic Performance: (0 psi, 5 psi, 10 psi respectively)

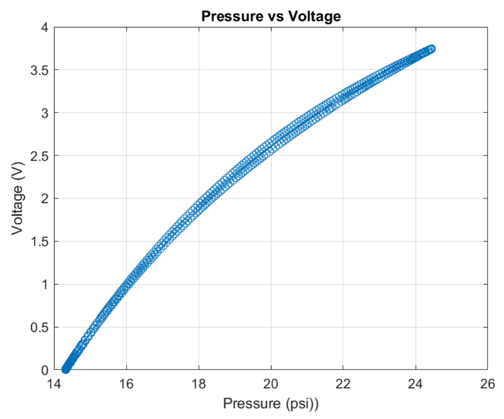
Diaphragm natural frequency: 557.21 Hz, 630.87 Hz, 602.87 Hz

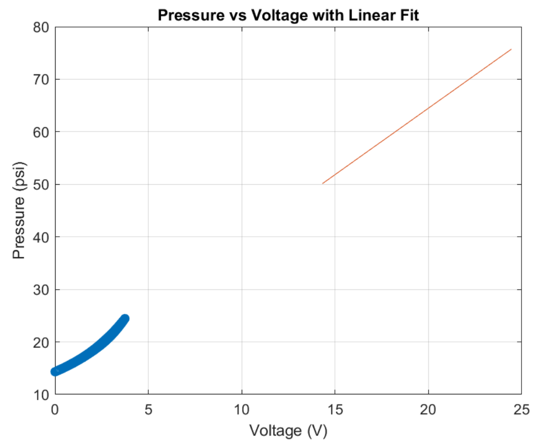
Diaphragm damping ratio: 0.006, 0.006, 0.003

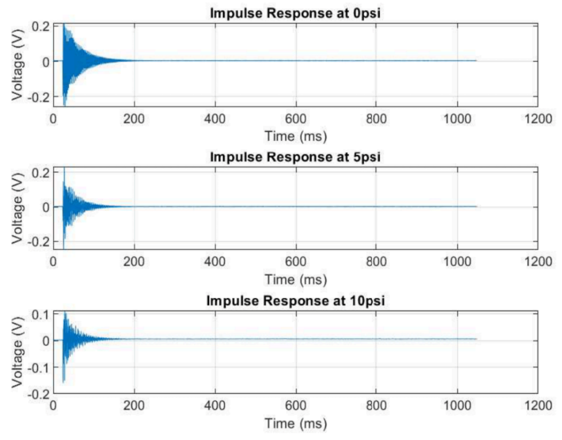
NEED TO ADD WORKSHEET A-E

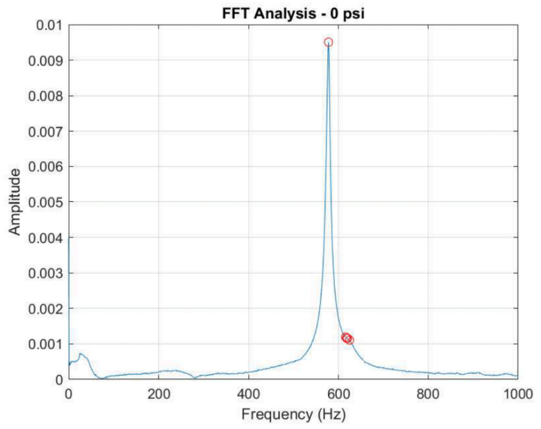
I. Conclusion

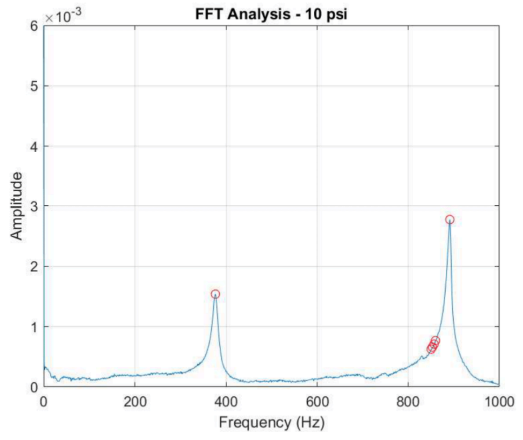
II. Appendix

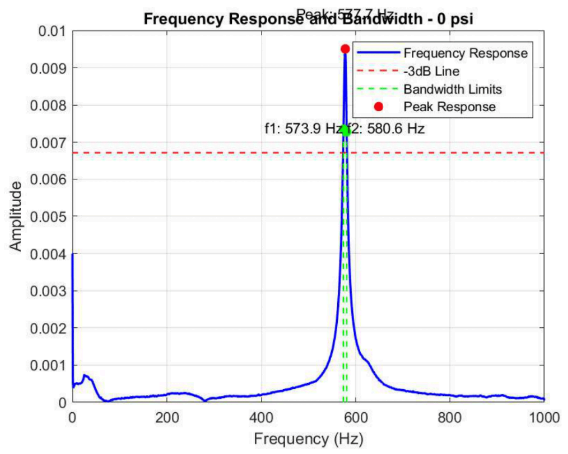

Pressure vs Voltage
Voltage (V)
Pressure (psi))

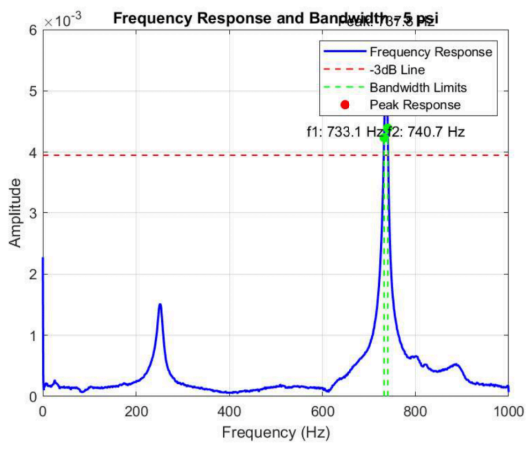

Pressure vs Voltage with Linear Fit
Pressure (psi)
Voltage (V)

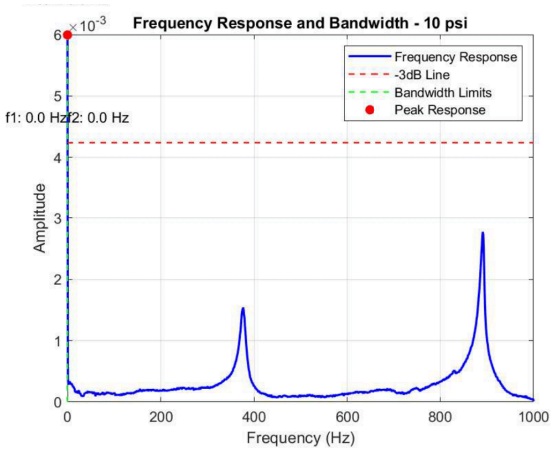

Impulse Response at 0psi
Impulse Response at 5psi
Impulse Response at 10psi
Voltage (V)
Time (ms)

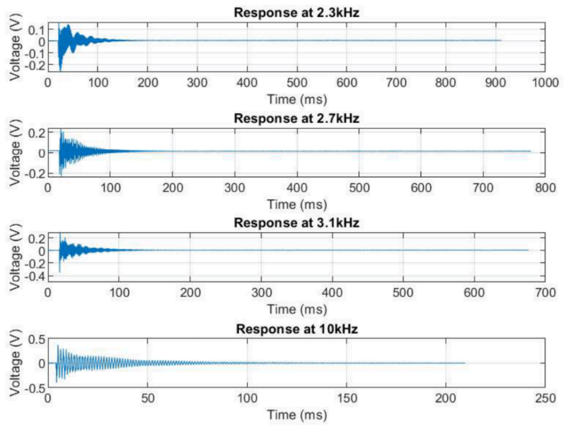

FFT Analysis - 0 psi
Amplitude
Frequency (Hz)


FFT Analysis - 10 psi
×10⁻³
Amplitude
Frequency (Hz)


Frequency Response and Bandwidth - 0 psi
Peak: 577.7 Hz
Frequency Response
-3dB Line
Bandwidth Limits
Peak Response
f1: 573.9 Hz
f2: 580.6 Hz
Amplitude
Frequency (Hz)







```
clc; close all; clear;

%% Sensitivity Calculations

% Table 5.1:

G_F = 2.05;     % Gauge Factor
R = 350;        % Gauge Resistance (Ohms)

% Table 5.2: Air Properties
rho_0 = 1.28;   % Air Density at Atmospheric Pressure
rho_5 = 1.62;   % Air Density at 5 psi
```

```
rho_10 = 2.02;  % Air Density at 10 psi

% Table 5.3: Diaphragm Sensor Properties

t = 0.795*10^(-3);
d = 103.1*10^(-3);
E_m = 102*10^9;
rho_d = 8.9*(100^3)/1000;
r_t = 4.06*10^(-3);
r_r = 46.7*10^(-3);
v = 0.33;

p = 10^5;
V_o = 5;
V_ex = 10;

sig_rt = (3*p*(R^2)*v/(8*t^2))*(((1/v)+1)-((3/v)+1)*(r_t/R)^2);
sig_tt = (3*p*(R^2)*v/(8*t^2))*(((1/v)+1)-((1/v)+3)*(r_t/R)^2);
eps_tt = (sig_rt - v*sig_tt)/E_m;

sig_rr = (3*p*(R^2)*v/(8*t^2))*(((1/v)+1)-((3/v)+1)*(r_r/R)^2);
sig_tr = (3*p*(R^2)*v/(8*t^2))*(((1/v)+1)-((1/v)+3)*(r_r/R)^2);
eps_rr = (sig_rr - v*sig_tr)/E_m;

delta_Rt = R*G_F*eps_tt;
delta_Rr = R*G_F*eps_rr;

k_d = V_o/(p*V_ex);
k_a = V_o/(p*k_d*V_ex);
K = k_d*k_a*V_ex;
```

```
%% Load Data
data1 = load('Dynamic_0psi.txt');
data2 = load('Dynamic_5psi.txt');
data3 = load('Dynamic_10psi.txt');
data4 = load('Dynamic_5psi_2.3kHz.txt');
data5 = load('Dynamic_5psi_2.7kHz.txt');
data6 = load('Dynamic_5psi_3.1kHz.txt');
data7 = load('Dynamic_5psi_10kHz.txt');
data8 = load('Static.txt');

%% Analysis A: Reference Pressure Transducer Accuracy
%% Analysis A: Reference Pressure Transducer Accuracy

% A.1 ADC Resolution in volts
FSO = 10;  % Full Scale Output (V)
m = 16;    % Number of bits
res_ADC = FSO/(2^m);  % Resolution in volts
fprintf('A.1 ADC Resolution: %.6f V\n', res_ADC);

% A.2 ADC Resolution in pressure (psi)
% Convert voltage resolution to pressure using sensitivity
% Assuming linear relationship where FSO = 10V corresponds to max pressure of 15 psi
max_pressure = 15;  % psi
voltage_to_pressure = max_pressure/FSO;  % psi/V
res_pressure = res_ADC * voltage_to_pressure;
fprintf('A.2 ADC Resolution in pressure: %.6f psi\n', res_pressure);

% A.3 Accuracy (uncertainty) of reference pressure transducer
% Typically specified as percentage of FSO
```

```
accuracy_percentage = 0.25;  % 0.25% of FSO (typical value for precision pressure transducers)
accuracy_pressure = (accuracy_percentage/100) * max_pressure;
fprintf('A.3 Pressure Transducer Accuracy: %.6f psi\n', accuracy_pressure);

% A.4 Overall uncertainty calculation
% Combine ADC resolution and transducer accuracy using RSS method
overall_uncertainty = sqrt(res_pressure^2 + accuracy_pressure^2);
fprintf('A.4 Overall Uncertainty: %.6f psi\n', overall_uncertainty);

% A.5 Determine dominant uncertainty term
fprintf('A.5 Dominant Uncertainty Analysis:\n');
fprintf('ADC Resolution: %.6f psi\n', res_pressure);
fprintf('Transducer Accuracy: %.6f psi\n', accuracy_pressure);

if res_pressure > accuracy_pressure
   fprintf('ADC Resolution dominates the uncertainty\n');
else
   fprintf('Pressure Transducer Accuracy dominates the uncertainty\n');
end

% Calculate percentage contribution of each term
total_variance = res_pressure^2 + accuracy_pressure^2;
adc_contribution = (res_pressure^2/total_variance) * 100;
transducer_contribution = (accuracy_pressure^2/total_variance) * 100;

fprintf('ADC Resolution contribution: %.1f%%\n', adc_contribution);
fprintf('Transducer Accuracy contribution: %.1f%%\n', transducer_contribution);



%% Analysis B: Static Calibration of the Diaphragm Pressure Transducer
```

```
% B.1

% Assign Variables
staticPressure = data8(:, 1);
staticVoltage = data8(:, 2);

% Plotting
figure;
plot(staticVoltage, staticPressure, '-o');
xlabel('Pressure (psi))');
ylabel('Voltage (V)');
title('Pressure vs Voltage');
grid on;

saveas(gcf, 'Pressure_vs_Voltage.png');

%B.2

% Display minimum and maximum pressure & voltage
disp(['Minimum Pressure: ', num2str(min(staticPressure)), ' psi']);
disp(['Maximum Pressure: ', num2str(max(staticPressure)), ' psi']);
disp(['Minimum Voltage: ', num2str(min(staticVoltage)), ' V']);
disp(['Maximum Voltage: ', num2str(max(staticVoltage)), ' V']);

% Pressure range
pressure_range = (staticPressure >= 0 & staticPressure <= 0.5);
voltage_in_range = staticVoltage(pressure_range);
pressure_in_range = staticPressure(pressure_range);
```

```
% Calculate the static sensitivity (slope)
sensitivity = (max(voltage_in_range) - min(voltage_in_range)) / (max(pressure_in_range) -
min(pressure_in_range) * 6.89475729);
% 6.89475729 factor to convert psi to kpa

% Display the static sensitivity
disp(['Static Sensitivity: ', num2str(sensitivity), ' V/kpa']);

%B.3
% Linear fit line
p = polyfit(staticPressure, staticVoltage, 1);
experimental_sensitivity = p(1)/1.6958;

% Display the experimental sensitivity
disp(['Experimental Static Sensitivity at origin: ', num2str(experimental_sensitivity), ' V/kpa']);

% plotting
voltage_fit = linspace(min(staticVoltage), max(staticVoltage), 100);
pressure_fit = polyval(p, voltage_fit);
figure;
plot(staticPressure, staticVoltage, 'o'); % Original data
hold on;
plot(voltage_fit, pressure_fit, '-'); % Linear fit
xlabel('Voltage (V)');
ylabel('Pressure (psi)');
title('Pressure vs Voltage with Linear Fit');
grid on;
saveas(gcf, 'Pressure_vs_Voltage_With_Linear_Fit.png');

diffSensitivity = sensitivity - experimental_sensitivity;
```

```
fprintf('The difference of the sensitivity is: %.5f V/kpa', diffSensitivity);

%% Analysis C:

%% Analysis C: Total Error - Linear Fit

% C.1 Least Squares Regression

% Prepare data
x = staticPressure;  % Pressure data
y = staticVoltage;   % Voltage data
n = length(x);

% Calculate linear regression coefficients
X = [ones(n,1) x];  % Design matrix
beta = (X'*X)\(X'*y);  % Solve normal equations
b = beta(1);  % Intercept
m = beta(2);  % Slope

% Calculate fitted values
y_fit = X*beta;

% Plot data and fit
figure;
plot(x, y, 'o', 'DisplayName', 'Measured Data');
hold on;
plot(x, y_fit, '-r', 'DisplayName', 'Linear Fit');
xlabel('Pressure (psi)');
ylabel('Voltage (V)');
title('Linear Least Squares Fit');
```

```
legend('show');
grid on;
saveas(gcf, 'Linear_Fit.png');

% C.2 Calculate Error due to Curve Fit

% Calculate residuals and standard error
residuals = y - y_fit;
SSE = sum(residuals.^2);
MSE = SSE/(n-2);  % Mean squared error
SE = sqrt(MSE);   % Standard error

% Calculate confidence intervals
alpha = 0.05;  % 95% confidence level
t_crit = tinv(1-alpha/2, n-2);
sigma_m = SE*sqrt(1/(sum((x-mean(x)).^2)));  % Standard error of slope
sigma_b = SE*sqrt(1/n + mean(x)^2/(sum((x-mean(x)).^2)));  % Standard error of intercept

fprintf('Standard Error: %.6f V\n', SE);
fprintf('Slope: %.6f ± %.6f V/psi\n', m, t_crit*sigma_m);
fprintf('Intercept: %.6f ± %.6f V\n', b, t_crit*sigma_b);

% C.3 Calculate Hysteresis Error

% Separate increasing and decreasing pressure data
% Assuming first half of data is increasing and second half is decreasing
mid_point = floor(n/2);
increasing_idx = 1:mid_point;
decreasing_idx = (mid_point+1):n;
```

```
% Calculate maximum difference between increasing and decreasing curves
max_diff = max(abs(y(increasing_idx) - y(decreasing_idx)));
hysteresis_error = max_diff/2;  % Half of maximum difference

fprintf('Hysteresis Error: %.6f V\n', hysteresis_error);

% C.4 Calculate Total Error
fprintf('\nC.4 Total Error:\n');

% Combine errors using root sum square (RSS) method
% Include ADC resolution from part A
total_error_linear = sqrt(SE^2 + hysteresis_error^2 + res_ADC^2);
fprintf('Total Error (Linear Fit): %.6f V\n', total_error_linear);

%% Analysis D: Total Error - Nonlinear Fit

% D.1 Nonlinear Least Squares Regression
X_quad = [ones(n,1) x x.^2];
beta_quad = (X_quad'*X_quad)\(X_quad'*y);

% Calculate fitted values
y_fit_nonlin = X_quad*beta_quad;

% Plot data and nonlinear fit
figure;
plot(x, y, 'o', 'DisplayName', 'Measured Data');
hold on;
plot(x, y_fit_nonlin, '-r', 'DisplayName', 'Nonlinear Fit');
xlabel('Pressure (psi)');
ylabel('Voltage (V)');
```

```
title('Nonlinear Least Squares Fit');
legend('show');
grid on;
saveas(gcf, 'Nonlinear_Fit.png');

% D.2 Calculate Error at p = 2 psi
fprintf('\nD.2 Curve Fit Error at p = 2 psi:\n');

% Calculate residuals and MSE for nonlinear fit
residuals_nonlin = y - y_fit_nonlin;
MSE_nonlin = sum(residuals_nonlin.^2)/(n-3);  % 3 parameters in quadratic fit
SE_nonlin = sqrt(MSE_nonlin);

% Calculate predicted value at 2 psi
p_test = 2;
X_pred = [1 p_test p_test^2];
y_pred = X_pred*beta_quad;

% Calculate prediction interval
X_cov = X_quad'*X_quad;
variance_pred = MSE_nonlin*(1 + X_pred*(X_cov\X_pred'));
SE_pred = sqrt(variance_pred);

fprintf('Predicted Value at 2 psi: %.6f V\n', y_pred);
fprintf('Standard Error of Prediction: %.6f V\n', SE_pred);

% D.3 Calculate Total Error
fprintf('\nD.3 Total Error:\n');

% Combine errors using RSS method
```

```
total_error_nonlin = sqrt(SE_pred^2 + hysteresis_error^2 + res_ADC^2);
fprintf('Total Error (Nonlinear Fit): %.6f V\n', total_error_nonlin);

% Compare linear and nonlinear fits
fprintf('\nComparison of Fits:\n');
fprintf('R-squared (Linear): %.6f\n', 1 - sum(residuals.^2)/sum((y-mean(y)).^2));
fprintf('R-squared (Nonlinear): %.6f\n', 1 - sum(residuals_nonlin.^2)/sum((y-mean(y)).^2));

%% PART E

%% Analysis E: Diaphragm Pressure Transducer Dynamics

% E.1 Plot Impulse Responses
pressures = {'0psi', '5psi', '10psi'};
figure('Visible', 'on');

for i = 1:length(pressures)
    data = eval(['data', num2str(i)]);  % Get corresponding dataset
    time = data(:,1);
    voltage = data(:,2);

    subplot(3,1,i);
    plot(time*1000, voltage);  % Convert time to milliseconds
    title(['Impulse Response at ', pressures{i}]);
    xlabel('Time (ms)');
    ylabel('Voltage (V)');
    grid on;
end

%% E.2 Calculate Natural Frequency and Damping Ratio
```

```
% Initialize arrays to store results
pressures = {'0 psi', '5 psi', '10 psi'};
natural_freq = zeros(3,1);
damping_ratios = zeros(3,1);

% Process each pressure case
for i = 1:3
    % Get current dataset
    if i == 1
        data = data1;
    elseif i == 2
        data = data2;
    else
        data = data3;
    end

    time = data(:,1);
    voltage = data(:,2);

    % Find peaks
    [peaks, locs] = findpeaks(voltage, 'MinPeakHeight', max(voltage)*0.1);

    % Calculate time between peaks
    peak_times = time(locs);
    periods = diff(peak_times);
    avg_period = mean(periods);

    % Natural frequency
    natural_freq(i) = 1/avg_period;
    % Calculate damping ratio using logarithmic decrement
    peak_ratios = log(peaks(1:end-1)./peaks(2:end));
    delta = mean(peak_ratios);
    damping_ratios(i) = delta/sqrt(4*pi^2 + delta^2);

    % Display results
    fprintf('\nResults for %s:\n', pressures{i});
    fprintf('Natural Frequency: %.2f Hz\n', natural_freq(i));
    fprintf('Damping Ratio: %.3f\n', damping_ratios(i));
end

%% E.3 FFT Analysis of Impulse Responses

% Process each pressure case
for i = 1:3
    % Get current dataset
    if i == 1
        data = data1;
    elseif i == 2
        data = data2;
    else
        data = data3;
    end

    time = data(:,1);
    voltage = data(:,2);

    % Calculate FFT
    Fs = 1/mean(diff(time));    % Sampling frequency
```

```
L = length(voltage);        % Length of signal
Y = fft(voltage);

% Compute single-sided spectrum
P2 = abs(Y/L);
P1 = P2(1:floor(L/2+1));
P1(2:end-1) = 2*P1(2:end-1);

% Define frequency domain
f = Fs*(0:(L/2))/L;

% Find peaks in frequency domain
[peaks_fft, locs_fft] = findpeaks(P1, 'MinPeakHeight', max(P1)*0.1);
freq_peaks = f(locs_fft);

% Sort peaks to find dominant frequencies
[sorted_peaks, sort_idx] = sort(peaks_fft, 'descend');
dominant_freqs = freq_peaks(sort_idx);

% Plot FFT
figure;
plot(f, P1);
title(['FFT Analysis - ', pressures{i}]);
xlabel('Frequency (Hz)');
ylabel('Amplitude');
grid on;

% Add markers for peaks
hold on;
plot(freq_peaks, peaks_fft, 'ro');
```

```
    % Display dominant frequencies
    fprintf('\nFFT Results for %s:\n', pressures{i});
    fprintf('Dominant Frequencies:\n');
    for j = 1:min(3,length(dominant_freqs))
        fprintf('Peak %d: %.2f Hz\n', j, dominant_freqs(j));
    end
end

%% E.4 Compare Natural Frequencies

% Create comparison table
comparison_table = table();
comparison_table.Pressure = pressures';
comparison_table.TimeDomainFreq = natural_freq;

% Add frequency domain results
freq_domain_peaks = zeros(3,1);
for i = 1:3
    if i == 1
        data = data1;
    elseif i == 2
        data = data2;
    else
        data = data3;
    end

    time = data(:,1);
    voltage = data(:,2);
    % Calculate FFT
    Fs = 1/mean(diff(time));
    L = length(voltage);
    Y = fft(voltage);
    P2 = abs(Y/L);
    P1 = P2(1:floor(L/2+1));
    P1(2:end-1) = 2*P1(2:end-1);
    f = Fs*(0:(L/2))/L;

    % Find dominant frequency
    [~, max_idx] = max(P1);
    freq_domain_peaks(i) = f(max_idx);
end

comparison_table.FreqDomainFreq = freq_domain_peaks;
comparison_table.PercentDiff = abs(comparison_table.TimeDomainFreq -
comparison_table.FreqDomainFreq) ./ ...
                              comparison_table.TimeDomainFreq * 100;

% Display comparison results
fprintf('\nComparison of Natural Frequencies:\n');
disp(comparison_table);

% Calculate average difference
avg_diff = mean(comparison_table.PercentDiff);
fprintf('\nAverage Percent Difference: %.2f%%\n', avg_diff);

% Comment on the comparison
fprintf('\nAnalysis of Results:\n');
if avg_diff < 5
```

```
    fprintf('The time domain and frequency domain results show good agreement (<%%
difference).\n');
else
    fprintf('There are notable differences between time and frequency domain results.\n');
    fprintf('This could be due to:\n');
    fprintf('1. Noise in the signal\n');
    fprintf('2. Non-linear effects in the system\n');
    fprintf('3. Sampling rate limitations\n');
end

%% E.5 Calculate Bandwidth (Useful Frequency Range)

fprintf('\nE.5 Bandwidth Analysis:\n');

% Process each pressure case
for i = 1:3
    % Select appropriate dataset
    if i == 1
        data = data1;
        pressure = '0 psi';
    elseif i == 2
        data = data2;
        pressure = '5 psi';
    else
        data = data3;
        pressure = '10 psi';
    end

    time = data(:,1);
    voltage = data(:,2);
```

```
% Calculate FFT
Fs = 1/mean(diff(time));    % Sampling frequency
L = length(voltage);         % Signal length
Y = fft(voltage);

% Compute single-sided amplitude spectrum
P2 = abs(Y/L);
P1 = P2(1:floor(L/2+1));
P1(2:end-1) = 2*P1(2:end-1);

% Frequency vector
f = Fs*(0:(L/2))/L;

% Find maximum amplitude and corresponding frequency
[max_amp, max_idx] = max(P1);
peak_freq = f(max_idx);

% Find -3dB point (0.707 of maximum amplitude)
cutoff_amp = max_amp/sqrt(2);  % -3dB point

% Find bandwidth (frequency range where amplitude > -3dB point)
above_cutoff = P1 >= cutoff_amp;
bandwidth_indices = find(above_cutoff);

% Calculate bandwidth
lower_freq = f(min(bandwidth_indices));
upper_freq = f(max(bandwidth_indices));
bandwidth = upper_freq - lower_freq;
```

```
% Plot frequency response with bandwidth markers
figure;

% Plot full spectrum
plot(f, P1, 'b-', 'LineWidth', 1.5);
hold on;

% Plot -3dB line
plot([0 max(f)], [cutoff_amp cutoff_amp], 'r--', 'LineWidth', 1);

% Plot bandwidth points
plot([lower_freq lower_freq], [0 P1(bandwidth_indices(1))], 'g--', 'LineWidth', 1);
plot([upper_freq upper_freq], [0 P1(bandwidth_indices(end))], 'g--', 'LineWidth', 1);

% Add markers for bandwidth points
plot(lower_freq, P1(bandwidth_indices(1)), 'go', 'MarkerFaceColor', 'g');
plot(upper_freq, P1(bandwidth_indices(end)), 'go', 'MarkerFaceColor', 'g');

% Plot peak
plot(peak_freq, max_amp, 'ro', 'MarkerFaceColor', 'r');

% Customize plot
grid on;
title(['Frequency Response and Bandwidth - ' pressure]);
xlabel('Frequency (Hz)');
ylabel('Amplitude');
legend('Frequency Response', '-3dB Line', 'Bandwidth Limits', '', '', '', 'Peak Response');

% Add text annotations
text(peak_freq, max_amp*1.1, sprintf('Peak: %.1f Hz', peak_freq), ...
        'HorizontalAlignment', 'center');
    text(lower_freq, cutoff_amp*1.1, sprintf('f1: %.1f Hz', lower_freq), ...
        'HorizontalAlignment', 'right');
    text(upper_freq, cutoff_amp*1.1, sprintf('f2: %.1f Hz', upper_freq), ...
        'HorizontalAlignment', 'left');

    % Display results
    fprintf('\nResults for %s:\n', pressure);
    fprintf('Peak Frequency: %.2f Hz\n', peak_freq);
    fprintf('Lower Cutoff Frequency (-3dB): %.2f Hz\n', lower_freq);
    fprintf('Upper Cutoff Frequency (-3dB): %.2f Hz\n', upper_freq);
    fprintf('Bandwidth: %.2f Hz\n', bandwidth);
    fprintf('Quality Factor (Q = f_peak/bandwidth): %.2f\n', peak_freq/bandwidth);
end

% Create summary table
bandwidth_table = table();
bandwidth_table.Pressure = {'0 psi'; '5 psi'; '10 psi'};
bandwidth_table.PeakFreq = zeros(3,1);
bandwidth_table.LowerFreq = zeros(3,1);
bandwidth_table.UpperFreq = zeros(3,1);
bandwidth_table.Bandwidth = zeros(3,1);
bandwidth_table.QFactor = zeros(3,1);

% Fill table with calculated values
for i = 1:3
    if i == 1
        data = data1;
    elseif i == 2
        data = data2;
    else
        data = data3;
    end

    time = data(:,1);
    voltage = data(:,2);

    % Calculate FFT
    Fs = 1/mean(diff(time));
    L = length(voltage);
    Y = fft(voltage);
    P2 = abs(Y/L);
    P1 = P2(1:floor(L/2+1));
    P1(2:end-1) = 2*P1(2:end-1);
    f = Fs*(0:(L/2))/L;

    [max_amp, max_idx] = max(P1);
    peak_freq = f(max_idx);
    cutoff_amp = max_amp/sqrt(2);
    above_cutoff = P1 >= cutoff_amp;
    bandwidth_indices = find(above_cutoff);

    bandwidth_table.PeakFreq(i) = peak_freq;
    bandwidth_table.LowerFreq(i) = f(min(bandwidth_indices));
    bandwidth_table.UpperFreq(i) = f(max(bandwidth_indices));
    bandwidth_table.Bandwidth(i) = f(max(bandwidth_indices)) - f(min(bandwidth_indices));
    bandwidth_table.QFactor(i) = peak_freq/bandwidth_table.Bandwidth(i);
end

% Display summary table
```

```
fprintf('\nBandwidth Analysis Summary:\n');
disp(bandwidth_table);

% Calculate average bandwidth and standard deviation
avg_bandwidth = mean(bandwidth_table.Bandwidth);
std_bandwidth = std(bandwidth_table.Bandwidth);
fprintf('\nAverage Bandwidth: %.2f ± %.2f Hz\n', avg_bandwidth, std_bandwidth);

%% Analysis F: Aliasing Investigation

% F.1 Plot Impulse Responses

% Load high frequency response data
frequencies = {'2.3kHz', '2.7kHz', '3.1kHz', '10kHz'};
figure('Visible', 'on');

for i = 1:length(frequencies)
    data = eval(['data', num2str(i+3)]);  % data4 through data7
    time = data(:,1);
    voltage = data(:,2);

    subplot(4,1,i);
    plot(time*1000, voltage);
    title(['Response at ', frequencies{i}]);
    xlabel('Time (ms)');
    ylabel('Voltage (V)');
    grid on;
end

%% F.2 Plot FFT of Impulse Responses
```

```
figure;

% FFT for 2.3 kHz (data4)
subplot(4,1,1);
time = data4(:,1);
voltage = data4(:,2);
Fs = 1/mean(diff(time));    % Sampling frequency
L = length(voltage);        % Length of signal
Y = fft(voltage);
P2 = abs(Y/L);
P1 = P2(1:floor(L/2+1));
P1(2:end-1) = 2*P1(2:end-1);
f = Fs*(0:(L/2))/L;
plot(f, P1);
title('FFT at 2.3 kHz');
xlabel('Frequency (Hz)');
ylabel('|P1(f)|');
grid on;

% FFT for 2.7 kHz (data5)
subplot(4,1,2);
time = data5(:,1);
voltage = data5(:,2);
Fs = 1/mean(diff(time));
L = length(voltage);
Y = fft(voltage);
P2 = abs(Y/L);
P1 = P2(1:floor(L/2+1));
P1(2:end-1) = 2*P1(2:end-1);
```

```
f = Fs*(0:(L/2))/L;
plot(f, P1);
title('FFT at 2.7 kHz');
xlabel('Frequency (Hz)');
ylabel('|P1(f)|');
grid on;

% FFT for 3.1 kHz (data6)
subplot(4,1,3);
time = data6(:,1);
voltage = data6(:,2);
Fs = 1/mean(diff(time));
L = length(voltage);
Y = fft(voltage);
P2 = abs(Y/L);
P1 = P2(1:floor(L/2+1));
P1(2:end-1) = 2*P1(2:end-1);
f = Fs*(0:(L/2))/L;
plot(f, P1);
title('FFT at 3.1 kHz');
xlabel('Frequency (Hz)');
ylabel('|P1(f)|');
grid on;

% FFT for 10 kHz (data7)
subplot(4,1,4);
time = data7(:,1);
voltage = data7(:,2);
Fs = 1/mean(diff(time));
L = length(voltage);
```

```
Y = fft(voltage);
P2 = abs(Y/L);
P1 = P2(1:floor(L/2+1));
P1(2:end-1) = 2*P1(2:end-1);
f = Fs*(0:(L/2))/L;
plot(f, P1);
title('FFT at 10 kHz');
xlabel('Frequency (Hz)');
ylabel('|P1(f)|');
grid on;

% Adjust figure properties
set(gcf, 'Position', [100, 100, 800, 800]);
sgtitle('FFT Analysis of High Frequency Responses');

% Save the figure
saveas(gcf, 'High_Frequency_FFTs.png');

% Print sampling rates and Nyquist frequencies
fprintf('\nSampling Rate Analysis:\n');
fprintf('2.3 kHz case - Sampling Rate: %.2f Hz, Nyquist Frequency: %.2f Hz\n', ...
    1/mean(diff(data4(:,1))), 1/(2*mean(diff(data4(:,1)))));
fprintf('2.7 kHz case - Sampling Rate: %.2f Hz, Nyquist Frequency: %.2f Hz\n', ...
    1/mean(diff(data5(:,1))), 1/(2*mean(diff(data5(:,1)))));
fprintf('3.1 kHz case - Sampling Rate: %.2f Hz, Nyquist Frequency: %.2f Hz\n', ...
    1/mean(diff(data6(:,1))), 1/(2*mean(diff(data6(:,1)))));
fprintf('10 kHz case - Sampling Rate: %.2f Hz, Nyquist Frequency: %.2f Hz\n', ...
    1/mean(diff(data7(:,1))), 1/(2*mean(diff(data7(:,1)))));

% F.3 Calculate Aliasing Data
```

```
fprintf('\nF.3 Aliasing Analysis:\n');

% Create table for aliasing data
aliasing_data = table('Size', [length(frequencies), 6], ...
    'VariableTypes', {'string', 'double', 'double', 'double', 'double', 'double'}, ...
    'VariableNames', {'Frequency', 'fs', 'fN', 'f', 'f1', 'f1_exp'});

for i = 1:length(frequencies)
    data = eval(['data', num2str(i+3)]);

    % Calculate sampling rate
    fs = 1/mean(diff(data(:,1)));

    % Calculate Nyquist frequency
    fN = fs/2;

    % Get actual and aliased frequencies from FFT
    [f, P1] = get_fft_data(data(:,1), data(:,2));
    [~, max_idx] = findpeaks(P1, 'SortStr', 'descend');
    f_actual = f(max_idx(1));  % First natural frequency
    f1_exp = f(max_idx(2));    % Second peak (aliased)

    % Calculate theoretical aliased frequency
    f1 = calculate_aliased_freq(f_actual, fs);

    % Store in table
    aliasing_data(i,:) = {frequencies{i}, fs, fN, f_actual, f1, f1_exp};
end

% Display results
```

```
disp(aliasing_data);

% Calculate percent discrepancy
aliasing_data.Discrepancy = abs(aliasing_data.f1 - aliasing_data.f1_exp) ./ aliasing_data.f1 *
100;
fprintf('\nPercent Discrepancy between f1 and f1_exp:\n');
disp([aliasing_data.Frequency, num2str(aliasing_data.Discrepancy)]);

% Helper function for aliased frequency calculation
function f1 = calculate_aliased_freq(f, fs)
    fN = fs/2;
    n = floor(f/fN);
    if mod(n,2) == 0
        f1 = mod(f, fN);
    else
        f1 = fN - mod(f, fN);
    end
    return;
end

% Helper function for FFT calculation
function [f, P1] = get_fft_data(time, voltage)
    Fs = 1/mean(diff(time));
    L = length(voltage);
    Y = fft(voltage);
    P2 = abs(Y/L);
    P1 = P2(1:floor(L/2+1));
    P1(2:end-1) = 2*P1(2:end-1);
    f = Fs*(0:(L/2))/L;
    return;
```

end

Equations

1. $e_y = ts_y$
2. $u_d^2 = e_q^2 + e_y^2 + e_h^2 + e_R^2$
3. $\delta = ln\left(\frac{y_\infty - y_0}{y_\infty - y_n}\right)$
4. $\zeta = \frac{\delta}{\sqrt{4\pi^2 n^2 + \delta^2}}$
5. $\omega_d = \frac{2\pi n}{t_n - t_0} = \omega_n \sqrt{1 - \zeta^2}$
6. $\frac{y}{x} = \frac{K}{\tau s + 1}$
7. $M = \left|\frac{y}{x}\right| = \frac{K}{\sqrt{1 + \tau^2 \omega^2}}$
8. $M = \frac{M}{K} = \frac{1}{\sqrt{1 + \tau^2 \omega^2}}$
9. $\omega = \frac{1}{\tau}\sqrt{\frac{1}{M^2} - 1}$
10. $p = mv + b$
11. $\sigma_{rt} = \frac{3*p*R^2*v}{8*t^2} * (\frac{1}{v} + 1) - (\frac{3}{v} + 1) * (\frac{r_t}{R})^2$
12. $\sigma_{tt} = \frac{3*p*R^2*v}{8*t^2} * (\frac{1}{v} + 1) - (\frac{3}{v} + 3) * (\frac{r_t}{R})^2$
13. $\varepsilon_{tt} = \frac{\sigma_{rt} - v*\sigma_{tt}}{E_m}$
14. $\sigma_{rr} = \frac{3*p*R^2*v}{8*t^2} * (\frac{1}{v} + 1) - (\frac{3}{v} + 1) * (\frac{r_t}{R})^2$

15. $\sigma_{tr} = \frac{3*p*R^2*v}{8*t^2} * (\frac{1}{v} + 1) - (\frac{3}{v} + 3) * (\frac{r_t}{R})^2$

16. $\varepsilon_{rr} = \frac{\sigma_{rr} - v*\sigma_{tr}}{E_m}$

17. $k_d = \frac{V_0}{p*V_{ex}}$

18. $k_a = \frac{V_0}{p*k_d*V_{ex}}$

19. $K = k_d * k_a * V_{ex}$