Table of Contents

..... 1
Task 1 1
Task 1.A 1
Task 1.B 3
Task 1.C 4
Task 1.D 5
Task 1.E 6
Task 1.F 7
Task 2 8
Task 2.A 9
Task 2.B 10
Task 2.B.1 11
Task 2.C 12
Task 2.D 13
Task 2.D.1 14
Task 2.E 16
Task 2.F 22
Task 2.G 23
Task 2.G.1 24
Task 2.H 26
Task 2.H.1 27


```
% Ruipu Ji
% SE 265
% Homework #1

clc; clear; close all;

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

Task 1

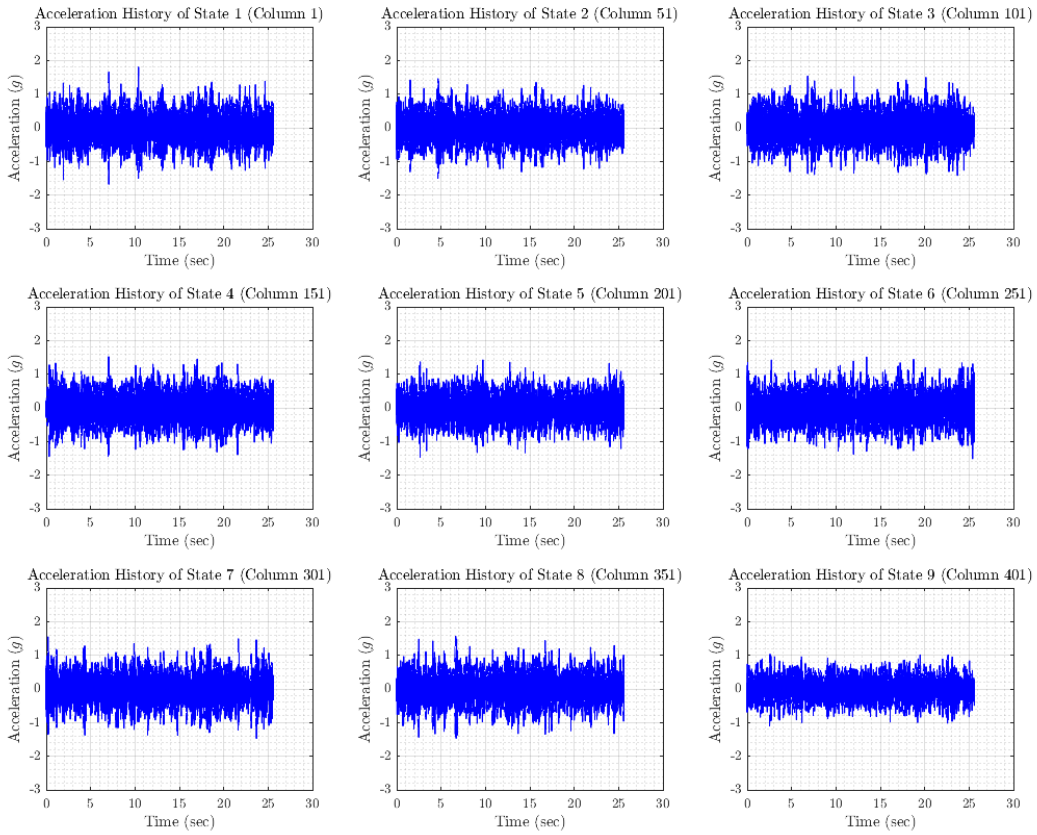
```
load('4-Story Structure Data/data3SS2009.mat'); % Load the data file.
inputData = squeeze(dataset(:,1,:)); % inputData = Data from channel 1
(input time history from the load cell).
testingData = squeeze(dataset(:,5,:)); % testingData = Data from channel 5
(acceleration from the top floor).
% squeeze() is to remove the dimension with length of 1.
n = size(inputData,1); % n = Number of data points in each set of signal.
```

Task 1.A

Generate the time history vector.

```
samplingFrequency = 320; % samplingFrequency = Sampling frequency in Hz.
timeVector = 0: 1/samplingFrequency : (n-1)/samplingFrequency; % Create the
```

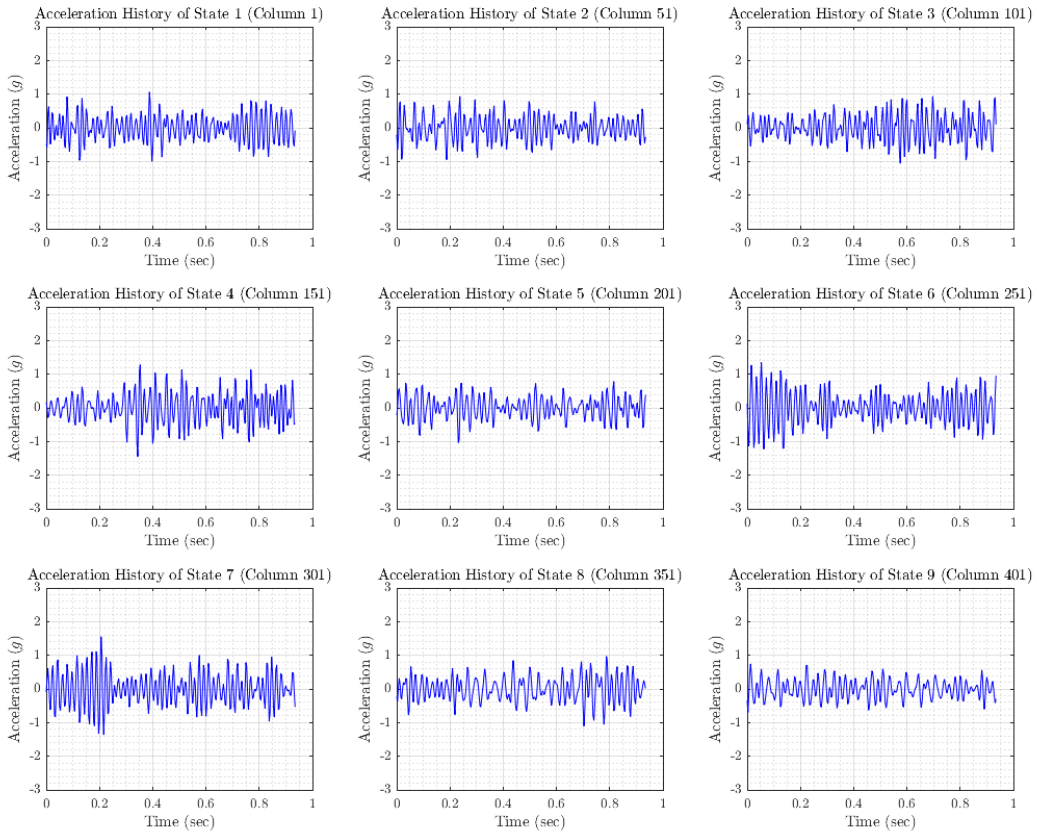
```
time vector.

set(0, 'DefaultAxesFontSize', 10);
set(0, 'DefaultTextFontSize', 10);

figure('Renderer', 'painters', 'Position', [10 10 1200 900]);

% Loop over the 9 undamaged states, plot the acceleration history.
for state = 1:9
    subplot(3,3,state);
    hold on;
    plot(timeVector, testingData(:,50*(state-1)+1), 'Color', 'b',
'LineWidth', 0.1);
    grid on;
    grid minor;
    box on;
    xlim([0 ceil(max(timeVector)/5)*5]);
    ylim([-ceil(max(abs(testingData),[],"all")) ceil(max(abs(testingData),
[],"all"))]);
    xticks(0:5:ceil(max(timeVector)/5)*5);
    yticks(-ceil(max(abs(testingData),[],"all")):1:ceil(max(abs(testingData),
[],"all")));
    xlabel('Time (sec)');
    ylabel('Acceleration ($g$)');
    title(sprintf(['Acceleration History of State ', num2str(state),
' (Column ', num2str(50*(state-1)+1), ')']));
end
```



Task 1.B

```
figure('Renderer', 'painters', 'Position', [10 10 1200 900]);

% Loop over the 9 undamaged states, plot the acceleration history (first 300
data points).
for state = 1:9
    subplot(3,3,state);
    hold on;
    plot(timeVector(1:300), testingData(1:300,50*(state-1)+1), 'Color', 'b',
'LineWidth', 0.1);
    grid on;
    grid minor;
    box on;
    xlim([0 ceil(max(timeVector(1:300))/0.2)*0.2]);
    ylim([-ceil(max(abs(testingData),[],"all")) ceil(max(abs(testingData),
[],"all"))]);
    xticks(0:0.2:ceil(max(timeVector(1:300))/0.2)*0.2);
    yticks(-ceil(max(abs(testingData),[],"all")):1:ceil(max(abs(testingData),
[],"all")));
    xlabel('Time (sec)');
    ylabel('Acceleration ($g$)');
```

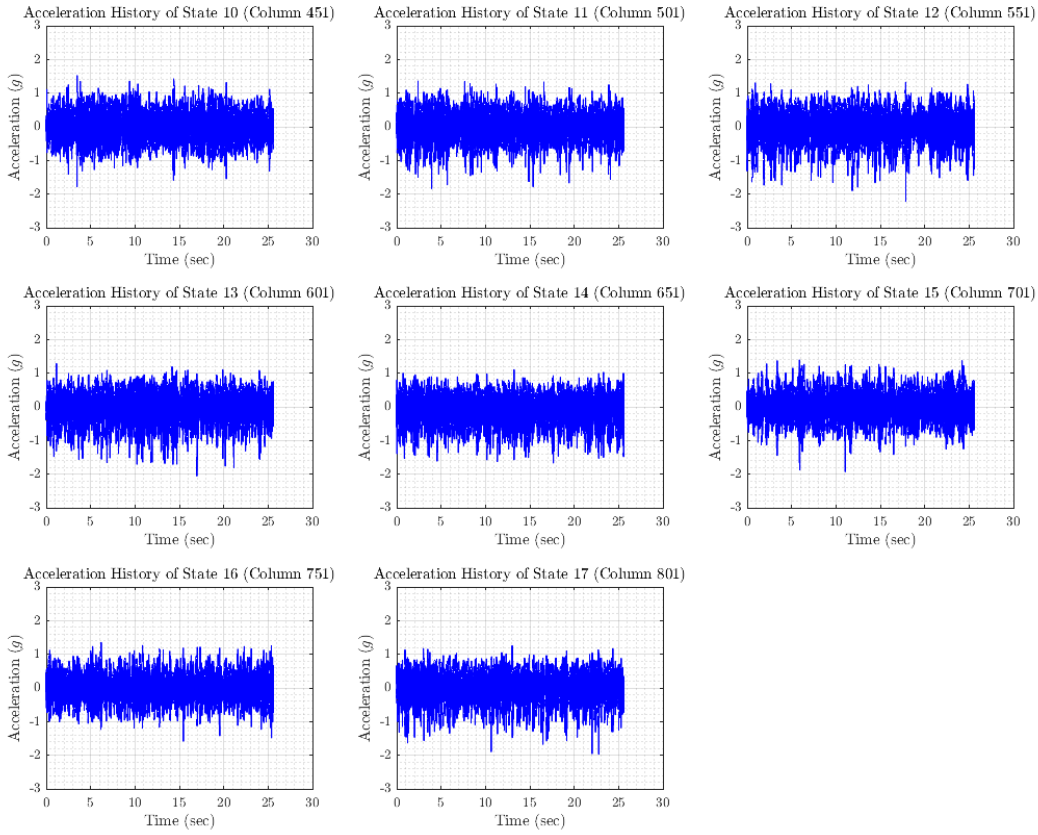
```
    title(sprintf(['Acceleration History of State ', num2str(state),
' (Column ', num2str(50*(state-1)+1), ')']));
end
```



Task 1.C

```
figure('Renderer', 'painters', 'Position', [10 10 1200 900]);

% Loop over the 8 damaged states, plot the acceleration history.
for state = 1:8
    subplot(3,3,state);
    hold on;
    plot(timeVector, testingData(:,50*(state+8)+1), 'Color', 'b',
'LineWidth', 0.1);
    grid on;
    grid minor;
    box on;
    xlim([0 ceil(max(timeVector)/5)*5]);
    ylim([-ceil(max(abs(testingData),[],"all")) ceil(max(abs(testingData),
[],"all"))]);
    xticks(0:5:ceil(max(timeVector)/5)*5);
    yticks(-ceil(max(abs(testingData),[],"all")):1:ceil(max(abs(testingData),
[],"all")));
```

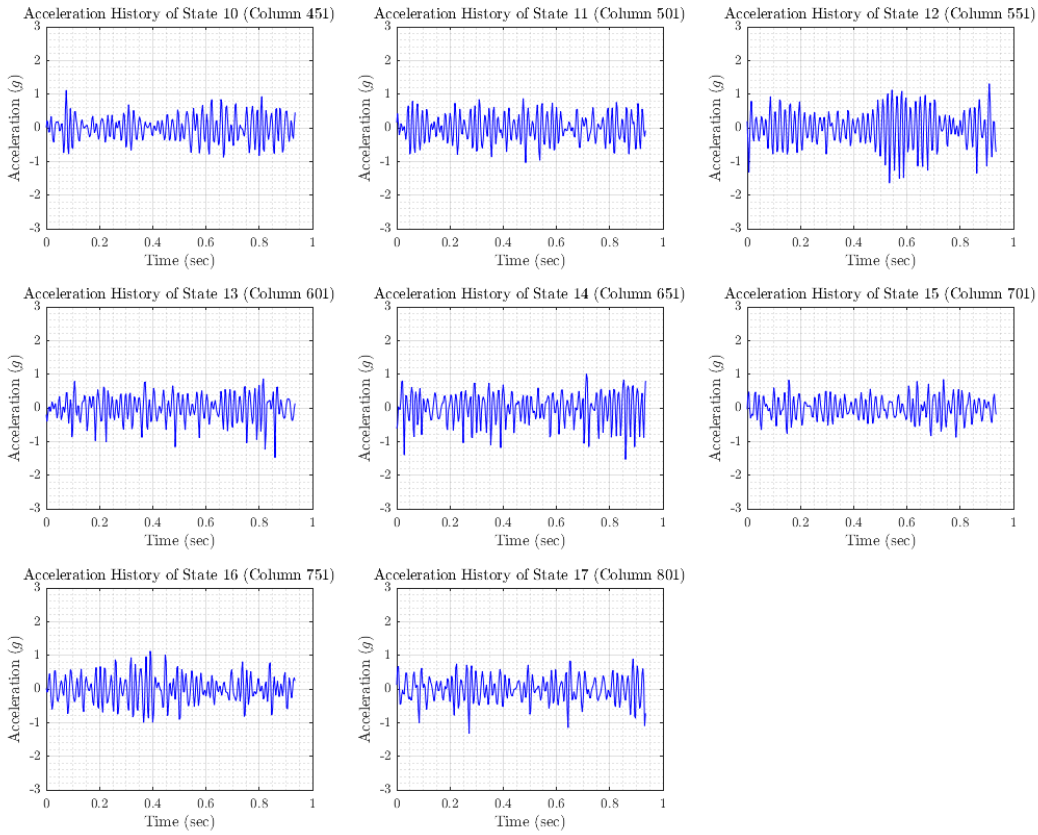
```
    xlabel('Time (sec)');
    ylabel('Acceleration ($g$)');
    title(sprintf(['Acceleration History of State ', num2str(state+9),
' (Column ', num2str(50*(state+8)+1), ')']));
end
```



Task 1.D

```
figure('Renderer', 'painters', 'Position', [10 10 1200 900]);

% Loop over the 8 damaged states, plot the acceleration history (first 300
data points).
for state = 1:8
    subplot(3,3,state);
    hold on;
    plot(timeVector(1:300), testingData(1:300,50*(state+8)+1), 'Color', 'b',
'LineWidth', 0.1);
    grid on;
    grid minor;
    box on;
    xlim([0 ceil(max(timeVector(1:300))/0.2)*0.2]);
    ylim([-ceil(max(abs(testingData),[],"all")) ceil(max(abs(testingData),
[],"all"))]);
```

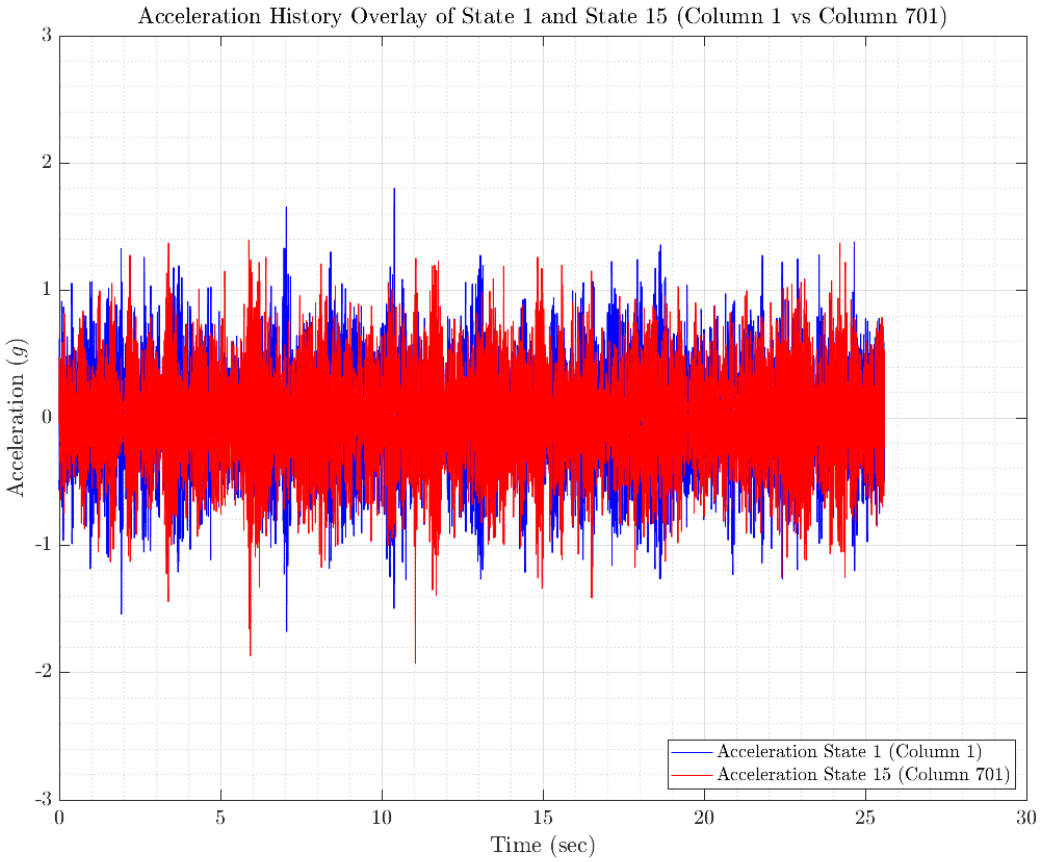
```
    xticks(0:0.2:ceil(max(timeVector(1:300))/0.2)*0.2);
    yticks(-ceil(max(abs(testingData),[],"all")):1:ceil(max(abs(testingData),
[],"all")));
    xlabel('Time (sec)');
    ylabel('Acceleration ($g$)');
    title(sprintf(['Acceleration History of State ', num2str(state+9),
' (Column ', num2str(50*(state+8)+1), ')']));
end
```



Task 1.E

```
set(0, 'DefaultAxesFontSize', 15);
set(0, 'DefaultTextFontSize', 15);

% Plot the acceleration history overlay of state 1 and state 15.
figure('Renderer', 'painters', 'Position', [10 10 1200 900]);
hold on;
plot(timeVector, testingData(:,1), 'Color', 'b', 'LineWidth', 0.1); % State
1.
plot(timeVector, testingData(:,701), 'Color', 'r', 'LineWidth', 0.1); %
State 15.
grid on;
grid minor;
```

```
box on;
xlim([0 ceil(max(timeVector)/5)*5]);
ylim([-ceil(max(abs(testingData),[],"all")) ceil(max(abs(testingData),
[],"all"))]);
xticks(0:5:ceil(max(timeVector)/5)*5);
yticks(-ceil(max(abs(testingData),[],"all")):1:ceil(max(abs(testingData),
[],"all")));
xlabel('Time (sec)');
ylabel('Acceleration ($g$)');
legend('Acceleration State 1 (Column 1)', 'Acceleration State
15 (Column 701)', 'Location', 'southeast');
title(sprintf('Acceleration History Overlay of State 1 and State
15 (Column 1 vs Column 701)'));
```

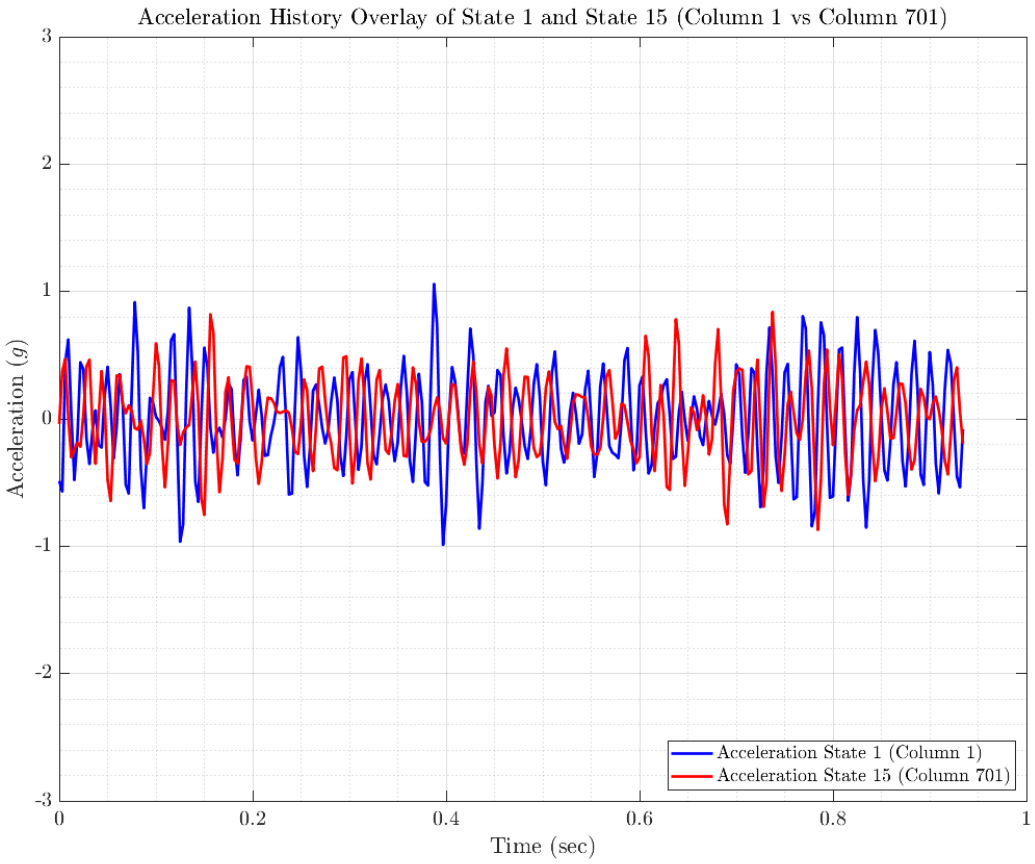


Task 1.F

Plot the acceleration history overlay of state 1 and state 15 (first 300 data points).

```
figure('Renderer', 'painters', 'Position', [10 10 1200 900]);
hold on;
plot(timeVector(1:300), testingData(1:300,1), 'Color', 'b', 'LineWidth', 2);
% State 1.
plot(timeVector(1:300), testingData(1:300,701), 'Color', 'r', 'LineWidth',
```

```
2); % State 15.
grid on;
grid minor;
box on;
xlim([0 ceil(max(timeVector(1:300))/0.2)*0.2]);
ylim([-ceil(max(abs(testingData),[],"all")) ceil(max(abs(testingData),
[],"all"))]);
xticks(0:0.2:ceil(max(timeVector(1:300))/0.2)*0.2);
yticks(-ceil(max(abs(testingData),[],"all")):1:ceil(max(abs(testingData),
[],"all")));
xlabel('Time (sec)');
ylabel('Acceleration ($g$)');
legend('Acceleration State 1 (Column 1)', 'Acceleration State
15 (Column 701)', 'Location', 'southeast');
title(sprintf('Acceleration History Overlay of State 1 and State
15 (Column 1 vs Column 701)'));
```



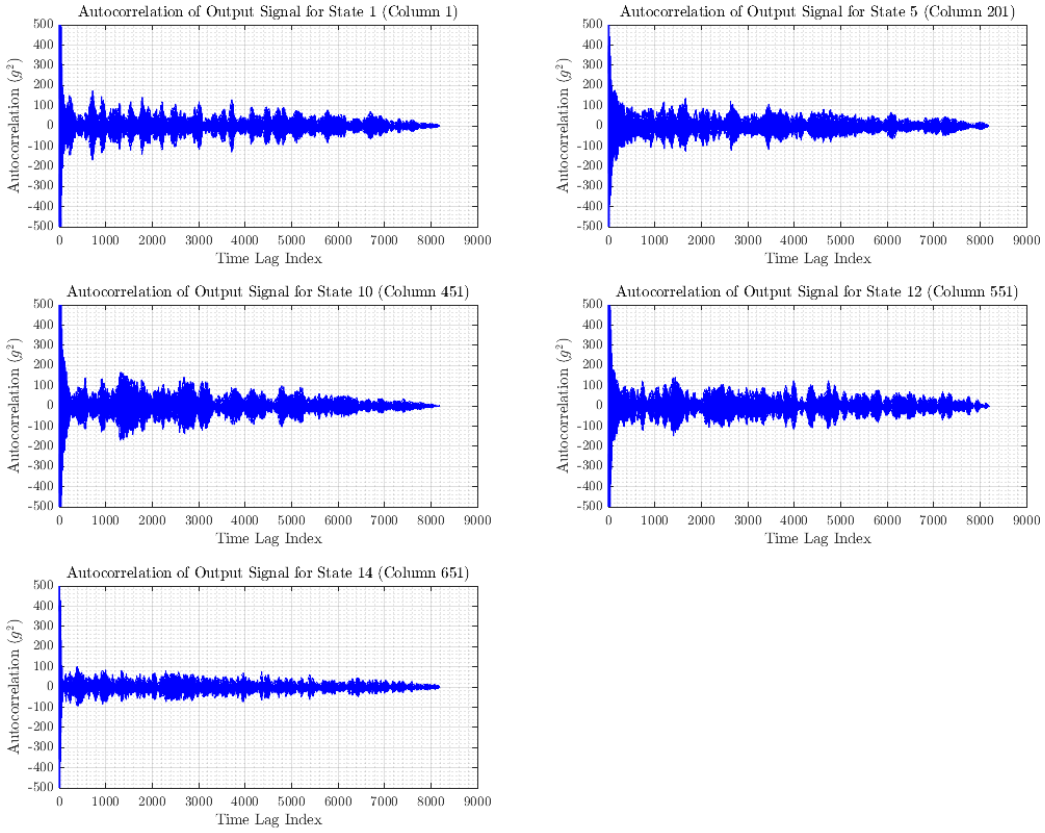
Task 2

```
state = [1, 5, 10, 12, 14]; % Create a vector for the state number.
column = 50*(state-1)+1; % Create a vector for the column number.
```

```
NyquistFrequency = samplingFrequency/2; % Calculate the Nyquist frequency
based on the sampling frequency.
```

Task 2.A

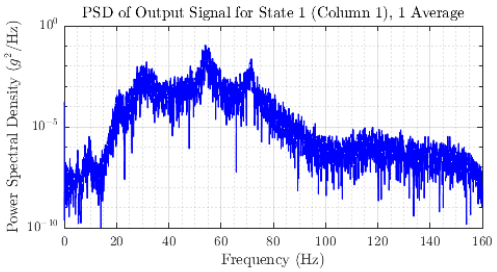
```
set(0, 'DefaultAxesFontSize', 10);
set(0, 'DefaultTextFontSize', 10);

figure('Renderer', 'painters', 'Position', [10 10 1200 900]);

% Loop over the 5 different states:
for i = 1:5
    [r, lags] = xcorr(testingData(:,column(i))); % Calculate the
autocorrelation of the signal.
    subplot(3,2,i);
    plot(lags, r, 'Color', 'b', 'LineWidth', 0.1); % Plot the
autocorrelation function vs time lag.
    grid on;
    grid minor;
    box on;
    xlim([0 ceil(n/1000)*1000]);
    ylim([-500 500]);
    xticks(0:1000:ceil(n/1000)*1000);
    yticks(-500:100:500);
    xlabel('Time Lag Index');
    ylabel('Autocorrelation ($g^2$)');
    title(sprintf(['Autocorrelation of Output Signal for State ',
num2str(state(i)), ' (Column ', num2str(column(i)), ')']));
end
```

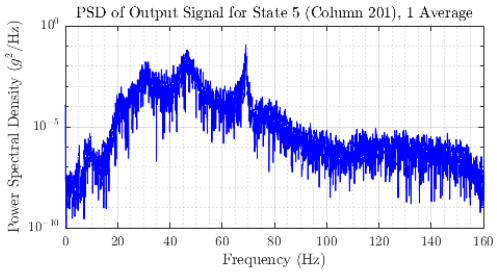


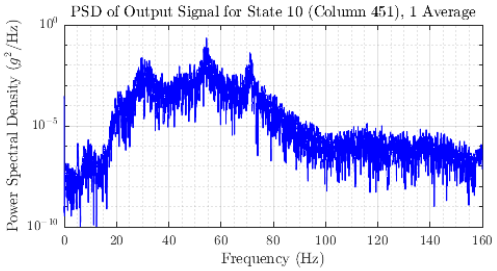
Task 2.B

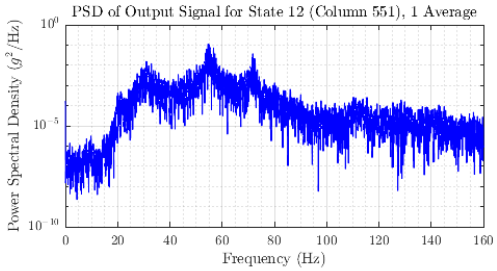
```
figure('Renderer', 'painters', 'Position', [10 10 1200 900]);

% Loop over the 5 different states:
for i = 1:5
    % Calculate the power spectral density of the signal.
    % Hanning window averaging the entire signal and zero overlap is applied
here.
    [psd, f] = pwelch(testingData(:,column(i)), hann(n), 0, [],
samplingFrequency);
    subplot(3,2,i);
    semilogy(f, psd, 'Color', 'b', 'LineWidth', 0.1); % Plot the power
spectral density vs frequency in the log scale.
    grid on;
    grid minor;
    box on;
    xlim([0 ceil(NyquistFrequency/20)*20]);
    ylim([1e-10 1e0]);
    xticks(0:20:ceil(NyquistFrequency/20)*20);
    yticks(10.^(-10:5:0));
    xlabel('Frequency (Hz)');
```

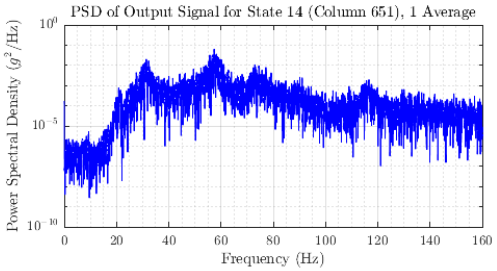
```
    ylabel('Power Spectral Density ($g^2$/Hz)');
    title(sprintf(['PSD of Output Signal for State ', num2str(state(i)),
' (Column ', num2str(column(i)), '), 1 Average']));
end
```







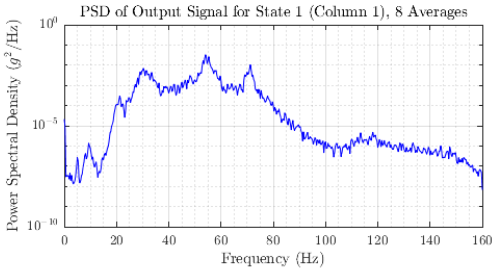


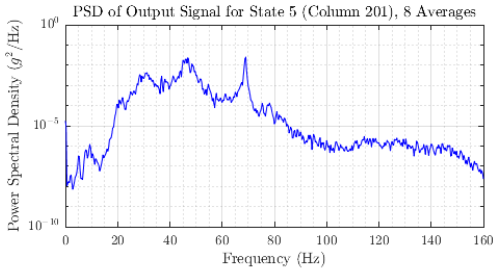


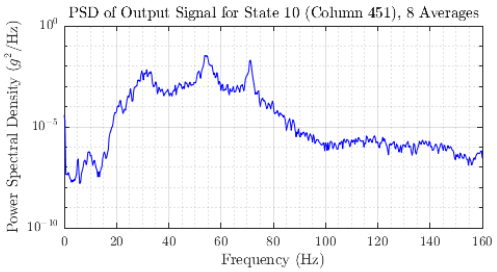
Task 2.B.1

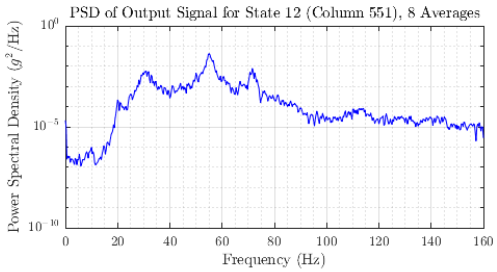
```
figure('Renderer', 'painters', 'Position', [10 10 1200 900]);

% Loop over the 5 different states:
for i = 1:5
    % Calculate the power spectral density of the signal.
    % Hanning window averaging in 8 different blocks and zero overlap is
applied here.
    [psd, f] = pwelch(testingData(:,column(i)), hann(n/8), 0, [],
samplingFrequency);
    subplot(3,2,i);
    semilogy(f, psd, 'Color', 'b', 'LineWidth', 0.1); % Plot the power
spectral density vs frequency in the log scale.
    grid on;
    grid minor;
    box on;
    xlim([0 ceil(NyquistFrequency/20)*20]);
```

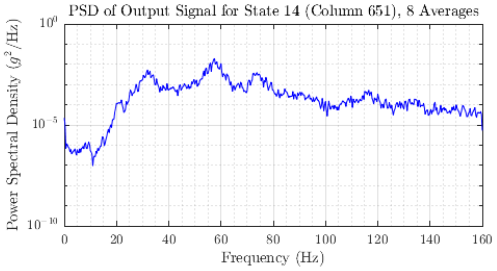
```
    ylim([1e-10 1e0]);
    xticks(0:20:ceil(NyquistFrequency/20)*20);
    yticks(10.^(-10:5:0));
    xlabel('Frequency (Hz)');
    ylabel('Power Spectral Density ($g^2$/Hz)');
    title(sprintf(['PSD of Output Signal for State ', num2str(state(i)),
' (Column ', num2str(column(i)), '), 8 Averages']));
end
```







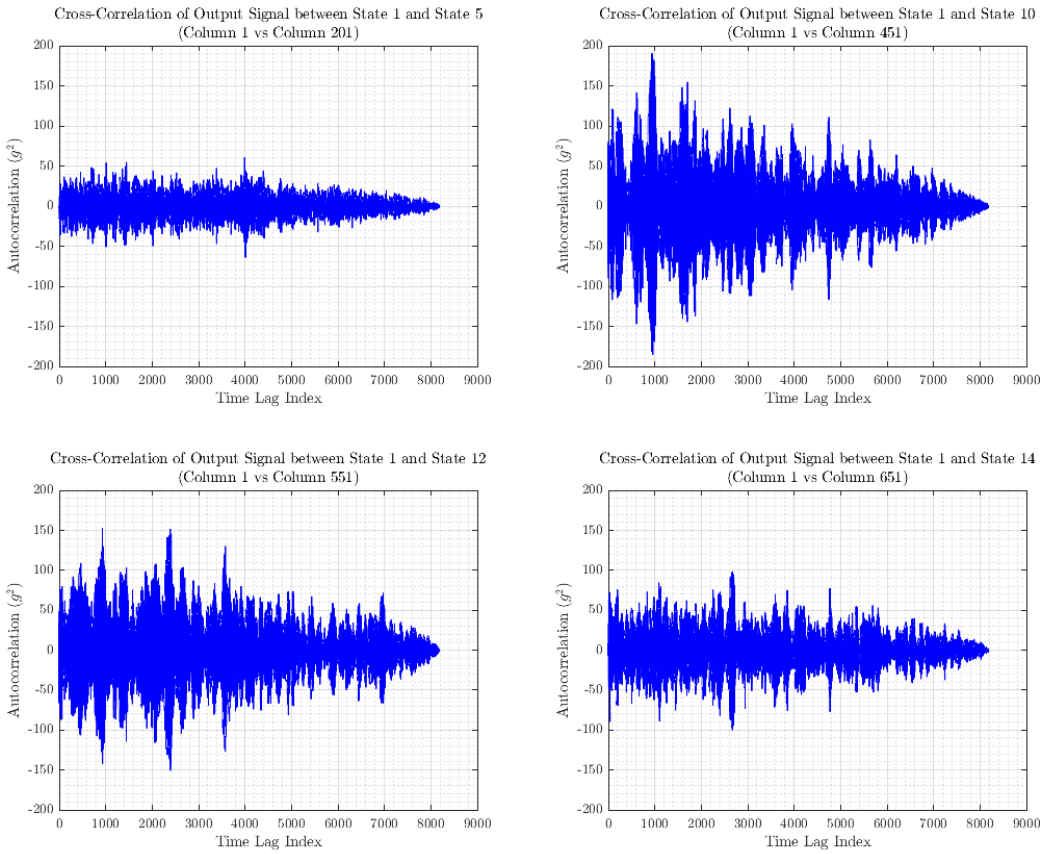




Task 2.C

```
figure('Renderer', 'painters', 'Position', [10 10 1200 900]);

% Loop over the 4 different states:
for i = 1:4
    [r, lags] = xcorr(testingData(:,column(1)), testingData(:,column(i+1)));
% Calculate the cross-correlation of the signal.
    subplot(2,2,i);
    plot(lags, r, 'Color', 'b', 'LineWidth', 0.1); % Plot the cross-
correlation function vs time lag.
    grid on;
    grid minor;
    box on;
```

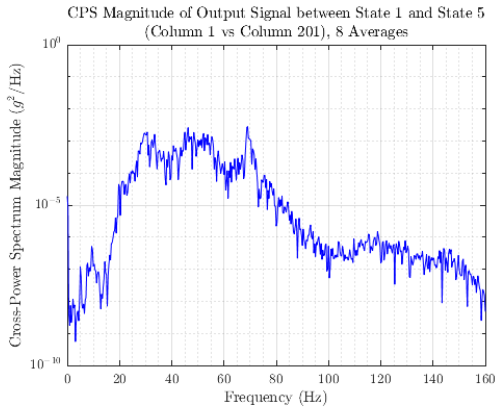
```
    xlim([0 ceil(n/1000)*1000]);
    ylim([-200 200]);
    xticks(0:1000:ceil(n/1000)*1000);
    yticks(-200:50:200);
    xlabel('Time Lag Index');
    ylabel('Autocorrelation ($g^2$)');
    title(sprintf(['Cross-Correlation of Output Signal between State
1 and State ', num2str(state(i+1)),'\n','(Column 1 vs Column ',
num2str(column(i+1)), ')']));
end
```

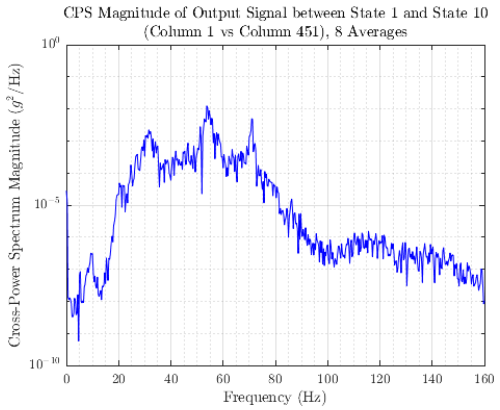


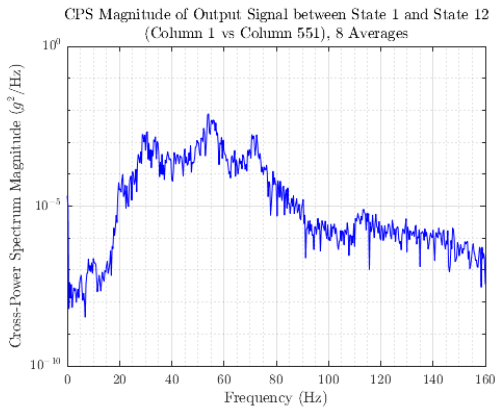
Task 2.D

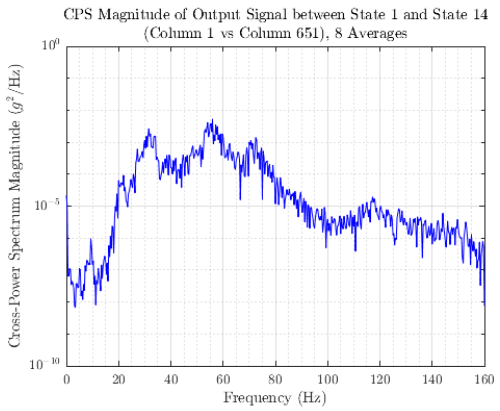
```
figure('Renderer', 'painters', 'Position', [10 10 1200 900]);

% Loop over the 4 different states:
for i = 1:4
    % Calculate the cross-power spectral density of the signal.
    % Hanning window averaging in 8 different blocks and zero overlap is
applied here.
    [psd, f] = cpsd(testingData(:,column(1)), testingData(:,column(i+1)),
hann(n/8), 0, [], samplingFrequency);
    subplot(2,2,i);
```

```
    semilogy(f, abs(psd), 'Color', 'b', 'LineWidth', 0.1); % Plot the cross-
power spectrum magnitude vs frequency in the log scale.
    grid on;
    grid minor;
    box on;
    xlim([0 ceil(NyquistFrequency/20)*20]);
    ylim([1e-10 1e0]);
    xticks(0:20:ceil(NyquistFrequency/20)*20);
    yticks(10.^(-10:5:0));
    xlabel('Frequency (Hz)');
    ylabel('Cross-Power Spectrum Magnitude ($g^2$/Hz)');
    title(sprintf(['CPS Magnitude of Output Signal between State 1 and State
', num2str(state(i+1)),'\n','(Column 1 vs Column ', num2str(column(i+1)),
'), 8 Averages']));
end
```





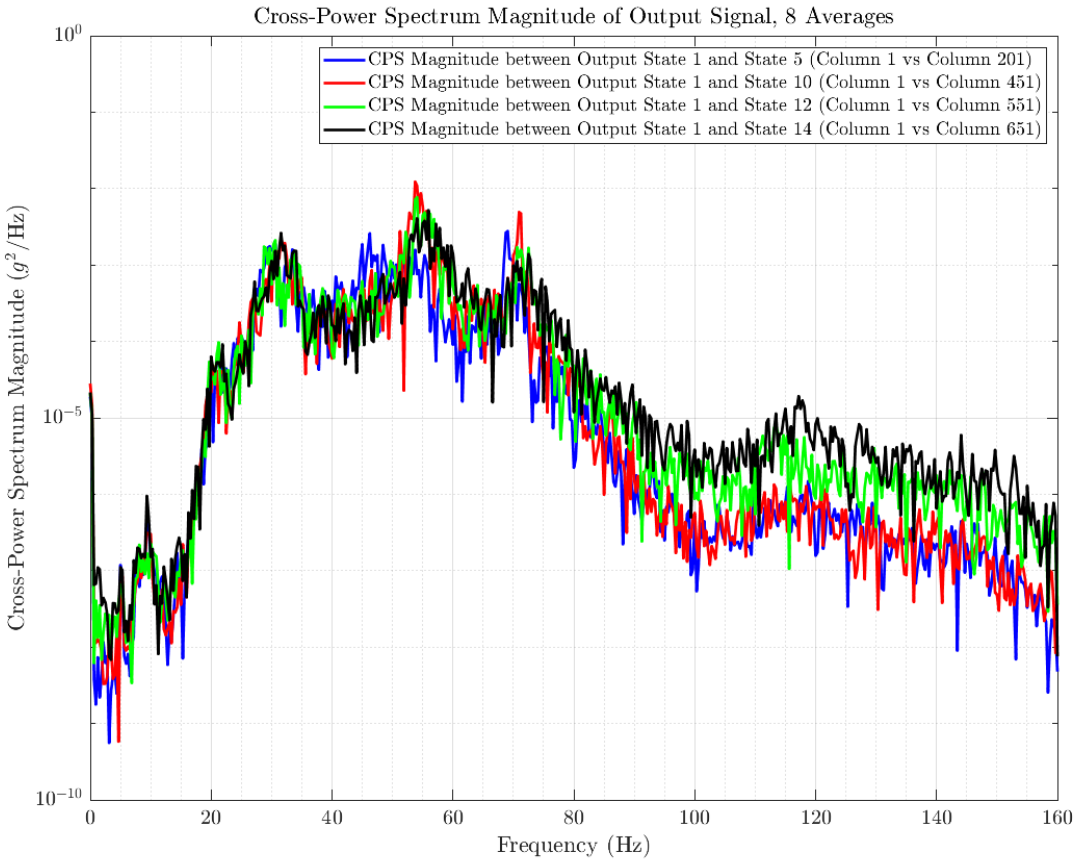




Task 2.D.1

```
set(0, 'DefaultAxesFontSize', 15);
set(0, 'DefaultTextFontSize', 15);

color = ['b', 'r', 'g', 'k', 'm']; % Create a vector for colors in the plot.
```

```
figure('Renderer', 'painters', 'Position', [10 10 1200 900]);

% Loop over the 4 different states:
for i = 1:4
    % Calculate the cross-power spectral density of the signal.
    % Hanning window averaging in 8 different blocks and zero overlap is
applied here.
    [psd, f] = cpsd(testingData(:,column(1)), testingData(:,column(i+1)),
hann(n/8), 0, [], samplingFrequency);
    semilogy(f, abs(psd), 'Color', color(i), 'LineWidth', 2); % Plot the
cross-power spectrum magnitude vs frequency in the log scale.
    hold on;
end

grid on;
grid minor;
box on;
xlim([0 ceil(NyquistFrequency/20)*20]);
ylim([1e-10 1e0]);
xticks(0:20:ceil(NyquistFrequency/20)*20);
yticks(10.^(-10:5:0));
xlabel('Frequency (Hz)');
ylabel('Cross-Power Spectrum Magnitude ($g^2$/Hz)');

legend('CPS Magnitude between Output State 1 and State 5 (Column 1 vs Column
201)', 'CPS Magnitude between Output State 1 and State 10 (Column 1 vs
Column 451)', 'CPS Magnitude between Output State 1 and State 12 (Column 1
vs Column 551)', 'CPS Magnitude between Output State 1 and State 14 (Column
1 vs Column 651)', 'Location', 'northeast');
title(sprintf('Cross-Power Spectrum Magnitude of Output Signal, 8
Averages'));
```



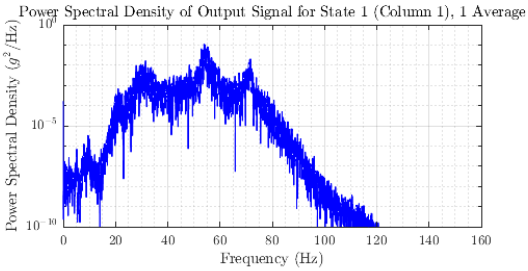
Task 2.E

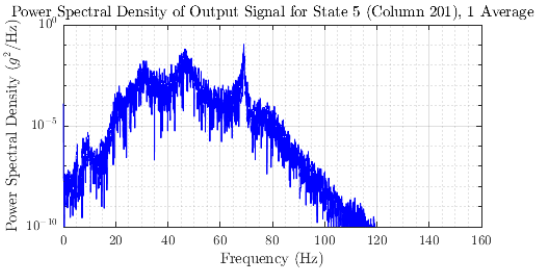
Create the 6th order Butterworth low-pass filter. [coefficients, filter] = butter(order, frequency). Note that the frequency in the input should be the normalized cut-off frequency.

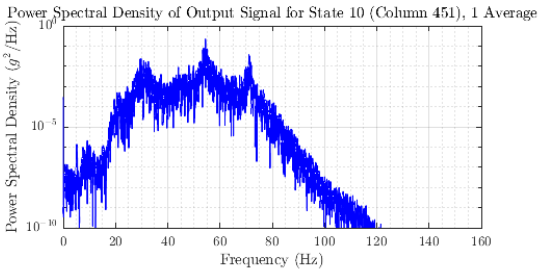
```
CutOffFrequency = 80; % Cut-off frequency in Hz.
[c, f] = butter(6, CutOffFrequency/NyquistFrequency);

% Filter the testing data with the generated 6th order Butterworth low-pass
filter.
testingDataFiltered = filter(c, f, testingData);

% Repeat Task 2.B ------------------------------------------------------------
set(0, 'DefaultAxesFontSize', 10);
set(0, 'DefaultTextFontSize', 10);

figure('Renderer', 'painters', 'Position', [10 10 1200 900]);

% Loop over the 5 different states:
for i = 1:5
    % Calculate the power spectral density of the signal.
    % Hanning window averaging the entire signal and zero overlap is applied
here.
```

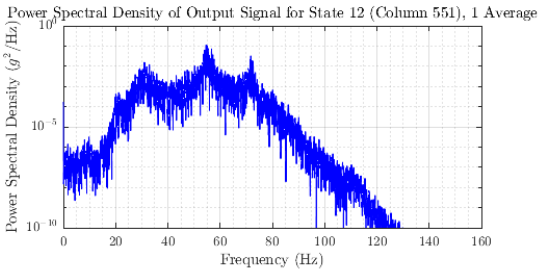
```
    [psd, f] = pwelch(testingDataFiltered(:,column(i)), hann(n), 0, [],
samplingFrequency);
    subplot(3,2,i);
    semilogy(f, psd, 'Color', 'b', 'LineWidth', 0.1); % Plot the power
spectral density vs frequency in the log scale.
    grid on;
    grid minor;
    box on;
    xlim([0 ceil(NyquistFrequency/20)*20]);
    ylim([1e-10 1e0]);
    xticks(0:20:ceil(NyquistFrequency/20)*20);
    yticks(10.^(-10:5:0));
    xlabel('Frequency (Hz)');
    ylabel('Power Spectral Density ($g^2$/Hz)');
    title(sprintf(['Power Spectral Density of Output Signal for State ',
num2str(state(i)), ' (Column ', num2str(column(i)), '), 1 Average']));
end

% Repeat Task 2.B.1 ------------------------------------------------------
figure('Renderer', 'painters', 'Position', [10 10 1200 900]);

% Loop over the 5 different states:
for i = 1:5
    % Calculate the power spectral density of the signal.
    % Hanning window averaging in 8 different blocks and zero overlap is
applied here.
    [psd, f] = pwelch(testingDataFiltered(:,column(i)), hann(n/8), 0, [],
samplingFrequency);
    subplot(3,2,i);
    semilogy(f, psd, 'Color', 'b', 'LineWidth', 0.1); % Plot the power
spectral density vs frequency in the log scale.
    grid on;
    grid minor;
    box on;
    xlim([0 ceil(NyquistFrequency/20)*20]);
    ylim([1e-10 1e0]);
    xticks(0:20:ceil(NyquistFrequency/20)*20);
    yticks(10.^(-10:5:0));
    xlabel('Frequency (Hz)');
    ylabel('Power Spectral Density ($g^2$/Hz)');
    title(sprintf(['Power Spectral Density of Output Signal for State ',
num2str(state(i)), ' (Column ', num2str(column(i)), '), 8 Average']));
end

% Repeat Task 2.D --------------------------------------------------------
figure('Renderer', 'painters', 'Position', [10 10 1200 900]);

% Loop over the 4 different states:
for i = 1:4
    % Calculate the cross-power spectral density of the signal.
    % Hanning window averaging in 8 different blocks and zero overlap is
applied here.
    [psd, f] = cpsd(testingDataFiltered(:,column(1)),
testingDataFiltered(:,column(i+1)), hann(n/8), 0, [], samplingFrequency);
```

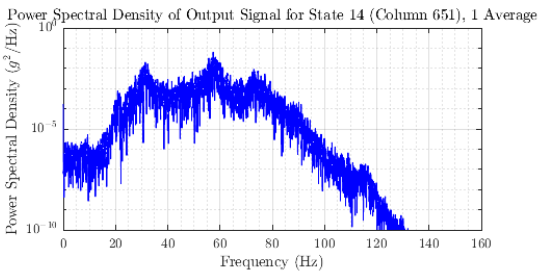
```
    subplot(2,2,i);
    semilogy(f, abs(psd), 'Color', 'b', 'LineWidth', 0.1); % Plot the cross-
power spectrum magnitude vs frequency in the log scale.
    grid on;
    grid minor;
    box on;
    xlim([0 ceil(NyquistFrequency/20)*20]);
    ylim([1e-10 1e0]);
    xticks(0:20:ceil(NyquistFrequency/20)*20);
    yticks(10.^(-10:5:0));
    xlabel('Frequency (Hz)');
    ylabel('Cross-Power Spectrum Magnitude ($g^2$/Hz)');
    title(sprintf(['CPS Magnitude of Output Signal between State 1 and State
', num2str(state(i+1)),'\n','(Column 1 vs Column ', num2str(column(i+1)),
'), 8 Averages']));
end

% Repeat Task 2.D.1 ------------------------------------------------------
set(0, 'DefaultAxesFontSize', 15);
set(0, 'DefaultTextFontSize', 15);

figure('Renderer', 'painters', 'Position', [10 10 1200 900]);

% Loop over the 4 different states:
for i = 1:4
    % Calculate the cross-power spectral density of the signal.
    % Hanning window averaging in 8 different blocks and zero overlap is
applied here.
    [psd, f] = cpsd(testingDataFiltered(:,column(1)),
testingDataFiltered(:,column(i+1)), hann(n/8), 0, [], samplingFrequency);
    semilogy(f, abs(psd), 'Color', color(i), 'LineWidth', 2); % Plot the
cross-power spectrum magnitude vs frequency in the log scale.
    hold on;
end

grid on;
grid minor;
box on;
xlim([0 ceil(NyquistFrequency/20)*20]);
ylim([1e-10 1e0]);
xticks(0:20:ceil(NyquistFrequency/20)*20);
yticks(10.^(-10:5:0));
xlabel('Frequency (Hz)');
ylabel('Cross-Power Spectrum Magnitude ($g^2$/Hz)');

legend('CPS Magnitude between Output State 1 and State 5 (Column 1 vs Column
201)', 'CPS Magnitude between Output State 1 and State 10 (Column 1 vs
Column 451)', 'CPS Magnitude between Output State 1 and State 12 (Column 1
vs Column 551)', 'CPS Magnitude between Output State 1 and State 14 (Column
1 vs Column 651)', 'Location', 'northeast');
title(sprintf('Cross-Power Spectrum Magnitude of Output Signal, 8
Averages'));
```

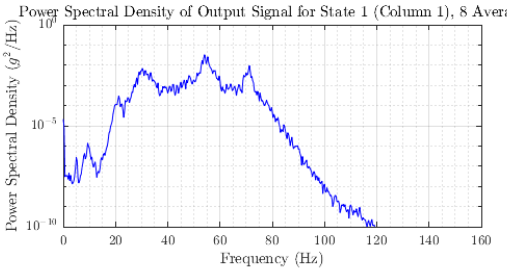

Power Spectral Density of Output Signal for State 1 (Column 1), 1 Average
Power Spectral Density (g^2/Hz)
Frequency (Hz)

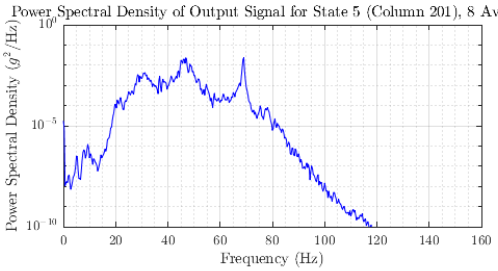

Power Spectral Density of Output Signal for State 5 (Column 201), 1 Average
Power Spectral Density (g^2/Hz)
Frequency (Hz)

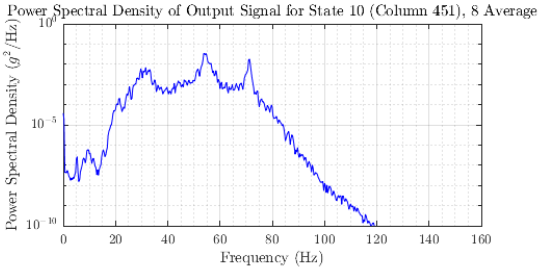

Power Spectral Density of Output Signal for State 10 (Column 451), 1 Average
Power Spectral Density (g^2/Hz)
Frequency (Hz)

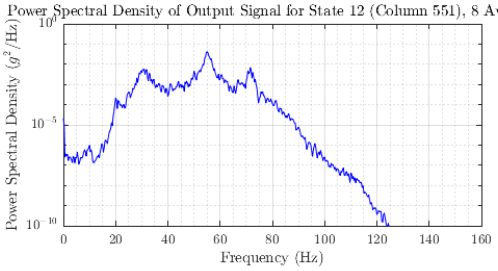

Power Spectral Density of Output Signal for State 12 (Column 551), 1 Average
Power Spectral Density (g^2/Hz)
Frequency (Hz)

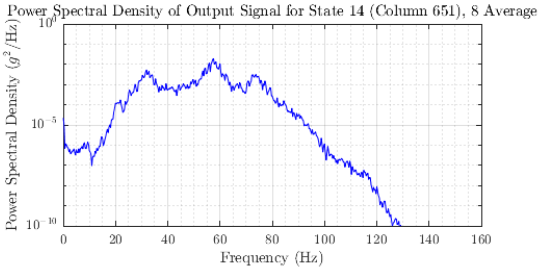

Power Spectral Density of Output Signal for State 14 (Column 651), 1 Average
Power Spectral Density (g^2/Hz)
Frequency (Hz)

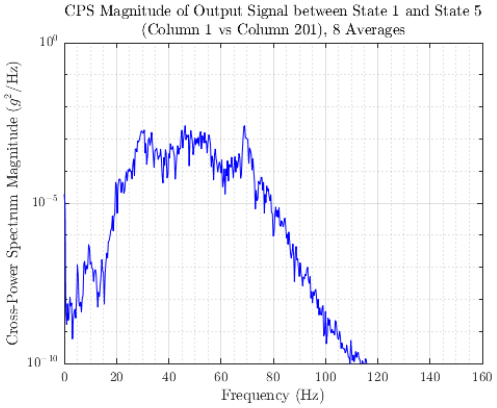

Power Spectral Density of Output Signal for State 1 (Column 1), 8 Average
Power Spectral Density (g^2/Hz)
Frequency (Hz)

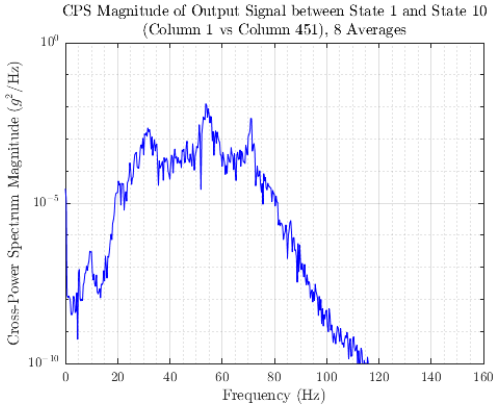

Power Spectral Density of Output Signal for State 5 (Column 201), 8 Average
Power Spectral Density (g^2/Hz)
Frequency (Hz)

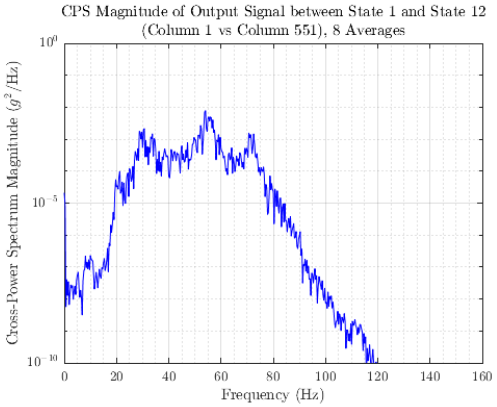

Power Spectral Density of Output Signal for State 10 (Column 451), 8 Average
Power Spectral Density (g^2/Hz)
Frequency (Hz)

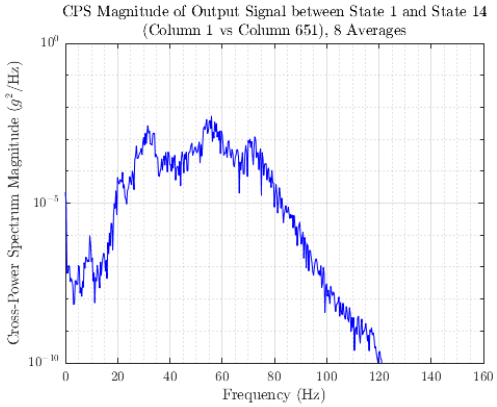

Power Spectral Density of Output Signal for State 12 (Column 551), 8 Average
Power Spectral Density (g^2/Hz)
Frequency (Hz)

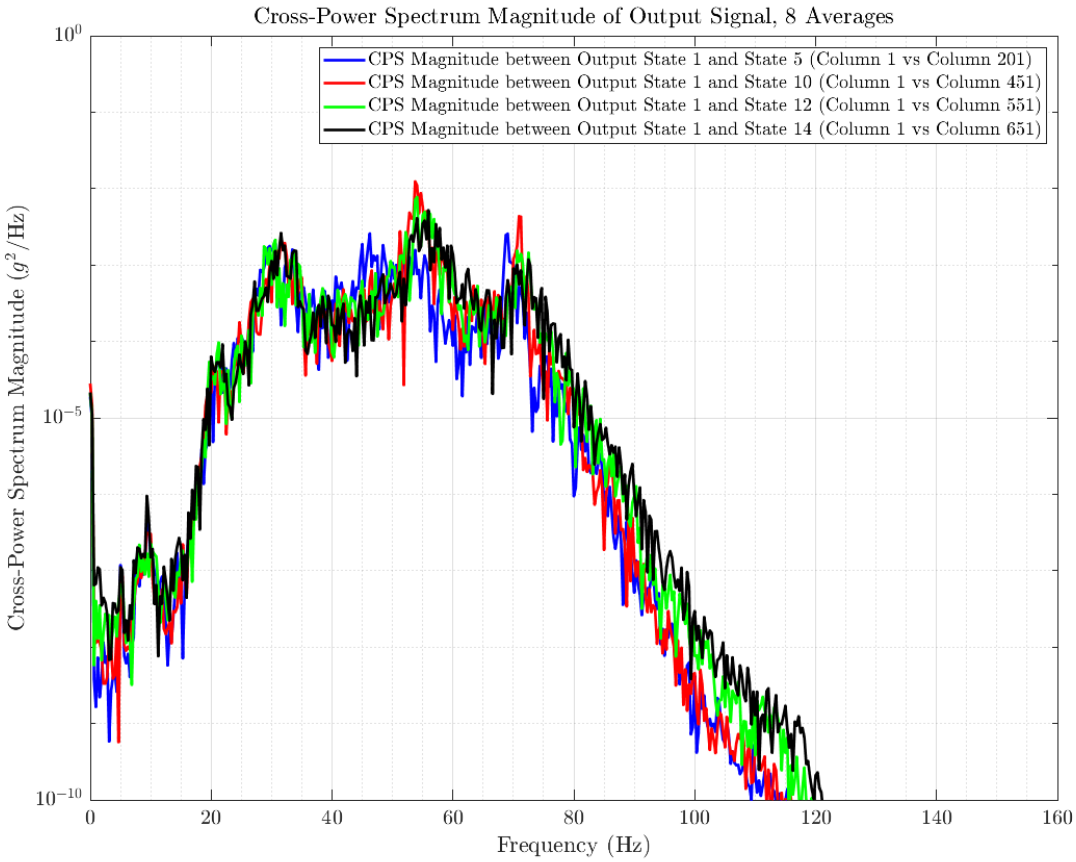

Power Spectral Density of Output Signal for State 14 (Column 651), 8 Average
Power Spectral Density (g^2/Hz)
Frequency (Hz)


CPS Magnitude of Output Signal between State 1 and State 5
(Column 1 vs Column 201), 8 Averages
Cross-Power Spectrum Magnitude (g^2/Hz)
Frequency (Hz)


CPS Magnitude of Output Signal between State 1 and State 10
(Column 1 vs Column 451), 8 Averages
Cross-Power Spectrum Magnitude (g^2/Hz)
Frequency (Hz)


CPS Magnitude of Output Signal between State 1 and State 12
(Column 1 vs Column 551), 8 Averages
Cross-Power Spectrum Magnitude (g^2/Hz)
Frequency (Hz)


CPS Magnitude of Output Signal between State 1 and State 14
(Column 1 vs Column 651), 8 Averages
Cross-Power Spectrum Magnitude (g^2/Hz)
Frequency (Hz)



Task 2.F

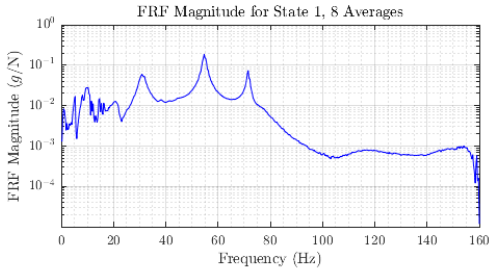
```
figure('Renderer', 'painters', 'Position', [10 10 1200 900]);

% Loop over the 5 different states:
for i = 1:5
    % Calculate the power spectral density of the signal.
    % Hanning window averaging in 8 different blocks and zero overlap is
applied here.
    [psd, f] = pwelch(inputData(:,column(i)), hann(n/8), 0, [],
samplingFrequency);
    semilogy(f, psd, 'Color', color(i), 'LineWidth', 2); % Plot the power
spectral density vs frequency in the log scale.
    hold on;
end

grid on;
grid minor;
box on;
xlim([0 ceil(NyquistFrequency/20)*20]);
ylim([1e-5 1e2]);
xticks(0:20:ceil(NyquistFrequency/20)*20);
```

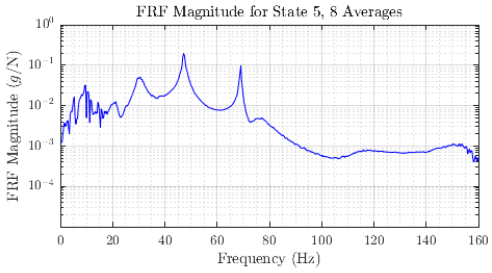
```
yticks(10.^(-5:1:2));
xlabel('Frequency (Hz)');
ylabel('Power Spectral Density (N$^2$/Hz)');
legend('Power Spectral Density of Input State 1 (Column 1)', 'Power Spectral
Density of Input State 5 (Column 201)', 'Power Spectral Density of Input
State 10 (Column 451)', 'Power Spectral Density of Input State 12 (Column
551)', 'Power Spectral Density of Input State 14 (Column 651)', 'Location',
'southeast');
title(sprintf('Power Spectral Density of Input Signal, 8 Averages'));
```

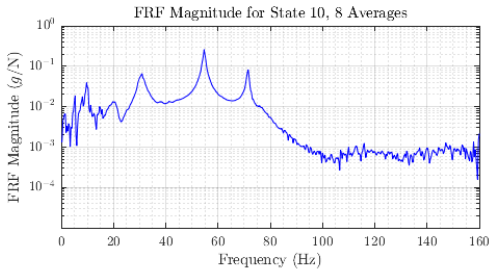
Task 2.G

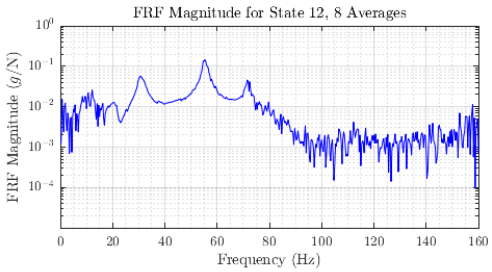
```
set(0, 'DefaultAxesFontSize', 10);
set(0, 'DefaultTextFontSize', 10);

figure('Renderer', 'painters', 'Position', [10 10 1200 900]);

% Loop over the 5 different states:
for i = 1:5
    % Calculate the frequency response function of each state.
    % Hanning window averaging in 8 different blocks and zero overlap is
applied here.
    [frf, f] = tfestimate(inputData(:,column(i)), testingData(:,column(i)),
```

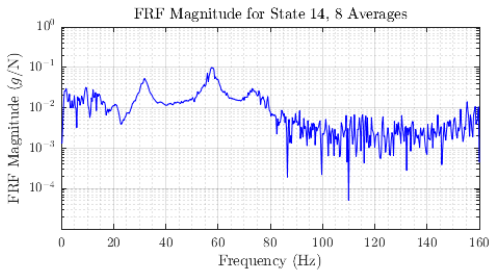
```
hann(n/8), 0, [], samplingFrequency);
    subplot(3,2,i);
    semilogy(f, abs(frf), 'Color', 'b', 'LineWidth', 0.1); % Plot the
frequency response function magnitude vs frequency in the log scale.
    grid on;
    grid minor;
    box on;
    xlim([0 ceil(NyquistFrequency/20)*20]);
    ylim([1e-5 1e0]);
    xticks(0:20:ceil(NyquistFrequency/20)*20);
    yticks(10.^(-5:1:0));
    xlabel('Frequency (Hz)');
    ylabel('FRF Magnitude ($g$/N)');
    title(sprintf(['FRF Magnitude for State ', num2str(state(i)), ', 8
Averages']));
end
```







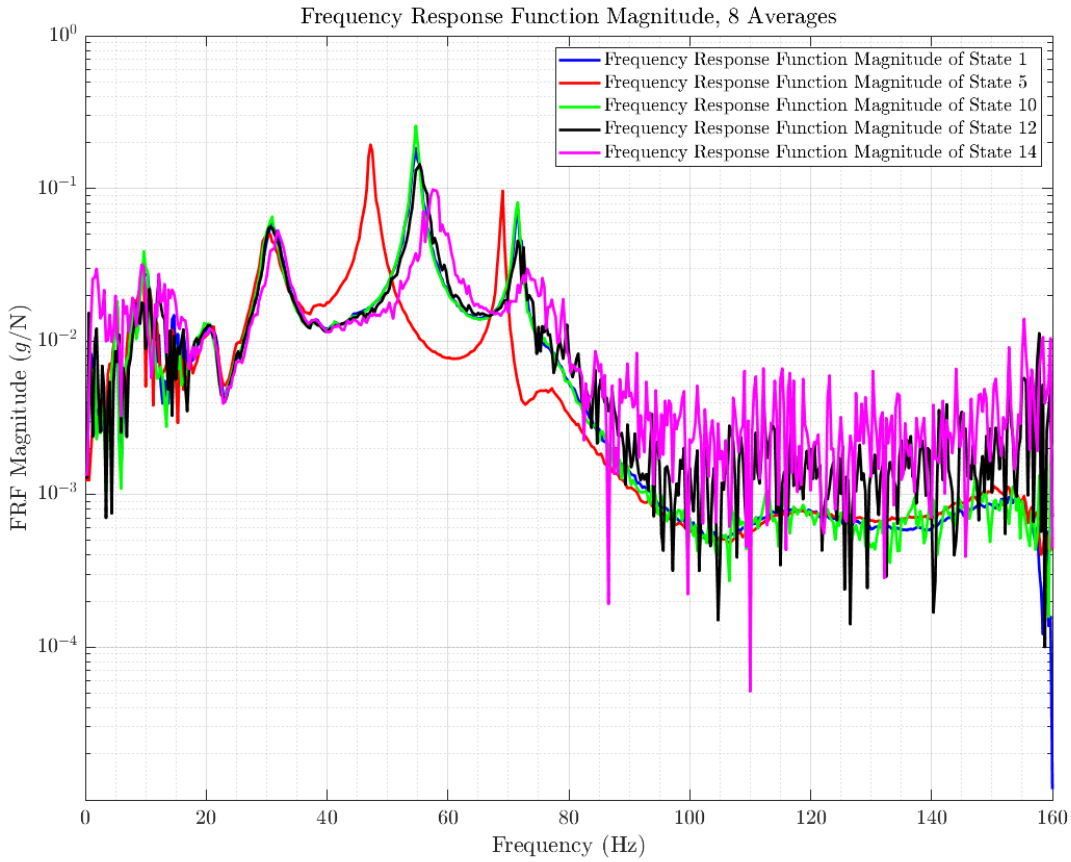




Task 2.G.1

```
set(0, 'DefaultAxesFontSize', 15);
set(0, 'DefaultTextFontSize', 15);

figure('Renderer', 'painters', 'Position', [10 10 1200 900]);
```

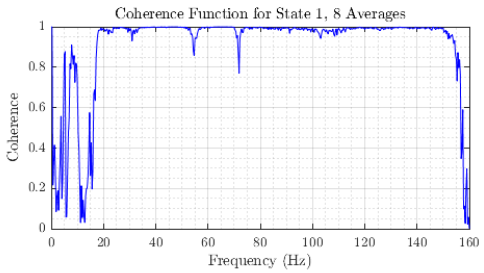
```
% Loop over the 5 different states:
for i = 1:5
    % Calculate the frequency response function of each state.
    % Hanning window averaging in 8 different blocks and zero overlap is
applied here.
    [frf, f] = tfestimate(inputData(:,column(i)), testingData(:,column(i)),
hann(n/8), 0, [], samplingFrequency);
    semilogy(f, abs(frf), 'Color', color(i), 'LineWidth', 2); % Plot the
frequency response function magnitude vs frequency in the log scale.
    hold on;
end

grid on;
grid minor;
box on;
xlim([0 ceil(NyquistFrequency/20)*20]);
ylim([1e-5 1e0]);
xticks(0:20:ceil(NyquistFrequency/20)*20);
yticks(10.^(-5:1:0));
xlabel('Frequency (Hz)');
ylabel('FRF Magnitude ($g$/N)');
legend('Frequency Response Function Magnitude of State 1 ', 'Frequency
Response Function Magnitude of State 5', 'Frequency Response Function
Magnitude of State 10', 'Frequency Response Function Magnitude of State
12', 'Frequency Response Function Magnitude of State 14', 'Location',
'northeast');
title(sprintf('Frequency Response Function Magnitude, 8 Averages'));
```

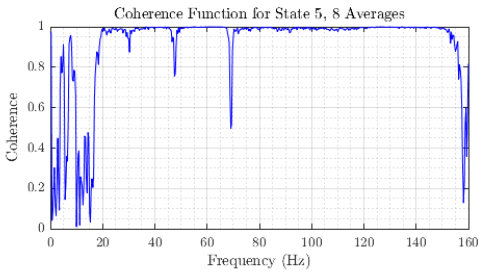


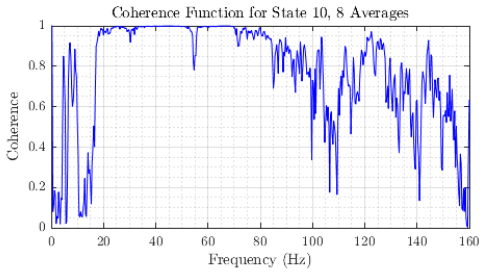
Task 2.H

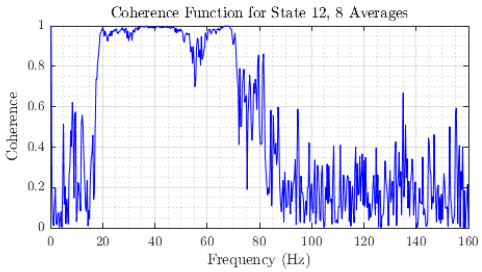
```
set(0, 'DefaultAxesFontSize', 10);
set(0, 'DefaultTextFontSize', 10);

figure('Renderer', 'painters', 'Position', [10 10 1200 900]);

% Loop over the 5 different states:
for i = 1:5
    % Calculate the coherence function of each state.
    % Hanning window averaging in 8 different blocks and zero overlap is
applied here.
    [coh, f] = mscohere(inputData(:,column(i)), testingData(:,column(i)),
hann(n/8), 0, [], samplingFrequency);
    subplot(3,2,i);
    plot(f, coh, 'Color', 'b', 'LineWidth', 0.1); % Plot the coherence
function vs frequency.
    grid on;
    grid minor;
    box on;
    xlim([0 ceil(NyquistFrequency/20)*20]);
    ylim([0 1]);
```

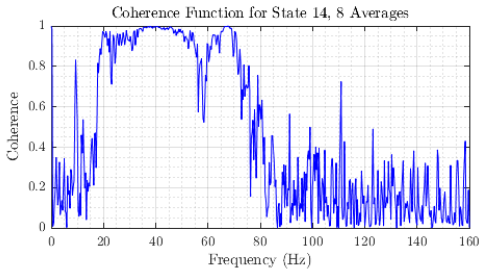
```
    xticks(0:20:ceil(NyquistFrequency/20)*20);
    yticks(0:0.2:1);
    xlabel('Frequency (Hz)');
    ylabel('Coherence');
    title(sprintf(['Coherence Function for State ', num2str(state(i)), ', 8
Averages']));
end
```







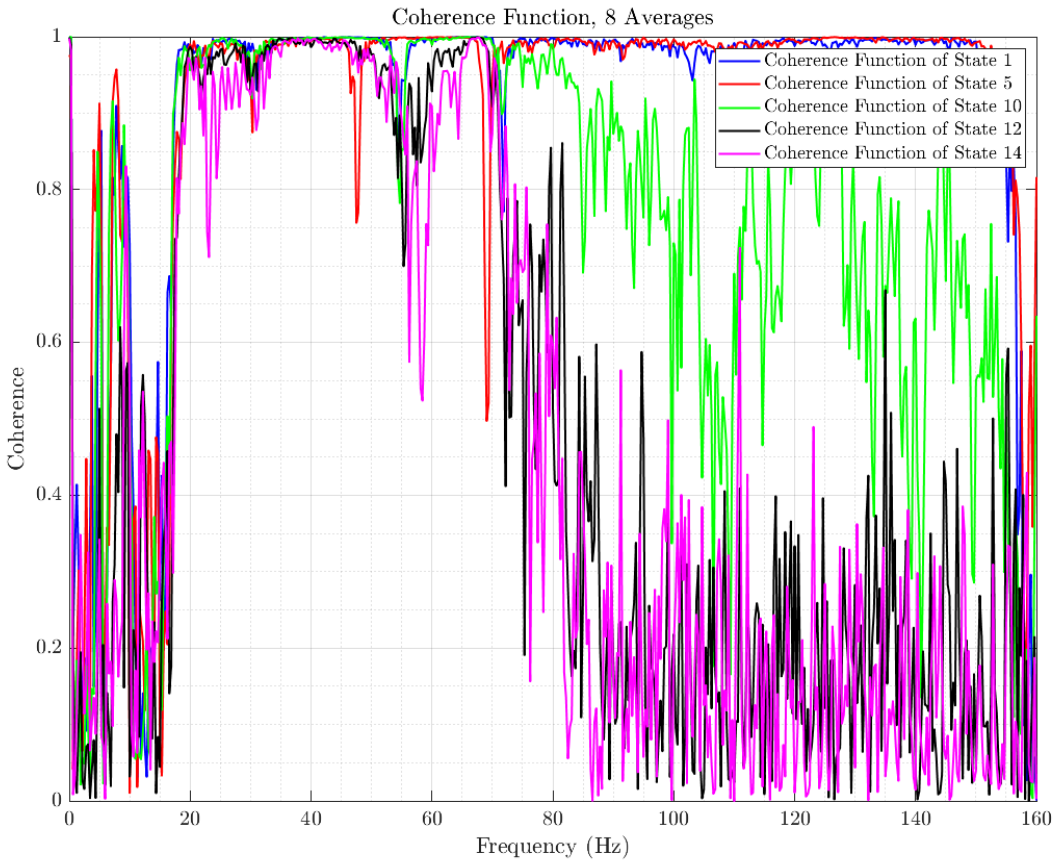




Task 2.H.1

```
set(0, 'DefaultAxesFontSize', 15);
set(0, 'DefaultTextFontSize', 15);

figure('Renderer', 'painters', 'Position', [10 10 1200 900]);

% Loop over the 5 different states:
for i = 1:5
    % Calculate the coherence function of each state.
    % Hanning window averaging in 8 different blocks and zero overlap is
applied here.
    [coh, f] = mscohere(inputData(:,column(i)), testingData(:,column(i)),
hann(n/8), 0, [], samplingFrequency);
    plot(f, coh, 'Color', color(i), 'LineWidth', 1.5); % Plot the coherence
```

```
function vs frequency.
    hold on;
end

grid on;
grid minor;
box on;
xlim([0 ceil(NyquistFrequency/20)*20]);
ylim([0 1]);
xticks(0:20:ceil(NyquistFrequency/20)*20);
yticks(0:0.2:1);
xlabel('Frequency (Hz)');
ylabel('Coherence');
legend('Coherence Function of State 1 ', 'Coherence Function of State
5', 'Coherence Function of State 10', 'Coherence Function of State 12',
'Coherence Function of State 14', 'Location', 'northeast');
title(sprintf('Coherence Function, 8 Averages'));
```



Published with MATLAB® R2023b