BME Faculty of Mechanical Engineering	EXPERIMENTAL METHODS	Name: Kelle Gergő
Dep. of Applied Mechanics	2nd HOMEWORK	Neptun code: GBBNUL
2024/25 I.	Deadline: see Moodle	Late submission ☒ Correction ☐
Declaration: With my signature I declare, that I did my homework myself, everything written in there is my knowledge.		Signature: Kelle Gergő

We only correct homeworks that correspond with formal requirements. Correction or late submission is only possible until the late submission deadline.

Problem

The natural frequencies and the modeshapes of a rectangular plate with thickness 4 mm and additional mass are investigated (see Fig. 1/a). During the impulse test a PCB 086C03 modal hammer is used and 3 piezoelectric CCLD accelerometers. The sensors were connected to a NI-9234 C modul and a cDAQ-9174, CompactDAQ chassis. The data is collected in matlab. The modal analysis is performed a $n \times m$ grid (see Fig. 1/b). The location ($Pfig$) of the excitation, the forces signal and the 3 acceleration signals are collected and stored in a Matlab struct.

The link for the measurement data can be found at the bottom of the page!

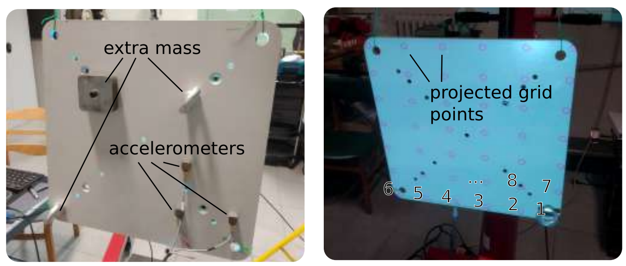

Figure 1: Measurement layout and the grid on measurement points. The figure is just an illustration.

Task

- Analyse the force signal and the signal of the 1st accelerometer. Create a proper pre-triggering and use a 0.25 seconds long signal for the analysis!
- Determine the frequency response functions for each impulse and plot them together on a logarithmic scale!
- Remove the false measurement points (caused by e.g.: prall)!
- Determine the first 5-9 natural frequencies. (Only the well separated, clearly visible frequency peaks shall be analyzed).
- Plot the modeshapes in a triangulated grid of the proper measurement points and highline the nodal lines!

Formal requirements

Create a PDF report on the measurement results and the post-processing using document processor software (e.g. Word, LaTeX). The first page must be the signed cover sheet. Please also attach the commented program code of the data analysis. Short comments and explanations must be given to each step, to such an extent that the work may be reproduced even by colleagues who are not specialized in the given topic. All questions must be answered! Do not upload pure program codes, it won't be accepted.

Download time signals: https://www.mm.bme.hu/edu/msc-en/expmeth/0hw2/10x10c.mat

MŰEGYETEM 1782

Budapest University of Technology and Economics
Faculty of Mechanical Engineering

Experimental methods in solid mechanics
II. Homework

Kelle Gergő

December 10, 2024

Contents

1 Force Signal Analysis

Task description: Analyse the force signal and the signal of the 1st accelerometer. Create a proper pre-triggering and use a 0.25 seconds long signal for the analysis!

Based on the force signal plot (Figure 1), it is observed that the force signal rises almost immediately, within the first 1 ms. Therefore, a pre-triggering mechanism is unnecessary, as the force signal does not have a noticeable delay before its onset. The corresponding acceleration signal for a duration of 0.25 s is shown in Figure 2. The acceleration exhibits a rapid response following the force application and gradually fade away over time.

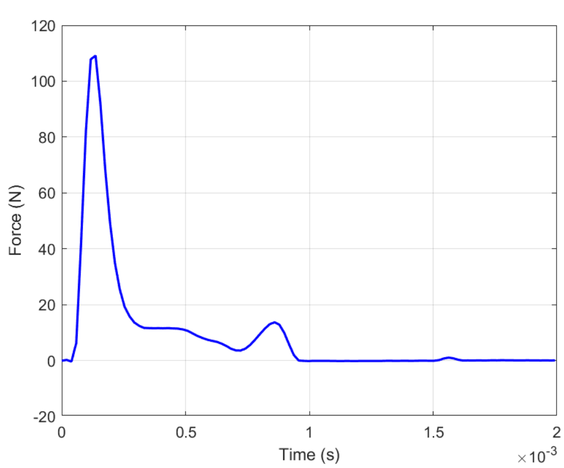

Figure 1: Force vs. Time plot for the first 2 ms.

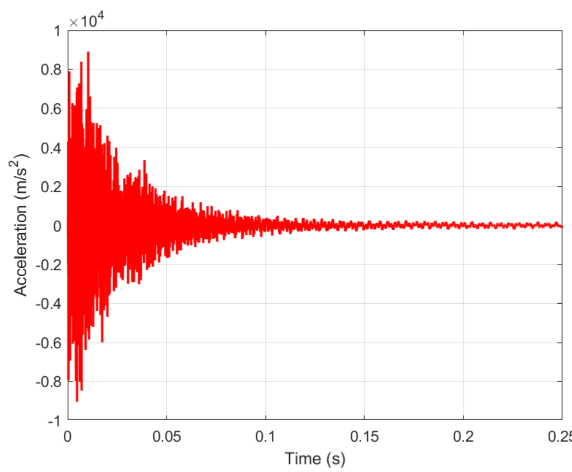

Figure 2: Acceleration vs. Time plot for 0.25 s.

2 Fequency Response Function

Task description: Determine the frequency response functions for each impulse and plot them together on a logarithmic scale!

To compute the frequency response functions (FRF) for each of the 100 measuring points, the following approach was used:

- The force signal and acceleration signal from the first accelerometer were extracted for each measuring point.
- The Fast Fourier Transform (FFT) was applied to both the force (F_{force}) and acceleration (F_{acc}) signals:

$$F_{\text{force}} = \text{FFT}(\text{force}), \quad F_{\text{acc}} = \text{FFT}(\text{acceleration}).$$

- The Frequency Response Function (FRF) for each measuring point was calculated as the ratio of the FFT of the acceleration to the FFT of the force:

$$\text{FRF} = \frac{F_{\text{acc}}}{F_{\text{force}}}.$$

- The magnitude of the FRF (|FRF|) was plotted for all measuring points across the frequency range on a logarithmic scale.

The sampling frequency (F_s) was determined using the time intervals of the signal, while the limit frequency range was determined as $F_s/2$ (Nyquist frequency).

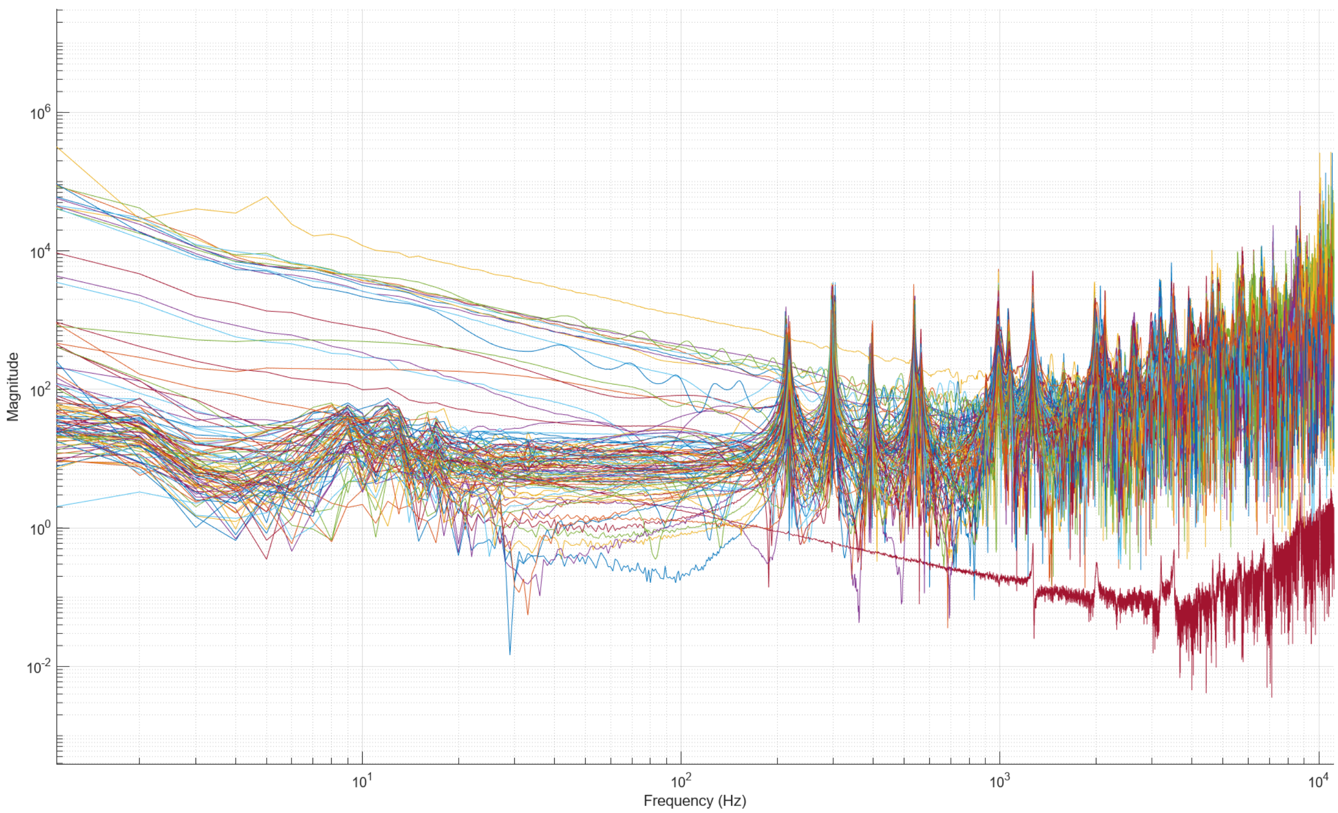

Figure 3: Frequency Response Functions (FRF) for 100 Measuring Points on a logarithmic scale.

Key observations about the FRF plot (Figure 3):

- Peaks in the FRF magnitude indicate natural frequencies of the structure. These peaks are clearly visible in the interval of 200 to 1400 Hz.
- The results confirm consistent system behavior across the 100 measuring points, with variations that may indicate local modal effects or measurement differences and mistakes.

3 Data Clearing

Task description: Remove the false measurement points (caused by e.g.: prall)!

To identify and remove false measurement points, I've plottet the force signal for each of the 100 measurement points and searched manually for anomalies. Each plot displayed the force signal restricted to the time domain $0\,\text{s} \leq t \leq 0.05\,\text{s}$, allowing for a clear view of the initial force behavior.

Points with irregular force signals, such as the presence of prall effects or noise, were flagged as bad measurements. An example of such a bad measurement is shown in Figure 4, corresponding to Point 91. The signal exhibits erratic behavior inconsistent with valid force responses, making it unsuitable for further analysis.

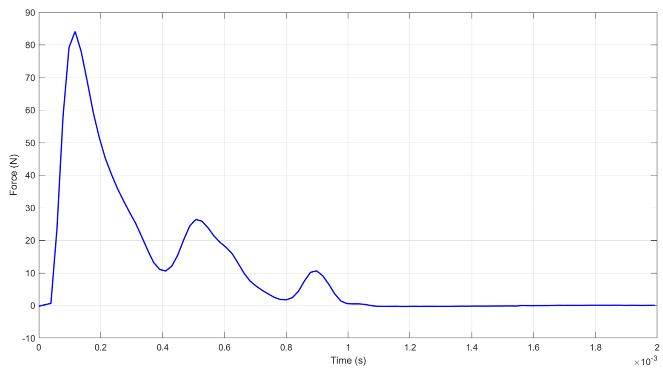

Figure 4: Force signal for Point 91 showing prall effects.

Once identified, the indices of the bad measurement points were stored in an array and removed from the dataset:

$$\text{Bad Points: } [1, 2, 5, 6, 10, 30, 57, 68, 78, 83, 90, 91, 92, 98, 100].$$

The cleaned dataset was saved to a new MATLAB file (`10x10c_cleaned.mat`), ensuring that subsequent analyses use only valid data.

4 Natural Frequencies

Task description: etermine the first 5-9 natural frequencies. (Only the well separated, clearly visible frequency peaks shall be analyzed).

To identify the natural frequencies of the system, the following steps were performed:

- The frequency response functions (FRF) for the cleaned dataset were calculated the sam way as before.
- Frequency ranges for potential natural frequencies were defined based on Figure 3:

$$\text{First: } 200 - 230\,\text{Hz}, \quad \text{Second: } 280 - 320\,\text{Hz},$$

$$\text{Third: } 370 - 420\,\text{Hz}, \quad \text{Fourth: } 490 - 560\,\text{Hz},$$

$$\text{Fifth: } 900 - 1050\,\text{Hz}, \quad \text{Sixth: } 1200 - 1300\,\text{Hz}.$$

- Within each frequency range, the maximum FRF magnitude was located across all measuring points, and the corresponding frequency was recorded as the natural frequency.
- The first 6 natural frequencies were marked in the plot alongside the FRF.

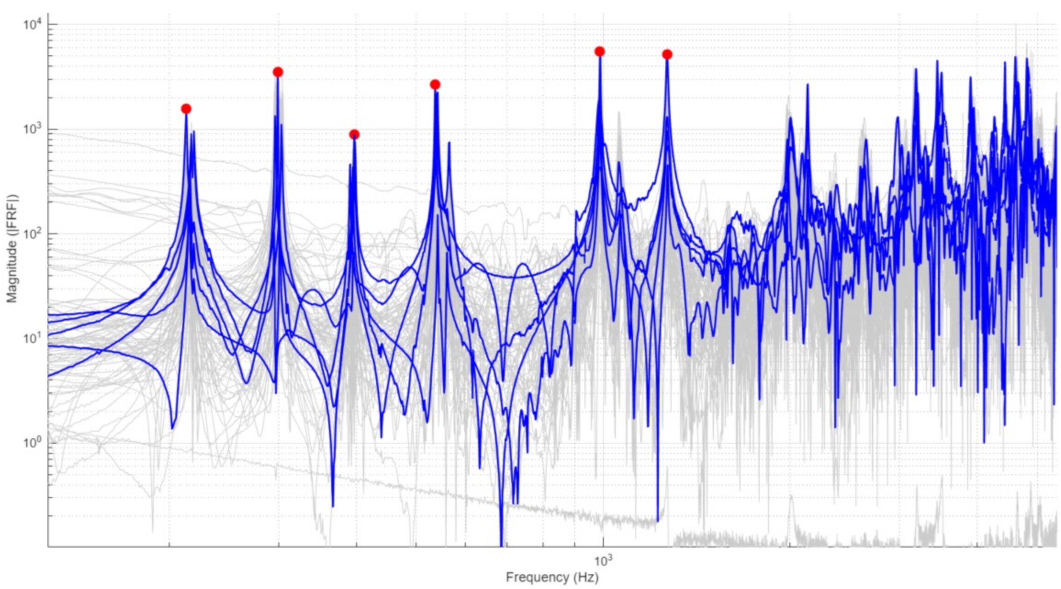

Figure 5: Frequency Response Functions (FRFs) with Natural Frequencies Highlighted.

On Figure 5 the peaks corresponding to natural frequencies are marked with red circles, and their associated FRFs are highlighted with blue. The red circles identify the following natural frequencies:

$$\text{Natural Frequencies: } 213\,\text{Hz}, 299\,\text{Hz}, 397\,\text{Hz}, 536\,\text{Hz}, 988\,\text{Hz}, 1267\,\text{Hz}.$$

With this method I could easily identify the natural frequencies by focusing on the clearly visible peaks in the FRF plot.

5 Modeshapes

Task description: Plot the modeshapes in a triangulated grid of the proper measurement points and highligh the nodal lines!

The mode shapes at the identified natural frequencies were computed using the imaginary part of the Frequency Response Functions (FRFs). Each mode shape was plotted as a 3D surface map illustrating amplitude variations with detailed contour lines highlighting nodal lines.

To account for potential errors in frequency measurement due to numerical inaccuracies, a frequency range of ±30 Hz around each natural frequency was considered. Within this range, the local maxima or minima of the imaginary part of the FRF were identified for each measuring point.

The measurement points were mapped using their `Pspace` positions, as shown in Figure 6. The modeshapes were plotted using a triangulated grid with the highlighted nodal lines showed in Figure 6

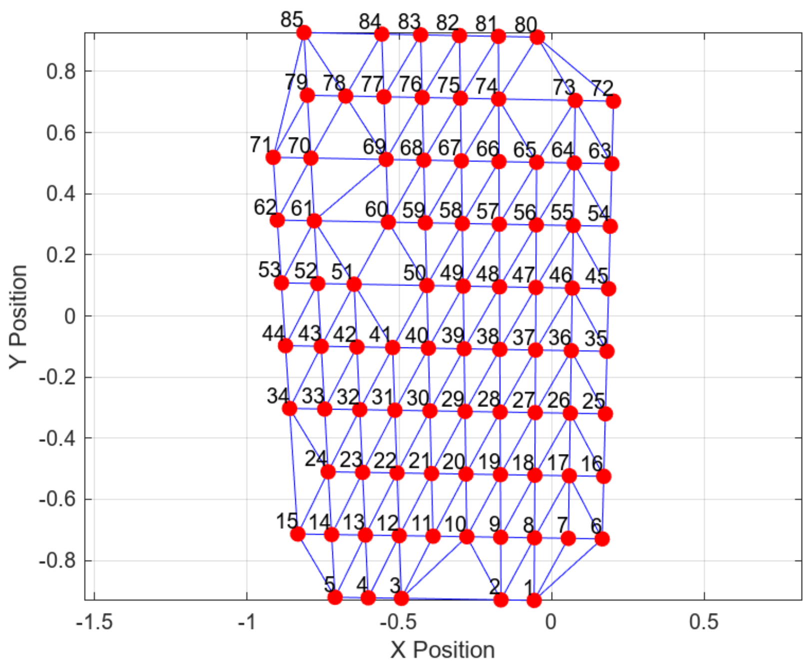

Figure 6: Triangulated grid of measurement points.

The following Figures illustrate the mode shapes at the identified natural frequencies, highlighting unique vibration patterns and nodal lines. The $\pm$ 30 Hz range ensures precise identification of local extrema in the imaginary part, accounting for measurement variations.

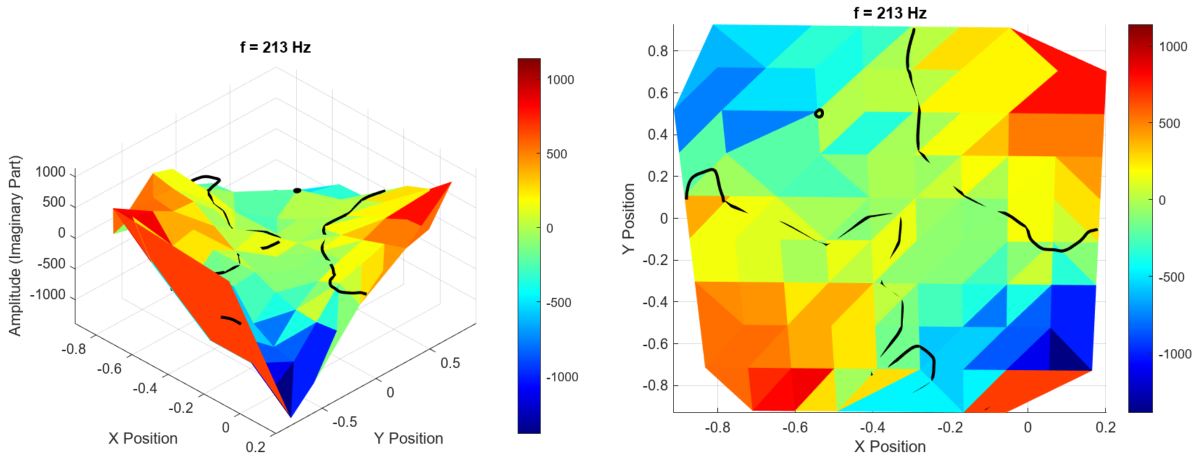

Figure 7: Mode shape at 213 Hz.

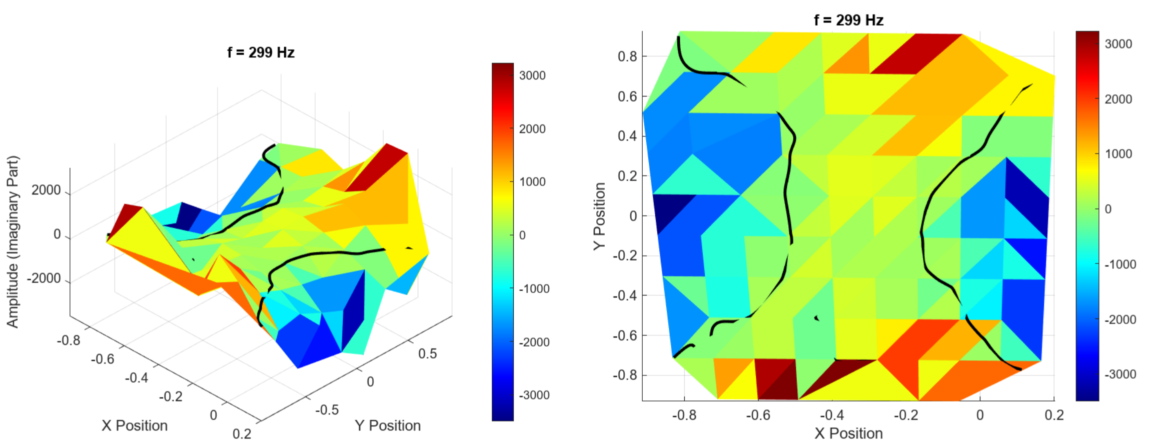

Figure 8: Mode shape at 299 Hz.

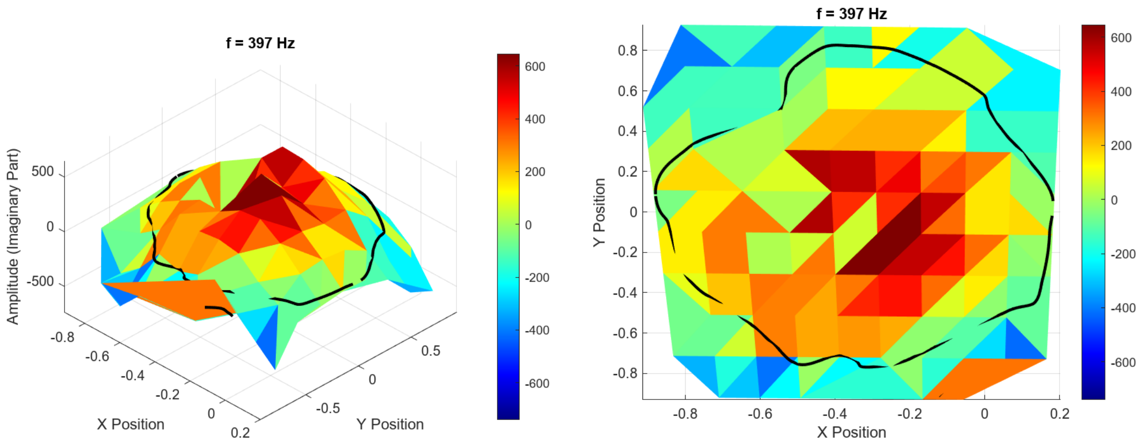

Figure 9: Mode shape at 397 Hz.

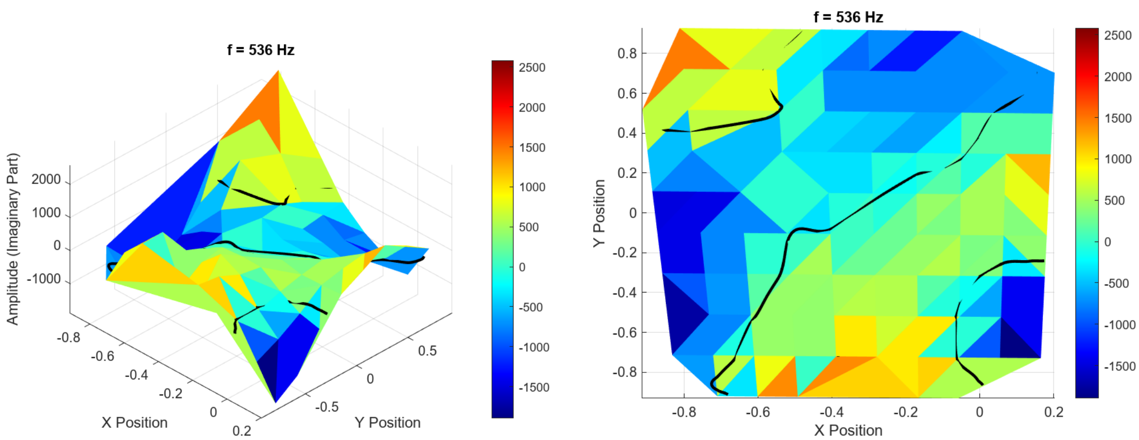

Figure 10: Mode shape at 536 Hz.

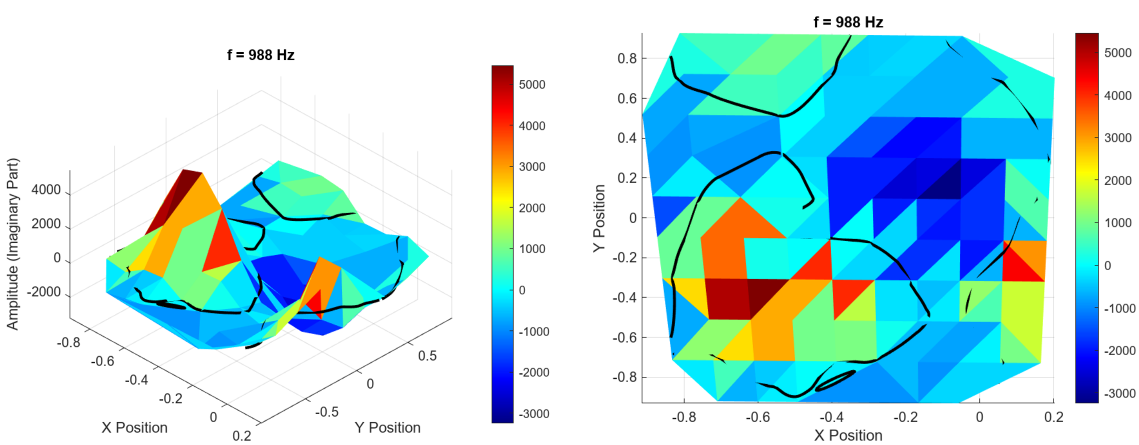

Figure 11: Mode shape at 988 Hz.

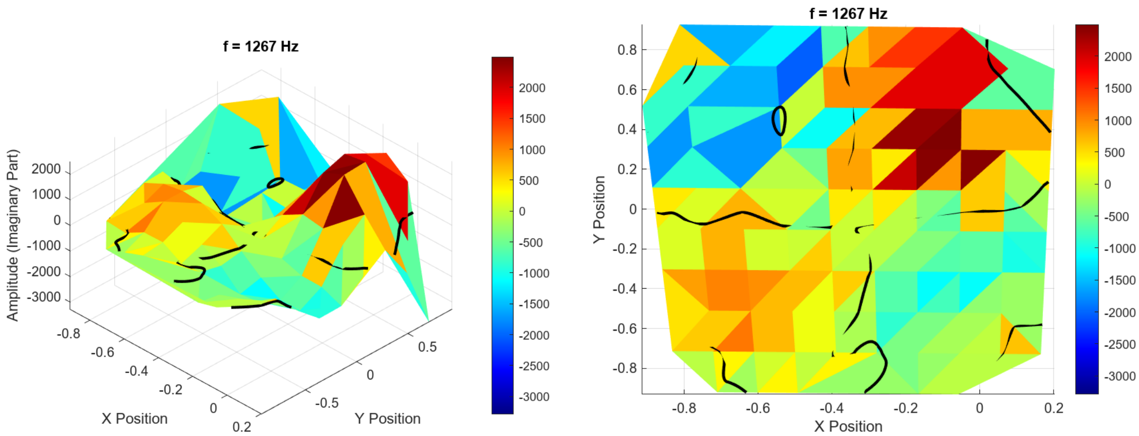

Figure 12: Mode shape at 1267 Hz.

The triangulated grid and ±30 Hz range proved necessary for accurate mapping and peak identification. While the resolution of the measurement points limits the smoothness of the mode shapes, the key characteristics and patterns remain clearly identifiable.

6 Appendix

The following MATLAB code was used for the homework:

```
clear all
%%
% Load the MAT file
load('10x10c.mat');

time = points(1).time;
force = points(1).force;
acc = points(1).acc(:, 1); % 1st accelerometer

% Parameters
Fs = 1 / (time(2) - time(1)); % Sampling frequency
pre_trigger_time = 0.00; % pre-trigger if necessary
signal_duration = 0.25;

% Convert times to samples
pre_trigger_samples = round(pre_trigger_time * Fs);
total_samples = round(signal_duration * Fs);

% Find the trigger point (peak of force signal)
[~, trigger_index] = max(abs(force));

% Define start and end indices for the segment
start_index = max(1, trigger_index - pre_trigger_samples);
end_index = min(length(time), start_index + total_samples - 1);

% Extract the segments
time_segment = time(start_index:end_index);
force_segment = force(start_index:end_index);
acc_segment = acc(start_index:end_index);

% Plot the results
figure;
subplot(2, 1, 1);
plot(time_segment, force_segment);
title('Force Signal (Segment)');
xlabel('Time (s)');
ylabel('Force');

subplot(2, 1, 2);
plot(time_segment, acc_segment);
title('Acceleration Signal (Segment)');
xlabel('Time (s)');
ylabel('Acceleration');
%%
time_range = (time >= 0 & time <= 0.002);
time_segment = time(time_range);
force_segment = force(time_range);

% Plot the results
figure;
```

```
plot(time_segment, force_segment, 'b-', 'LineWidth', 1.5);
xlabel('Time (s)');
ylabel('Force (N)');
grid on;

% Save the plot as a PNG
saveas(gcf, 'force_time_plot_0_to_0_002.png');

% Restrict to the time domain 0 to 0.25 seconds
time_range = (time >= 0 & time <= 0.25);
time_segment = time(time_range);
acc_segment = acc(time_range);

% Plot the results
figure;
plot(time_segment, acc_segment, 'r-', 'LineWidth', 1.5);
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');
grid on;

% Save the plot as a PNG
saveas(gcf, 'acceleration_time_plot_0_to_0_25.png');
%%
% Define parameters
Fs = 1 / (points(1).time(2) - points(1).time(1)); % Sampling frequency
N = length(points(1).time); % Number of samples
f = (0:N-1)*(Fs/N); % Frequency vector
FRFs = zeros(N, 100);

% Measuring points
for i = 1:100
    force = points(i).force;
    acc = points(i).acc(:, 1); % 1st accelerometer

    F_force = fft(force);
    F_acc = fft(acc);

    % FRF
    FRFs(:, i) = F_acc ./ F_force;
end

figure('Position', [100, 100, 1600, 900]); % 16:9 aspect ratio

hold on;
for i = 1:100
    plot(f, abs(FRFs(:, i)));
end
hold off;

% Formatting the plot
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
set(gca, 'XScale', 'log', 'YScale', 'log');
```

```
xlim([0 Fs/2]); % Limit frequency range to Nyquist frequency

% Save the plot as a PNG
print(gcf, 'FRF_plot_high_res.png', '-dpng', '-r300');
%%
plotsPerRow = 5;
numPoints = length(points);

% Measurement points
for i = 1:plotsPerRow:numPoints
    figure('Position', [100, 100, 1600, 400]);

    for j = 0:(plotsPerRow - 1)
        idx = i + j; % Current measurement point index
        if idx > numPoints
            break;
        end

        % Extract data for the current point
        time = points(idx).time;
        force = points(idx).force;

        % Restrict to the time domain 0 to 0.002 seconds
        timeDomain = (time >= 0 & time <= 0.002);
        timeSegment = time(timeDomain);
        forceSegment = force(timeDomain);

        % Create a subplot
        subplot(1, plotsPerRow, j + 1);
        plot(timeSegment, forceSegment);
        title(['Point #', num2str(idx)]);
        xlabel('Time (s)');
        ylabel('Force');
        grid on;
    end
end
%%
% Extract data for Point #91
point_idx = 91;
time = points(point_idx).time;
force = points(point_idx).force;

% Restrict to the time domain 0 to 0.002 seconds
timeDomain = (time >= 0 & time <= 0.002);
timeSegment = time(timeDomain);
forceSegment = force(timeDomain);

% Create the plot
figure('Position', [100, 100, 1200, 600]);
plot(timeSegment, forceSegment, 'b-', 'LineWidth', 1.5);
xlabel('Time (s)');
ylabel('Force (N)');
grid on;

```

```
% Save the plot as a PNG
saveas(gcf, 'Point_91_Force_Plot.png');
%%
load('10x10c.mat');

% Bad measurement points to delete
badPoints = [1, 2, 5, 6, 10, 30, 57, 68, 78, 83, 90, 91, 92, 98, 100];
points(badPoints) = [];

save('10x10c_cleaned.mat', 'points');
%%
% Load the cleaned MAT file
load('10x10c_cleaned.mat');

Fs = 1 / (points(1).time(2) - points(1).time(1)); % Sampling frequency
N = length(points(1).time); % Number of samples
f = (0:N-1)*(Fs/N); % Frequency vector

% Frequency ranges for natural frequencies
freq_ranges = [
    200, 230;  % First natural frequency
    280, 320;  % Second natural frequency
    370, 420;  % Third natural frequency
    490, 560;  % Fourth natural frequency
    900, 1050; % Fifth natural frequency
    1200, 1300 % Sixth natural frequency
];

natural_frequencies = zeros(size(freq_ranges, 1), 1);

% Frequency ranges
for k = 1:size(freq_ranges, 1)
    f_min = freq_ranges(k, 1);
    f_max = freq_ranges(k, 2);

    range_indices = (f >= f_min & f <= f_max);

    max_FRF_amplitude = -Inf;
    max_FRF_frequency = NaN;

    % Measurement points (to find the maximum in this range)
    for i = 1:length(points)
        force = points(i).force;
        acc = points(i).acc(:, 1); % 1st accelerometer

        F_force = fft(force);
        F_acc = fft(acc);

        % Compute the FRF
        FRF = F_acc ./ F_force;

        [peak_value, peak_idx] = max(abs(FRF(range_indices)));
        if peak_value > max_FRF_amplitude
            max_FRF_amplitude = peak_value;
```

```
            max_FRF_frequency = f(range_indices);
            max_FRF_frequency = max_FRF_frequency(peak_idx);
        end
    end

    % Store the maximum frequency for this range
    natural_frequencies(k) = max_FRF_frequency;
end
%%
Fs = 1 / (points(1).time(2) - points(1).time(1)); % Sampling frequency
N = length(points(1).time); % Number of samples
f = (0:N-1)*(Fs/N); % Frequency vector

% Frequency ranges for natural frequencies
freq_ranges = [
    200, 230;  % First natural frequency
    280, 320;  % Second natural frequency
    370, 420;  % Third natural frequency
    490, 560;  % Fourth natural frequency
    900, 1050; % Fifth natural frequency
    1200, 1300 % Sixth natural frequency
];

FRFs = zeros(N, length(points));

% ?easuring points
for i = 1:length(points)
    force = points(i).force;
    acc = points(i).acc(:, 1); % 1st accelerometer

    F_force = fft(force);
    F_acc = fft(acc);

    % Compute the FRF
    FRFs(:, i) = abs(F_acc ./ F_force);
end

natural_frequencies = zeros(size(freq_ranges, 1), 1);
peak_values = zeros(size(freq_ranges, 1), 1);
best_FRFs = zeros(N, size(freq_ranges, 1)); % Store the FRF with the peak for
   highlighting

figure('Position', [100, 100, 1600, 900]); % 16:9 aspect ratio

hold on;

% Plot all FRFs as background
for i = 1:length(points)
    plot(f, FRFs(:, i), 'Color', [0.8, 0.8, 0.8]); % Light gray for background
end

% Mark peaks
for k = 1:size(freq_ranges, 1)
    f_min = freq_ranges(k, 1);
```

```
    f_max = freq_ranges(k, 2);

    range_indices = (f >= f_min & f <= f_max);

    % Search for the global peak across all measuring points
    max_FRF_amplitude = -Inf;
    max_FRF_frequency = NaN;
    best_FRF_index = 0; % To track the point with the peak
    for i = 1:length(points)
        [peak_value, peak_idx] = max(FRFs(range_indices, i));
        if peak_value > max_FRF_amplitude
            max_FRF_amplitude = peak_value;
            max_FRF_frequency = f(range_indices);
            max_FRF_frequency = max_FRF_frequency(peak_idx);
            best_FRF_index = i;
        end
    end

    natural_frequencies(k) = max_FRF_frequency;
    peak_values(k) = max_FRF_amplitude;

    % Highlight the FRF of the point where the peak was found
    best_FRFs(:, k) = FRFs(:, best_FRF_index);
    plot(f, best_FRFs(:, k), 'b-', 'LineWidth', 1.5);

    % Mark the peak
    plot(max_FRF_frequency, max_FRF_amplitude, 'ro', 'MarkerSize', 8, '
   MarkerFaceColor', 'r'); % Peak
end

xlabel('Frequency (Hz)');
ylabel('Magnitude (|FRF|)');
grid on;
set(gca, 'XScale', 'log', 'YScale', 'log'); % Logarithmic x- and y-axes
xlim([0 Fs/2]); % Limit frequency range to Nyquist frequency
legend('FRFs', 'FRF of Peak Point', 'Found Peak', 'Location', 'Best');

hold off;

% Save the plot as PNG
print(gcf, 'FRF_plot_corrected_with_peaks.png', '-dpng', '-r300'); % Save at
   300 DPI
%%
% Load the cleaned MAT file
load('10x10c_cleaned.mat');

% Extract positions from Pspace
x = cell2mat(arrayfun(@(p) p.Pspace(1), points, 'UniformOutput', false))';
y = cell2mat(arrayfun(@(p) p.Pspace(2), points, 'UniformOutput', false))';

% Triangulate the grid to connect the dots
tri = delaunay(x, y);

% Plot the grid and connect the dots
```

```
figure;
triplot(tri, x, y, 'b');
hold on;

% Highlight the points
scatter(x, y, 50, 'filled', 'r');
xlabel('X Position');
ylabel('Y Position');
grid on;
axis equal;

% Annotate the points with their indices
for i = 1:length(x)
    text(x(i), y(i), ['  ', num2str(i)], 'VerticalAlignment', 'bottom', '
    HorizontalAlignment', 'right');
end

hold off;
%%
natural_frequencies = [213.0, 299.0, 397.0, 536.0, 988.0, 1267.0];

Fs = 1 / (points(1).time(2) - points(1).time(1));
N = length(points(1).time);
f = (0:N-1)*(Fs/N);

modeshapes = zeros(length(points), length(natural_frequencies));

% Natural frequencies
for k = 1:length(natural_frequencies)
    freq = natural_frequencies(k);

    % Define a frequency range around the natural frequency
    freq_range = [freq - 30, freq + 30]; % Adjustable...
    freq_indices = (f >= freq_range(1) & f <= freq_range(2));

    for i = 1:length(points)
        force = points(i).force;
        acc = points(i).acc(:, 1); % 1st accelerometer

        F_force = fft(force);
        F_acc = fft(acc);

        % Compute the FRF
        FRF = F_acc ./ F_force;
        imag_part = imag(FRF(freq_indices));
        % Find the local extrema (maximum or minimum) of the imaginary part
        [max_val, max_idx] = max(imag_part);
        [min_val, min_idx] = min(imag_part);

        % Select the extremum with the larger magnitude
        if abs(max_val) > abs(min_val)
            modeshapes(i, k) = max_val;
        else
            modeshapes(i, k) = min_val;
```

```
        end
    end
end

x = cell2mat(arrayfun(@(p) p.Pspace(1), points, 'UniformOutput', false))';
y = cell2mat(arrayfun(@(p) p.Pspace(2), points, 'UniformOutput', false))';
tri = delaunay(x, y);
[Xgrid, Ygrid] = meshgrid(linspace(min(x), max(x), 200), linspace(min(y), max(y
    ), 200)); % Finer grid

for k = 1:length(natural_frequencies)
    figure;

    % Extract mode shape data for this mode
    mode_data = modeshapes(:, k);
    Zgrid = griddata(x, y, mode_data, Xgrid, Ygrid, 'cubic');

    % Plot the mode shape
    trisurf(tri, x, y, mode_data, 'EdgeColor', 'none');
    colormap jet;
    colorbar;
    hold on;

    % Highlight the lines where Zgrid == 0
    contour(Xgrid, Ygrid, Zgrid, [0, 0], 'k', 'LineWidth', 2);

    title(['f = ', num2str(natural_frequencies(k)), ' Hz']);
    xlabel('X Position');
    ylabel('Y Position');
    zlabel('Amplitude (Imaginary Part)');
    grid on;
    view(45,45);
    hold off;
end
%%
% TOP VIEW
for k = 1:length(natural_frequencies)
    figure;
    mode_data = modeshapes(:, k);
    Zgrid = griddata(x, y, mode_data, Xgrid, Ygrid, 'cubic');

    trisurf(tri, x, y, mode_data, 'EdgeColor', 'none');
    colormap jet;
    colorbar;
    hold on;

    contour(Xgrid, Ygrid, Zgrid, [0, 0], 'k', 'LineWidth', 2);

    title(['f = ', num2str(natural_frequencies(k)), ' Hz']);
    xlabel('X Position');
    ylabel('Y Position');
    zlabel('Amplitude (Imaginary Part)');
    grid on;
    view(0,90);
```

```
    hold off;
end
```