Table of Contents


```
% Ruipu Ji
% SE 265
% Homework #6

clc; clear; close all;

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

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

Task 1: Load data and create arrays.

```
load('4-Story Structure Data/data3SS2009.mat'); % Load the data file.
Input = squeeze(dataset(:,1,:)); % Input = Data from channel 1 (input time
history from the load cell).
Response = dataset(:,2:5,:); % Response = Data from channel 2-5
(acceleration response for each level).
% squeeze() is to remove the dimension with length of 1.
NumOfPoints = size(Input,1); % NumOfPoints = Number of data points in each
set of signal.
NumOfTests = size(Input,2); % NumOfTests = Number of tests for each floor.
NumOfLevels = size(Response,2); % NumOfLevels = Number of levels (number of
input-sensor pairs) in the structure.
```

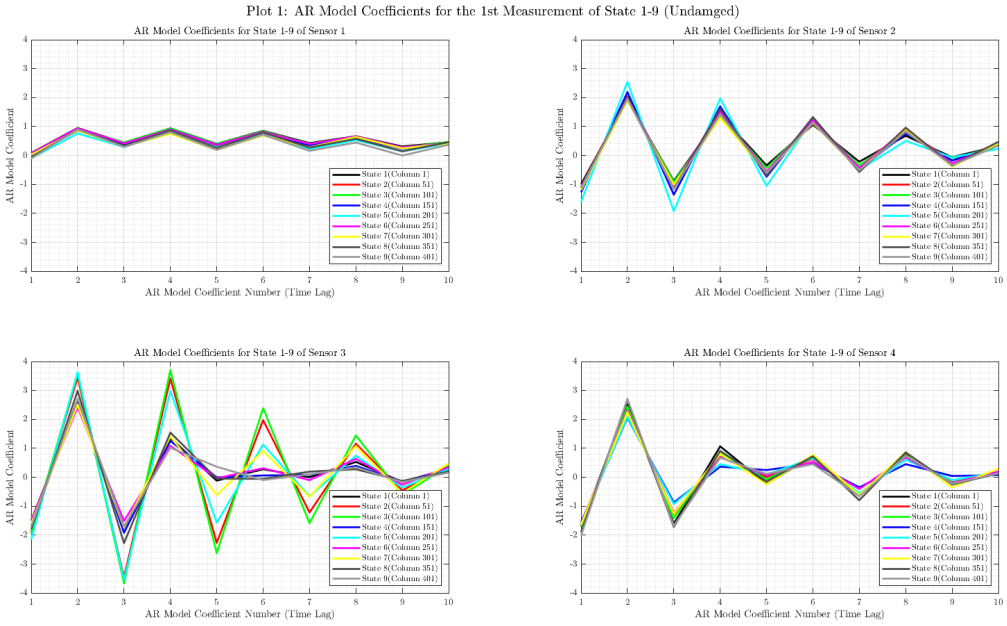
Task 2: Fit an auto-regressive (AR) model to all the response time-histories.

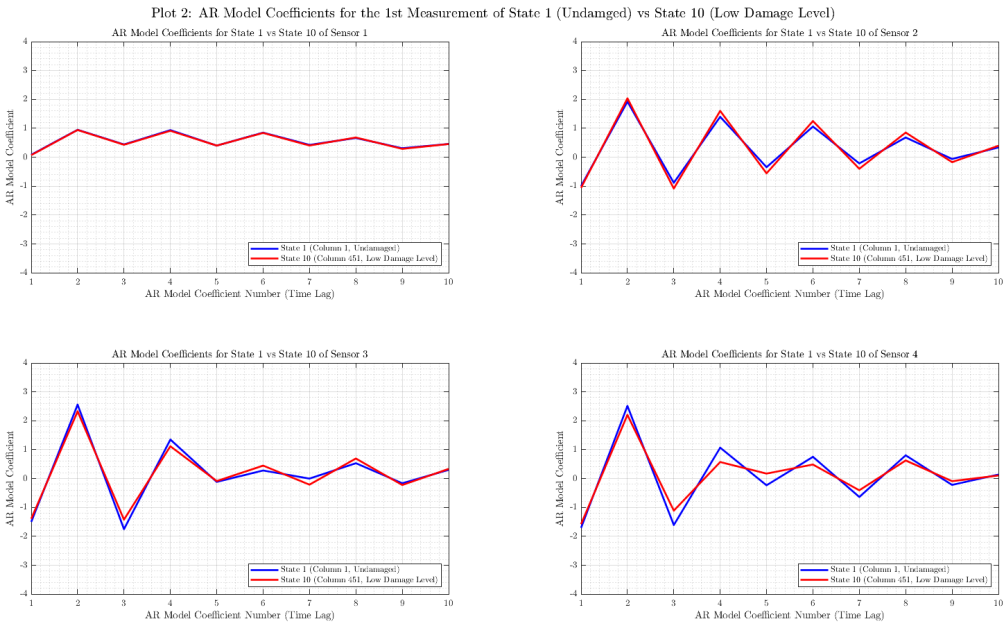
```
Order = 10; % Define the order of the AR model.

% Initialize a 3-D matrix for the coefficients of the AR model.
% Dimension-1 = 4, which represents the total number of levels in the
structure.
% Dimension-2 = 850, which represents the number of tests for each floor.
% Dimension-3 = 10, which represents the order of the AR model.
Coefficients = zeros(NumOfLevels, NumOfTests, Order);
```

```
% Calculate the coefficients for a 10-th order linear AR model.
for Level = 1:NumOfLevels % Loop over all the levels.
    % First create a temperory matrix to store the result including the 1 in
the first column.
    % lpc(x, p) finds the coefficients of a p-th order linear predictor and
returns to a 1-D row vector.
    % If x is a 2-D matrix, then the function will treat each column as a
separate channel.
    % The input x must be in double precision.
    Coefficients_temp = lpc(double(squeeze(Response(:,Level,:))), Order);

    % Remove the 1 in the first column and store the coefficient matrix in
the final output matrix.
    Coefficients_temp(:,1) = [];
    Coefficients(Level,:,:) = Coefficients_temp;
end

% Plot 1: AR model coefficients for the first measurement of State 1-9
(undamaged states).
% -------------------------------------------------------------------------
% Create the color map and the plot legend cell array 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];
LegendState = cell(9,1);

figure('Renderer', 'painters', 'Position', [10 10 1800 1000]);

for Level = 1:NumOfLevels
    subplot(2,2,Level);
    hold on;

    for NumOfState = 1:9
        plot(squeeze(Coefficients(Level, (NumOfState-1)*50+1, :)), 'Color',
color(NumOfState,:), 'LineWidth', 2); % Plot the AR model coefficients.
        LegendState{NumOfState} = sprintf(['State ', num2str(NumOfState),
'(Column ', num2str((NumOfState-1)*50+1), ')']);
    end

    grid on;
    grid minor;
    box on;
    xlim([1 Order]);
    ylim([-4 4]);
    xticks(1:1:Order);
    yticks(-4:1:4);
    xlabel('AR Model Coefficient Number (Time Lag)');
    ylabel('AR Model Coefficient');
    legend(LegendState, 'Location', 'southeast');
    title(sprintf(['AR Model Coefficients for State 1-9 of Sensor ',
num2str(Level)]));
end
```

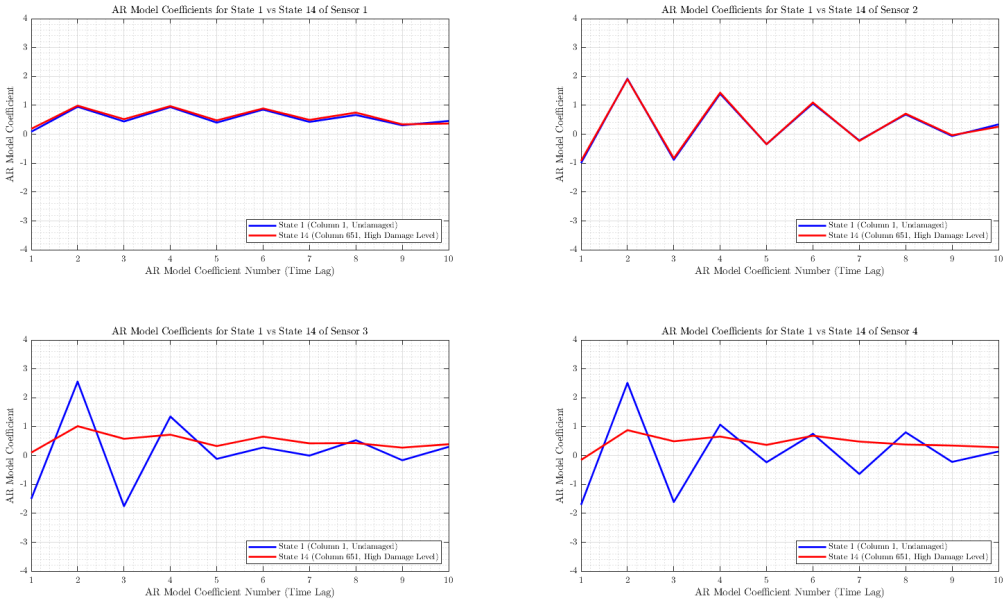
```
sgtitle('Plot 1: AR Model Coefficients for the 1st Measurement of State 1-9
(Undamged)');

% Plot 2: AR model coefficients for the first measurement of state 1
(undamged) vs state 10 (low damage level).
% -------------------------------------------------------------------------
figure('Renderer', 'painters', 'Position', [10 10 1800 1000]);

for Level = 1:NumOfLevels
    subplot(2,2,Level);
    hold on;
    plot(squeeze(Coefficients(Level, 1, :)), 'b', 'LineWidth', 2); % Plot
the AR model coefficients for the 1st measurement of state 1.
    plot(squeeze(Coefficients(Level, 451, :)), 'r', 'LineWidth', 2); % Plot
the AR model coefficients for the 1st measurement of state 10.
    grid on;
    grid minor;
    box on;
    xlim([1 Order]);
    ylim([-4 4]);
    xticks(1:1:Order);
    yticks(-4:1:4);
    xlabel('AR Model Coefficient Number (Time Lag)');
    ylabel('AR Model Coefficient');
    legend('State 1 (Column 1, Undamaged)', 'State 10 (Column 451, Low
Damage Level)', 'Location', 'southeast');
    title(sprintf(['AR Model Coefficients for State 1 vs State 10 of Sensor
', num2str(Level)]));
end

sgtitle('Plot 2: AR Model Coefficients for the 1st Measurement of State 1
(Undamged) vs State 10 (Low Damage Level)');

% Plot 3: AR model coefficients for the first measurement of state 1
(undamged) vs state 14 (high damage level).
% -------------------------------------------------------------------------
figure('Renderer', 'painters', 'Position', [10 10 1800 1000]);

for Level = 1:NumOfLevels
    subplot(2,2,Level);
    hold on;
    plot(squeeze(Coefficients(Level, 1, :)), 'b', 'LineWidth', 2); % Plot
the AR model coefficients for the 1st measurement of state 1.
    plot(squeeze(Coefficients(Level, 651, :)), 'r', 'LineWidth', 2); % Plot
the AR model coefficients for the 1st measurement of state 14.
    grid on;
    grid minor;
    box on;
    xlim([1 Order]);
    ylim([-4 4]);
    xticks(1:1:Order);
    yticks(-4:1:4);
    xlabel('AR Model Coefficient Number (Time Lag)');
```

```
    ylabel('AR Model Coefficient');
    legend('State 1 (Column 1, Undamaged)', 'State 14 (Column 651, High
Damage Level)', 'Location', 'southeast');
    title(sprintf(['AR Model Coefficients for State 1 vs State 14 of Sensor
', num2str(Level)]));
end

sgtitle('Plot 3: AR Model Coefficients for the 1st Measurement of State 1
(Undamged) vs State 14 (High Damage Level)');
```





Plot 3: AR Model Coefficients for the 1st Measurement of State 1 (Undamged) vs State 14 (High Damage Level)



Task 3 Develop training and testing AR coefficient data sets.

Initialization.

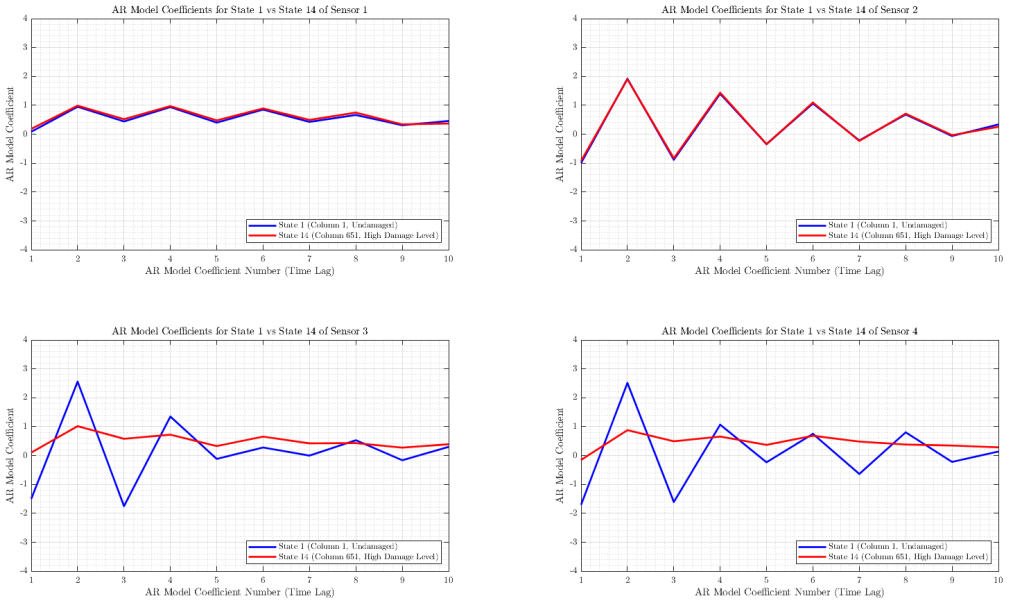
```
ARtrain = zeros(NumOfLevels,225,Order);
ARtest = zeros(NumOfLevels,625,Order);

for Level = 1:NumOfLevels % Loop over all the 4 levels.
    % Loop over the 9 undamaged states (State 1-9):
    % The first 25 measurements in each state are selected as the training
data
    % The second 25 measurements in each state are selected as the testing
data.
    for NumOfState = 1:9
        ARtrain(Level, (NumOfState-1)*25+1:NumOfState*25, :) =
squeeze(Coefficients(Level, (NumOfState-1)*50+1:(NumOfState-1)*50+25, :));
        ARtest(Level, (NumOfState-1)*25+1:NumOfState*25, :) =
squeeze(Coefficients(Level, (NumOfState-1)*50+26:NumOfState*50, :));
    end

    % Loop over the 8 damaged states (State 10-17):
    % All the 50 measurements in each state are selected as the testing data.
    for NumOfState = 10:17
        ARtest(Level, 225+(NumOfState-10)*50+1:225+(NumOfState-9)*50, :) =
squeeze(Coefficients(Level, (NumOfState-1)*50+1:NumOfState*50, :));
    end
end
```

Task 4: Calculate the Mahalanobis distances (MD) for all the testing data based on the mean and covariance of the training data.

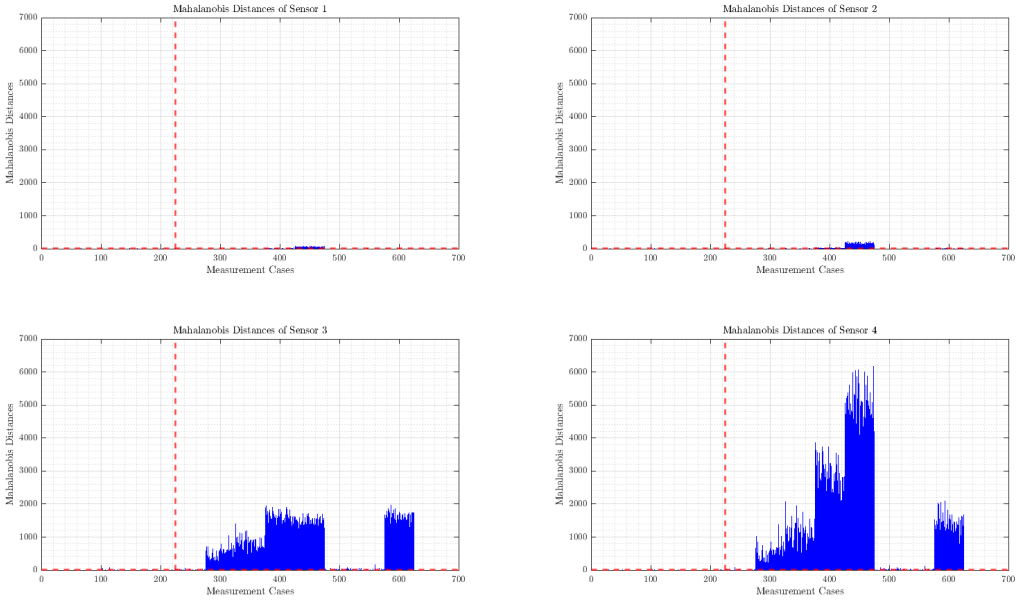
Initialization of a matrix for the MD values.

```
MD = zeros(size(ARtest,2), NumOfLevels);

% Loop over all the 4 levels, calculate the MD values for all the testing
data of each level.
for Level = 1:NumOfLevels
    MD(:,Level) = mahal(squeeze(ARtest(Level,:,:)),
squeeze(ARtrain(Level,:,:)));
end

% Plot 4: Plot the MD values for all the testing data of each level.
% ------------------------------------------------------------------------
figure('Renderer', 'painters', 'Position', [10 10 1800 1000]);

for Level = 1:NumOfLevels
    subplot(2,2,Level);
    hold on;
    bar(MD(:,Level), 1, 'blue');
    grid on;
    grid minor;
    box on;
    xlim([0 ceil(size(MD,1)/100)*100]);
    ylim([0 ceil(max(MD, [], 'all')/1000)*1000]);
    xticks(0:100:ceil(size(MD,1)/100)*100);
    yticks(0:1000:ceil(max(MD, [], 'all')/1000)*1000);
    xlabel('Measurement Cases');
    ylabel('Mahalanobis Distances');
    title(sprintf(['Mahalanobis Distances of Sensor ', num2str(Level)]));

    % Plot a vertical red line that separates the undamaged cases (1-225)
and damaged cases (226-625).
    xline(225.5, 'r--', 'LineWidth', 2);

    % Plot a horizontal red line corresponding to the lowest value of all
the damaged cases (226-625).
    yline(min(MD(226:625,Level)), 'r--', 'LineWidth', 2);

end

sgtitle('Plot 4: Mahalanobis Distances for Testing Data');
```

Plot 3: AR Model Coefficients for the 1st Measurement of State 1 (Undamged) vs State 14 (High Damage Level)



Plot 4: Mahalanobis Distances for Testing Data

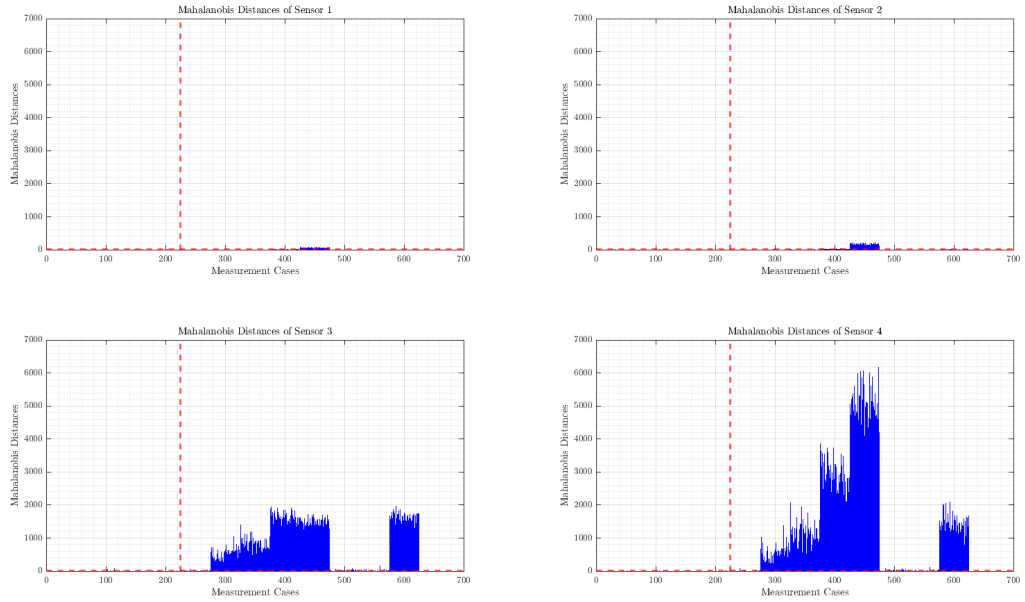


Task 5: Calculate the Receiver-Operating Characteristic (ROC) Curves for the testing data.

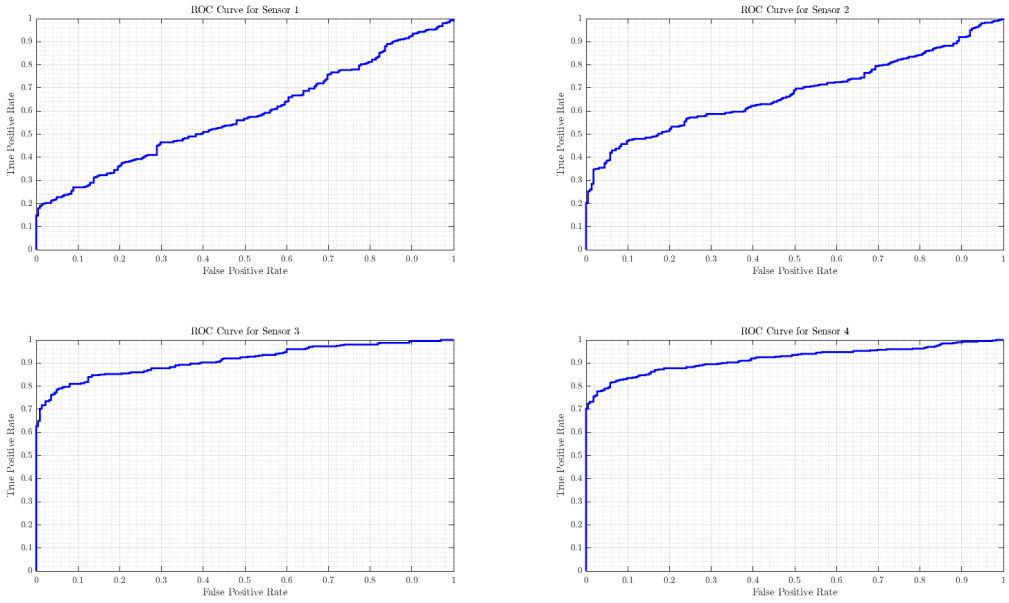
Create a label vector to indicate the undamaged cases and damaged cases. The components corresponding to undamaged cases (index 1-225) have the value of 0. The components corresponding to damaged cases (index 226-625) have the value of 1.

```
Label = zeros(size(MD,1),1);
Label(226:end) = 1;

% Plot 5: Plot the ROC curve for all the testing data of each level.
% ------------------------------------------------------------------------
figure('Renderer', 'painters', 'Position', [10 10 1800 1000]);

for Level = 1:NumOfLevels
    subplot(2,2,Level);
    hold on;

    % Calculate the false positive rate X and true positive rate Y.
    % [X,Y] = perfcurve(labels, scores, positive-class)
    [X, Y] = perfcurve(Label, MD(:,Level), 1);

    plot(X, Y, 'b', 'LineWidth', 2);
    grid on;
    grid minor;
    box on;
    xlim([0 1]);
    ylim([0 1]);
    xticks(0:0.1:1);
    yticks(0:0.1:1);
    xlabel('False Positive Rate');
    ylabel('True Positive Rate');
    title(sprintf(['ROC Curve for Sensor ', num2str(Level)]));
end

sgtitle('Plot 5: Receiver-Operating Characteristic (ROC) Curves for the
Testing Data');
```

Plot 4: Mahalanobis Distances for Testing Data



Plot 5: Receiver-Operating Characteristic (ROC) Curves for the Testing Data



Published with MATLAB® R2023b