Mechatronic Systems Identification

Lab 8 - Identification of a mechanical system – modal analysis

Khaldoun Fayad - 409597

Witold Surdej - 407100

26.04.2024

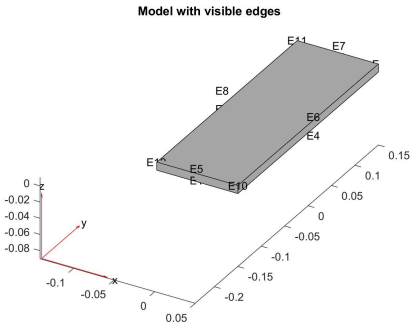
Description

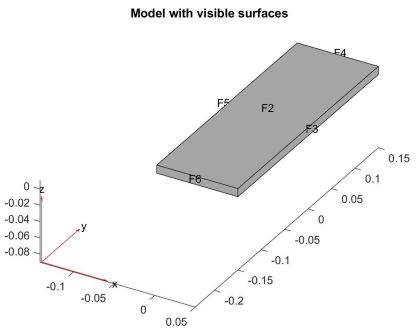
The aim of this laboratory is for us to learn the basics of model analysis in the field of identification of mechanical systems.

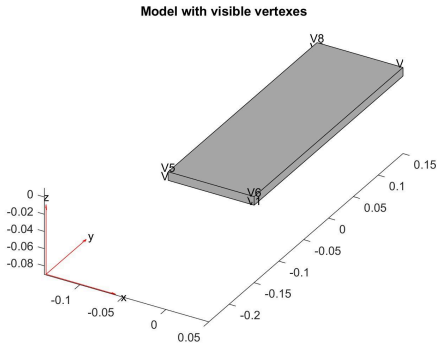
Task 1

In this lab exercise, we utilize MATLAB's Partial Differential Equation (PDE) Toolbox to perform a modal simulation of a plate with specified dimensions and material properties. The objective is to determine the natural frequencies and corresponding modes of vibration for the plate, fixed along one edge. This study provides insights into the dynamic behavior of mechanical structures and highlights the application of finite element methods (FEM) in engineering analysis. Understanding these aspects is crucial for the design and safety of mechanical systems.

```
%% Task 1
X = 0.1;
Y = 0.3;
Z = 0.01;
E = 210*10^9;
rho = 7800;
nu = 0.3;
modelM = createpde('structural','modal-solid');
gm = multicuboid(X,Y,Z);
modelM.Geometry = gm;
figure(1)
pdegplot(modelM,'EdgeLabels','on');
axis equal
title 'Model with visible edges'
figure(2)
pdegplot(modelM,'FaceLabels','on');
axis equal
title 'Model with visible surfaces'
figure(3)
pdegplot(modelM,'VertexLabels','on');
axis equal
title 'Model with visible vertexes'
hmax = 8e-3;
msh = generateMesh(modelM,'Hmax',hmax);
figure(4)
pdeplot3D(modelM);
axis equal
```

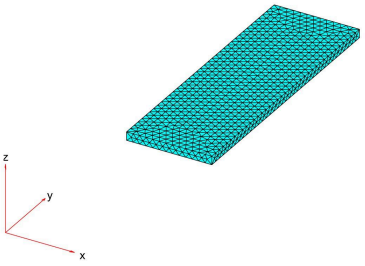






```
structuralProperties(modelM,'YoungsModulus',E,'PoissonsRatio',nu,'MassDe
nsity',rho);
structuralBC(modelM,'Face',6,'Constraint','fixed');
resModal = solve(modelM,'FrequencyRange',[0,1000]*2*pi);
modeID = 1:numel(resModal.NaturalFrequencies);
tmodalResults = table(modeID.',resModal.NaturalFrequencies/(2*pi));
tmodalResults.Properties.VariableNames = {'Mode','Frequency'};
disp(tmodalResults)
ModeNumber = 1;
FrameRate = 30;
AnimateModeShape(resModal,ModeNumber,FrameRate)
```

Mode	Frequency
1	94.804
2	565.48
3	589.72
4	865.07



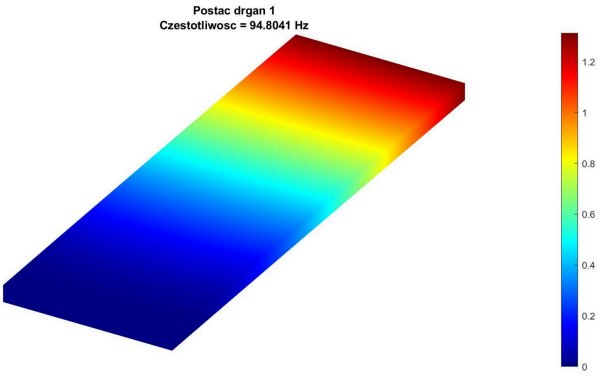


fig 1: The simulation of the deflection of the slab

Conclusions: We can observe that the simulation provides a confirmation of the ability of the PDE Toolbox to accurately predict the natural frequencies of a fixed plate. The calculated frequencies and observed vibration modes can be compared against theoretical values or simulations from other software to verify the accuracy of MATLAB's modal analysis tools. Furthermore, this simulation was able to detect and calculate the modes of the current system. It deduced 4 modes, each of a different frequency. The exercise demonstrates the impact of boundary conditions on the modal frequencies of structures. By fixing the plate on one side, the results show how constraints can significantly alter vibration patterns, which is essential for understanding structural behavior under various support conditions.

Task 2

In this second lab task, we will extend our investigation of mechanical system dynamics by conducting a frequency response simulation on the same plate structure previously analyzed for natural frequencies. By utilizing MATLAB's Partial Differential Equation Toolbox with a focus on structural and frequency-solid modules, this exercise aims to calculate the frequency response of the plate when subjected to a force along the Z-axis. We will assess the plate's behavior under vibrational excitation and identify how its natural frequencies manifest in the amplitude-frequency and phase-frequency displacement plots. This task not only deepens our understanding of modal interactions and damping effects but also explores the computational efficiency of leveraging previous modal analysis results in frequency response simulations. Through this analysis, we hope to elucidate the relationship between natural frequencies, anti-resonances, and their phase changes, providing valuable insights into the mechanical behavior of the system.

Task 2.1

In this task, we explore the frequency response of a mechanical plate system subjected to dynamic loading. This simulation, performed using MATLAB's PDE Toolbox, will measure the response at specific points on the plate under a force applied along the Z-axis. The task will analyze the amplitude-frequency and phase-frequency characteristics, highlighting visible eigenfrequencies and investigating why some frequencies are not apparent in the results.

```
%% Task 2
E = 210*10^9;
rho = 7800;
nu = 0.3;
%Ex. 2.1
X = 0.1;
Y = 0.3;
Z = 0.01;
A = [-0.05, 0.15, 0.01];
B = [-0.05, 0.0866, 0.01];
C = [0, 0.15, 0.01];
modelFR = createpde('structural','frequency-solid');
modelFR.Geometry = modelM.Geometry;
modelFR.Mesh = modelM.Mesh;
structuralProperties(modelFR,'YoungsModulus',E,'PoissonsRatio',nu,'MassD
ensity', rho);
structuralBC(modelFR,'Face',6,'Constraint','fixed');
structuralDamping(modelFR,'Zeta',0.001);
structuralBoundaryLoad(modelFR,'Vertex',8,'Force',[0,0,1]);
flist = linspace(0,999,1000)*2*pi;
resFreq = solve(modelFR,flist,'ModalResults',resModal);
[FRFAd,FreqAd] = DisplayFRF(resFreq,A,'displacement')
[FRFBd,FreqBd] = DisplayFRF(resFreq,B,'displacement');
[FRFCd,FreqCd] = DisplayFRF(resFreq,C,'displacement');
figure()
pdeplot3D(modelM);
hold on
```

```
plot3(A(1),A(2),A(3),'ro')
plot3(B(1),B(2),B(3),'bo')
plot3(C(1),C(2),C(3),'yo')
hold off
axis equal
[FRFAv,FreqAv] = DisplayFRF(resFreq,A,'velocity');
[FRFBv,FreqBv] = DisplayFRF(resFreq,B,'velocity');
[FRFCv,FreqCv] = DisplayFRF(resFreq,C,'velocity');
figure()
pdeplot3D(modelM);
hold on
plot3(A(1),A(2),A(3),'ro')
plot3(B(1),B(2),B(3),'bo')
plot3(C(1),C(2),C(3),'yo')
hold off
axis equal
[FRFAa,FreqAa] = DisplayFRF(resFreq,A,'acceleration');
[FRFBa,FreqBa] = DisplayFRF(resFreq,B,'acceleration');
[FRFCa,FreqCa] = DisplayFRF(resFreq,C,'acceleration');
figure()
pdeplot3D(modelM);
hold on
plot3(A(1),A(2),A(3),'ro')
plot3(B(1),B(2),B(3),'bo')
plot3(C(1),C(2),C(3),'yo')
hold off
axis equal
% Extract complex components (assuming FRF is already complex)
FRF_mag = abs(FRFAd);  % Magnitude of FRF
FRF_phase = angle(FRFAd) * (180/pi);  % Phase of FRF in degrees
% Plot Bode plot (magnitude and phase)
figure();
% Magnitude plot (dB)
subplot(2, 1, 1);
semilogx(FreqAd, 20*log10(FRF_mag));  % Convert magnitude to dB
title(['Magnitude']);
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
grid on;
xlim([10^1 10^3])
legend('X','Y','Z')
% Phase plot (degrees)
subplot(2, 1, 2);
semilogx(FreqAd, FRF_phase);
title(['Phase']);
xlabel('Frequency (Hz)');
ylabel('Phase (degrees)');
grid on;
xlim([10^1 10^3])
legend('X','Y','Z')
sgtitle('Bode plot of the displacement at point A')
% Extract complex components (assuming FRF is already complex)
FRF_mag = abs(FRFBd);  % Magnitude of FRF
```

```
FRF_phase = angle(FRFBd) * (180/pi);  % Phase of FRF in degrees
% Plot Bode plot (magnitude and phase)
figure();
% Magnitude plot (dB)
subplot(2, 1, 1);
semilogx(FreqBd, 20*log10(FRF_mag));  % Convert magnitude to dB
title(['Magnitude']);
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
grid on;
xlim([10^1 10^3])
legend('X','Y','Z')
% Phase plot (degrees)
subplot(2, 1, 2);
semilogx(FreqBd, FRF_phase);
title(['Phase']);
xlabel('Frequency (Hz)');
ylabel('Phase (degrees)');
grid on;
xlim([10^1 10^3])
legend('X','Y','Z')
sgtitle('Bode plot of the displacement at point B')
% Extract complex components (assuming FRF is already complex)
FRF_mag = abs(FRFCd);  % Magnitude of FRF
FRF_phase = angle(FRFCd) * (180/pi);  % Phase of FRF in degrees
% Plot Bode plot (magnitude and phase)
figure();
% Magnitude plot (dB)
subplot(2, 1, 1);
semilogx(FreqCd, 20*log10(FRF_mag));  % Convert magnitude to dB
title(['Magnitude']);
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
grid on;
xlim([10^1 10^3])
legend('X','Y','Z')
% Phase plot (degrees)
subplot(2, 1, 2);
semilogx(FreqCd, FRF_phase);
title(['Phase']);
xlabel('Frequency (Hz)');
ylabel('Phase (degrees)');
grid on;
xlim([10^1 10^3])
legend('X','Y','Z')
sgtitle('Bode plot of the displacement
at point C')
```

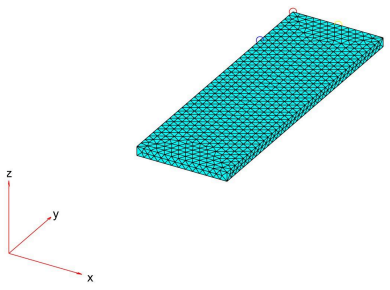


Fig 2: The points present on the slab at the proper dimensions

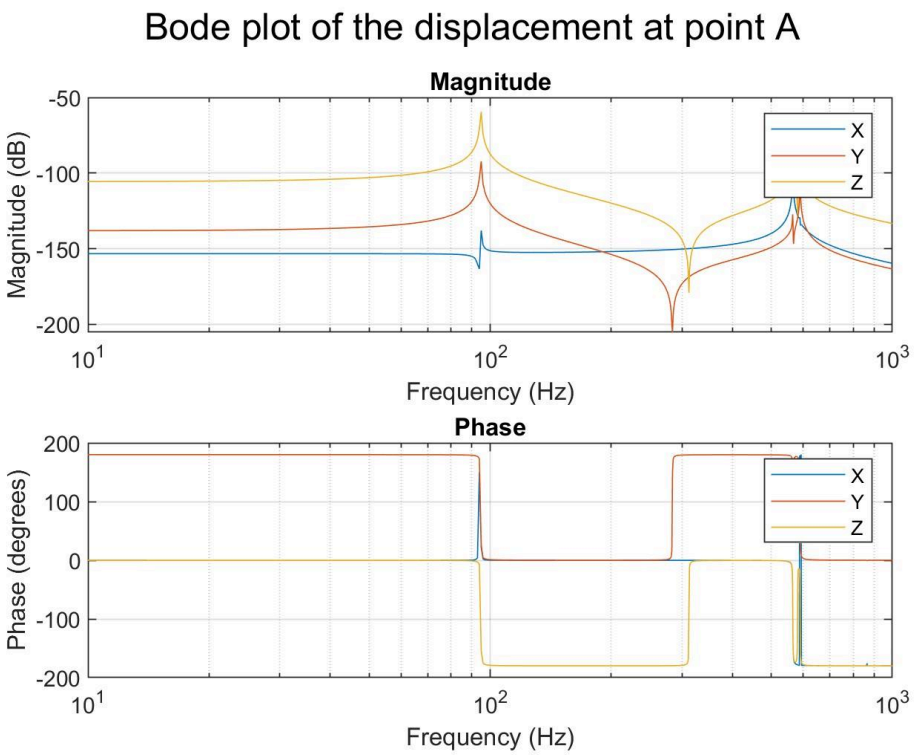


Fig 3: The bode plots of the displacement at point A

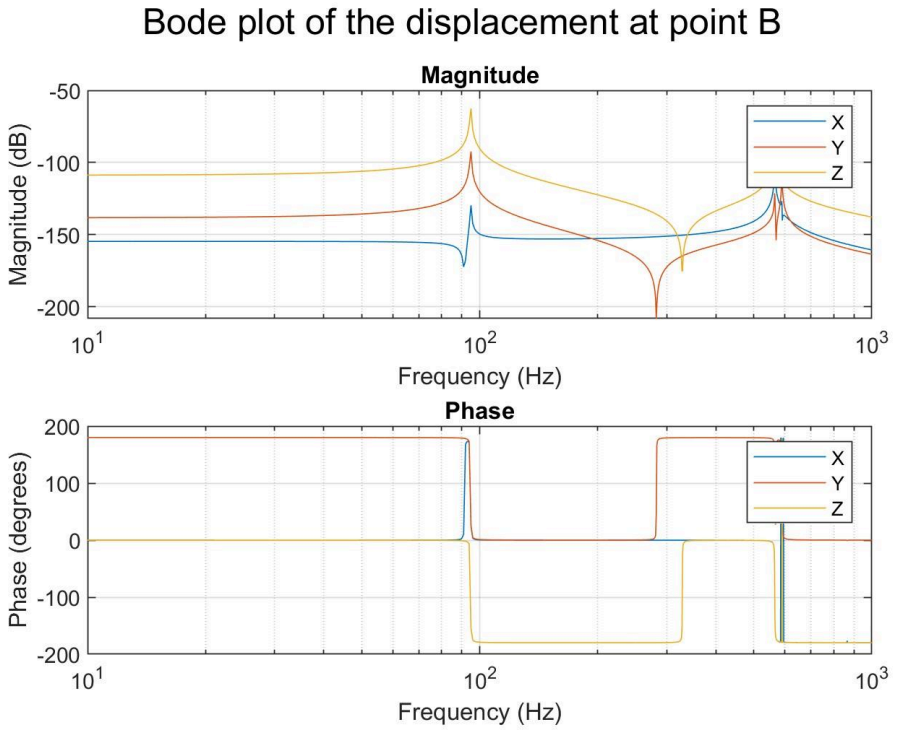


Fig 3: The bode plots of the displacement at point B

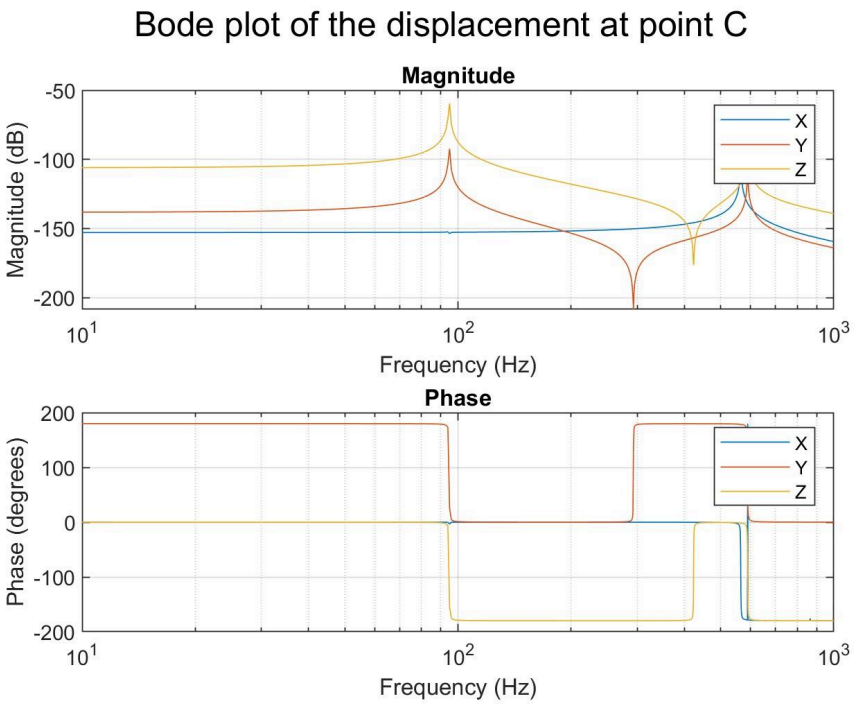


Fig 3: The bode plots of the displacement at point C

Conclusions: We can observe that the frequency response simulation in this exercise effectively demonstrated the plate's dynamic behavior under vibrational excitation. Key findings include the identification of visible eigenfrequencies in the amplitude-frequency plots, which correlate with the natural frequencies determined in earlier modal analyses. The absence of certain frequencies can be attributed to modal damping and the specific dynamics at measured points. Phase changes around the natural frequencies provided insights into the system's resonance and anti-resonance behaviors. We can notice that the phase change at the natural frequency is almost 360 degrees. This exercise reinforces the critical role of accurate frequency response analysis in predicting system behavior under operational conditions, offering a foundational understanding necessary for optimizing mechanical design and enhancing system performance.

Task 2.2

This section explores the amplitude-frequency and phase-frequency characteristics of displacement, velocity, and acceleration measured at point A. The aim is to analyze the differences between these characteristics and derive displacement and acceleration characteristics from velocity using frequency domain operations. Finally, we will compare the derived values with those obtained from the simulation. This task aims to provide insights into the relationships and behaviors of these physical quantities within the given system.

```
% Extract complex components (assuming FRF is already complex)
FRF_mag = abs(FRFAv);  % Magnitude of FRF
FRF_phase = angle(FRFAv) * (180/pi);  % Phase of FRF in degrees
% Plot Bode plot (magnitude and phase)
figure();
% Magnitude plot (dB)
subplot(2, 1, 1);
semilogx(FreqAv, 20*log10(FRF_mag));  % Convert magnitude to dB
title(['Magnitude']);
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
grid on;
xlim([10^1 10^3])
legend('X','Y','Z')
% Phase plot (degrees)
subplot(2, 1, 2);
semilogx(FreqAv, FRF_phase);
title(['Phase']);
xlabel('Frequency (Hz)');
ylabel('Phase (degrees)');
grid on;
xlim([10^1 10^3])
legend('X','Y','Z')
sgtitle('Bode plot of the velocity at point A')
% Extract complex components (assuming FRF is already complex)
FRF_mag = abs(FRFAa);  % Magnitude of FRF
FRF_phase = angle(FRFAa) * (180/pi);  % Phase of FRF in degrees
% Plot Bode plot (magnitude and phase)
figure();
% Magnitude plot (dB)
subplot(2, 1, 1);
semilogx(FreqAa, 20*log10(FRF_mag));  % Convert magnitude to dB
title(['Magnitude']);
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
grid on;
xlim([10^1 10^3])
legend('X','Y','Z')
% Phase plot (degrees)
subplot(2, 1, 2);
semilogx(FreqAa, FRF_phase);
title(['Phase']);
xlabel('Frequency (Hz)');
```

```
ylabel('Phase (degrees)');
grid on;
xlim([10^1 10^3])
legend('X','Y','Z')
sgtitle('Bode plot of the acceleration at point A')
% Expand omega to match the dimensions of FRFAv
omega_matrix = repmat(omega.', [1, 3]);  % Transpose omega and replicate
across columns
% Displacement from Velocity (Integration)
Xf = FRFAv ./ (1j * omega_matrix);  % Ensure omega_matrix is used here
Xf_magnitude = abs(Xf);
Xf_phase = angle(Xf) * (180/pi);
% Acceleration from Velocity (Differentiation)
Af = FRFAv .* (1j * omega_matrix);  % Ensure omega_matrix is used here
Af_magnitude = abs(Af);
Af_phase = angle(Af) * (180/pi);

% Plotting the derived displacement
figure;
subplot(2, 1, 1);
semilogx(FreqAv, 20*log10(Xf_magnitude));  % dB conversion for magnitude
title('Magnitude of Derived Displacement');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
grid on;
subplot(2, 1, 2);
semilogx(FreqAv, Xf_phase);
title('Phase of Derived Displacement');
xlabel('Frequency (Hz)');
ylabel('Phase (degrees)');
grid on;
% Plotting the derived acceleration
figure;
subplot(2, 1, 1);
semilogx(FreqAv, 20*log10(Af_magnitude));  % dB conversion for magnitude
title('Magnitude of Derived Acceleration');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
grid on;
subplot(2, 1, 2);
semilogx(FreqAv, Af_phase);
title('Phase of Derived Acceleration');
xlabel('Frequency (Hz)');
ylabel('Phase (degrees)');
grid on;
```

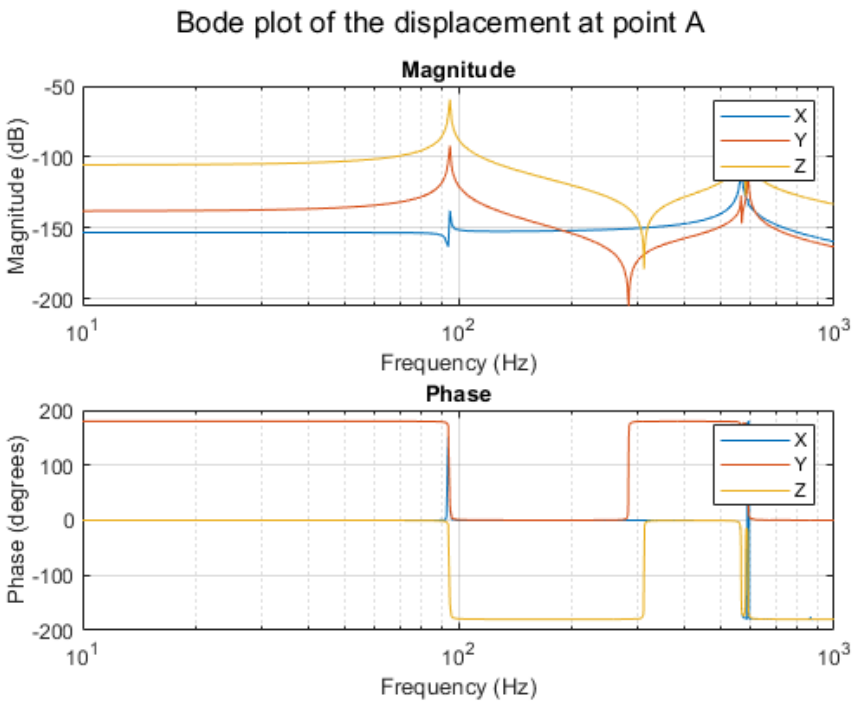


Fig 4: The bode plots of the displacement at point A

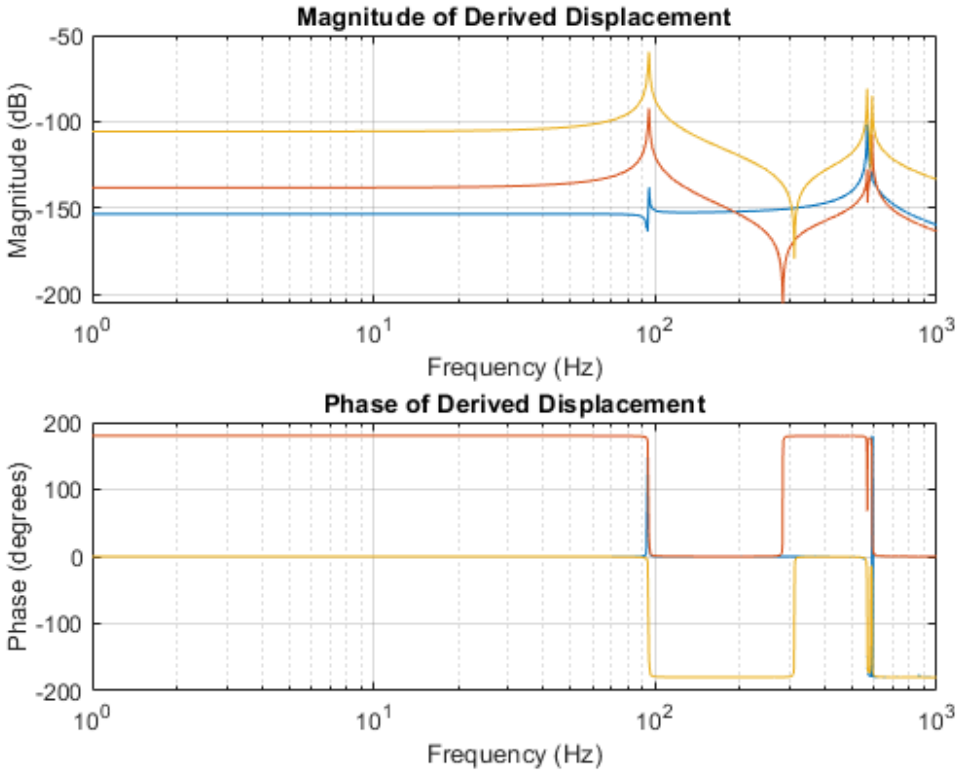


Fig 5: The derived bode plots of the displacement at point A

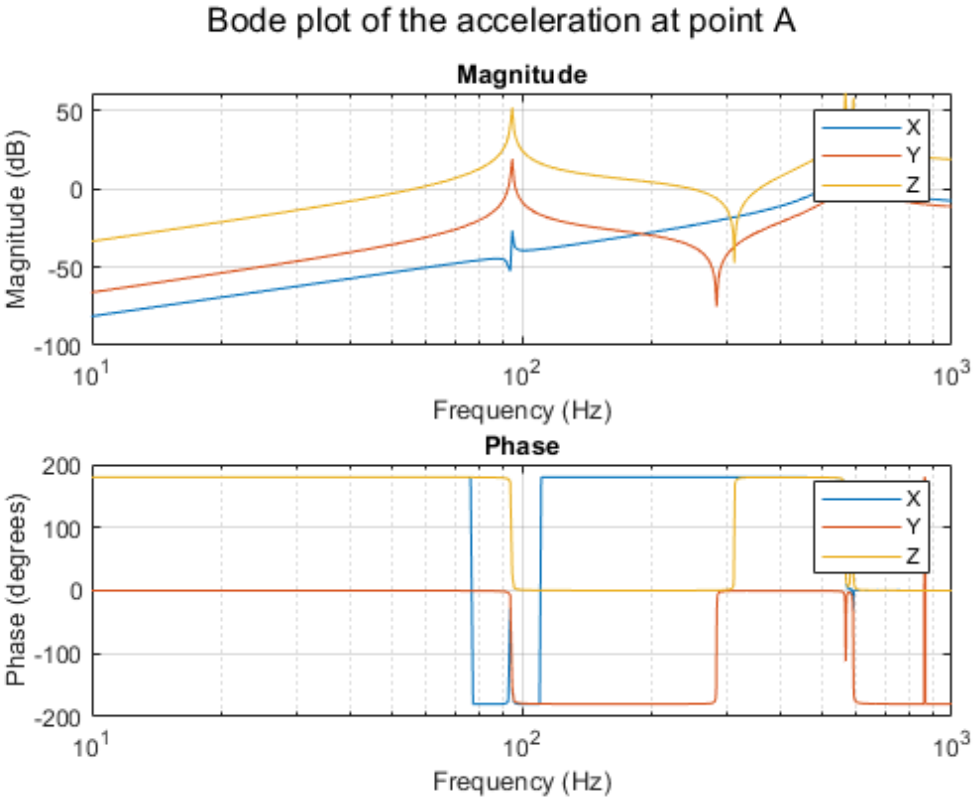


Fig 6: The bode plots of the acceleration at point A

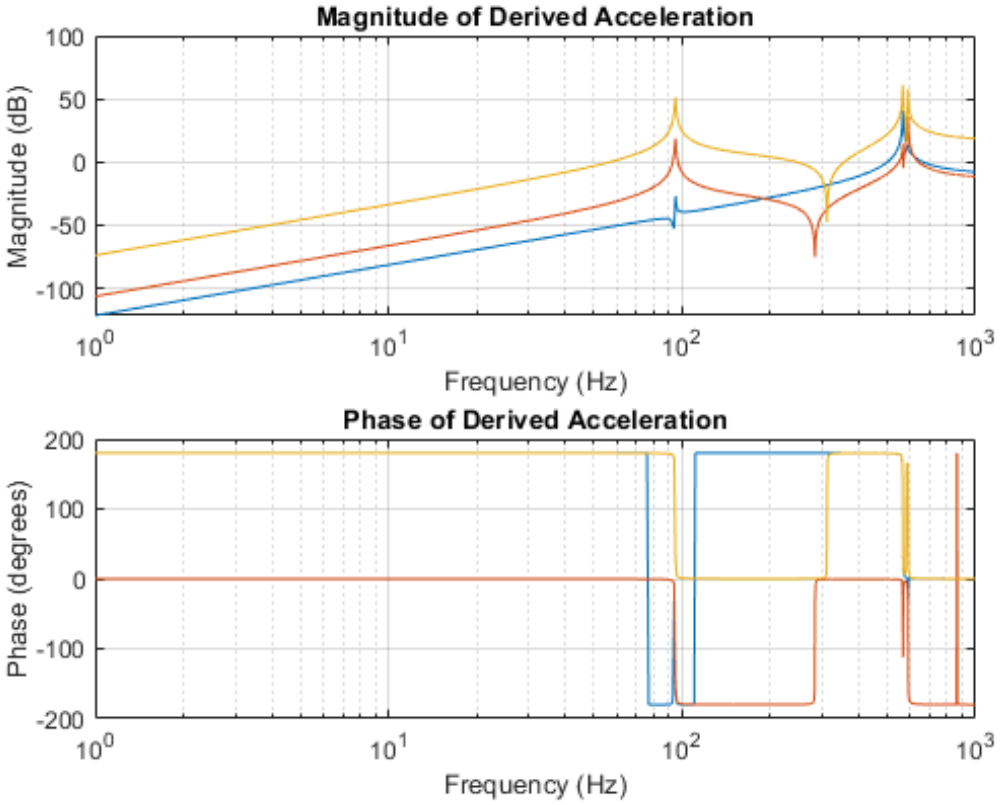


Fig 7: The derived bode plots of the acceleration at point A

Conclusions: This task has proven that by leveraging frequency domain operations like integration and derivative, we can successfully derive displacement and acceleration characteristics from velocity data, with the resulting plots aligning with simulation results. This validation underscores the practical utility of frequency domain analysis in understanding and predicting system dynamics, particularly for engineering applications related to system design, control, and optimization. Overall, this example highlights the critical role of frequency domain techniques in effectively analyzing and modeling dynamic systems for practical engineering purposes.

Task 2.3

This task focuses on utilizing frequency responses to visualize vibration modes corresponding to the first three eigenfrequencies for a specific condition where Zlayer = 0.01m. The objective is to analyze and display the vibration shapes associated with these eigenfrequencies using the `DisplayMode` function. This function allows us to extract displacement data along the Z axis for a chosen frequency from the simulation results and the frequency simulation model.

```
[~,ModeShape,Xv,Yv,Zv] = DisplayMode(resFreq,modelFR,0.01,3,94.804);
ModeZ = imag(ModeShape);
scale = max(Xv)./max(abs(ModeZ(:)));
figure
tri = delaunay(Xv,Yv);
trisurf(tri,Xv,Yv,ModeZ*scale);
axis equal
shading flat
xlabel('x (m)'); ylabel('y (m)'); zlabel('imag (FRF)');
title('Mode Shape at 94.804 Hz')
[~,ModeShape,Xv,Yv,Zv] = DisplayMode(resFreq,modelFR,0.01,3,565.48);
ModeZ = imag(ModeShape);
scale = max(Xv)./max(abs(ModeZ(:)));
figure
tri = delaunay(Xv,Yv);
trisurf(tri,Xv,Yv,ModeZ*scale);
axis equal
shading flat
xlabel('x (m)'); ylabel('y (m)'); zlabel('imag (FRF)');
title('Mode Shape at 565.48 Hz')
[~,ModeShape,Xv,Yv,Zv] = DisplayMode(resFreq,modelFR,0.01,3,589.72);
ModeZ = imag(ModeShape);
scale = max(Xv)./max(abs(ModeZ(:)));
figure
tri = delaunay(Xv,Yv);
trisurf(tri,Xv,Yv,ModeZ*scale);
axis equal
shading flat
xlabel('x (m)'); ylabel('y (m)'); zlabel('imag (FRF)');
title('Mode Shape at 589.72 Hz')
```

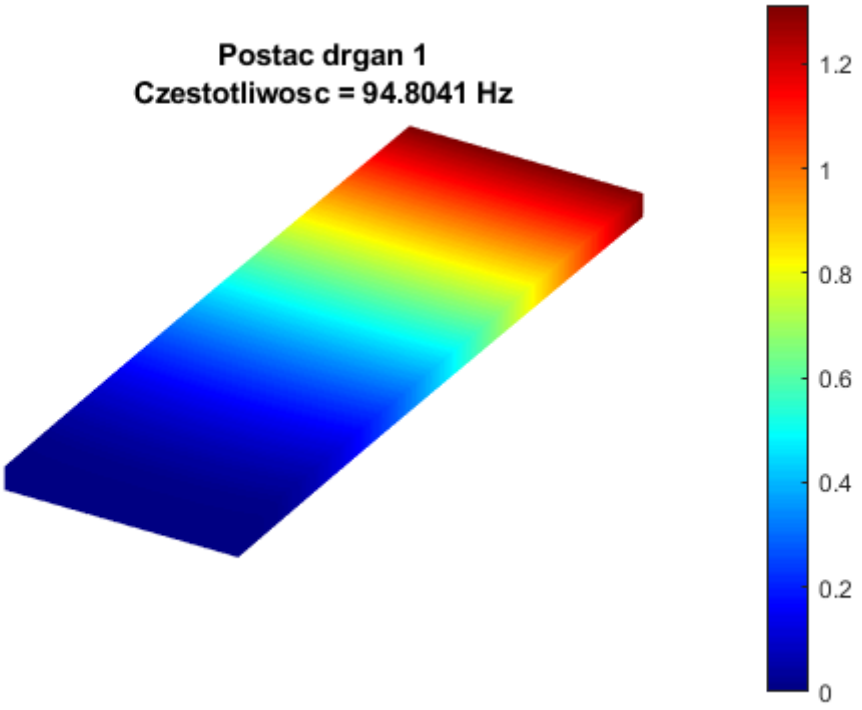


Fig 8: Simulated Mode Shape 1

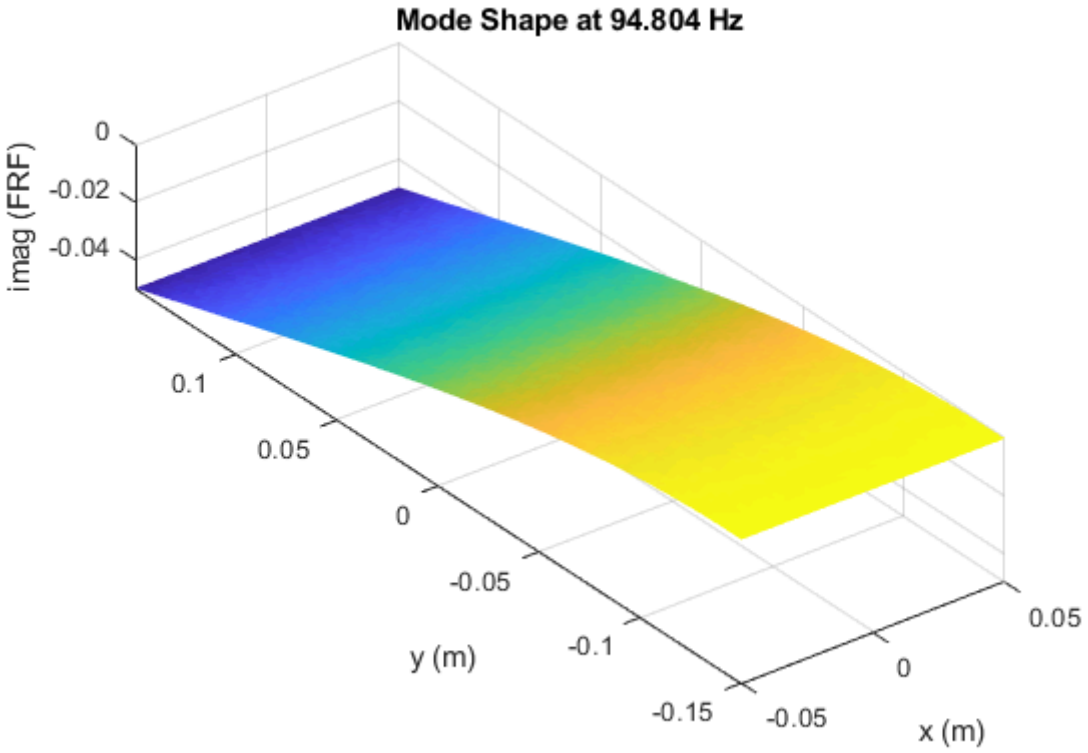


Fig 9: Theoretical Mode Shape form 1

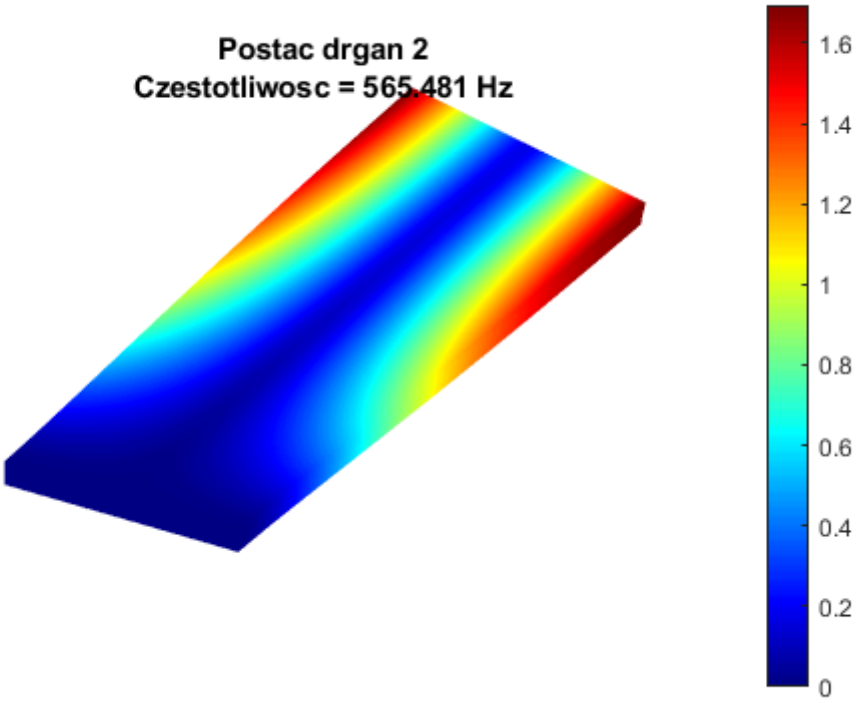


Fig 10: Simulated Mode Shape form 2

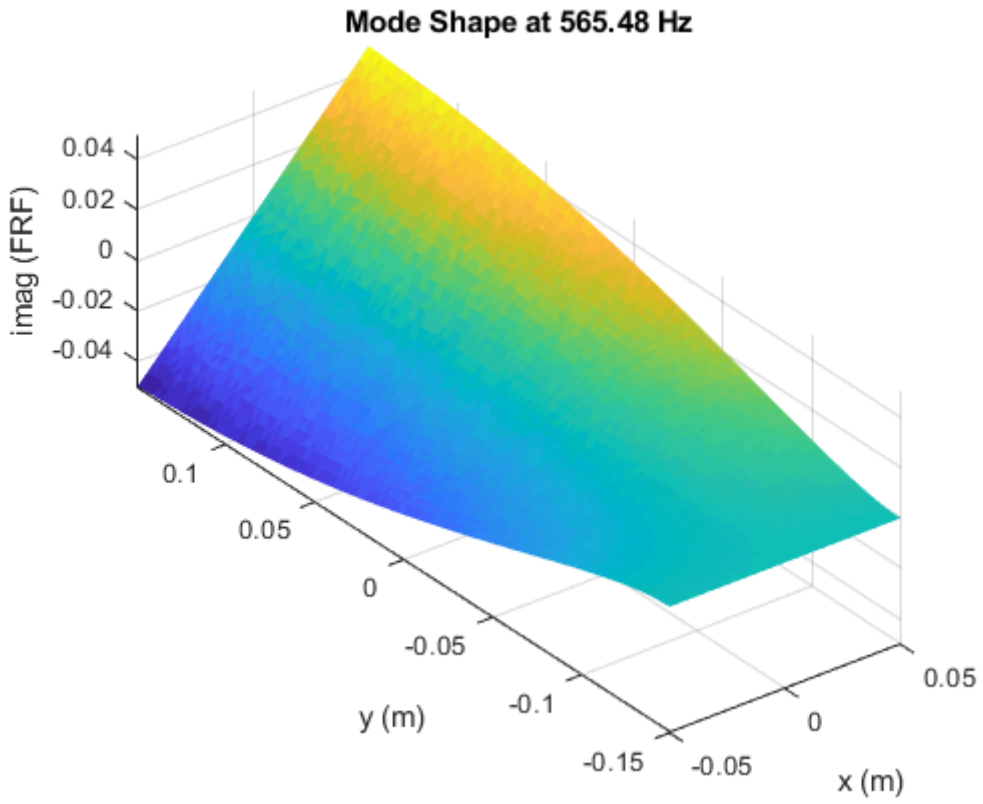


Fig 11: Theoretical Mode Shape form 2

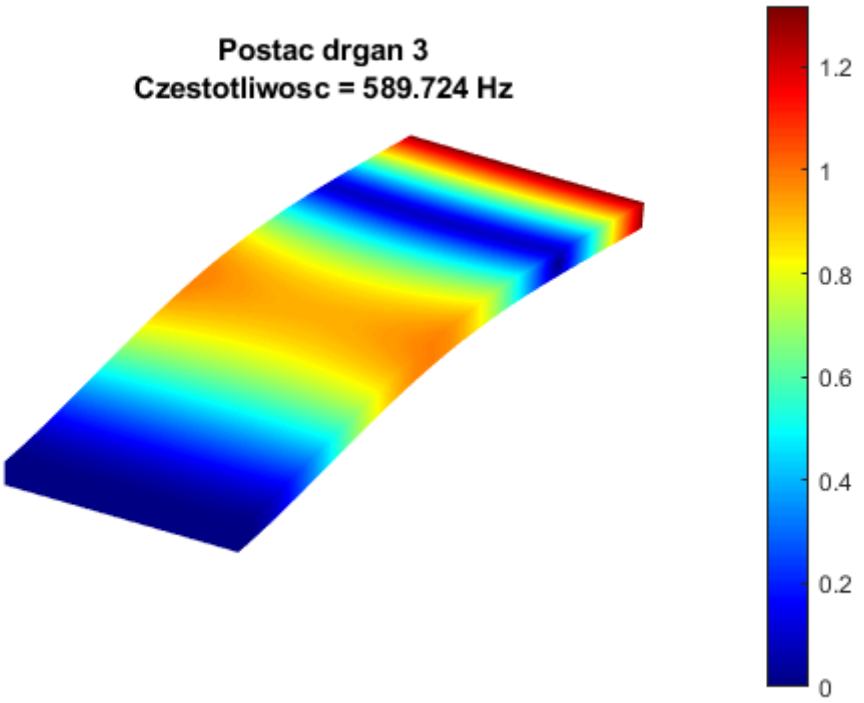


Fig 12: Simulated Mode Shape form 3

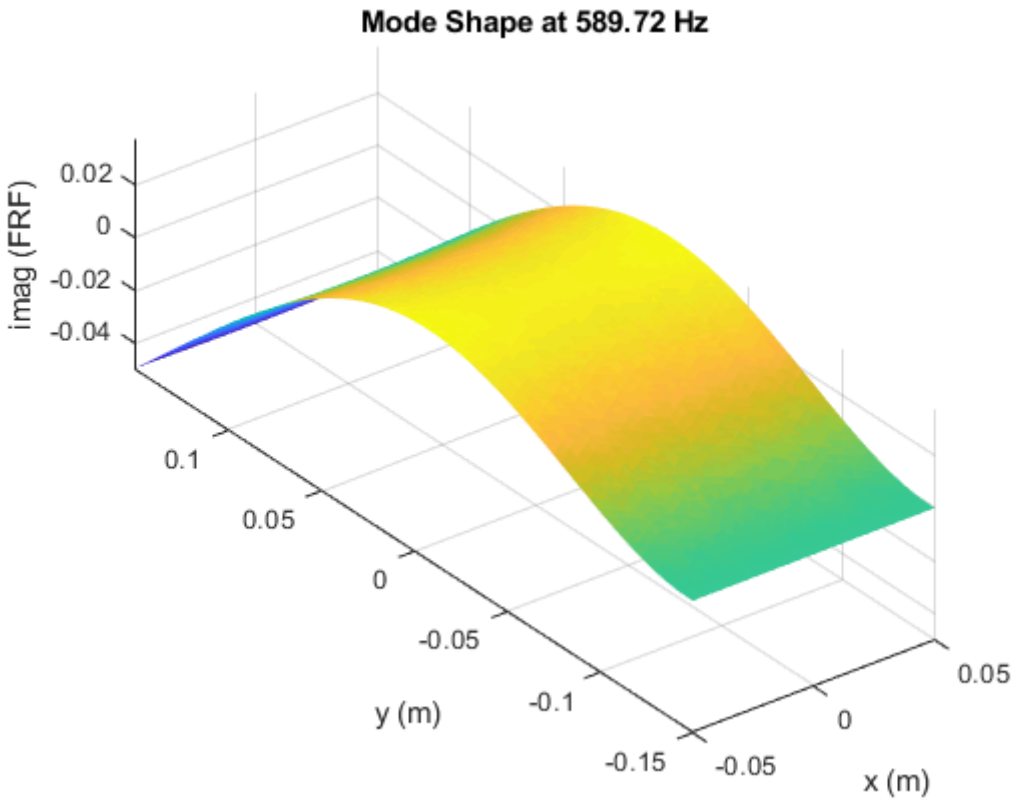


Fig 13: Theoretical Mode Shape form 3

Conclusions: The results presented above showcase that the modal forms obtained from the spectral transfer function are largely congruent to those obtained from model simulation. The discrepancies observed in the plot above largely stem from the animation of the simulated plots and slight inaccuracies of the techniques. In general this task showcases that the two methods are largely accurate in simulating the behavior of the system.

Task 2.4

In this task we revisit the tasks 2.1 and 2.3, this time incorporating a damping ratio of 3% of critical damping. The goal is to compare the resulting vibration shapes and response graphs obtained with this damping assumption against those from the previous exercises.

The discrepancies observed between the graphs are attributed to the coupling effect between adjacent eigenfrequencies, leading to interconnected shapes as per the principle of modal superposition. To analyze non-conjugate vibration forms, fitting functions to Frequency Response Function (FRF) data becomes essential. In MATLAB, the `modalfit` function is utilized for this purpose, which fits functions to FRF data and provides vibration diagrams corresponding to individual modes of natural vibrations.

The exercise entails viewing the mode shapes (MS) obtained using `modalfit` and assessing their similarity to those obtained in Task 1. It also involves evaluating which forms benefit from the `modalfit` function compared to the direct method employed in Task 2.3, aiming to gain insights into the effectiveness of fitting functions for analyzing complex vibration characteristics. This approach allows for a comprehensive comparison of vibration responses under different damping conditions, aiding in understanding system behavior and mode shapes.

The code used for completing this task is largely the same as for the previous tasks. Therefore we only include new additions.

```
    fc = first_three_freq(i); % Current frequency
    [~, ModeShape, Xv, Yv, Zv] = DisplayMode(resFreq, modelFR, Zlayer,
Axis, fc); % Display mode shape
    fcs=[94.804 565.48 589.72];
    [FN,DR,MS] = modalfit(FRFmap,FreqAd,2000,4,'FreqRange',([0
1000]),'FitMethod','lsce','PhysFreq',fcs);
    % Use the imaginary part of the ModeShape
    ModeZ = imag(MS);
    scale = max(Xv) / max(abs(ModeZ(:,3))); % Scale factor for
visualization
    % Plotting the mode shape
    figure;
    tri = delaunay(Xv, Yv); % Create a Delaunay triangulation for
plotting
    trisurf(tri, Xv, Yv, ModeZ(:,3) * scale); % Plot the triangulated
surface
    axis equal;
    shading flat;
    xlabel('x (m)');
```

```
    ylabel('y (m)');
    zlabel('imag (FRF)');
    title(sprintf('Mode Shape at 3rd', fc/(2*pi))); % Title with
frequency in Hz
```

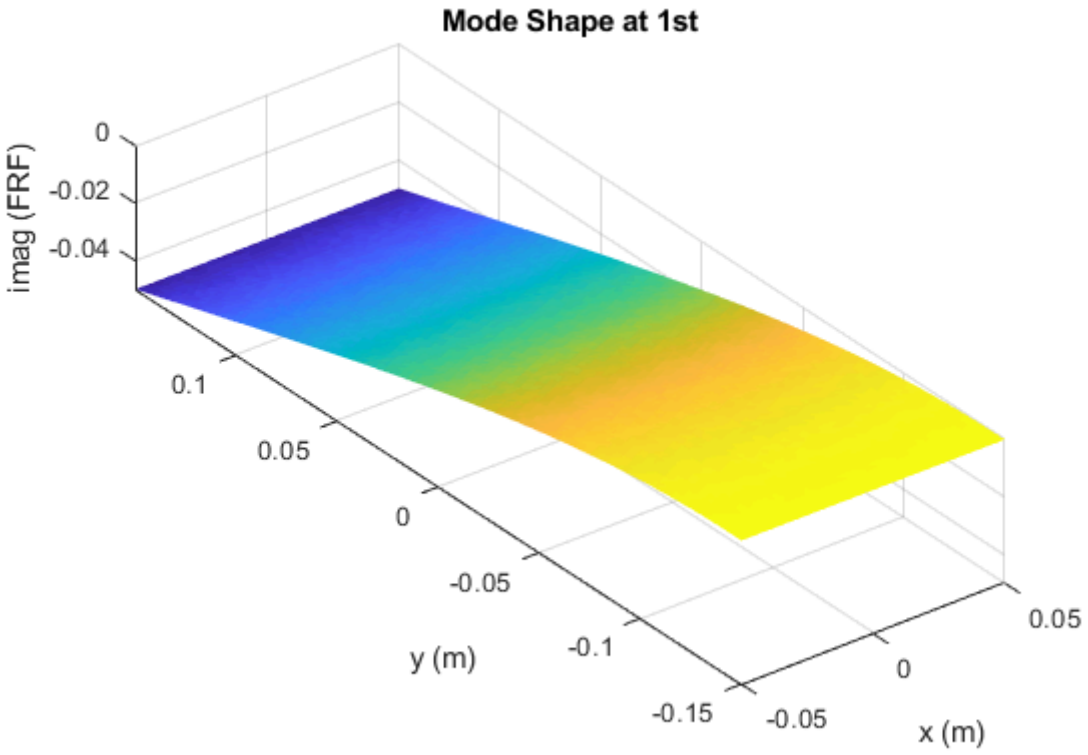


Fig 14: The Mode Shape form 1 corrected using modalfit

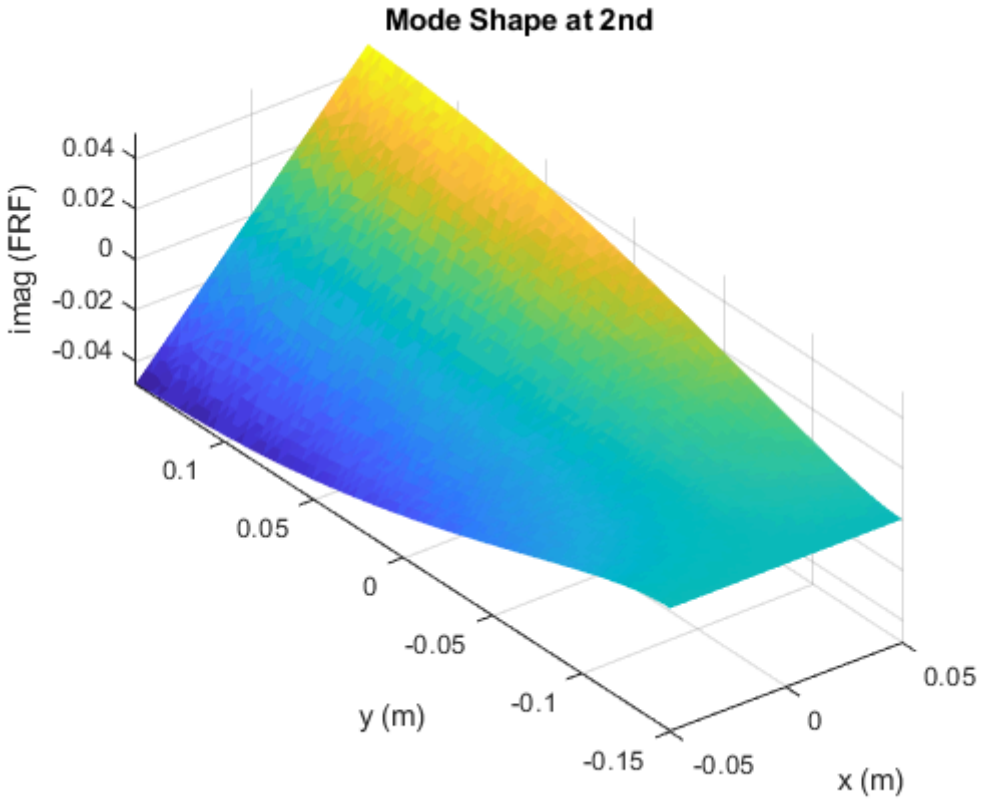


Fig 15: The Mode Shape form 2 corrected using modalfit

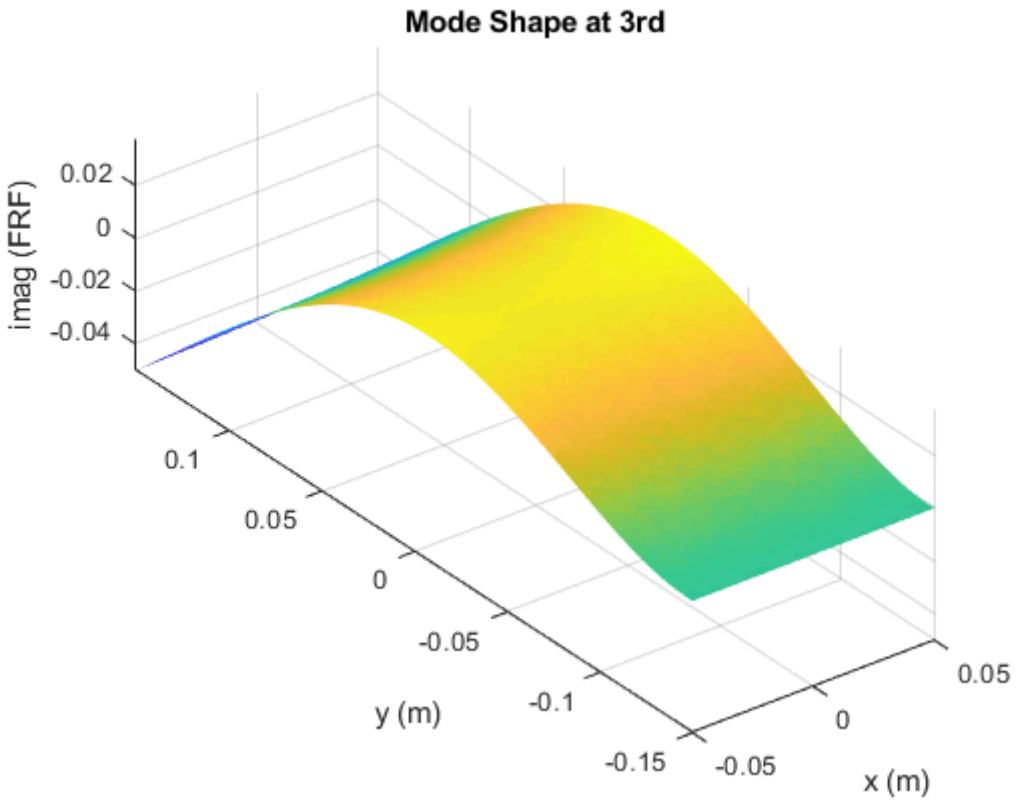


Fig 16: The Mode Shape form 3 corrected using modalfit

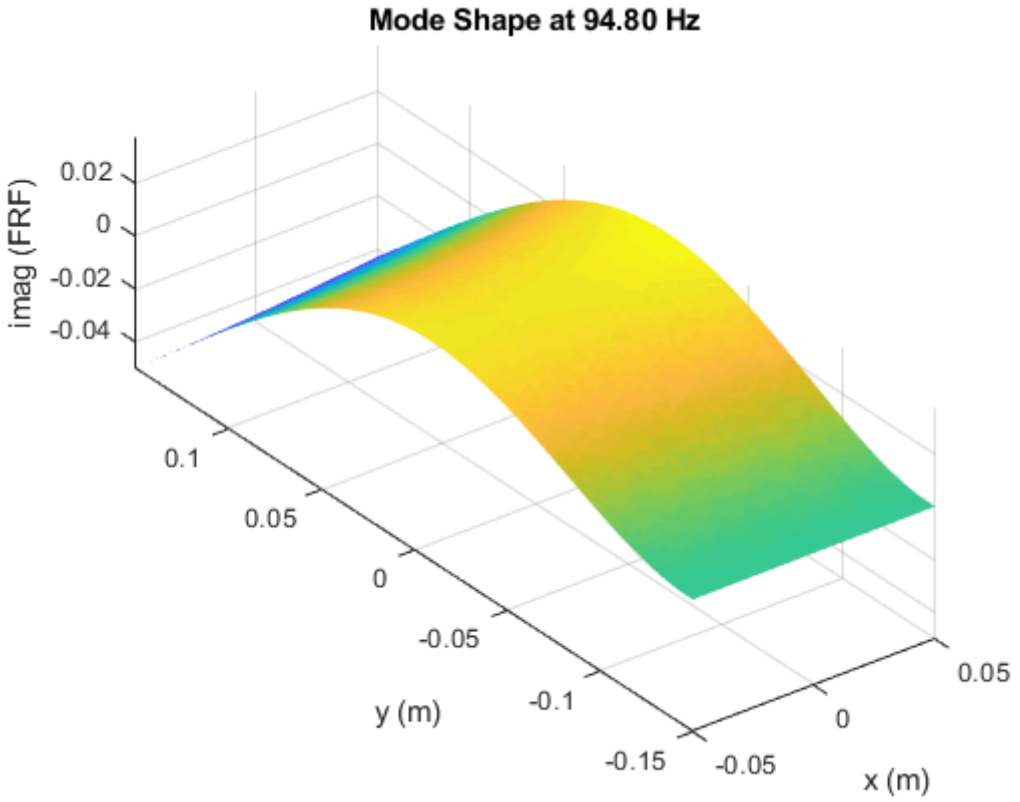


Fig 17: The Mode Shape form 1 without correction

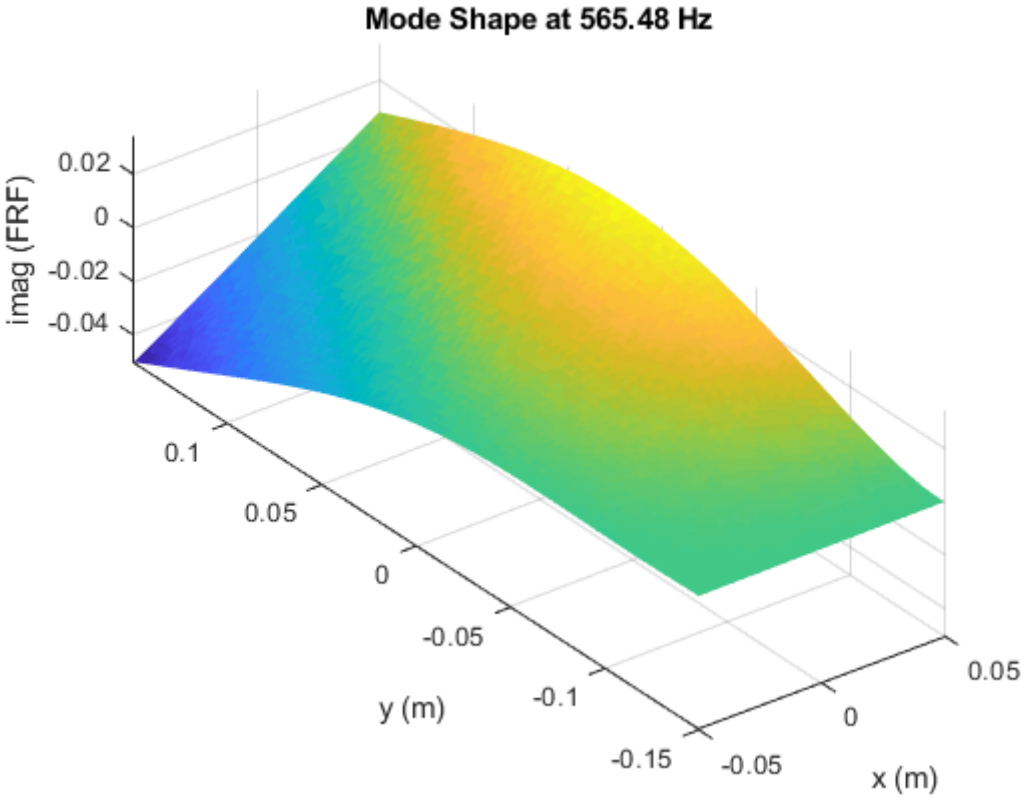


Fig18: The Mode Shape form 2 without correction

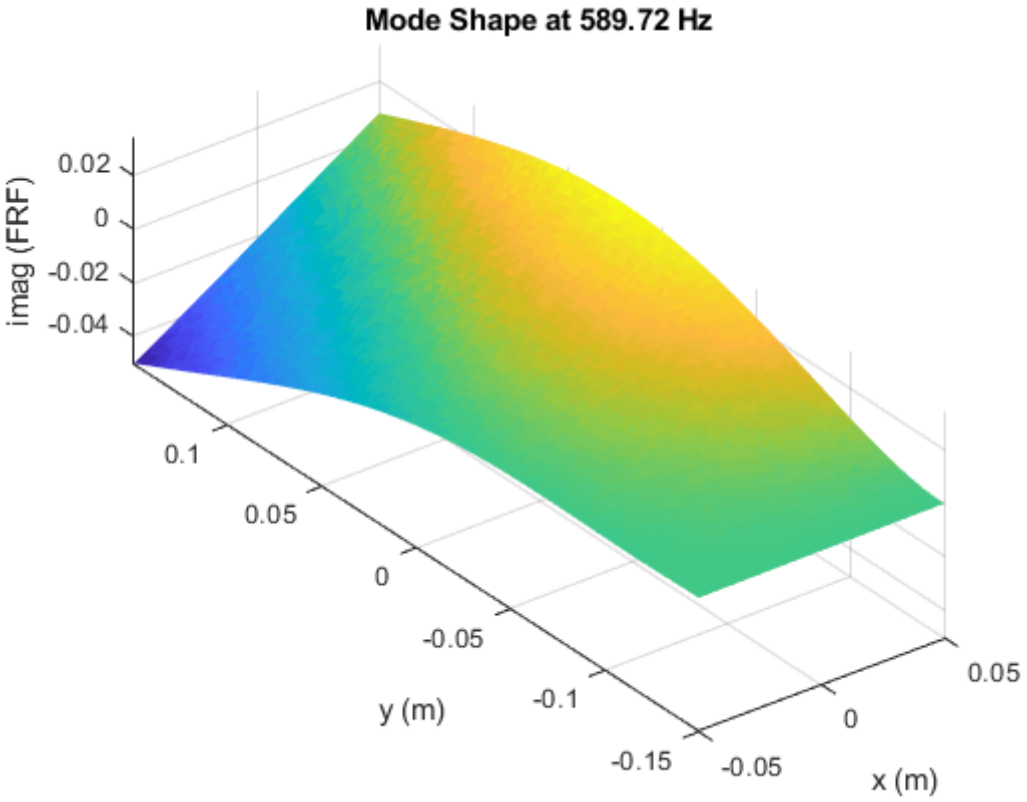


Fig 19: The Mode Shape form 3 without correction

Conclusions: The task showcases that the Mode Shapes obtained without any corrections for a system with 3% damping are overlapping and distorted due to the coupling between adjacent eigenfrequencies. Such an issue makes us unable to use the obtained data for further analysis, therefore techniques such as matlab's "modalfit" to solve this issue were developed.
Using the `modalfit` function has proven capable of correcting mode shape overlapping and distortion resulting from coupling between adjacent eigenfrequencies. By fitting functions to Frequency Response Function (FRF) data within specified frequency ranges, modalfit effectively separates and identifies individual mode shapes, reducing interference effects and providing clearer representations of system dynamics. This approach enhances the accuracy and clarity of mode shape analysis, facilitating a more insightful understanding of vibration behavior in complex dynamic systems.