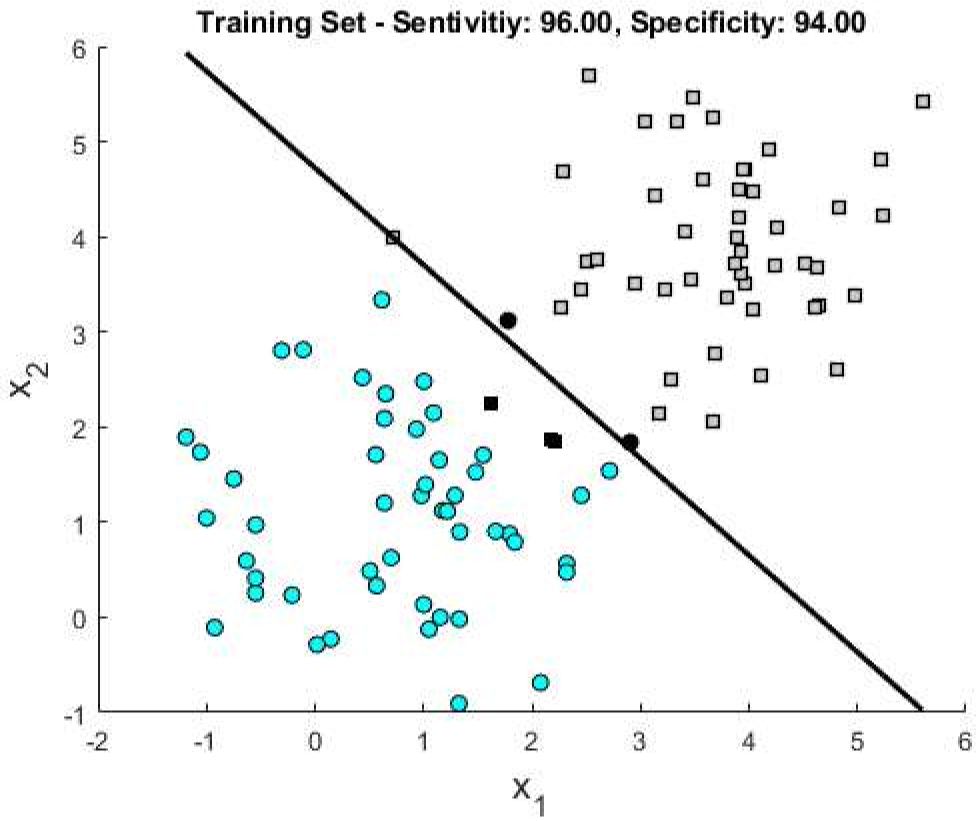
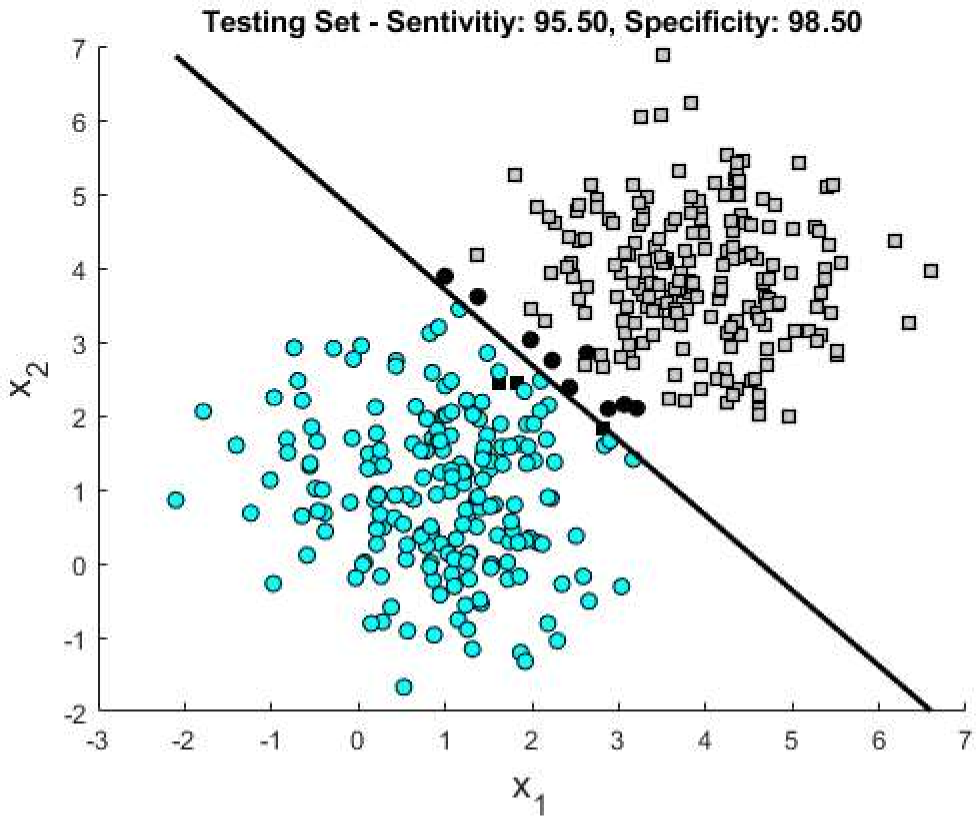
Faranak Dayyani - student number: 1002373674

Contents

Question 1:

```
%Loading the signal data
s = load('prob16_1.mat');
% training set variables:
Xt = s.Xt;
Xt(:,3) = ones;
dt = s.dt;
N1 = 100;
% test set variables:
X = s.X;
X(:,3) = ones;
d = s.d;
N2 = 400;
% LLSQ:
% training the model
part11 = ((Xt.')*Xt)^(-1);
part22 = (Xt.')*(dt.');
w2 = part11 * part22;

figure;
[sen_t, spec_t] = linear_eval(Xt,dt,w2);
title(sprintf('Training Set - Sentivitiy: %0.2f, Specificity: %0.2f', sen_t, spec_t));

figure;
[sen, spec] = linear_eval(X,d,w2);
title(sprintf('Testing Set - Sentivitiy: %0.2f, Specificity: %0.2f', sen, spec));
```





Question 2**:

```
%Loading the signal data
s = load('prob16_4.mat');
% training set variables:
Xt = s.Xt;
Xt(:,5) = ones;
dt = s.dt;
N1 = 100;
```

```
% test set variables:
X = s.X;
X(:,5) = ones;
d = s.d;
N2 = 400;

part1 = ((Xt.')*Xt)^(-1);
part2 = (Xt.')*(dt);
wt = part1 * part2;

sprintf(' ------------ Training Set Values --------------')
[sen_t, spec_t] = linear_eval_Q2(Xt,dt,wt);
sprintf('Sentivitiy: %0.2f, Specificity: %0.2f', sen_t, spec_t)

sprintf(' ------------ Testing Set Values --------------')
[sen, spec] = linear_eval_Q2(X,d,wt);
sprintf('Sentivitiy: %0.2f, Specificity: %0.2f', sen, spec)

% from the results, it can be seen that the sensitivity and specificity are
% lower for the testing test (82.50 & 79.00) compared to the training set
% (86.00 & 82.00) which is expected.

close all;
```

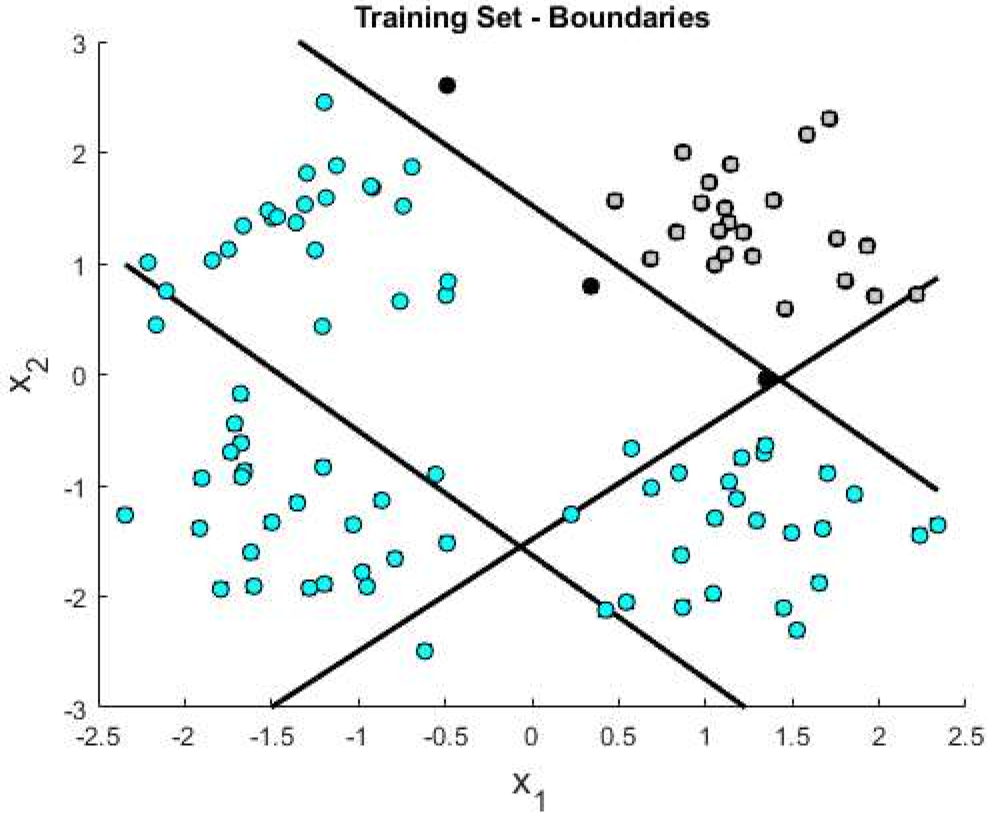
```
ans =

    ' ------------ Training Set Values --------------'


ans =

    'True Positive: 43'


ans =

    'True Negative: 41'


ans =

    'False Positive: 9'


ans =

    'False Negative: 7'


ans =

    'Sentivitiy: 86.00, Specificity: 82.00'


ans =

    ' ------------ Testing Set Values --------------'


ans =

    'True Positive: 165'
```

```
ans =

    'True Negative: 158'


ans =

    'False Positive: 42'


ans =

    'False Negative: 35'


ans =

    'Sentivitiy: 82.50, Specificity: 79.00'
```

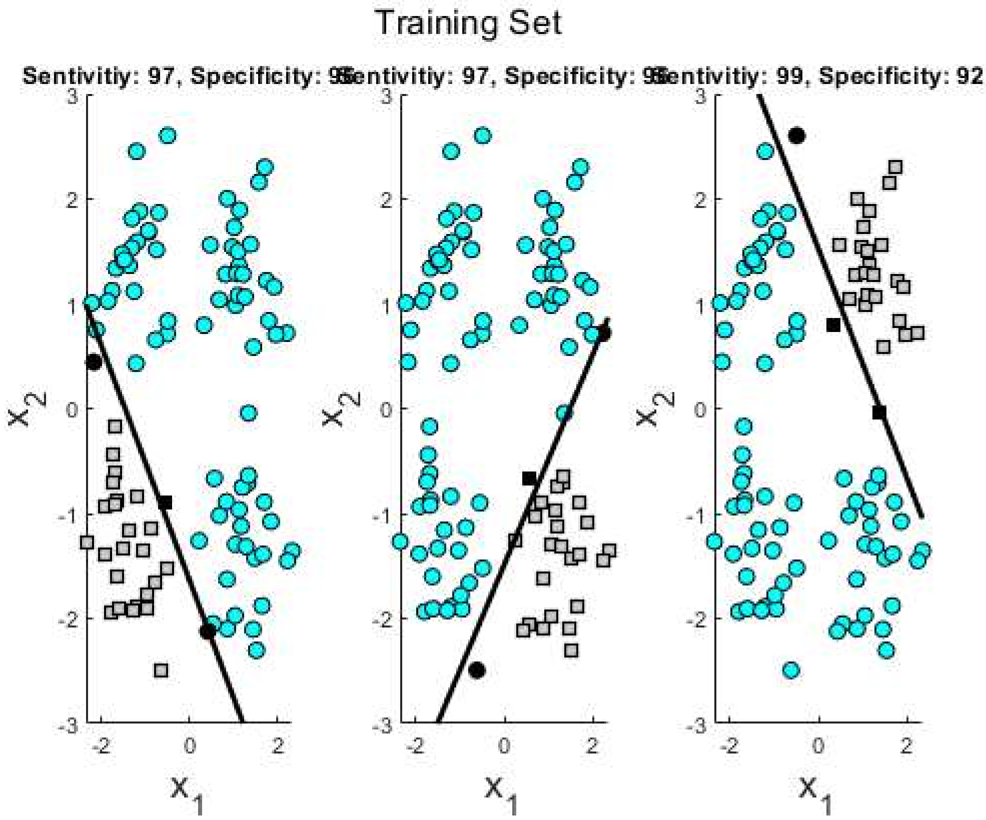
Question 3**:

```
%Loading the signal data
s = load('prob16_6.mat');

% training set variables:
Xt = s.Xt;
Xt(:,3) = ones;
Dt = s.Dt;
N1 = 100;

% creating Linear discriminators - Training set:
% plotting boundaries for each class
figure;
for i = 1:3
    dt = Dt(:,i);
    part1 = ((Xt.')*Xt)^(-1);
    part2 = (Xt.')*(dt);
    wt = part1 * part2;
    [sen_t, spec_t] = linear_eval(Xt,dt,wt);
    title('Training Set - Boundaries')

end

figure;
sgtitle('Training Set');
% subplotting each class
for i = 1:3
    dt = Dt(:,i);
    part1 = ((Xt.')*Xt)^(-1);
    part2 = (Xt.')*(dt);
    wt = part1 * part2;

    subplot(1,3,i);
    [sen_t, spec_t] = linear_eval(Xt,dt,wt);
    title(sprintf('Sentivitiy: %.0f, Specificity: %0.f', sen_t, spec_t));
end

% test set variables:
X = s.X;
X(:,3) = ones;
D = s.D;
N2 = 600;
```

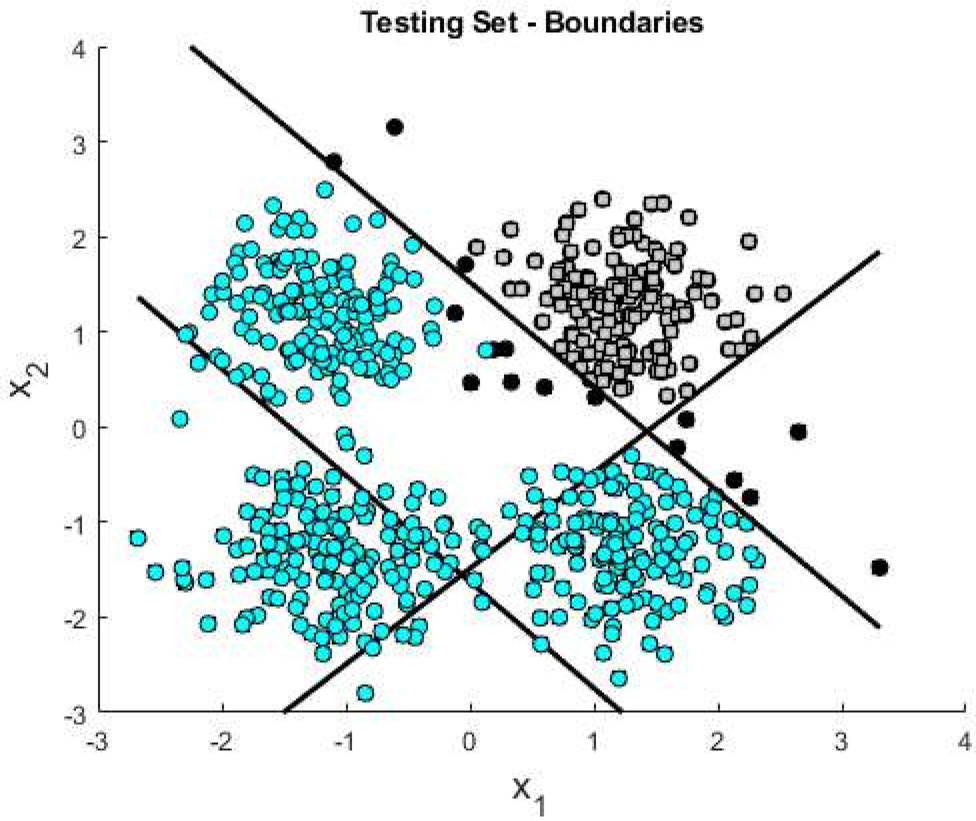
```
% creating Linear discriminators - Testing set:
% plotting boundaries for each class
figure;
for j = 1:3
    dt = Dt(:,j);
    d = D(:,j);
    part1 = ((Xt.')*Xt)^(-1);
    part2 = (Xt.')*(dt);
    wt = part1 * part2;
    [sen, spec] = linear_eval(X,d,wt);
    %[sen, spec] = linear_eval(X,d,wt);
    title('Testing Set - Boundaries')

end

figure;
sgtitle('Testing Set');
% subplotting each class
for j = 1:3
    dt = Dt(:,j);
    d = D(:,j);
    part1 = ((Xt.')*Xt)^(-1);
    part2 = (Xt.')*(dt);
    wt = part1 * part2;
    subplot(1,3,j);
    [sen, spec] = linear_eval(X,d,wt);
    title(sprintf('Sentivitiy: %0.2f, Specificity: %0.2f', sen, spec));
end
```

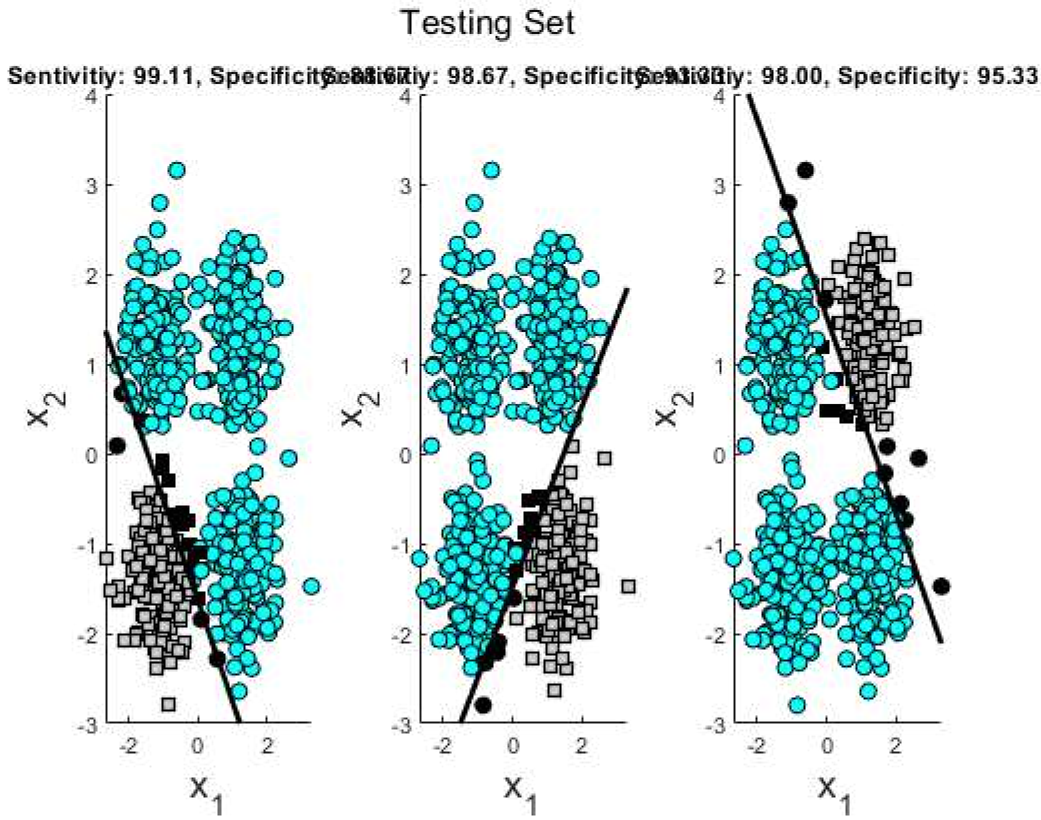


Training Set - Boundaries
x_2
x_1



Training Set
Sentivitiy: 97, Specificity: 95
Sentivitiy: 97, Specificity: 95
Sentivitiy: 99, Specificity: 92
x_2
x_1



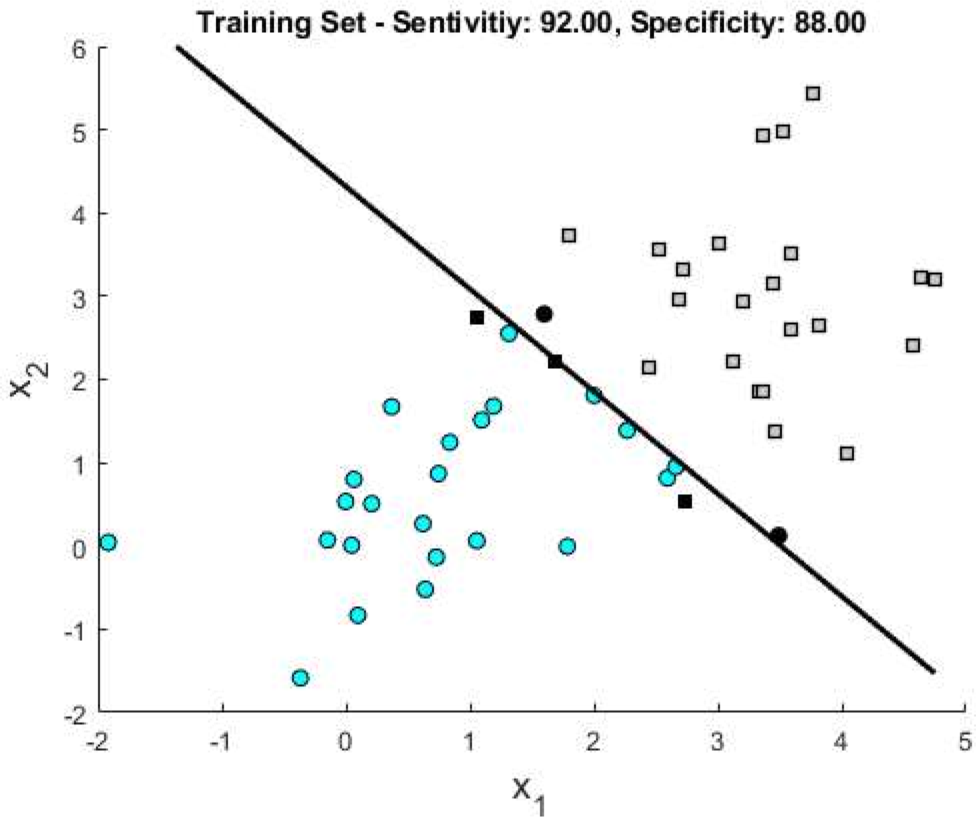


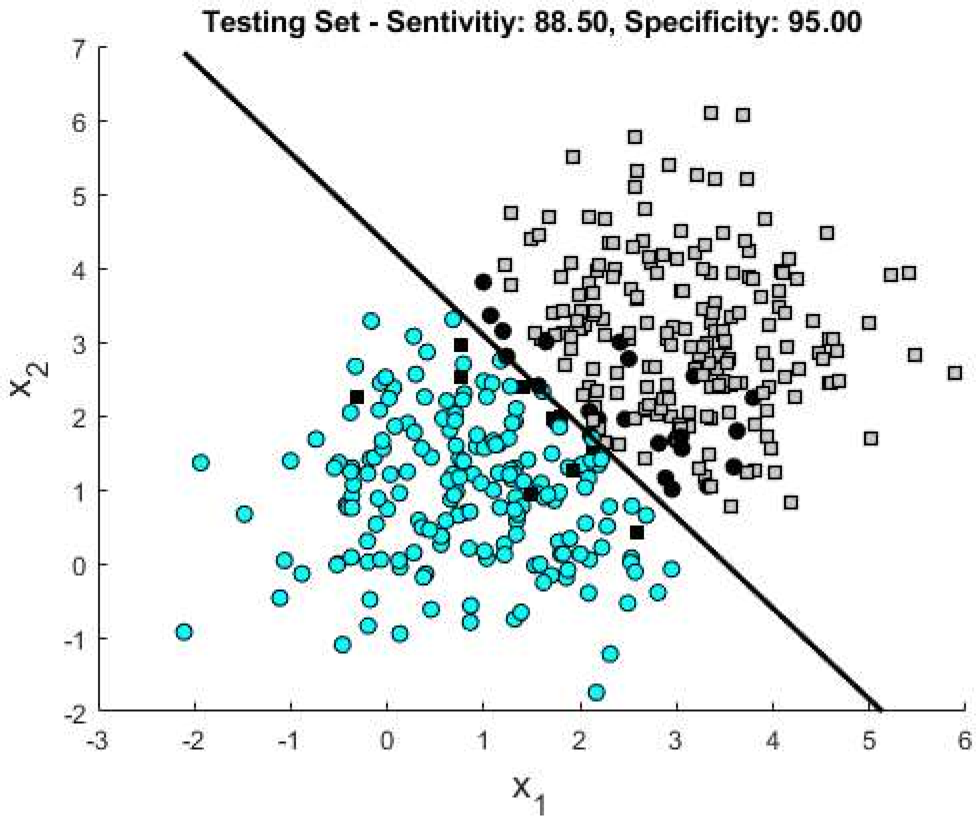
Question 4:

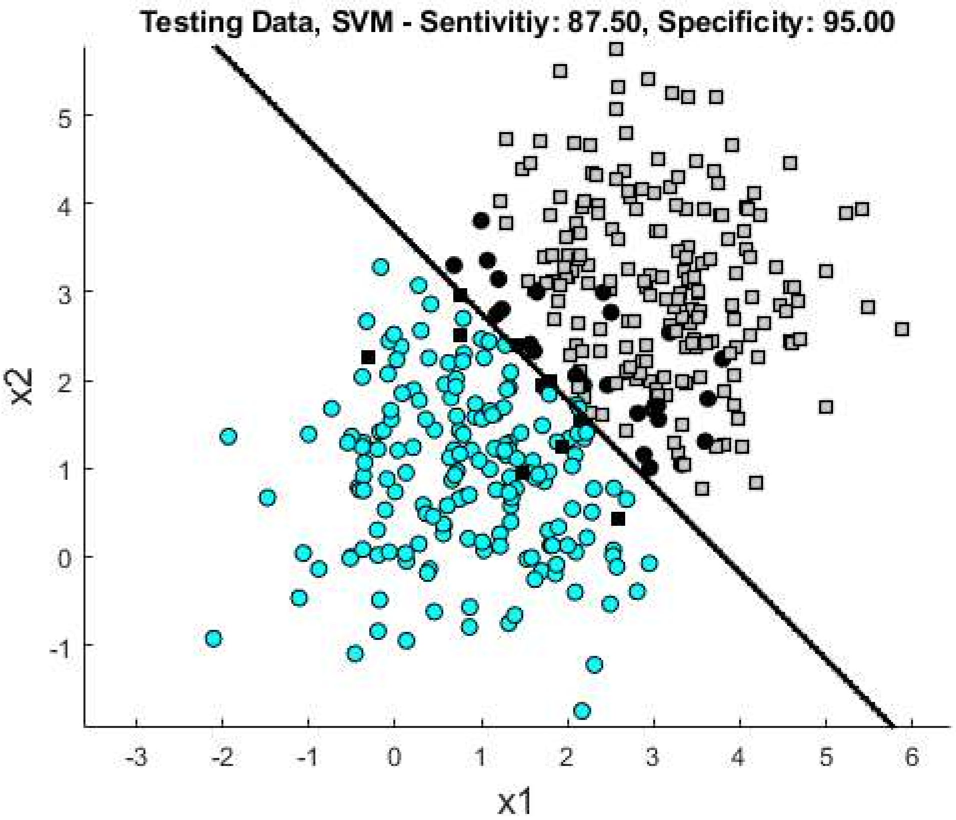
```
%Loading the signal data
s = load('prob16_7.mat');

% Training set:
N1 = 50;
Xt = s.Xt;
dt = s.dt; % class is indicated by -1 and 1
```

```
dtl = s.dtl; % class is indicated by 0 and 1

% Testing set:
N2 = 400;
X = s.X;
d = s.d; % class is indicated by -1 and 1
dl = s.dl; % class is indicated by 0 and 1

% Linear Least Squares:
%%%% Use linear_eval to plot the least squares training and test set results
% for training set
Xt(:,3) = ones;
part1_t = ((Xt.')*Xt)^(-1);
part2_t = (Xt.')*(dtl.');
wt = part1_t * part2_t;

% for testing set
X(:,3) = ones;

figure;
[sen_t, spec_t] = linear_eval(Xt,dtl,wt);
title(sprintf('Training Set - Sentivitiy: %0.2f, Specificity: %0.2f', sen_t, spec_t));
figure;
[sen, spec] = linear_eval(X,dl,wt);
title(sprintf('Testing Set - Sentivitiy: %0.2f, Specificity: %0.2f', sen, spec));

% SVM:
%%%% use svc, svcplot and plot_results and svcbound
% training LSVM:
[nsv, alpha, b0] = svc(Xt,dt','linear');
figure;
svcplot(Xt, dt','linear',alpha,b0); %plotting training results
title('Training data - SVM');

% plotting test results:
y = svcoutput(Xt, dt', X, 'linear', alpha, b0); % applying classifier
figure;
X = s.X;
[sen_svm, spec_svm] = plot_results(X, d, y, 0); %plotting data
svcbound(Xt, dt', 'linear', alpha, b0);  % plotting boundaries
title(sprintf('Testing Data, SVM - Sentivitiy: %0.2f, Specificity: %0.2f', sen_svm, spec_svm));
```

```
Support Vector Classification
_____________________________
Constructing ...
Optimising ...
The interior-point-convex algorithm does not accept an initial point.
Ignoring X0.

No feasible solution found.

quadprog stopped because it was unable to find a point that satisfies
the constraints within the value of the constraint tolerance.

Execution time:  0.2 seconds
Status : -2.000000e+00
|w0|^2    : 171317.539307
Margin    : 0.004832
Sum alpha : 70010928.258251
Support Vectors : 50 (100.0%)
```


Training Set - Sentivitiy: 92.00, Specificity: 88.00
6
5
4
3
2
1
0
-1
-2
x2
-2
-1
0
1
2
3
4
5
x1


Testing Set - Sentivitiy: 88.50, Specificity: 95.00
7
6
5
4
3
2
1
0
-1
-2
x2
-3
-2
-1
0
1
2
3
4
5
6
x1

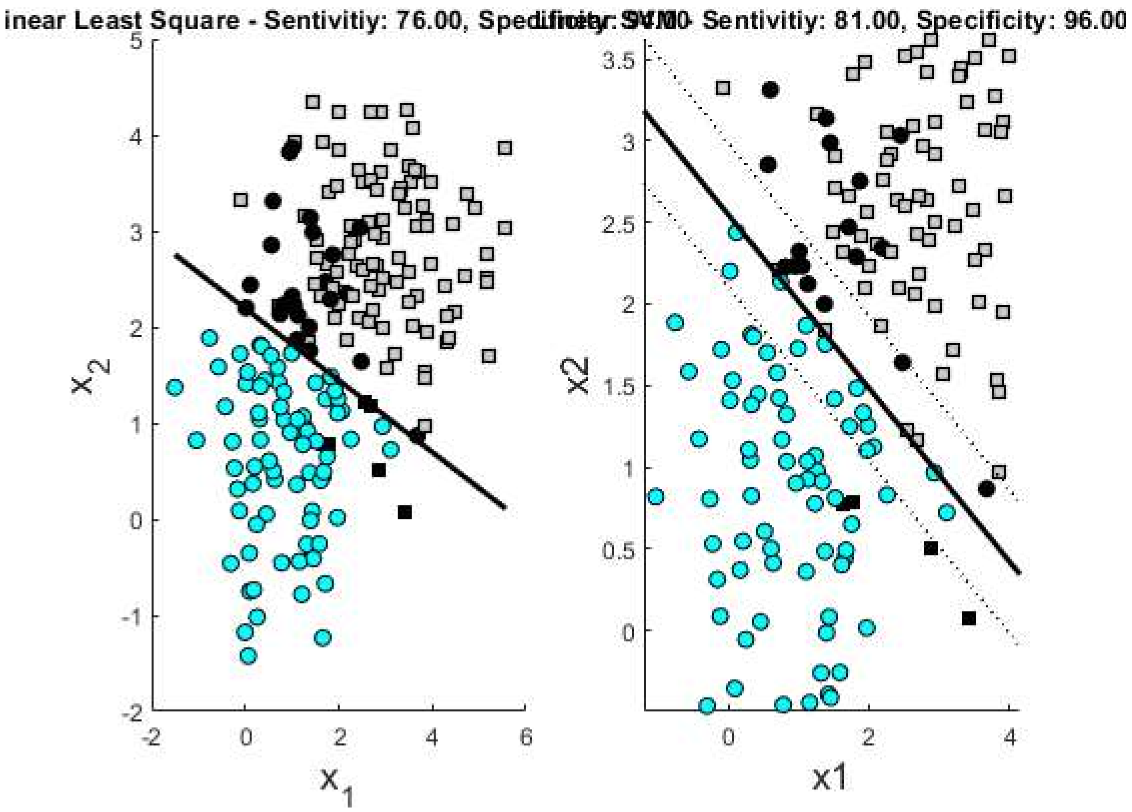


Question 5**:

```
%Loading the signal data
s = load('prob16_8.mat');

% Training set
N1 = 10;
dt = s.dt;
dtl = s.dtl;
```

```
Xt = s.Xt;

% Testing set
N2 = 200;
d = s.d;
dl = s.dl;
X = s.X;

% Linear Least Square
% for testing set
Xt(:,3) = ones;
X(:,3) = ones;
part1 = ((Xt.')*Xt)^(-1);
part2 = (Xt.')*(dtl.');
wt = part1 * part2;

figure;
subplot(1,2,1);
[sen_l, spec_l] = linear_eval(X,dl,wt);
title(sprintf('Linear Least Square - Sentivitiy: %0.2f, Specificity: %0.2f', sen_l, spec_l));

% SVM
[nsv, alpha, b0] = svc(Xt, dt', 'linear');
y = svcoutput(Xt, dt', X, 'linear', alpha, b0);
subplot(1,2,2);
X = s.X;
[sen_svm, spec_svm] = plot_results(X, d, y, 0);
svcbound(Xt, dt', 'linear', alpha, b0);  % plotting boundaries
title(sprintf('Linear SVM - Sentivitiy: %0.2f, Specificity: %0.2f', sen_svm, spec_svm));

% From the results, it can be seen that the Linear SVM approach has better
% performance (higher sensitivity and specificity) in comparison with the
% linear least square method. This is one of the strength of the SVM
% approach for this type of dataset size and can be supported by the
% results.
```

```
Support Vector Classification
_____________________________
Constructing ...
Optimising ...
The interior-point-convex algorithm does not accept an initial point.
Ignoring X0.

Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

Execution time:  0.0 seconds
Status :
|w0|^2    : 6.594742
Margin    : 0.778809
Sum alpha : 6.594742
Support Vectors : 3 (30.0%)
```

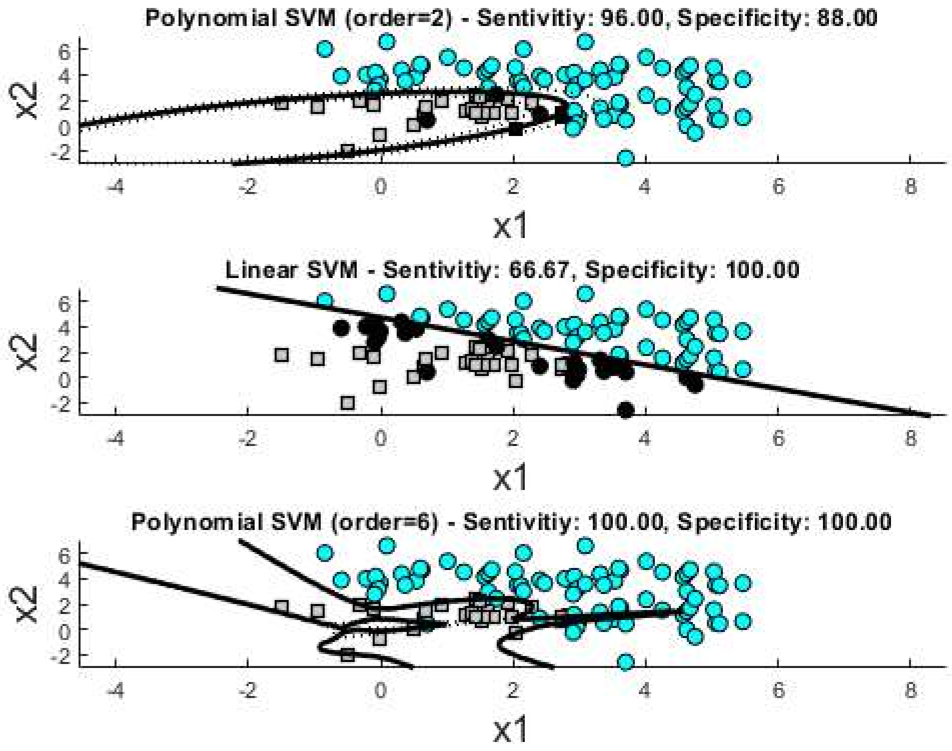


Question 6:

```
%Loading the signal data
s = load('prob16_10.mat');
Xt = s.Xt;
dt = s.dt;

% polynomial order = 2
global p1;
p1 = 2;
[nsv_p1, alpha_p1, b_p1] = svc(Xt, dt', 'poly', 10);
y_p1 = svcoutput(Xt, dt', Xt, 'poly', alpha_p1, b_p1);
figure;
subplot(3,1,1);
[sen_p1, spec_p1] = plot_results(Xt, dt, y_p1, 0);
svcbound(Xt, dt', 'poly', alpha_p1, b_p1);
title(sprintf('Polynomial SVM (order=2) - Sentivitiy: %0.2f, Specificity: %0.2f', sen_p1, spec_p1));

% Linear SVM
[nsv, alpha, b0] = svc(Xt, dt', 'linear');
y_l = svcoutput(Xt, dt', Xt, 'linear', alpha, b0);
subplot(3,1,2);
[sen_l, spec_l] = plot_results(Xt, dt, y_l, 0);
svcbound(Xt, dt', 'linear', alpha, b0);  % plotting boundaries
title(sprintf('Linear SVM - Sentivitiy: %0.2f, Specificity: %0.2f', sen_l, spec_l));

% polynomial order = 6
global p1;
p1 = 6;
[nsv_p2, alpha_p2, b_p2] = svc(Xt, dt', 'poly');
y_p2 = svcoutput(Xt, dt', Xt, 'poly', alpha_p2, b_p2);
subplot(3,1,3);
[sen_p2, spec_p2] = plot_results(Xt, dt, y_p2, 0);
svcbound(Xt, dt', 'poly', alpha_p2, b_p2);
title(sprintf('Polynomial SVM (order=6) - Sentivitiy: %0.2f, Specificity: %0.2f', sen_p2, spec_p2));

% as the complexity of the boundary increases, the performance also
```

```
% increases. This can be seen in the results shown from polynomial order 2
% compared with polynomial order 6 in which polynomial order 6 has higher
% sensitiviy and specificity values.
```

```
Support Vector Classification
_____________________________
Constructing ...
Optimising ...
The interior-point-convex algorithm does not accept an initial point.
Ignoring X0.

Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

Execution time:  0.0 seconds
Status :
|w0|^2    : 14.526203
Margin    : 0.524752
Sum alpha : 142.694010
Support Vectors : 17 (17.0%)
Support Vector Classification
_____________________________
Constructing ...
Optimising ...
The interior-point-convex algorithm does not accept an initial point.
Ignoring X0.

No feasible solution found.

quadprog stopped because it was unable to find a point that satisfies
the constraints within the value of the constraint tolerance.

Execution time:  0.0 seconds
Status : -2.000000e+00
|w0|^2    : 199929.043973
Margin    : 0.004473
Sum alpha : 91053740.043034
Support Vectors : 100 (100.0%)
Support Vector Classification
_____________________________
Constructing ...
Optimising ...
The interior-point-convex algorithm does not accept an initial point.
Ignoring X0.

Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

Execution time:  0.0 seconds
Status :
|w0|^2    : 41.467967
Margin    : 0.310580
Sum alpha : 41.467967
Support Vectors : 14 (14.0%)
```

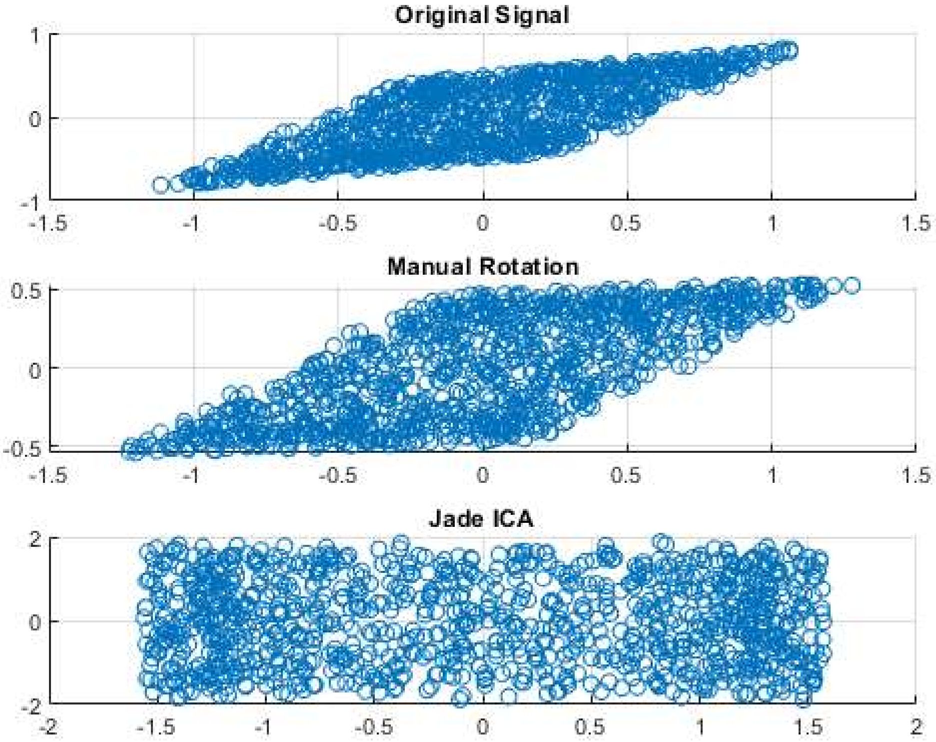


Question 7**:

```
%loading the data
s = load('p9_1_data.mat');
X = s.X;
x1 = X(1,:);
x2 = X(2,:);
N = length(x1);

% determine kurtosis for each of the two signals
k1 = kurtosis(x1)-3;
k2 = kurtosis(x2)-3;
sprintf('Signal 1 Kurtosis = %0.2f',k1)
sprintf('Signal 2 Kurtosis = %0.2f',k2)

% add the two kurtosis together
k_sum = k1 + k2;
sprintf('Kurtosis sum = %0.2f',k_sum)

figure;
subplot(3,1,1);
scatter(X(1,:),X(2,:)); title('Original Signal'); grid on;

r1 = 0.867;
r2 = 0.5;
r3 = 0.9;
r4 = 1.2;
r5 = 2.9;

y = rotation(X, r5);
subplot(3,1,2);
scatter(y(1,:), y(2,:)); title('Manual Rotation'); grid on;

% Apply the Jade ICA algorithm
B = jadeR(X,2);
y2 = B*X;
subplot(3,1,3);
```

```
scatter(y2(1,:), y2(2,:)); title('Jade ICA'); grid on;

% From the results, it can be seen that the manual and Jade ICA approach do
% not yield the same results.
% The kurtosis values for the data rotated by ICA are more negative than
% the kurtosis values of the ones found manually. This indicated high
% non-Gaussianity and therefore, maximizing independence between the two
% components.
% the result from the manually rotated data is less effective than the ICA
% algorithm in finding the maximum.
```

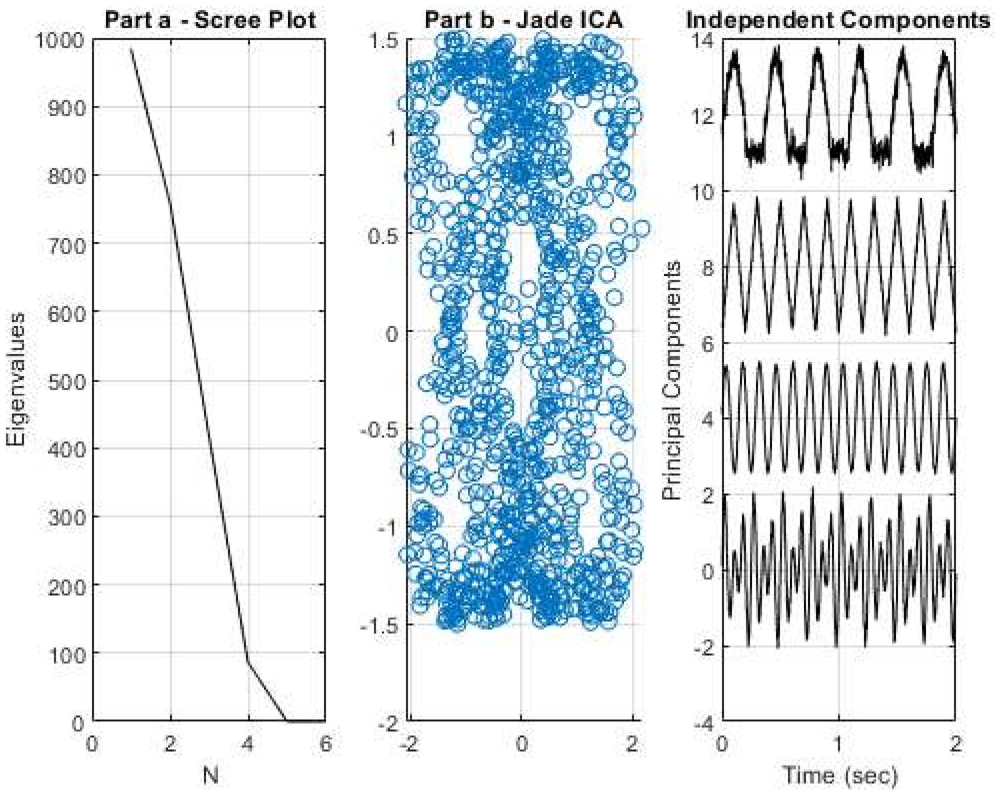
```
ans =

    'Signal 1 Kurtosis = -0.68'


ans =

    'Signal 2 Kurtosis = -1.23'


ans =

    'Kurtosis sum = -1.91'
```



Question 8**:

```
%loading the data
s = load('p9_2_data.mat');
X = s.X;
fs = 500;
N = length(X);
t = (1:N)/fs;
```

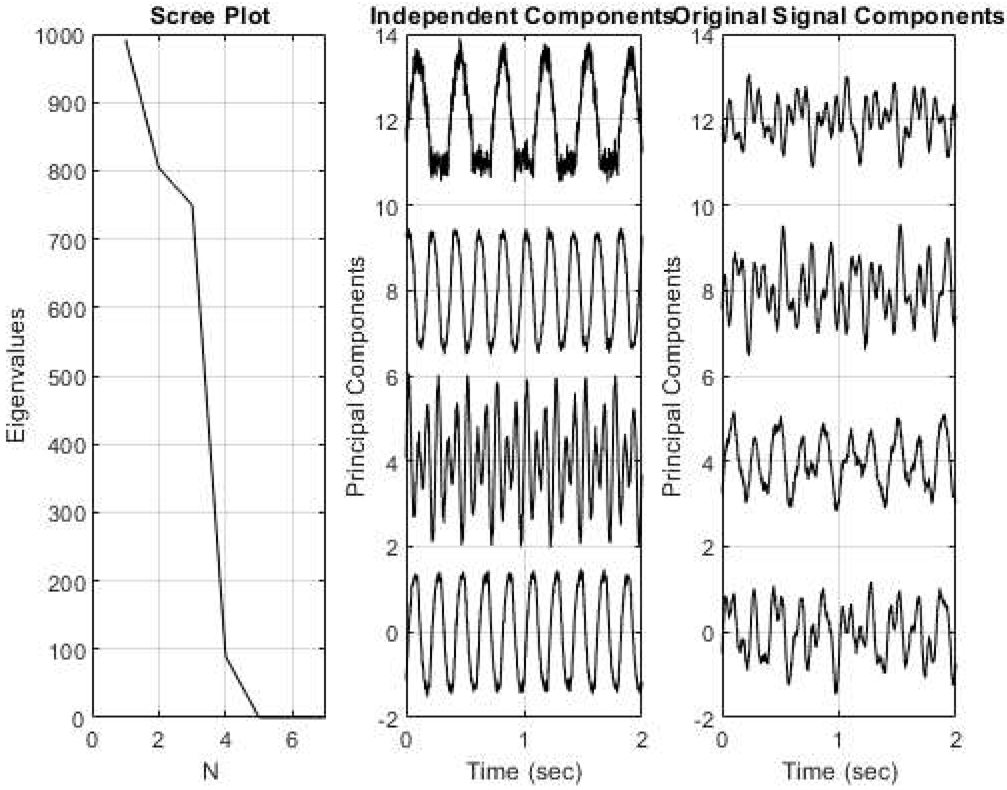
```
% part a: Determine the actual dimension of the data using PCA and the Scree plot.
%Find principal components
[U,S,pc]= svd(X,'econ');
eigen = diag(S).^2;
for i = 1:6
   pc(:,i) = pc(:,i) * sqrt(eigen(i));
end

figure;
subplot(1,3,1);
plot(eigen,'k');
xlabel('N');
ylabel('Eigenvalues');
title('Part a - Scree Plot'); grid on;

% By looking at the Scree plot, it can be seen that the point of inflection
% is at N=4 and then approaching zero for N>4. Therefore, the dimension can
% be estimated to be 4.

% part b: Perform an ICA analysis using either the "Jade" or "FastICA"
B = jadeR(X,4);
y = B*X;
subplot(1,3,2);
scatter(y(1,:), y(2,:)); title('Part b - Jade ICA'); grid on;
% Plot the independent components
subplot(1,3,3);
y = y';
plot(t,y(:,1),'k',t,y(:,2)+4,'k', t, y(:,3)+8,'k',t, y(:,4)+12,'k'); grid on;
xlabel('Time (sec)');
ylabel('Principal Components');
title('Independent Components');
```



Question 9**:

```
%loading the data
s = load('mix_sig3.mat');
```

```matlab
X = s.X;
X = X';
fs = 500;
N = length(X);
t = (1:N)/fs;

%Find principal components
[U,S,pc]= svd(X,'econ');
eigen = diag(S).^2;
for i = 1:7
   pc(:,i) = pc(:,i) * sqrt(eigen(i));
end

figure;
subplot(1,3,1);
plot(eigen,'k');
xlabel('N');
ylabel('Eigenvalues');
title('Scree Plot'); grid on;

% By looking at the Scree plot, it can be seen that the point of inflection
% is at N=4 and then approaching zero for N>4. Therefore, the dimension can
% be estimated to be 4.
% Apply ICA with Jade
B = jadeR(X,4);
y = B*X;
% Plot the independent components
subplot(1,3,2);
y = y';
plot(t,y(:,1),'k',t,y(:,2)+4,'k', t, y(:,3)+8,'k',t, y(:,4)+12,'k'); grid on;
xlabel('Time (sec)');
ylabel('Principal Components');
title('Independent Components');

X = X';
subplot(1,3,3);
plot(t,X(:,1),'k',t,X(:,2)+4,'k', t, X(:,3)+8,'k',t, X(:,4)+12,'k'); grid on;
xlabel('Time (sec)');
ylabel('Principal Components');
title('Original Signal Components');

% some of the reasons that could lead to ICA failing is the limited number
% of recordings making it an overcomplete problem (non-square ICA)
% Another reason could be  that the distribution of the signal is close to Gaussian
```

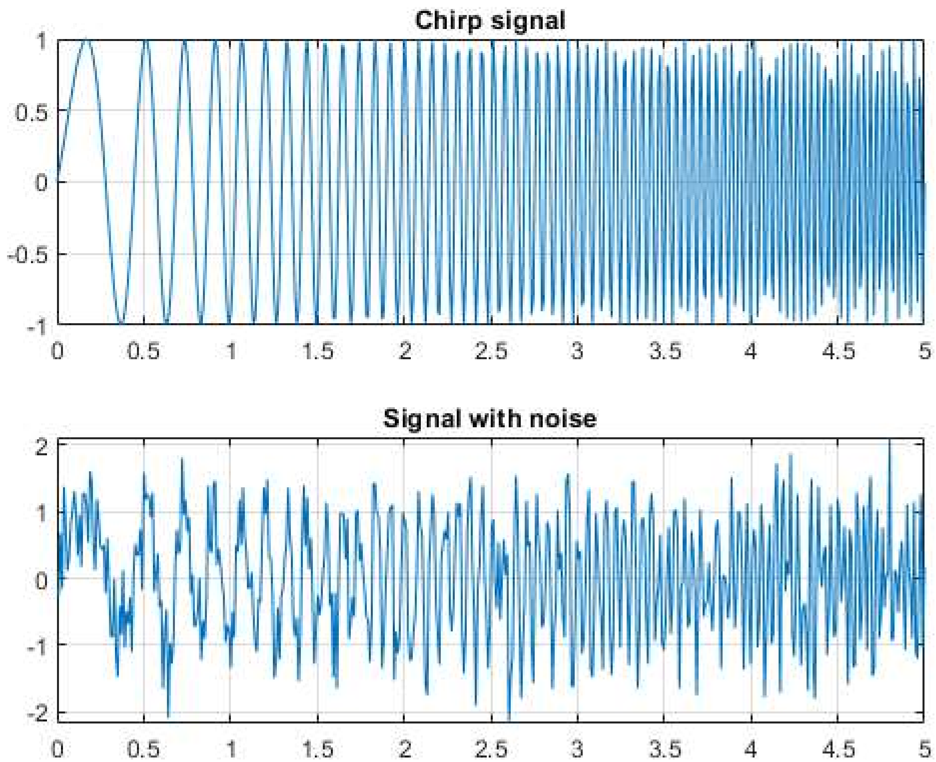


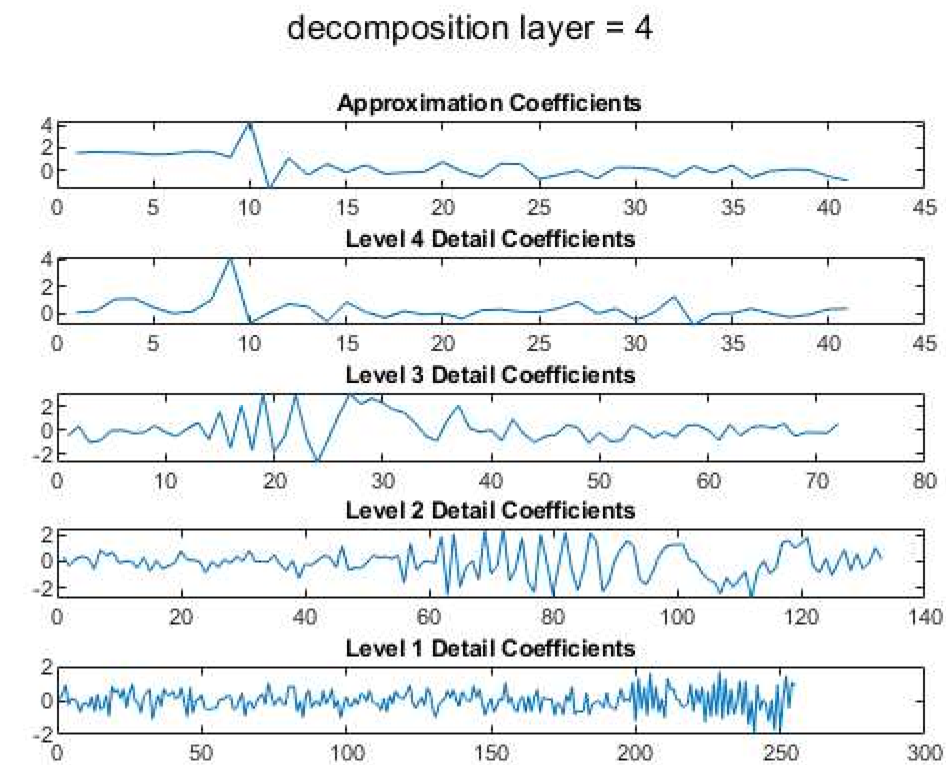
Question 10**:

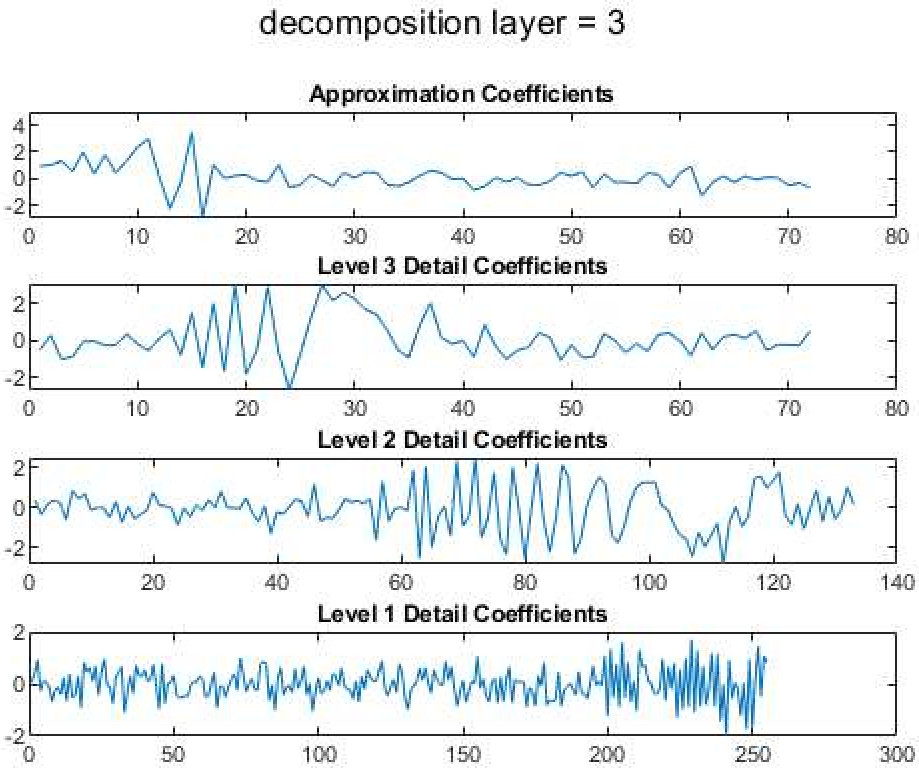
construct a chirp signal

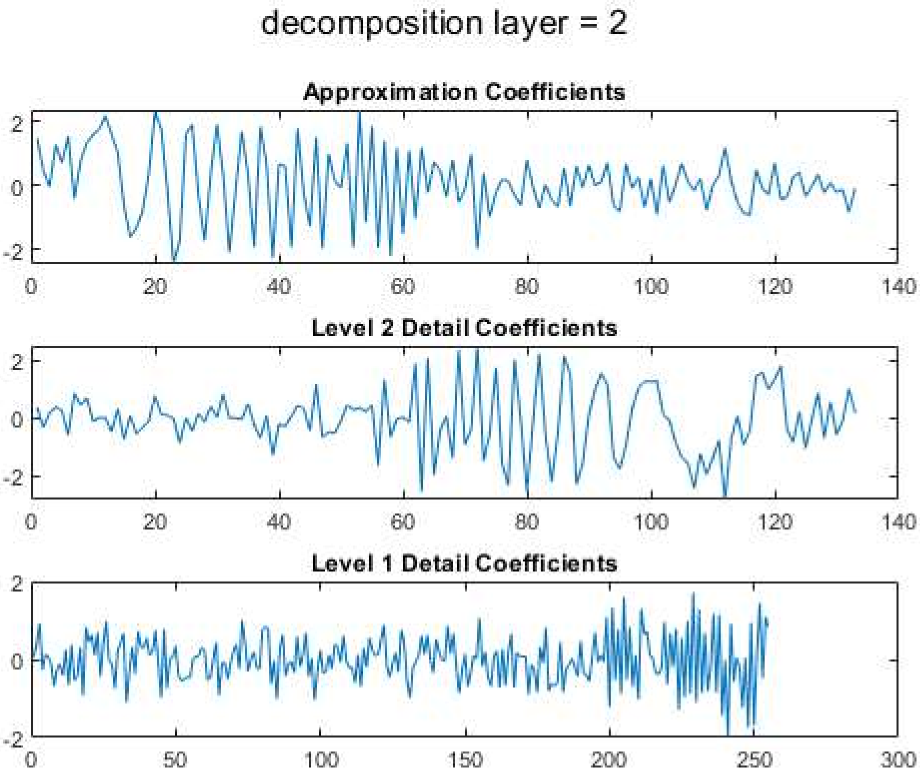
```
N = 500;
freq = linspace(2, 30, N);
time = linspace(0, 5, N);
x = sin(pi*time.*freq);   %chirp signal

figure;
subplot(2,1,1);
plot(time, x);
title('Chirp signal'); grid on;

% add noise
v = var(x);
x_noise = x + v*randn(1,length(x));
subplot(2,1,2);
plot(time,x_noise);
title('Signal with noise'); grid on;

% DWT - decomposition layer = 4
[c,l] = wavedec(x_noise,4,'db6');
approx = appcoef(c,l,'db6');
[cd1,cd2,cd3,cd4] = detcoef(c,l,[1 2 3 4]);

%Plot the coefficients.
figure;
sgtitle('decomposition layer = 4');
subplot(5,1,1);
plot(approx);
title('Approximation Coefficients');
subplot(5,1,2);
plot(cd4);
title('Level 4 Detail Coefficients');
subplot(5,1,3);
plot(cd3);
```

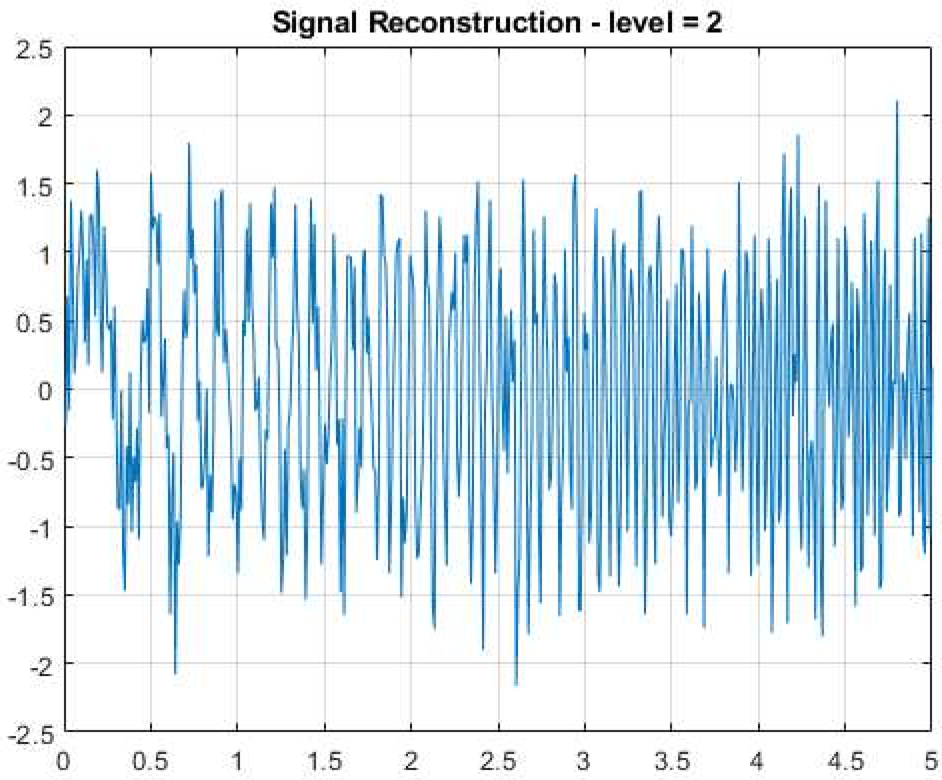
```
title('Level 3 Detail Coefficients');
subplot(5,1,4);
plot(cd2);
title('Level 2 Detail Coefficients');
subplot(5,1,5);
plot(cd1);
title('Level 1 Detail Coefficients');

% DWT - decomposition layer = 3
[c,l] = wavedec(x_noise,3,'db6');
approx = appcoef(c,l,'db6');
[cd1,cd2,cd3] = detcoef(c,l,[1 2 3]);

%Plot the coefficients.
figure;
sgtitle('decomposition layer = 3');
subplot(4,1,1);
plot(approx);
title('Approximation Coefficients');
subplot(4,1,2);
plot(cd3);
title('Level 3 Detail Coefficients');
subplot(4,1,3);
plot(cd2);
title('Level 2 Detail Coefficients');
subplot(4,1,4);
plot(cd1);
title('Level 1 Detail Coefficients');

% DWT - decomposition layer = 2
[c,l] = wavedec(x_noise,2,'db6');
approx = appcoef(c,l,'db6');
[cd1,cd2] = detcoef(c,l,[1 2]);

%Plot the coefficients.
figure;
sgtitle('decomposition layer = 2');
subplot(3,1,1);
plot(approx);
title('Approximation Coefficients');
subplot(3,1,2);
plot(cd2);
title('Level 2 Detail Coefficients');
subplot(3,1,3);
plot(cd1);
title('Level 1 Detail Coefficients');

% changing the decomposition levels gives us the same results since we're
% only removing the highest resolution that exists in the lowest level
% (level 1)

% reconstruction
x_rec3 = waverec(c,l,'db6');
figure;
plot(time,x_rec3);
title('Signal Reconstruction - level = 2'); grid on;
```


Chirp signal
Signal with noise


decomposition layer = 4
Approximation Coefficients
Level 4 Detail Coefficients
Level 3 Detail Coefficients
Level 2 Detail Coefficients
Level 1 Detail Coefficients


decomposition layer = 3
Approximation Coefficients
Level 3 Detail Coefficients
Level 2 Detail Coefficients
Level 1 Detail Coefficients

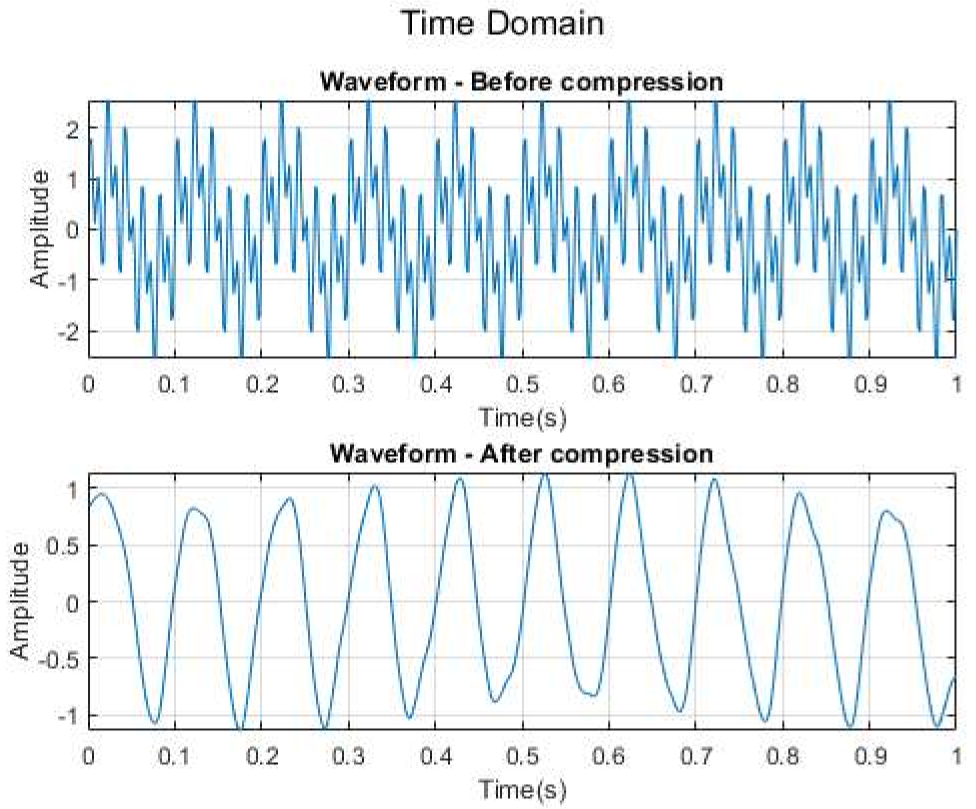

decomposition layer = 2
Approximation Coefficients
Level 2 Detail Coefficients
Level 1 Detail Coefficients

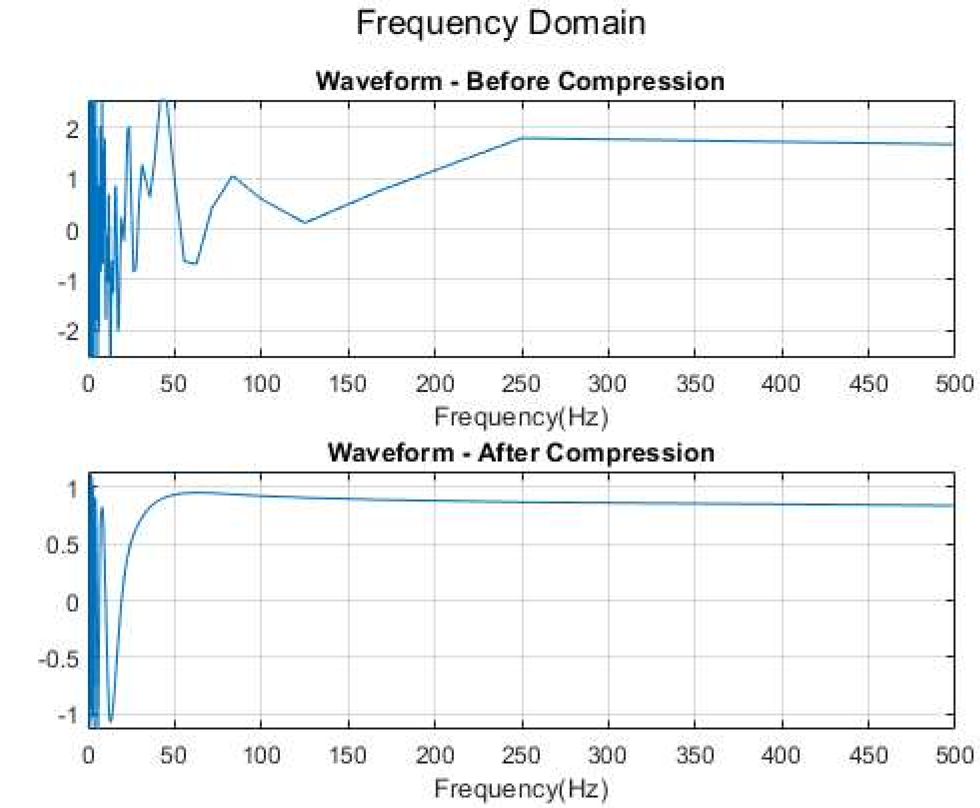


Question 11:

construct a chirp signal

```
f1 = 20;
f2 = 100;
f3 = 200;
fs = 500;
N = 500;
t = (1:N)/fs;
x = sin(pi*t*f1) + sin(pi*t*f2) + sin(pi*t*f3);
figure;
sgtitle('Time Domain');
subplot(2,1,1);
plot(t, x); xlabel('Time(s)'); ylabel('Amplitude');
title('Waveform - Before compression'); grid on;

% DWT - compression
[c,l] = wavedec(x,4,'db6');
nc = wthcoef("d",c,l,[1 2 3 4]);

% reconstruction
xrec = waverec(nc,l,'db6');
subplot(2,1,2);
plot(t,xrec); xlabel('Time(s)'); ylabel('Amplitude');
title('Waveform - After compression'); grid on;

f_t = 1./t;
figure;
sgtitle('Frequency Domain');
subplot(2,1,1);
plot(f_t, x); xlabel('Frequency(Hz)');
title('Waveform - Before Compression'); grid on;
subplot(2,1,2);
plot(f_t, xrec); xlabel('Frequency(Hz)');
title('Waveform - After Compression'); grid on;
```



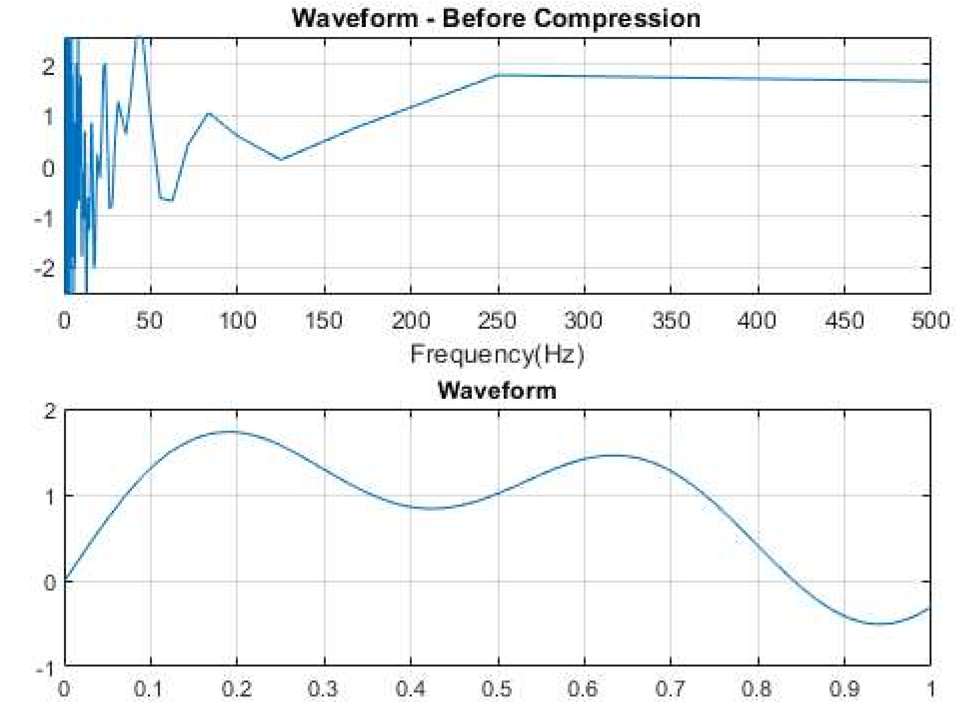


Question 12:

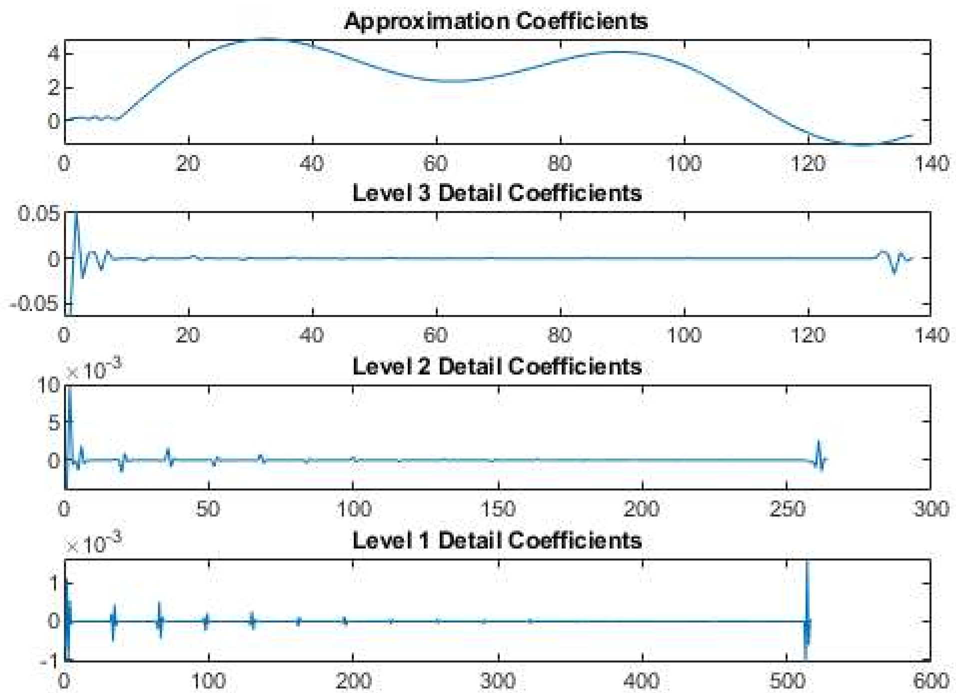
```
s = load('discontinunity_data.mat');
x = s.x;
N = length(x);
fs = 1024;
t = (1:N)/fs;
plot(t, x); grid on; title('Waveform');
% Decompose the waveform into three levels and examine and plot only the highest-resolution
```

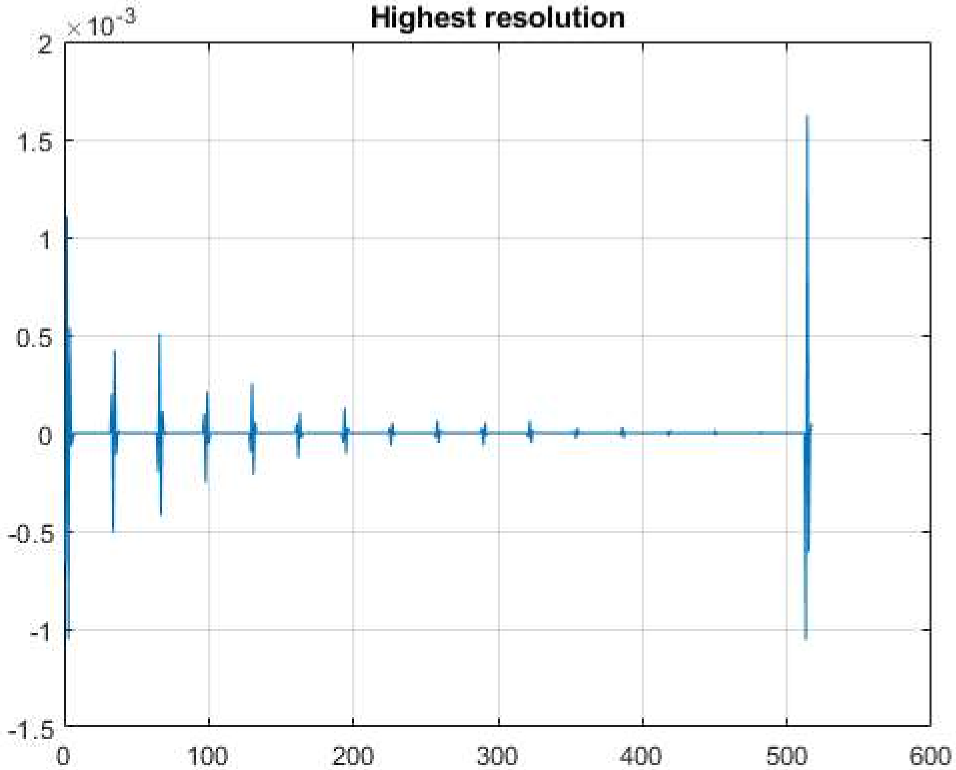
```matlab
% DWT - decomposition layer = 3
[c,l] = wavedec(x,3,'db6');
approx = appcoef(c,l,'db6');
[cd1,cd2,cd3] = detcoef(c,l,[1 2 3]);

%Plot the coefficients.
figure;
sgtitle('decomposition layer = 3');
subplot(4,1,1);
plot(approx);
title('Approximation Coefficients');
subplot(4,1,2);
plot(cd3);
title('Level 3 Detail Coefficients');
subplot(4,1,3);
plot(cd2);
title('Level 2 Detail Coefficients');
subplot(4,1,4);
plot(cd1);
title('Level 1 Detail Coefficients');

figure;
plot(cd1);
title('Highest resolution'); grid on;
```

Frequency Domain



decomposition layer = 3





Appendix:

%% linear_eval code slightly modified to display tp, tn, fp and fn values as below:

function [sensitivity, specificity] = linear_eval_Q2(X,d,w,threshold) [sensitivity, specificity] = linear_eval(X,d,w,threshold) Evaluates performance of a linear discriminator and plots results including the decision boundary Plots only two-variable input data, but has been expanded for three-variables in routine 'linear_eval3D' Uses the critera of > 0.5 to classify

Inputs X inputs (assumed a matrix where the number or rows are different) samples and the number of columns are the input dimension d correct outputs; i.e., targets (assumed a vector) w linear weights threshold decision boundary threshold (default 0.5) Outputs sensitivity Percent true positives specificity Percent true negative

if nargin < 4 threshold = .5; end tp = 0; % True positive count fp = 0; % False positive count tn = 0; % True negative count fn = 0; % False negative count [r,c] = size(X); % Determine linear response to X y = X*w; % Evaluate the output % % Plot the results hold on; % Assumes Class 0 is 0 (positive) and Class 1 is 1 (negative) Evaluates % each point for all four possibilities for i = 1:r if d(i) > threshold && y(i) > threshold plot(X(i,1),X(i,2),'sqk','MarkerFaceColor',[.8 .8 .8],'LineWidth',1); tn = tn + 1; %True negative elseif d(i) > threshold && y(i) <= threshold plot(X(i,1),X(i,2),'sqk','MarkerFaceColor','k'); fp = fp + 1; % False positive elseif d(i) <= threshold && y(i) <= threshold plot(X(i,1),X(i,2),'ok','MarkerFaceColor','c'); tp = tp + 1; % True positive elseif d(i) <= threshold && y(i) > threshold plot(X(i,1),X(i,2),'ok','MarkerFaceColor','k'); fn = fn + 1; % False negative end end % V = axis; % Get current axis % % Plot decision boundary W*x = .5 x1 = [min(X(:,1)),max(X(:,1))]; % Construct x1 over data range x2 = -w(1)*x1/w(2) + (-w(3)+threshold)/w(2); % Calculate x2 using Eq. (9) plot(x1,x2,'k','LineWidth',2); % Plot boundary line axis(V); % Restore axis xlabel('x_1','FontSize',14); ylabel('x_2','FontSize',14); % Evaluate sensitivity specificity = (tn/(tn+fp))*100; sensitivity = (tp/(tp+fn))*100;

sprintf('True Positive: %.0f', tp) sprintf('True Negative: %.0f', tn) sprintf('False Positive: %.0f', fp) sprintf('False Negative: %.0f', fn)

Published with MATLAB® R2022a