Table of Contents


```
% Ruipu Ji
% SE 265
% Homework #3

clc; clear; close all;

set(0, 'DefaultTextInterpreter', 'latex');
set(0, 'DefaultLegendInterpreter', 'latex');
set(0, 'DefaultAxesTickLabelInterpreter', 'latex');

set(0, 'DefaultAxesFontSize', 20);
set(0, 'DefaultTextFontSize', 20);
```

Task A. Form the mass matrix M.

```
nDOF = 8; % Define the number of DOFs for the system.
m = [0.4194 0.4194 0.4194 0.4194 0.4194 0.4194 0.4194 0.4194]; % Mass of each
 component (unit = kg).
M = zeros(nDOF,nDOF); % Initialization for the mass matrix.

% Assign the non-zero diagonal elements in the mass matrix.
for i = 1:nDOF
    M(i,i) = m(i);
end

% Display the mass matrix M in the command window.
disp('Mass matrix M:');
disp(M);
```

```
Mass matrix M:
  Columns 1 through 7
    0.4194         0         0         0         0         0         0
         0    0.4194         0         0         0         0         0
         0         0    0.4194         0         0         0         0
         0         0         0    0.4194         0         0         0
         0         0         0         0    0.4194         0         0
         0         0         0         0         0    0.4194         0
         0         0         0         0         0         0    0.4194
```

```
         0         0         0         0         0         0         0
  Column 8
         0
         0
         0
         0
         0
         0
         0
    0.4194
```

Task B. Form the stiffness matrix K.

```
k = [56700 56700 56700 56700 56700 56700 56700 56700]; % Stiffness of each
 spring (unit = N/m).
K = zeros(nDOF,nDOF); % Initialization for the stiffness matrix.

% Assign the non-zero elements in the stiffness matrix.
for i = 1:nDOF-1
    K(i,i) = k(i) + k(i+1);
    K(i,i+1) = -k(i+1);
    K(i+1,i) = -k(i+1);
end

K(nDOF, nDOF) = k(nDOF);

% Display the stiffness matrix K in the command window.
disp('Stiffness matrix K:');
disp(K);
```

```
Stiffness matrix K:
  Columns 1 through 6
      113400      -56700           0           0           0           0
      -56700      113400      -56700           0           0           0
           0      -56700      113400      -56700           0           0
           0           0      -56700      113400      -56700           0
           0           0           0      -56700      113400      -56700
           0           0           0           0      -56700      113400
           0           0           0           0           0      -56700
           0           0           0           0           0           0
  Columns 7 through 8
           0           0
           0           0
           0           0
           0           0
           0           0
      -56700           0
      113400      -56700
      -56700       56700
```

Task C. Solve for the system eigenvalues (natural frequencies) and eigenvectors (mode shapes).

Solve the square of eigenvalues (Lambda) and eigenvectors (Phi), only select select the real part.

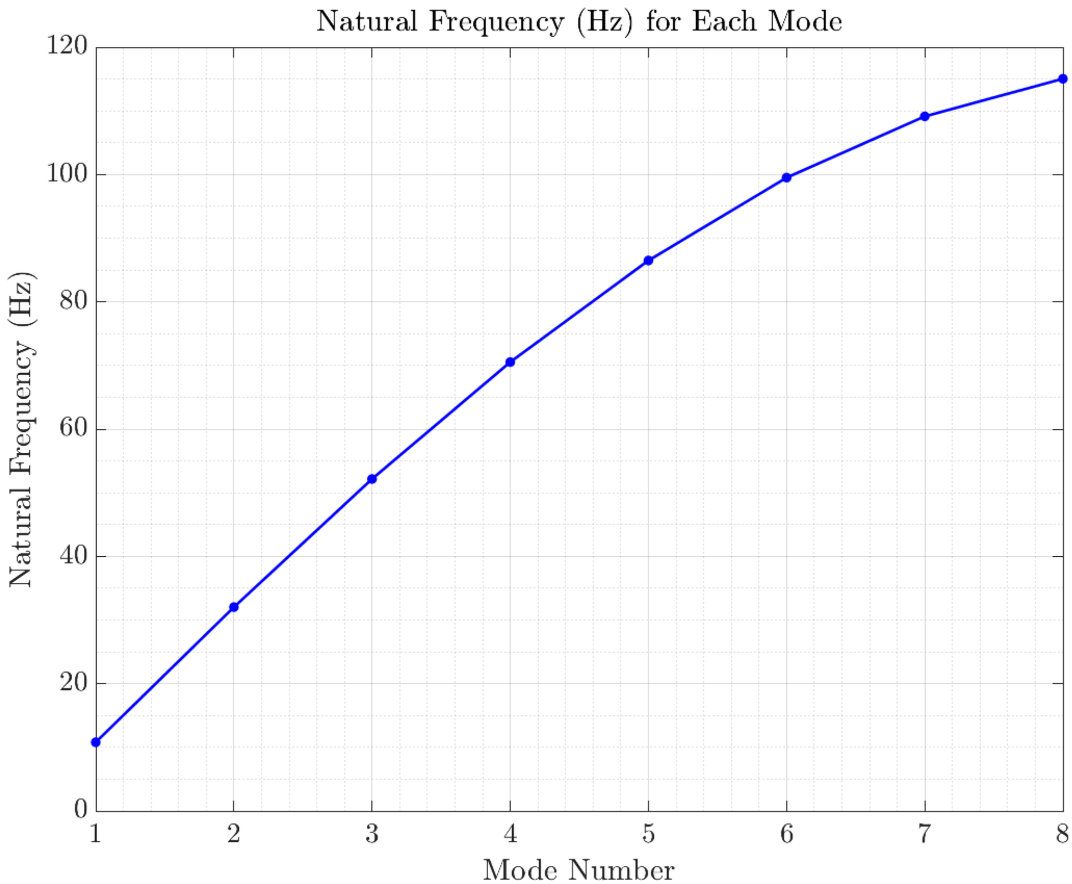
```
[Phi, Lambda] = eig(K, M);
Phi = real(Phi);
Lambda = real(Lambda);

% Save the square of eigenvalues in a new vector Lambda_d.
Lambda_d = zeros(nDOF, 1);
for i = 1:nDOF
    Lambda_d(i) = Lambda(i,i);
end

% Display the square of eigenvalues and eigenvectors.
disp('Square of eigenvalues:');
disp(Lambda_d);
disp('Eigenvectors (mode shapes):');
disp(Phi);

% Mass normalize the mode shapes.
Phi_norm = real(Phi / sqrt(Phi'*M*Phi));

% Display the mass-normalized mode shapes.
disp('Mass-normalized mode shapes:');
disp(Phi_norm);

% Orthogonality check.
disp('Orthogonality check:');
disp(real(Phi_norm' * M * Phi_norm)); % Should be the identity matrix I.
disp(real(Phi_norm' * K * Phi_norm)); % Should be the eigenvalues matrix
 Lambda.

% Calculate the natural frequencies in Hz.
Frequency = sqrt(Lambda_d)/(2*pi);

% Display the natural frequencies in Hz.
disp('Natural frequencies in Hz:');
disp(Frequency);

% Plot the natural frequencies vs mode number.
figure('Renderer', 'painters', 'Position', [10 10 1200 900]);

plot(Frequency, '-o', 'LineWidth', 2, 'MarkerSize',
 5, 'Color', 'b', 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'b');
grid on;
grid minor;
box on;
xlim([1 nDOF]);
```

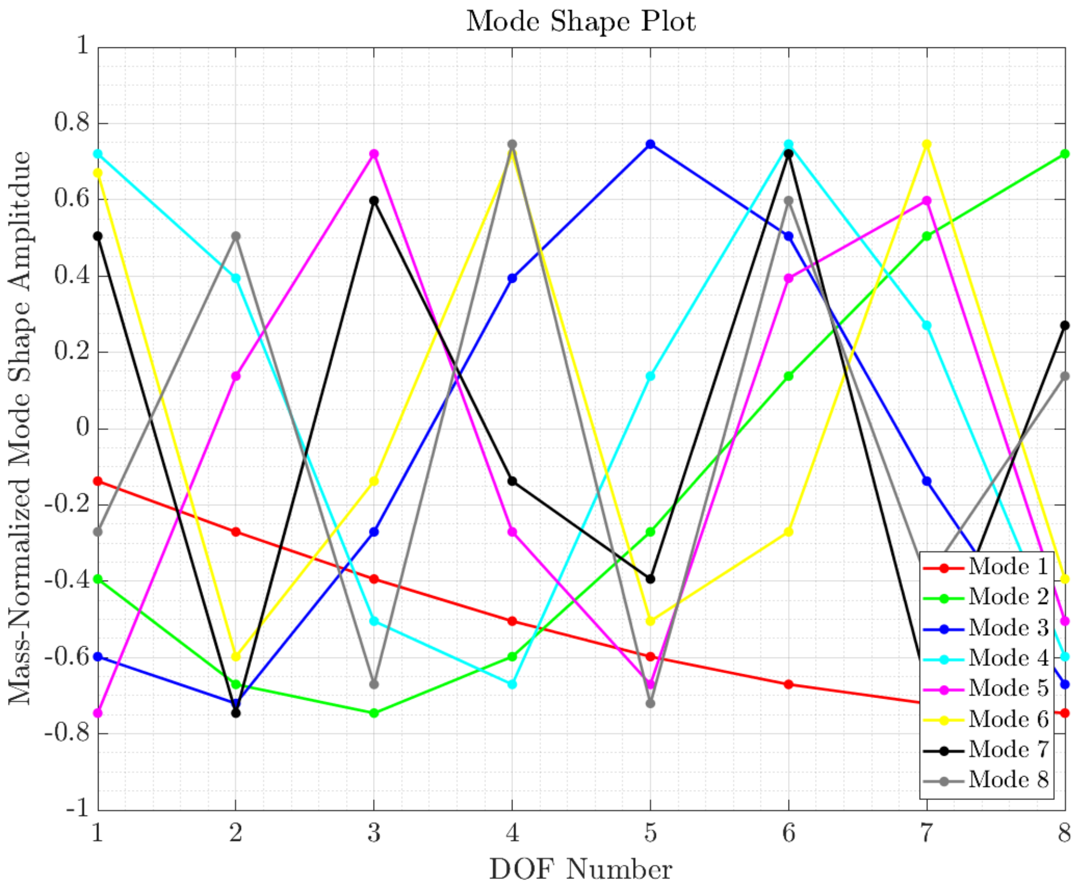
```
xticks(1:1:nDOF);
ylim([0 120]);
yticks(0:20:120);
xlabel('Mode Number');
ylabel('Natural Frequency (Hz)');
title('Natural Frequency (Hz) for Each Mode');

% Create a color map for the mode shape plot.
color = [1 0 0; 0 1 0; 0 0 1; 0 1 1;
         1 0 1; 1 1 0; 0 0 0; 0.5 0.5 0.5];

% Initialization for a cell array for the legend of each mode shape.
Legend = cell(nDOF,1);

% Plot the mode shapes.
figure('Renderer', 'painters', 'Position', [10 10 1200 900]);
hold on;

for i = 1:nDOF
    plot(Phi_norm(:,i), '-o', 'LineWidth', 2, 'MarkerSize', 5, 'Color',
 color(i,:), 'MarkerEdgeColor', color(i,:), 'MarkerFaceColor', color(i,:));
    Legend{i} = sprintf(['Mode ', num2str(i)]);
end

grid on;
grid minor;
box on;
xlim([1 nDOF]);
xticks(1:1:nDOF);
ylim([-1 1]);
yticks(-1:0.2:1);
xlabel('DOF Number');
ylabel('Mass-Normalized Mode Shape Amplitdue');
legend(Legend, 'Location', 'southeast');
title('Mode Shape Plot');

hold off;
```

```
Square of eigenvalues:
   1.0e+05 *
    0.0460
    0.4050
    1.0744
    1.9639
    2.9533
    3.9091
    4.7020
    5.2251
Eigenvectors (mode shapes):
  Columns 1 through 7
   -0.1376   -0.3943   -0.5977    0.7204   -0.7458    0.6705    0.5046
   -0.2706   -0.6705   -0.7204    0.3943    0.1376   -0.5977   -0.7458
   -0.3943   -0.7458   -0.2706   -0.5046    0.7204   -0.1376    0.5977
   -0.5046   -0.5977    0.3943   -0.6705   -0.2706    0.7204   -0.1376
```

```
   -0.5977   -0.2706    0.7458    0.1376   -0.6705   -0.5046   -0.3943
   -0.6705    0.1376    0.5046    0.7458    0.3943   -0.2706    0.7204
   -0.7204    0.5046   -0.1376    0.2706    0.5977    0.7458   -0.6705
   -0.7458    0.7204   -0.6705   -0.5977   -0.5046   -0.3943    0.2706
  Column 8
   -0.2706
    0.5046
   -0.6705
    0.7458
   -0.7204
    0.5977
   -0.3943
    0.1376
Mass-normalized mode shapes:
  Columns 1 through 7
   -0.1376   -0.3943   -0.5977    0.7204   -0.7458    0.6705    0.5046
   -0.2706   -0.6705   -0.7204    0.3943    0.1376   -0.5977   -0.7458
   -0.3943   -0.7458   -0.2706   -0.5046    0.7204   -0.1376    0.5977
   -0.5046   -0.5977    0.3943   -0.6705   -0.2706    0.7204   -0.1376
   -0.5977   -0.2706    0.7458    0.1376   -0.6705   -0.5046   -0.3943
   -0.6705    0.1376    0.5046    0.7458    0.3943   -0.2706    0.7204
   -0.7204    0.5046   -0.1376    0.2706    0.5977    0.7458   -0.6705
   -0.7458    0.7204   -0.6705   -0.5977   -0.5046   -0.3943    0.2706
  Column 8
   -0.2706
    0.5046
   -0.6705
    0.7458
   -0.7204
    0.5977
   -0.3943
    0.1376
Orthogonality check:
  Columns 1 through 7
    1.0000    0.0000   -0.0000   -0.0000   -0.0000    0.0000   -0.0000
    0.0000    1.0000   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000
   -0.0000   -0.0000    1.0000   -0.0000   -0.0000   -0.0000   -0.0000
   -0.0000   -0.0000   -0.0000    1.0000   -0.0000   -0.0000   -0.0000
   -0.0000   -0.0000   -0.0000   -0.0000    1.0000    0.0000   -0.0000
    0.0000   -0.0000   -0.0000   -0.0000    0.0000    1.0000   -0.0000
   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000    1.0000
   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000    0.0000
  Column 8
   -0.0000
   -0.0000
   -0.0000
   -0.0000
   -0.0000
   -0.0000
    0.0000
    1.0000
   1.0e+05 *
  Columns 1 through 7
    0.0460    0.0000    0.0000   -0.0000   -0.0000    0.0000   -0.0000
```

```
    0.0000    0.4050   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000
    0.0000    0.0000    1.0744    0.0000   -0.0000   -0.0000   -0.0000
   -0.0000   -0.0000    0.0000    1.9639   -0.0000   -0.0000   -0.0000
   -0.0000   -0.0000   -0.0000   -0.0000    2.9533    0.0000   -0.0000
    0.0000   -0.0000   -0.0000   -0.0000    0.0000    3.9091   -0.0000
   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000    4.7020
   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000    0.0000
  Column 8
   -0.0000
   -0.0000
   -0.0000
   -0.0000
   -0.0000
   -0.0000
    0.0000
    5.2251
Natural frequencies in Hz:
   10.7989
   32.0290
   52.1684
   70.5312
   86.4922
   99.5078
  109.1348
  115.0454
```


Natural Frequency (Hz) for Each Mode
Natural Frequency (Hz)
Mode Number
0
20
40
60
80
100
120
1
2
3
4
5
6
7
8



Task D. Analyze damaged system and compare with the modal properties from the undamaged system to those from the damage system.

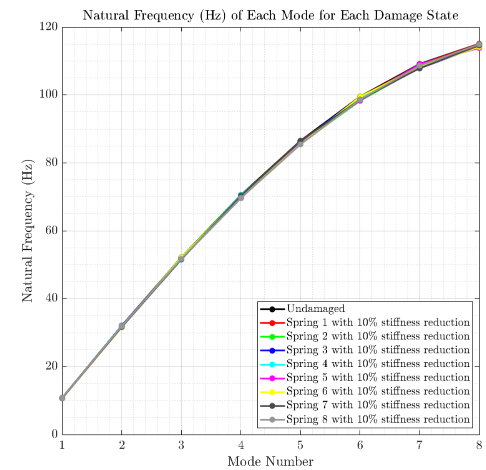
Initialization of matrices to store the frequency, mode shape 1 and mode shape 8 for each damaged conditions. Each column represents one damaged condition.

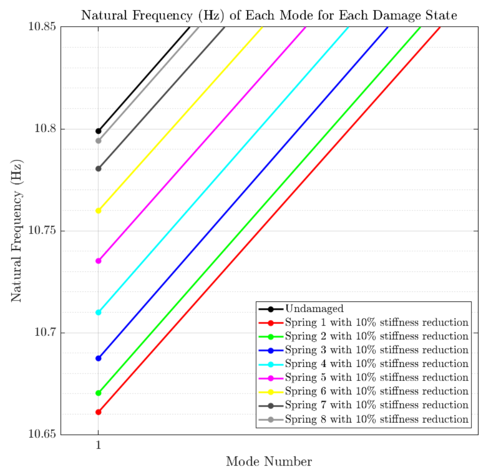
```
Frequency_damaged = zeros(nDOF, nDOF);
Phi_norm_damaged = [];
Mode1_damaged = zeros(nDOF, nDOF);
Mode8_damaged = zeros(nDOF, nDOF);

% Loop over all the 8 springs.
for i = 1:nDOF
    k_damaged = k;
    k_damaged(i) = 0.9 * k_damaged(i); % The i-th spring is damaged with
 stiffness reduced by 10%.

    % Form the stiffness matrix under damage K_damaged.
    K_damaged = zeros(nDOF,nDOF);
    for j = 1:nDOF-1
```

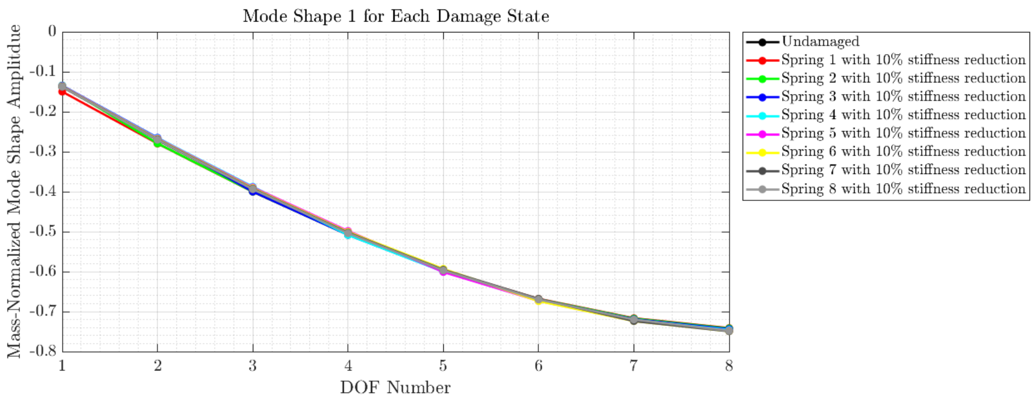
```
        K_damaged(j,j) = k_damaged(j) + k_damaged(j+1);
        K_damaged(j,j+1) = -k_damaged(j+1);
        K_damaged(j+1,j) = -k_damaged(j+1);
    end
    K_damaged(nDOF, nDOF) = k_damaged(nDOF);

    % Solve the square of eigenvalues (Lambda) and eigenvectors (Phi), only
 select select the real part.
    [Phi_damaged, Lambda_damaged] = eig(K_damaged, M);
    Phi_damaged = real(Phi_damaged);
    Lambda_damaged = real(Lambda_damaged);

    % Save the square of eigenvalues in a new vector Lambda_d.
    Lambda_d_damaged = zeros(nDOF, 1);
    for j = 1:nDOF
        Lambda_d_damaged(j) = Lambda_damaged(j,j);
    end

    % Mass normalize the mode shapes.
    Phi_norm_damaged = [Phi_norm_damaged, real(Phi_damaged /
 sqrt(Phi_damaged'*M*Phi_damaged))];

    % Save mode shape 1 and mode shape 8.
    Mode1_damaged(:,i) = Phi_norm_damaged(:,(i-1)*nDOF+1);
    Mode8_damaged(:,i) = Phi_norm_damaged(:,(i-1)*nDOF+8);

    % Calculate the natural frequencies in Hz and save the result.
    Frequency_damaged(:,i) = sqrt(Lambda_d_damaged)/(2*pi);
end

% Create a color map for the plot corresponding to different damage states.
color = [0 0 0; 1 0 0; 0 1 0;
         0 0 1; 0 1 1; 1 0 1;
         1 1 0; 0.3 0.3 0.3; 0.6 0.6 0.6];

% Create a cell array for the plot legend corresponding to different damage
 states.
LegendDamageState = cell(nDOF+1,1);
LegendDamageState{1} = 'Undamaged';
for i = 2:nDOF+1
    LegendDamageState{i} = sprintf(['Spring ', num2str(i-1), ' with 10\\%%
 stiffness reduction']);
end

% Plot the natural frequencies vs mode number for each damage state.
set(0, 'DefaultAxesFontSize', 15);
set(0, 'DefaultTextFontSize', 15);

figure('Renderer', 'painters', 'Position', [10 10 2000 800]);

subplot(1,2,1); % The full plot.
hold on;
plot(Frequency, '-o', 'LineWidth', 2, 'MarkerSize', 5, 'Color',
 color(1,:), 'MarkerEdgeColor', color(1,:), 'MarkerFaceColor', color(1,:));
```

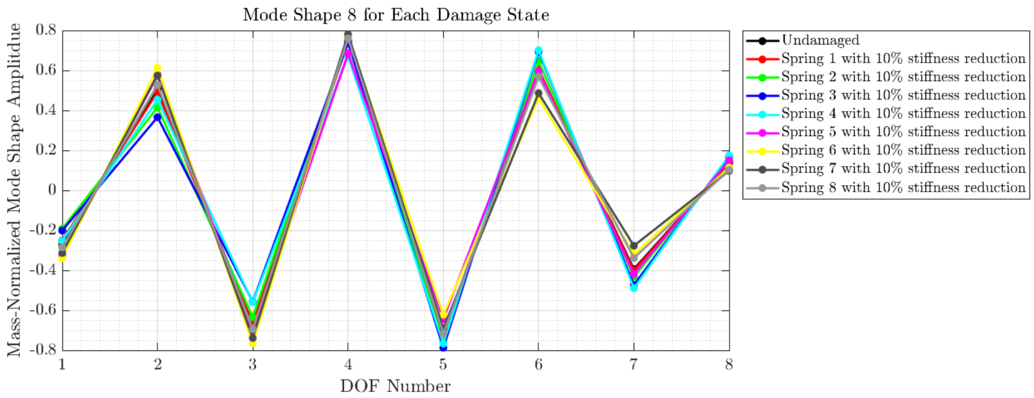
```
for i = 1:nDOF
    plot(Frequency_damaged(:,i), '-o', 'LineWidth', 2, 'MarkerSize',
 5, 'Color', color(i+1,:), 'MarkerEdgeColor', color(i+1,:), 'MarkerFaceColor',
 color(i+1,:));
end

grid on;
grid minor;
box on;
xlim([1 nDOF]);
xticks(1:1:nDOF);
ylim([0 120]);
yticks(0:20:120);
xlabel('Mode Number');
ylabel('Natural Frequency (Hz)');
legend(LegendDamageState, 'Location', 'southeast');
title('Natural Frequency (Hz) of Each Mode for Each Damage State');

subplot(1,2,2); % A zoom-in plot to show the difference.
hold on;

plot(Frequency, '-o', 'LineWidth', 2, 'MarkerSize', 5, 'Color',
 color(1,:), 'MarkerEdgeColor', color(1,:), 'MarkerFaceColor', color(1,:));

for i = 1:nDOF
    plot(Frequency_damaged(:,i), '-o', 'LineWidth', 2, 'MarkerSize',
 5, 'Color', color(i+1,:), 'MarkerEdgeColor', color(i+1,:), 'MarkerFaceColor',
 color(i+1,:));
end

grid on;
grid minor;
box on;
xlim([0.999 1.010]);
xticks(1:1:nDOF);
ylim([10.65 10.85]);
yticks(10.65:0.05:10.85);
xlabel('Mode Number');
ylabel('Natural Frequency (Hz)');
legend(LegendDamageState, 'Location', 'southeast');
title('Natural Frequency (Hz) of Each Mode for Each Damage State');

% Plot the 1st and 8th mode shapes for each damage state.
figure('Renderer', 'painters', 'Position', [10 10 1500 3000]);

subplot(2,1,1); % The 1st mode shape for each damage state.
hold on;

plot(Phi_norm(:,1), '-o', 'LineWidth', 2, 'MarkerSize', 5, 'Color',
 color(1,:), 'MarkerEdgeColor', color(1,:), 'MarkerFaceColor', color(1,:));

for i = 1:nDOF
```

```
    plot(Mode1_damaged(:,i), '-o', 'LineWidth', 2, 'MarkerSize', 5, 'Color',
 color(i+1,:), 'MarkerEdgeColor', color(i+1,:), 'MarkerFaceColor', color(i
+1,:));
end

grid on;
grid minor;
box on;
xlim([1 nDOF]);
xticks(1:1:nDOF);
ylim([-0.8 0]);
yticks(-0.8:0.1:0);
xlabel('DOF Number');
ylabel('Mass-Normalized Mode Shape Amplitdue');
legend(LegendDamageState, 'Location', 'bestoutside');
title('Mode Shape 1 for Each Damage State');

subplot(2,1,2); % The 8th mode shape for each damage state.
hold on;

plot(Phi_norm(:,8), '-o', 'LineWidth', 2, 'MarkerSize', 5, 'Color',
 color(1,:), 'MarkerEdgeColor', color(1,:), 'MarkerFaceColor', color(1,:));

for i = 1:nDOF
    plot(Mode8_damaged(:,i), '-o', 'LineWidth', 2, 'MarkerSize', 5, 'Color',
 color(i+1,:), 'MarkerEdgeColor', color(i+1,:), 'MarkerFaceColor', color(i
+1,:));
end

grid on;
grid minor;
box on;
xlim([1 nDOF]);
xticks(1:1:nDOF);
ylim([-0.8 0.8]);
yticks(-0.8:0.2:0.8);
xlabel('DOF Number');
ylabel('Mass-Normalized Mode Shape Amplitdue');
legend(LegendDamageState, 'Location', 'bestoutside');
title('Mode Shape 8 for Each Damage State');
```


Natural Frequency (Hz) of Each Mode for Each Damage State
Natural Frequency (Hz)
Mode Number
Undamaged
Spring 1 with 10% stiffness reduction
Spring 2 with 10% stiffness reduction
Spring 3 with 10% stiffness reduction
Spring 4 with 10% stiffness reduction
Spring 5 with 10% stiffness reduction
Spring 6 with 10% stiffness reduction
Spring 7 with 10% stiffness reduction
Spring 8 with 10% stiffness reduction


Natural Frequency (Hz) of Each Mode for Each Damage State
Natural Frequency (Hz)
Mode Number
Undamaged
Spring 1 with 10% stiffness reduction
Spring 2 with 10% stiffness reduction
Spring 3 with 10% stiffness reduction
Spring 4 with 10% stiffness reduction
Spring 5 with 10% stiffness reduction
Spring 6 with 10% stiffness reduction
Spring 7 with 10% stiffness reduction
Spring 8 with 10% stiffness reduction


Mode Shape 1 for Each Damage State
Mass-Normalized Mode Shape Amplitdue
DOF Number
Undamaged
Spring 1 with 10% stiffness reduction
Spring 2 with 10% stiffness reduction
Spring 3 with 10% stiffness reduction
Spring 4 with 10% stiffness reduction
Spring 5 with 10% stiffness reduction
Spring 6 with 10% stiffness reduction
Spring 7 with 10% stiffness reduction
Spring 8 with 10% stiffness reduction


Mode Shape 8 for Each Damage State
Mass-Normalized Mode Shape Amplitdue
DOF Number
Undamaged
Spring 1 with 10% stiffness reduction
Spring 2 with 10% stiffness reduction
Spring 3 with 10% stiffness reduction
Spring 4 with 10% stiffness reduction
Spring 5 with 10% stiffness reduction
Spring 6 with 10% stiffness reduction
Spring 7 with 10% stiffness reduction
Spring 8 with 10% stiffness reduction

Task E. Calculate the Modal Assurance Criteria (MAC) Metric to compare undamaged and damaged mode shapes.

```
PhiX = Phi_norm; % Phi_x = Mass-normalized mode shapes for the undamaged
 state.
PhiY = Phi_norm_damaged(:, 1:8); % Phi_y = Mass-normalized mode shapes for the
 case where 10% stiffness reduction occurs in spring 1.

MAC = zeros(nDOF, nDOF); % Initialization for the MAC matrix.

% Calculate the Modal Assurance Criteria (MAC) Metric.
for i = 1:nDOF
    for j = 1:nDOF
        MAC(i,j) = abs(PhiX(:,i)'*conj(PhiY(:,j)))^2 /
 (PhiX(:,i)'*conj(PhiX(:,i))*PhiY(:,j)'*conj(PhiY(:,j)));
    end
end

% Display the MAC result in the command window.
disp('Modal Assurance Criteria (MAC) Metric:');
disp(MAC);
```

```
Modal Assurance Criteria (MAC) Metric:
  Columns 1 through 7
    0.9999    0.0001    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0001    0.9993    0.0005    0.0001    0.0000    0.0000    0.0000
    0.0000    0.0004    0.9984    0.0008    0.0002    0.0001    0.0000
    0.0000    0.0001    0.0008    0.9978    0.0010    0.0002    0.0001
    0.0000    0.0000    0.0002    0.0009    0.9978    0.0009    0.0001
    0.0000    0.0000    0.0001    0.0002    0.0008    0.9983    0.0005
    0.0000    0.0000    0.0000    0.0001    0.0001    0.0005    0.9991
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0001    0.0002
  Column 8
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0001
    0.0002
    0.9997
```

Published with MATLAB® R2023a