Table of Contents


```
% Ruipu Ji
% SE 265
% Homework #2

clc; clear; close all;

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

Load the data.

```
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 A

Initializaiton. --

```
state = [1, 5, 10, 12, 14]; % Create a vector for the state number.

PeakMagnitude_vector = zeros(50, length(state));
Mean_vector = zeros(50, length(state));
MeanSquare_vector = zeros(50, length(state));
RootMeanSquare_vector = zeros(50, length(state));
StandardDeviation_vector = zeros(50, length(state));
Skewness_vector = zeros(50, length(state));
Kurtosis_vector = zeros(50, length(state));

% Use 2 embedded loops to calculate the statistics. -----------------------
% The outside loop is for the 5 different states.
% The inside loop is for the multiple measurements for each state.
```

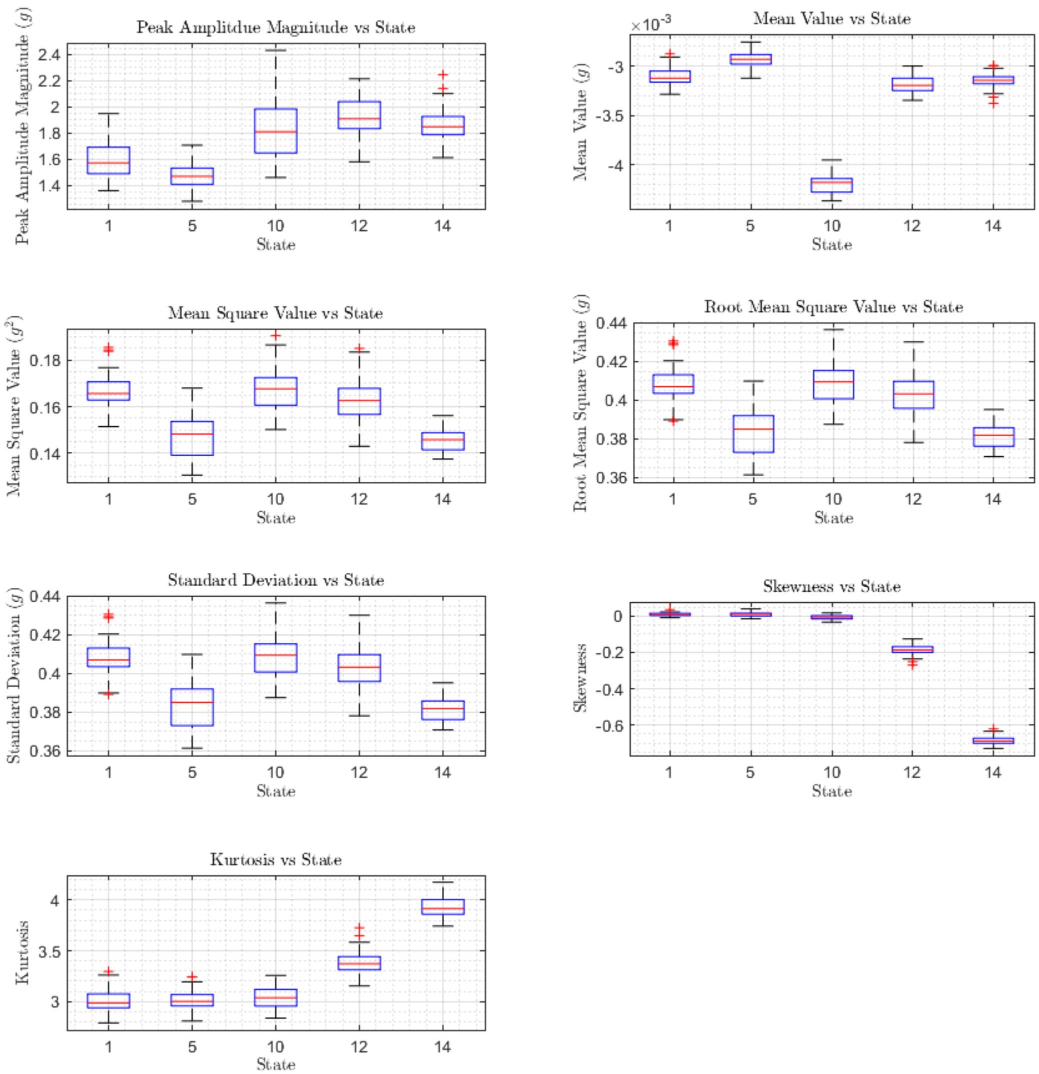
```
for NumOfState = 1:length(state)
    for NumOfMeasurement = 1:50
        PeakMagnitude_vector(NumOfMeasurement, NumOfState) =
max(abs(testingData(:, (state(NumOfState)-1)*50+NumOfMeasurement))); %
Calculate peak amplitdue magnitdue.
        Mean_vector(NumOfMeasurement, NumOfState) = mean(testingData(:,
(state(NumOfState)-1)*50+NumOfMeasurement)); % Calculate mean value.
        MeanSquare_vector(NumOfMeasurement, NumOfState) =
meansqr(testingData(:, (state(NumOfState)-1)*50+NumOfMeasurement)); %
Calculate mean square value.
        RootMeanSquare_vector(NumOfMeasurement, NumOfState) =
rms(testingData(:, (state(NumOfState)-1)*50+NumOfMeasurement)); % Calculate
root mean square value.
        StandardDeviation_vector(NumOfMeasurement, NumOfState) =
std(testingData(:, (state(NumOfState)-1)*50+NumOfMeasurement)); % Calculate
standard deviation.
        Skewness_vector(NumOfMeasurement, NumOfState) =
skewness(testingData(:, (state(NumOfState)-1)*50+NumOfMeasurement)); %
Calculate skewness.
        Kurtosis_vector(NumOfMeasurement, NumOfState) =
kurtosis(testingData(:, (state(NumOfState)-1)*50+NumOfMeasurement)); %
Calculate kurtosis.
    end
end
```

Task A.1

Generate the box plot for each statistic quantity.

```
set(0, 'DefaultAxesFontSize', 10);
set(0, 'DefaultTextFontSize', 10);
figure('Renderer', 'painters', 'Position', [10 10 1000 1000]);

% Box plot for the peak amplitude magnitude.
subplot(4,2,1);
boxplot(PeakMagnitude_vector, state);
grid on;
grid minor;
box on;
xlabel('State');
ylabel('Peak Amplitude Magnitude ($g$)');
title('Peak Amplitdue Magnitude vs State');

% Box plot for the mean value.
subplot(4,2,2);
boxplot(Mean_vector, state);
grid on;
grid minor;
box on;
xlabel('State');
ylabel('Mean Value ($g$)');
title('Mean Value vs State');
```

```
% Box plot for the mean square value.
subplot(4,2,3);
boxplot(MeanSquare_vector, state);
grid on;
grid minor;
box on;
xlabel('State');
ylabel('Mean Square Value ($g^2$)');
title('Mean Square Value vs State');

% Box plot for the root mean square value.
subplot(4,2,4);
boxplot(RootMeanSquare_vector, state);
grid on;
grid minor;
box on;
xlabel('State');
ylabel('Root Mean Square Value ($g$)');
title('Root Mean Square Value vs State');

% Box plot for the standard deviation.
subplot(4,2,5);
boxplot(StandardDeviation_vector, state);
grid on;
grid minor;
box on;
xlabel('State');
ylabel('Standard Deviation ($g$)');
title('Standard Deviation vs State');

% Box plot for the skewness.
subplot(4,2,6);
boxplot(Skewness_vector, state);
grid on;
grid minor;
box on;
xlabel('State');
ylabel('Skewness');
title('Skewness vs State');

% Box plot for the kurtosis.
subplot(4,2,7);
boxplot(Kurtosis_vector, state);
grid on;
grid minor;
box on;
xlabel('State');
ylabel('Kurtosis');
title('Kurtosis vs State');
```



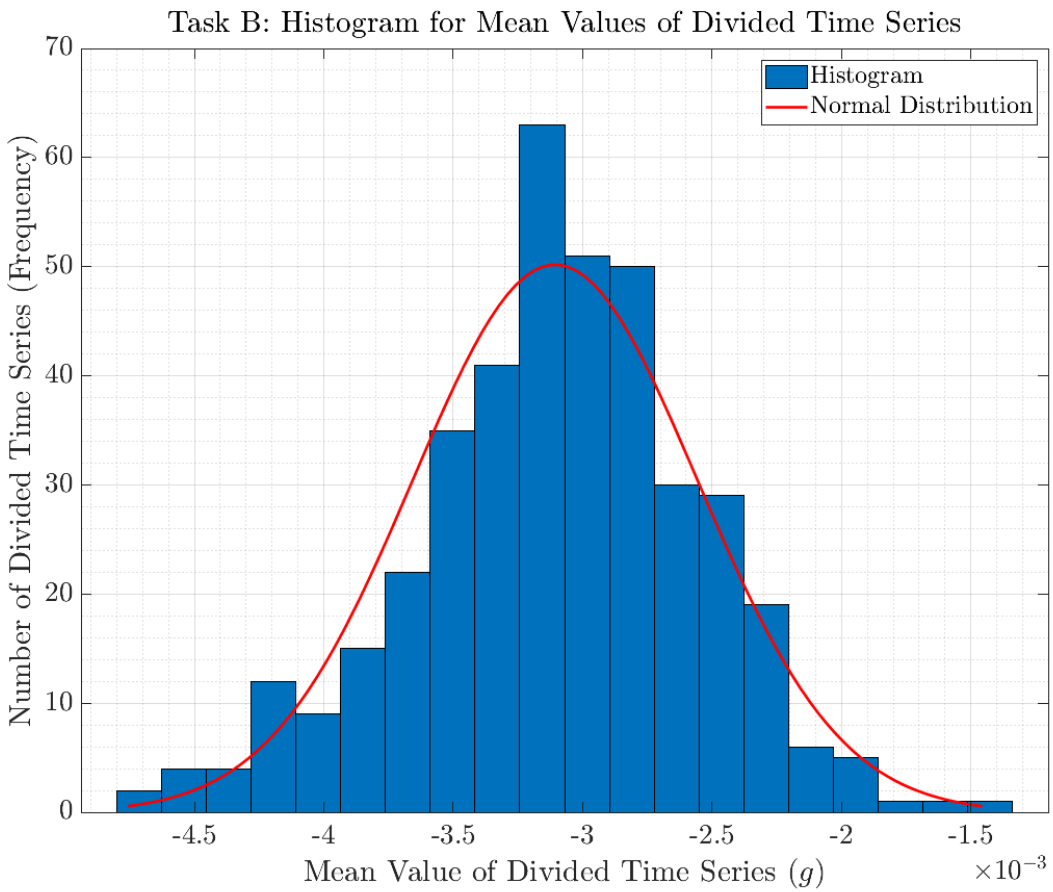
Task A.2

The peak amplitude magnitudes of the damaged cases are higher than the magnitudes of the undamged cases. However, the peak amplitude magnitude does not increase monotonically as the damage level increases. There is no obvious indication of damage for the mean value, mean square value, root mean square value and standard deviation because they are sensitive to the variability of the testing data (by comparing state 1 and state 5). Clear indication of damage is observed for skewness and kurtosis. As the damage level increases, the skewness decreases and the kurtosis increases for the testing data.

Task B

Break down each measurement into 8 1024-point time series. --------------

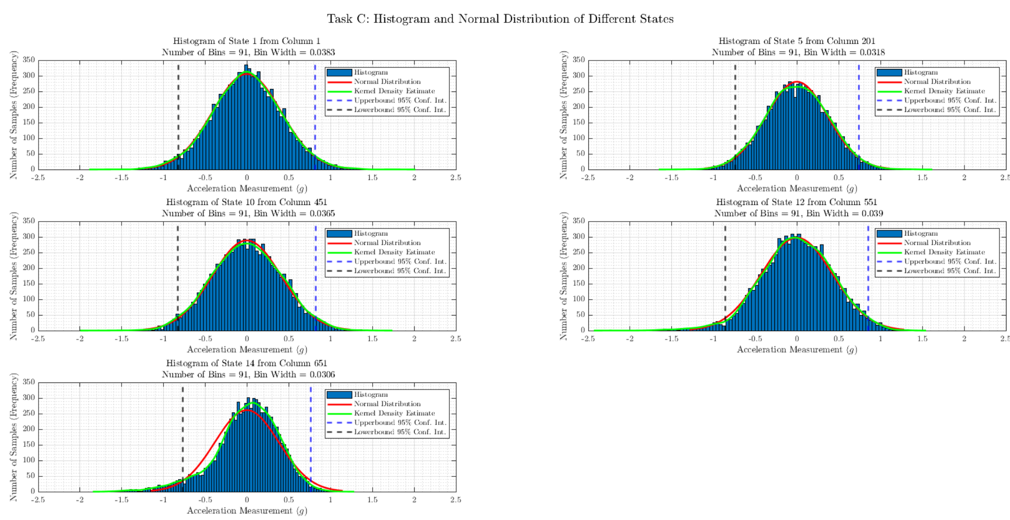
```matlab
State1_Measurement = zeros(n/8, 50*8);
for i = 1:50
    for j = 1:8
        State1_Measurement(:, (i-1)*8+j) = testingData((j-1)*n/8+1:j*n/8, i);
    end
end

% Calculate the average value of each column. -----------------------------
State1_Mean = zeros(1, size(State1_Measurement,2));
for i = 1:size(State1_Measurement,2)
    State1_Mean(i) = mean(State1_Measurement(:,i));
end

% Plot the histogram. -----------------------------------------------------
set(0, 'DefaultAxesFontSize', 20);
set(0, 'DefaultTextFontSize', 20);
figure('Renderer', 'painters', 'Position', [10 10 1200 900]);
histogram = histfit(State1_Mean);
grid on;
grid minor;
box on;
xlabel('Mean Value of Divided Time Series ($g$)');
ylabel('Number of Divided Time Series (Frequency)');
set(histogram(1), 'DisplayName', 'Histogram');
set(histogram(2), 'DisplayName', 'Normal Distribution');
legend('show', 'Location', 'northeast');
title('Task B: Histogram for Mean Values of Divided Time Series');
```



Task C

```
set(0, 'DefaultAxesFontSize', 10);
set(0, 'DefaultTextFontSize', 10);
figure('Renderer', 'painters', 'Position', [10 10 2000 900]);
sgtitle('Task C: Histogram and Normal Distribution of Different States');

for NumOfState = 1:length(state)
    subplot(3,2,NumOfState);
    hold on;
    % Task C.1 --------------------------------------------------------
    h = histfit(testingData(:, (state(NumOfState)-1)*50+1)); % Plot the
histogram with the normal distribution.
    NumOfBins = length(h(1,1).XData); % Calculate the number of bins in the
histogram.
    BinWidth = h(1,1).XData(2) - h(1,1).XData(1); % Calculate the bin width
in the histogram.
    set(h(1), 'DisplayName', 'Histogram');
    set(h(2), 'DisplayName', 'Normal Distribution');

    % Task C.2 --------------------------------------------------------
    [f, xi] = ksdensity(testingData(:, (state(NumOfState)-1)*50+1)); %
```

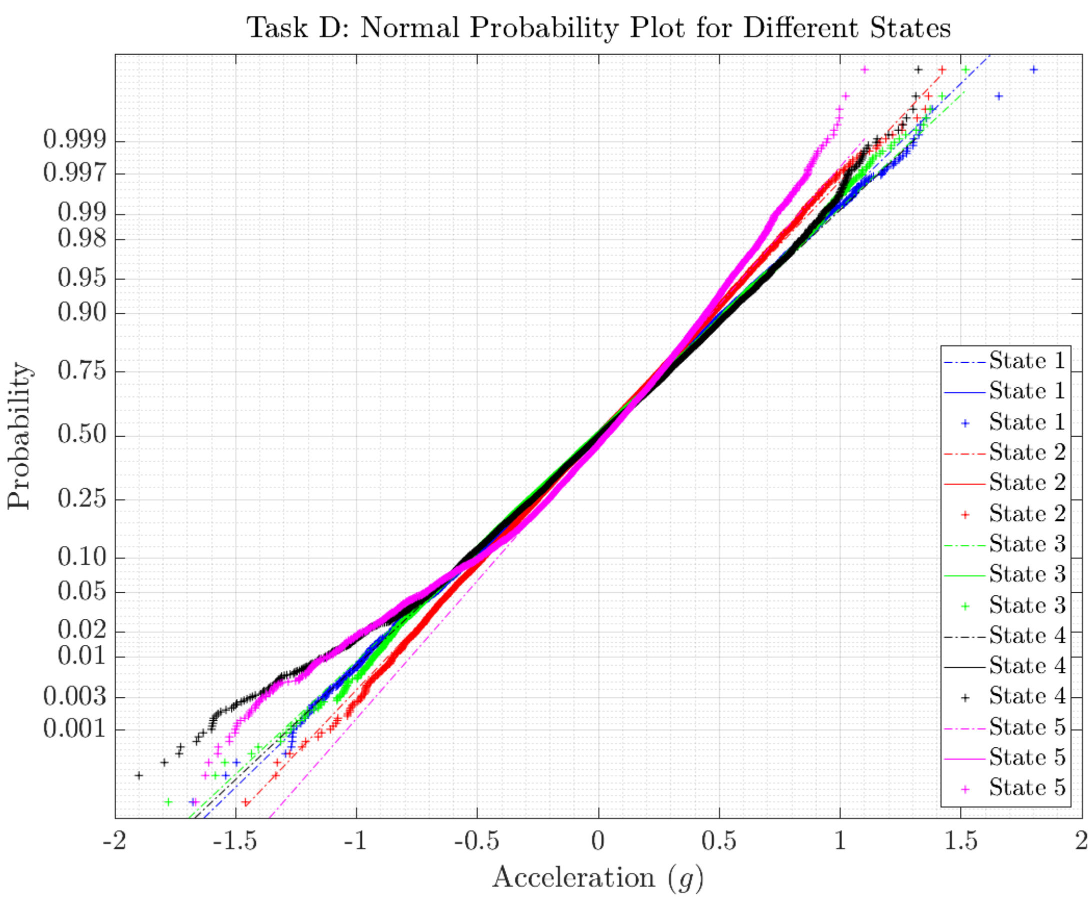
```
Calculate the kernel density estimate of density function.
    k = plot(xi,f*(n*BinWidth), 'Color', 'g', 'LineWidth', 2); % Plot with
the scale.
    set(k, 'DisplayName', 'Kernel Density Estimate');

    % Task C.3 ---------------------------------------------------------
    x1 = xline(Mean_vector(1, NumOfState) + 2*StandardDeviation_vector(1,
NumOfState), 'b--', 'LineWidth', 2); % Plot the upperbound of the 95%
confidence intervals.
    x2 = xline(Mean_vector(1, NumOfState) - 2*StandardDeviation_vector(1,
NumOfState), 'k--', 'LineWidth', 2); % Plot the lowerbound of the 95%
confidence intervals.
    set(x1, 'DisplayName', 'Upperbound 95\% Conf. Int.');
    set(x2, 'DisplayName', 'Lowerbound 95\% Conf. Int.');

    grid on;
    grid minor;
    box on;
    xlim([-2.5 2.5]);
    ylim([0 350]);
    xticks(-2.5:0.5:2.5);
    yticks(0:50:350);
    xlabel('Acceleration Measurement ($g$)');
    ylabel('Number of Samples (Frequency)');
    legend('show', 'Location', 'northeast');
    title(sprintf(['Histogram of State ', num2str(state(NumOfState)), ' from
Column ', num2str((state(NumOfState)-1)*50+1), '\n Number of Bins = ',
num2str(NumOfBins), ', Bin Width = ', num2str(BinWidth)]));
end
```



Task D

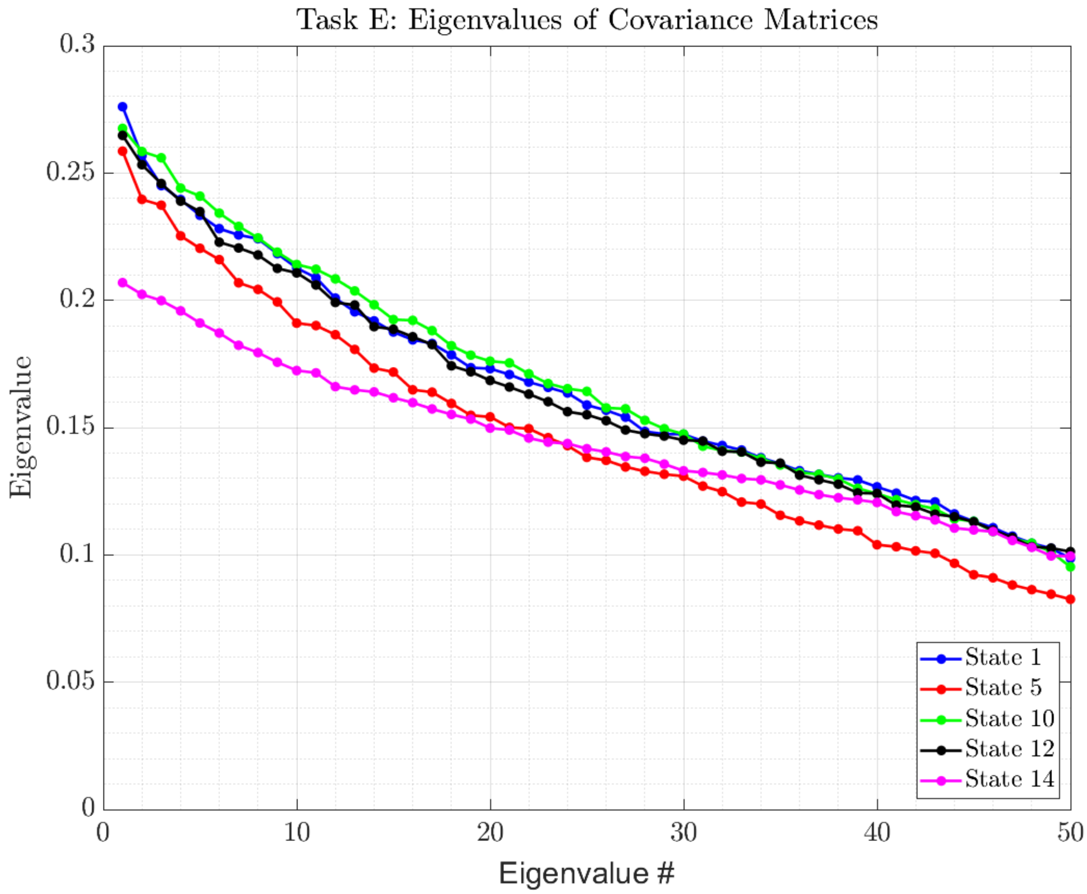
```
color = ['b', 'r', 'g', 'k', 'm']; % Create a vector for colors in the plot.

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

```
set(0, 'DefaultTextFontSize', 20);
figure('Renderer', 'painters', 'Position', [10 10 1200 900]);
hold on;

% Loop over the 5 different states, create the normal probability plot.
for NumOfState = 1:length(state)
    norm = normplot(testingData(:, (state(NumOfState)-1)*50+1));
    set(norm, 'Color', color(NumOfState), 'DisplayName', sprintf(['State ',
num2str(NumOfState)]));
end

grid on;
grid minor;
box on;
xlim([-2 2]);
xticks(-2:0.5:2);
xlabel('Acceleration ($g$)');
ylabel('Probability');
legend('show', 'Location', 'southeast');
title('Task D: Normal Probability Plot for Different States');
```



Task E

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

% Loop over the 5 different states:
for NumOfState = 1:length(state)
    d = testingData(:,(state(NumOfState)-1)*50+1:state(NumOfState)*50); %
Save the 50 measurements of each state in matrix [d].
    dm = d - Mean_vector(:,NumOfState)'; % Subtract the mean from each
measurement and save the result in matrix [dm].
    covariance = cov(dm); % Calculate the covariance.
    [Eigenvectors, EigenvaluesMatrix] = eig(covariance); % Calculate
eigenvalues and the corresponding eigenvectors of the covariance matrix.
    Eigenvalues = diag(EigenvaluesMatrix);
    [EigenvaluesDescending, Index] = sort(Eigenvalues, 'descend'); % Sort
the eigenvalues in the descending order and obtain the order index.
    EigenvectorsDescending = Eigenvectors(:, Index); % Sort the eigenvectors
based on the sequence of eigenvalues.
    plot(EigenvaluesDescending, '-o', 'LineWidth', 2, 'MarkerSize',
5, 'Color', color(NumOfState), 'MarkerEdgeColor', color(NumOfState),
'MarkerFaceColor', color(NumOfState));

    tn = d * EigenvectorsDescending(:,1); % Create a new time series [tn].

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

end

grid on;
grid minor;
box on;
xlim([0 50]);
ylim([0 0.3]);
xticks(0:10:50);
yticks(0:0.05:0.3);
xlabel('Eigenvalue #');
ylabel('Eigenvalue');
legend('State 1', 'State 5', 'State 10', 'State 12', 'State 14', 'Location',
'southeast');
title('Task E: Eigenvalues of Covariance Matrices');
```



Published with MATLAB® R2023b