Part 2

Report

Biological Signal Processing

Ardalan Gerami 99102112

6/26/2024

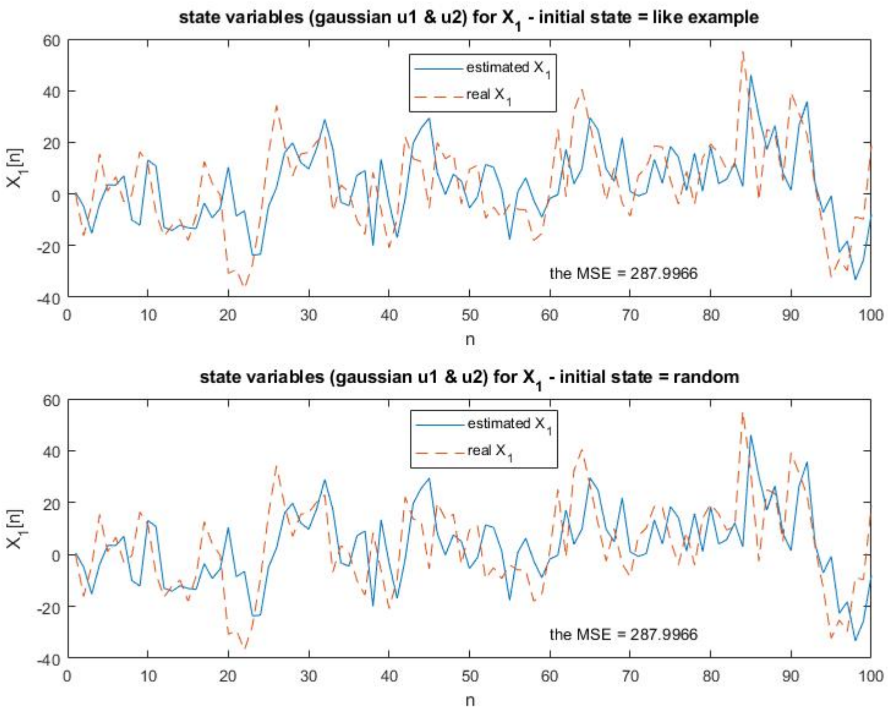
Contents

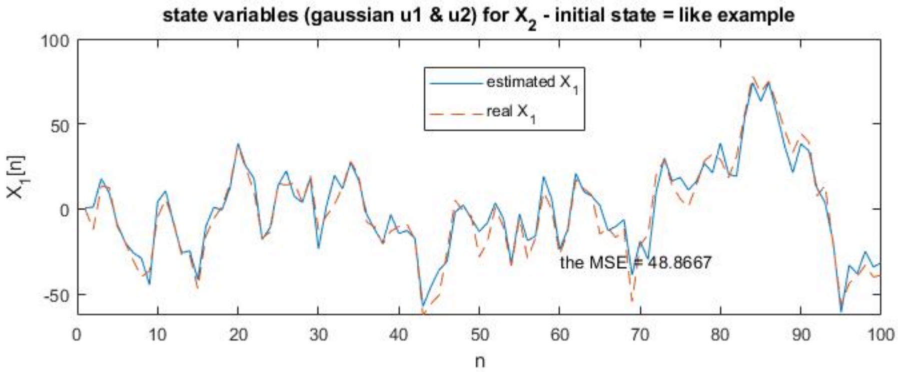
1 Question 1

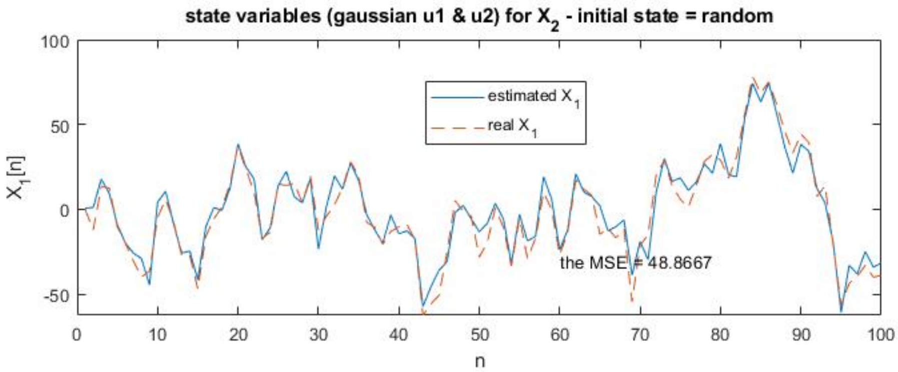
-Above all I should mention that I’ve answered the questions of part d and e in earlier three parts.
-Also the evaluation criteria is shown on the plot.
-At the end of the question I’ve explained the answers.

1.1 part a,d,e

- The estimation of state variables by initial states like ”example 2” and random initial states = [0.65,0.78] , MSE as Evaluation criteria:

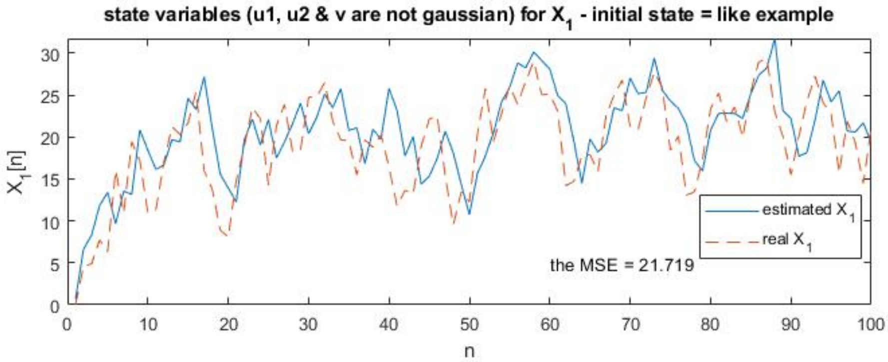


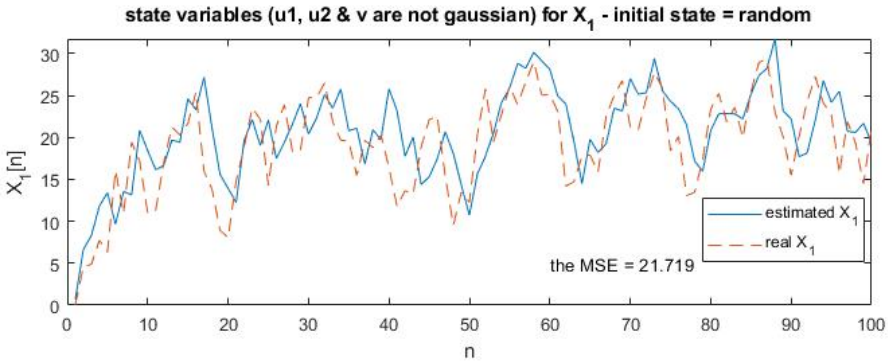


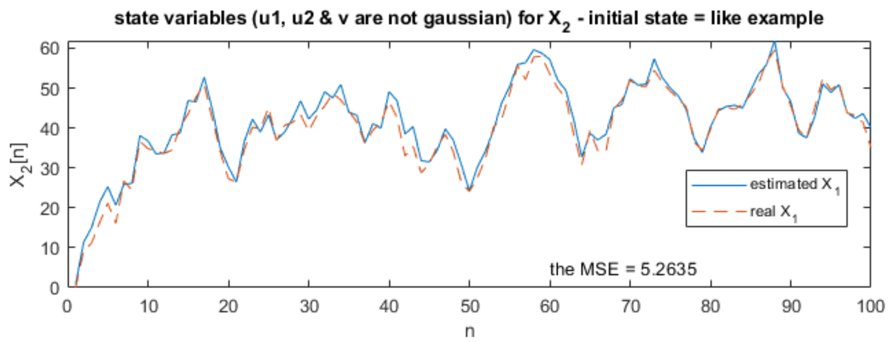


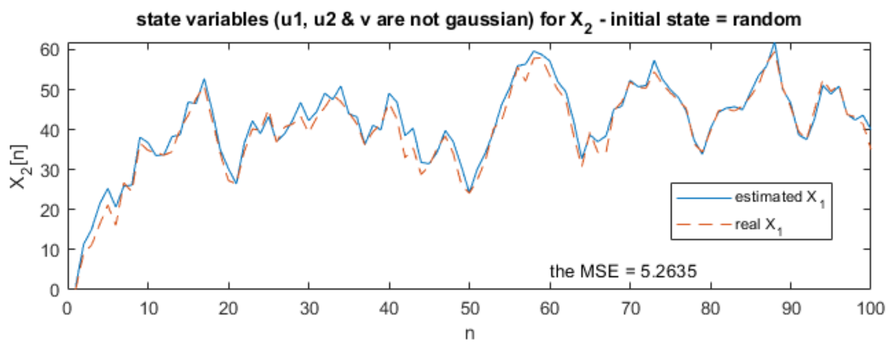
1.2 part b,d,e

- The estimation of state variables by initial states like ”example 2” and random initial states = [0.65,0.78] , MSE as Evaluation criteria when noises are not gaussian :


state variables (u1, u2 & v are not gaussian) for X_1 - initial state = like example
$X_1[n]$
n
estimated X_1
real X_1
the MSE = 21.719

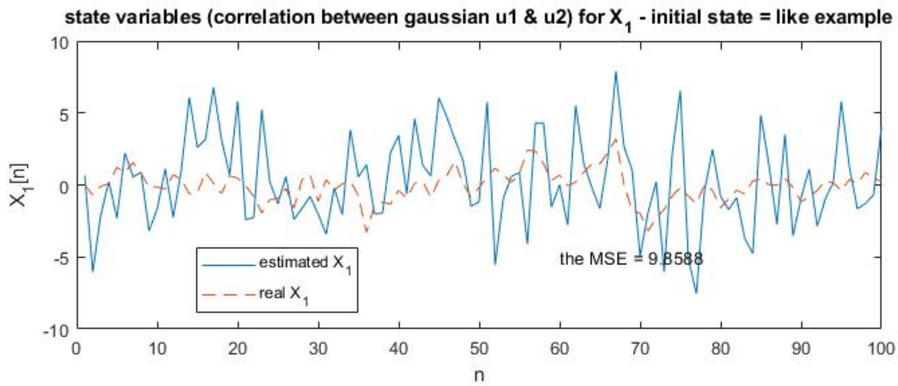

state variables (u1, u2 & v are not gaussian) for X_1 - initial state = random
$X_1[n]$
n
estimated X_1
real X_1
the MSE = 21.719

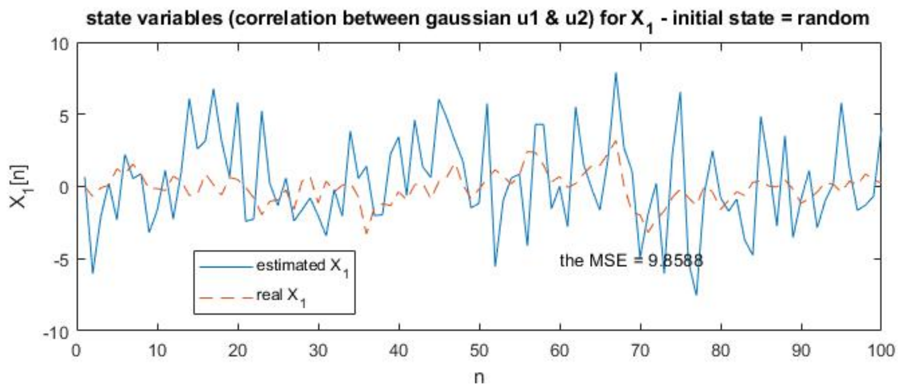

state variables (u1, u2 & v are not gaussian) for X_2 - initial state = like example
$X_2[n]$
n
estimated X_1
real X_1
the MSE = 5.2635

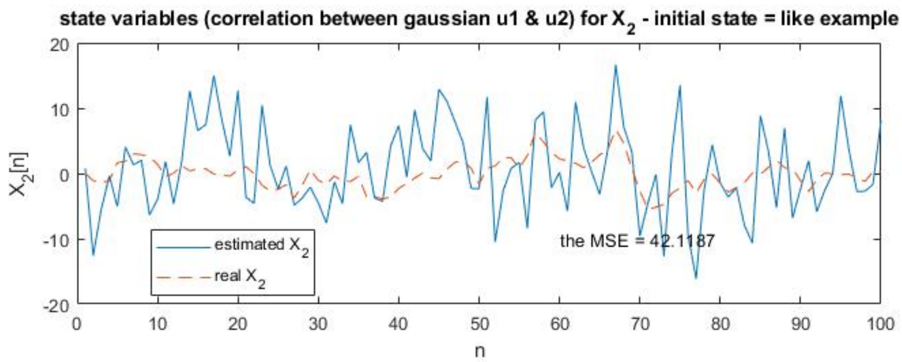

state variables (u1, u2 & v are not gaussian) for X_2 - initial state = random
$X_2[n]$
n
estimated X_1
real X_1
the MSE = 5.2635

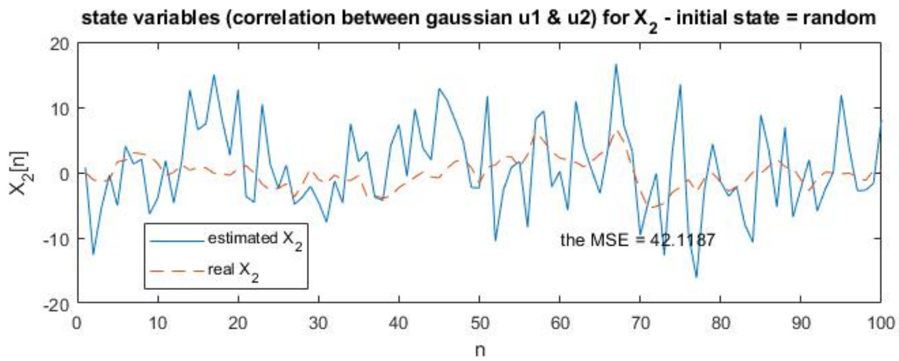
1.3 part c,d,e

- The estimation of state variables by initial states like ”example 2” and random initial states = [0.65,0.78] , MSE as Evaluation criteria when u1,u2 noises are correlated :









1.4 Explanation

Except for part C, where the noises u_1 and u_2 are correlated, the estimation of the state variables is excellent, and our Kalman filter's estimations closely follow the real values. The estimation of X_2 is particularly accurate, likely due to the fact that two filters are applied.

Matlab Code

```
clc; clear
%% inputs
a = 0.6;
b = 0.4;
u1_std = sqrt(12)*4;
u2_std = sqrt(12)*3;
```

```
v_std = sqrt(12)*2;
k = 100;
u1_a = u1_std * randn(1, k);
u2_a = u2_std * randn(1, k);
v_a = v_std * randn(1, k);
u1_b = u1_std * rand(1, k);
u2_b = u2_std * rand(1, k);
v_b = v_std * rand(1, k);
rho = 0.8;
x1 = randn(k, 1);
x2 = randn(k, 1);
R = [1 rho; rho 1];
L = chol(R, 'lower');
correlated_noises = L * [x1 x2]';
u1_c = correlated_noises(1, :);
u2_c = correlated_noises(2, :);
v_c = v_std * randn(1, k);
est_x0_handout = [0,0];
est_x0_random = [0.65,0.78];
%% part a
[~,~,est_x_a,x1_a,x2_a, ~, ~, ~, ~] = kalmanmanf_part(
    a, b, u1_std, u2_std, v_std, u1_a, u2_a, v_a,
    est_x0_handout);
[~,~,est_x_da,x1_da,x2_da, ~, ~, ~, ~] =
    kalmanmanf_part(a, b, u1_std, u2_std, v_std, u1_a,
    u2_a, v_a, est_x0_random);
figure;
subplot(2,1,1)
plot(est_x_a(1,:))
hold on
plot(x1_a,'--')
```

```
legend('estimated X_1','real X_1')
title('state variables (gaussian u1 & u2) for X_1 -
    initial state = like example')
ylabel('X_1[n]')
xlabel('n')
MSE = mean((est_x_a(1,:)-x1_a).^2,2);
text(60,-30,['the MSE = ' , num2str(MSE)])
subplot(2,1,2)
plot(est_x_da(1,:))
hold on
plot(x1_da,'--')
legend('estimated X_1','real X_1')
title('state variables (gaussian u1 & u2) for X_1 -
    initial state = random')
ylabel('X_1[n]')
xlabel('n')
MSE = mean((est_x_da(1,:)-x1_da).^2,2);
text(60,-30,['the MSE = ' , num2str(MSE)])
% part d
figure;
subplot(2,1,1)
plot(est_x_a(2,:))
hold on
plot(x2_a,'--')
legend('estimated X_1','real X_1')
title('state variables (gaussian u1 & u2) for X_2 -
    initial state = like example')
ylabel('X_1[n]')
xlabel('n')
MSE = mean((est_x_a(2,:)-x2_a).^2,2);
text(60,-30,['the MSE = ' , num2str(MSE)])
```

```
subplot(2,1,2)
plot(est_x_da(2,:))
hold on
plot(x2_da,'--')
legend('estimated X_1','real X_1')
title('state variables (gaussian u1 & u2) for X_2 - 
    initial state = random')
ylabel('X_1[n]')
xlabel('n')
MSE = mean((est_x_da(2,:)-x2_da).^2,2);
text(60,-30,['the MSE = ' , num2str(MSE)])
%% part b
[~,~,est_x_b,x1_b,x2_b, ~, ~, ~, ~] = kalmanmanf_part(
    a, b, u1_std, u2_std, v_std, u1_b, u2_b, v_b,
    est_x0_handout);
[~,~,est_x_db,x1_db,x2_db, ~, ~, ~, ~] =
    kalmanmanf_part(a, b, u1_std, u2_std, v_std, u1_b,
    u2_b, v_b, est_x0_random);
figure;
subplot(2,1,1)
plot(est_x_b(1,:))
hold on
plot(x1_b,'--')
legend('estimated X_1','real X_1')
title('state variables (u1, u2 & v are not gaussian)
    for X_1 - initial state = like example')
ylabel('X_1[n]')
xlabel('n')
MSE = mean((est_x_b(1,:)-x1_b).^2,2);
text(60,5,['the MSE = ' , num2str(MSE)])
subplot(2,1,2)
```

```
plot(est_x_db(1,:))
hold on
plot(x1_db,'--')
legend('estimated X_1','real X_1')
title('state variables (u1, u2 & v are not gaussian) for X_1 - initial state = random')
ylabel('X_1[n]')
xlabel('n')
MSE = mean((est_x_db(1,:)-x1_db).^2,2);
text(60,5,['the MSE = ' , num2str(MSE)])
%
figure;
subplot(2,1,1)
plot(est_x_b(2,:))
hold on
plot(x2_b,'--')
legend('estimated X_1','real X_1')
title('state variables (u1, u2 & v are not gaussian) for X_2 - initial state = like example')
ylabel('X_2[n]')
xlabel('n')
MSE = mean((est_x_b(2,:)-x2_b).^2,2);
text(60,5,['the MSE = ' , num2str(MSE)])
subplot(2,1,2)
plot(est_x_db(2,:))
hold on
plot(x2_db,'--')
legend('estimated X_1','real X_1')
title('state variables (u1, u2 & v are not gaussian) for X_2 - initial state = random')
ylabel('X_2[n]')
```

```
xlabel('n')
MSE = mean((est_x_db(2,:)-x2_db).^2,2);
text(60,5,['the MSE = ' , num2str(MSE)])
%% part c
[~,~,est_x_c,x1_c,x2_c, ~, ~, ~, ~] = kalmanmanf_partc
    (a, b, u1_std, u2_std, v_std, u1_c, u2_c, v_c,
    est_x0_handout,rho);
[~,~,est_x_dc,x1_dc,x2_dc, ~, ~, ~, ~] =
    kalmanmanf_partc(a, b, u1_std, u2_std, v_std, u1_c,
     u2_c, v_c, est_x0_random,rho);
figure;
subplot(2,1,1)
plot(est_x_c(1,:))
hold on
plot(x1_c,'--')
legend('estimated X_1','real X_1')
title('state variables (correlation between gaussian
    u1 & u2) for X_1 - initial state = like example')
ylabel('X_1[n]')
xlabel('n')
MSE = mean((est_x_c(1,:)-x1_c).^2,2);
text(60,-5,['the MSE = ' , num2str(MSE)])
subplot(2,1,2)
plot(est_x_dc(1,:))
hold on
plot(x1_dc,'--')
legend('estimated X_1','real X_1')
title('state variables (correlation between gaussian
    u1 & u2) for X_1 - initial state = random')
ylabel('X_1[n]')
xlabel('n')
```

```
MSE = mean((est_x_dc(1,:)-x1_dc).^2,2);
text(60,-5,['the MSE = ' , num2str(MSE)])
%
figure;
subplot(2,1,1)
plot(est_x_c(2,:))
hold on
plot(x2_c,'--')
legend('estimated X_2','real X_2')
title('state variables (correlation between gaussian
    u1 & u2) for X_2 - initial state = like example')
ylabel('X_2[n]')
xlabel('n')
MSE = mean((est_x_c(2,:)-x2_c).^2,2);
text(60,-5*2,['the MSE = ' , num2str(MSE)])
subplot(2,1,2)
plot(est_x_dc(2,:))
hold on
plot(x2_dc,'--')
legend('estimated X_2','real X_2')
title('state variables (correlation between gaussian
    u1 & u2) for X_2 - initial state = random')
ylabel('X_2[n]')
xlabel('n')
MSE = mean((est_x_dc(2,:)-x2_dc).^2,2);
text(60,-5*2,['the MSE = ' , num2str(MSE)])
%% the function for kalman filter
function [estimated_z,real_output,est_x,x1,x2,
    est_xminus, est_p, est_pminus, est_G] =
    kalmanmanf_part(a, b, std1, std2, stdv, u1, u2 ,v ,
    est_x0)
```

```
k = 100;
H1 = tf([1], [1 -a], 1);
H2 = tf([1], [1 -b], 1);
x1 = filter(H1.Numerator{1}, H1.Denominator{1}, u1);
x2 = filter(H2.Numerator{1}, H2.Denominator{1}, x1 + u2);
real_output = x2 + v;
est_x = zeros(2, k);
est_xminus = zeros(2, k);
est_p = zeros(2, 2, k);
est_pminus = zeros(2, 2, k);
est_G = zeros(2, 1, k);
est_p(:, :, 1) = [std1^2/(1-a^2), std1^2/((1-a^2)*(1-a*b)); std1^2/((1-a^2)*(1-a*b)), (std1^2+(1-a^2)*std2^2)/((1-a^2)*(1-b^2))];
est_x(:, 1) = est_x0;
est_x(:, 1) = [0.65; 0.78];
F = [a,0;a,b];
Q = [std1^2, std1^2; std1^2, std1^2 + std2^2];
R = stdv^2;
H = [0,1];
estimated_z = zeros(1,k);
estimated_z(1,1) = H*est_x(:,1) + v(1);
for i = 2:k
    est_xminus(:, i) = F * est_x(:, i-1);
    est_pminus(:, :, i) = F * est_p(:, :, i-1) * F' + Q;
    Gk = est_pminus(:, :, i) * H' / (H * est_pminus(:, :, i) * H' + R);
```

```
            est_x(:, i) = est_xminus(:, i) + Gk * (
                real_output(i) - H * est_xminus(:, i));
            est_p(:, :, i) = est_pminus(:, :, i) - Gk * H
                * est_pminus(:, :, i);
            est_G(:, :, i) = Gk;
            estimated_z(1,i) = H*est_x(:,i) + v(i);
        end
end

function [estimated_z,real_output,est_x,x1,x2,
    est_xminus, est_p, est_pminus, est_G] =
    kalmanmanf_partc(a, b, std1, std2, stdv, u1, u2 ,v
    ,est_x0,rho)
     k = 100;
     H1 = tf([1], [1 -a], 1);
     H2 = tf([1], [1 -b], 1);
     x1 = filter(H1.Numerator{1}, H1.Denominator{1}, u1
         );
     x2 = filter(H2.Numerator{1}, H2.Denominator{1}, x1
          + u2);
     real_output = x2 + v;
     est_x = zeros(2, k);
     est_xminus = zeros(2, k);
     est_p = zeros(2, 2, k);
     est_pminus = zeros(2, 2, k);
     est_G = zeros(2, 1, k);
     est_p(:, :, 1) = [std1^2/(1-a^2), std1*std2*rho
         /((1-a^2)*(1-a*b)); std1*std2*rho/((1-a^2)*(1-a
         *b)), (std1^2+std2^2)/((1-b^2))];
     est_x(:, 1) = est_x0;
     est_x(:, 1) = [0.65; 0.78];
```

```
    F = [a,0;a,b];
    Q = [std1^2, std1*std2*rho; std1*std2*rho, std1^2 + std2^2];
    R = stdv^2;
    H = [0,1];
    estimated_z = zeros(1,k);
    estimated_z(1,1) = H*est_x(:,1) + v(1);
    for i = 2:k
        est_xminus(:, i) = F * est_x(:, i-1);
        est_pminus(:, :, i) = F * est_p(:, :, i-1) * F' + Q;
        Gk = est_pminus(:, :, i) * H' / (H * est_pminus(:, :, i) * H' + R);
        est_x(:, i) = est_xminus(:, i) + Gk * (real_output(i) - H * est_xminus(:, i));
        est_p(:, :, i) = est_pminus(:, :, i) - Gk * H * est_pminus(:, :, i);
        est_G(:, :, i) = Gk;
        estimated_z(1,i) = H*est_x(:,i) + v(i);
    end
end
```

2 Question 2

- Explanation of each part located at the end of them.

2.1 part a

I’ve done it by below code and the vector at each feature :

code

```
feature1 = zeros(1,train_num);
feature2 = zeros(1,train_num);
feature3 = zeros(1,train_num);
mean_train = zeros(1,train_num);
for i = 1:train_num
    if i<=9
        field_name = ['ECG0' num2str(i)];
    else
        field_name = ['ECG' num2str(i)];
    end
    mean_train(i) = mean(ecg.(field_name));
    lower_than_m= [];
    higher_than_m = [];
    k = 0;
    l = 0;
    for j = 1:length(ecg.(field_name))
        if ecg.(field_name)(j)<mean_train(i)
            k = k+1;
            lower_than_m(k) = ecg.(field_name)(j);
        else
            l = l+1;
            higher_than_m(l) = ecg.(field_name)(j);
        end
    end
```

```
        feature1(i) = mean(lower_than_m) / min(ecg.(
            field_name));
        feature2(i) = mean(lower_than_m)/var(lower_than_m)
            ;
        feature3(i) = (mean(higher_than_m)-mean(
            lower_than_m))/(max(ecg.(field_name))-min(ecg.(
            field_name)));
end
```

feature1

```
feature1 =

  Columns 1 through 19

    0.3309    0.2829    0.3976    0.3090    0.2790    0.2422    0.3974    0.4048    0.2016    0.3003    0.2058    0.3149    0.2590    0.2603    0.2725    0.2552    0.2319    0.3388    0.1409

  Columns 20 through 38

    0.5242    0.3076    0.2748    0.1875    0.2671    0.1859    0.1964    0.2191    0.1775    0.1661    0.2521    0.1690    0.2510    0.2287    0.1933    0.1720    0.2039    0.2181    0.2100

  Columns 39 through 40

    0.3575    0.2794
```

feature2

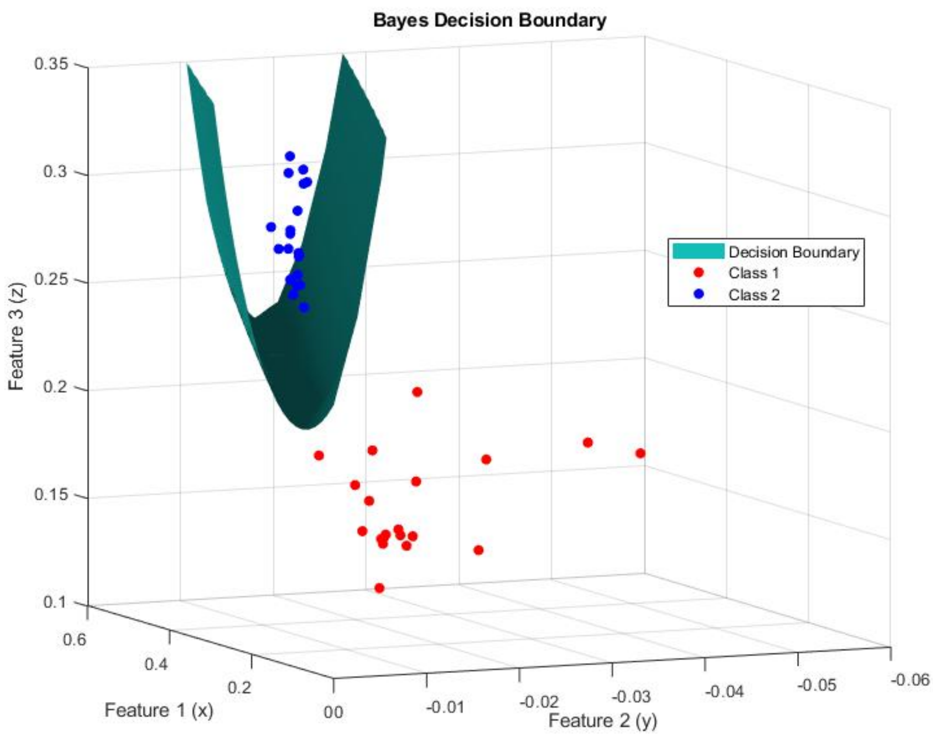
```
feature2 =

  Columns 1 through 19

   -0.0302   -0.0210   -0.0264   -0.0178   -0.0154   -0.0161   -0.0506   -0.0343   -0.0168   -0.0186   -0.0114   -0.0229   -0.0184   -0.0164   -0.0176   -0.0164   -0.0174   -0.0188   -0.0046

  Columns 20 through 38

   -0.0505   -0.0089   -0.0062   -0.0039   -0.0069   -0.0035   -0.0049   -0.0048   -0.0046   -0.0036   -0.0072   -0.0038   -0.0064   -0.0062   -0.0046   -0.0040   -0.0043   -0.0064   -0.0064

  Columns 39 through 40

   -0.0126   -0.0056
```

feature3

```
feature3 =

  Columns 1 through 19

    0.1337    0.1451    0.1627    0.1843    0.1491    0.1482    0.1698    0.1706    0.1464    0.1412    0.1756    0.2098    0.1503    0.1235    0.1473    0.1466    0.1493    0.1589    0.1947

  Columns 20 through 38

    0.1677    0.2889    0.2827    0.2669    0.3183    0.2741    0.2855    0.2863    0.3255    0.2860    0.2666    0.2866    0.3271    0.3030    0.2757    0.2724    0.2942    0.2587    0.3174

  Columns 39 through 40

    0.3068    0.2928
```

2.2 part b

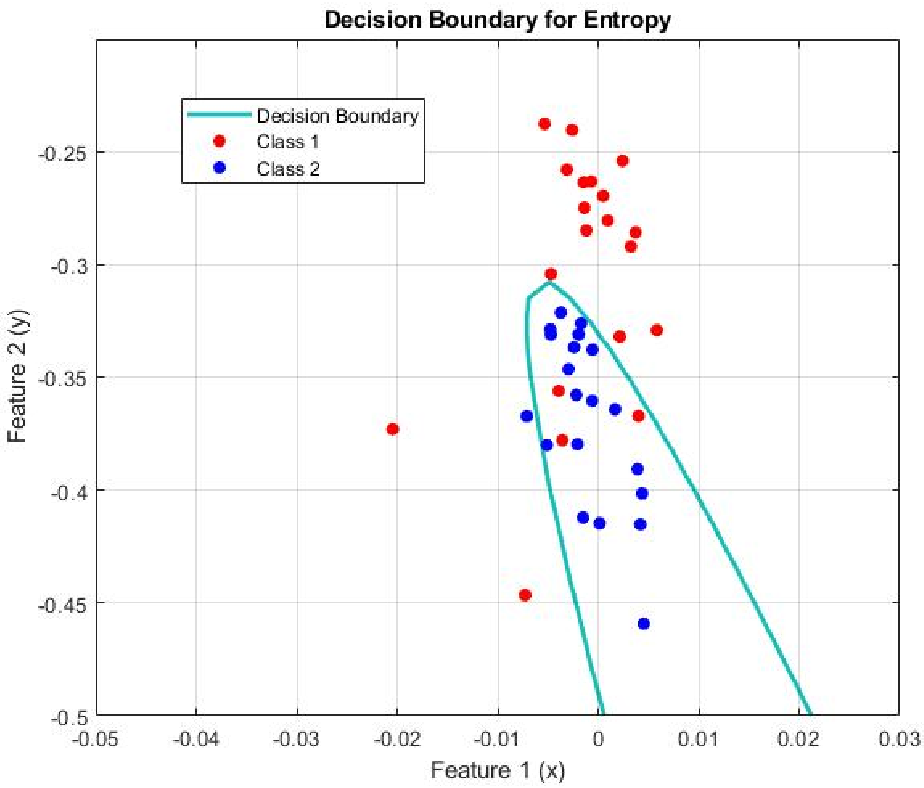
I have done classification by bayes decision boundary and using the equation from handout:



As we can see they are classified correctly.

2.3 part c

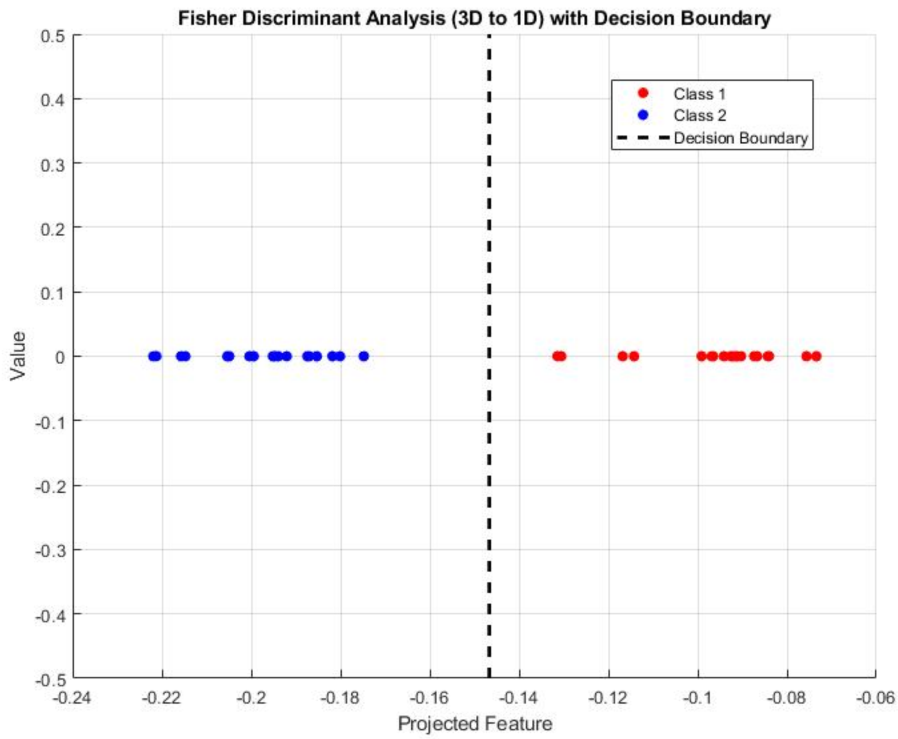
I have done classification by entropy reduction and using the equation from hand-out:



As we can see they are classified correctly but there are some features which wrongly labeled.

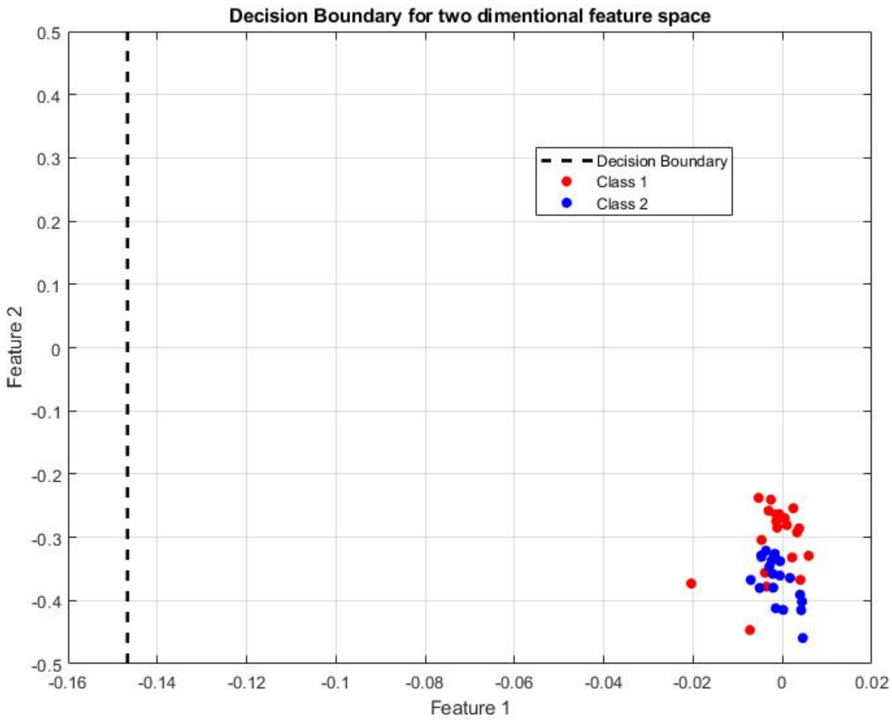
2.4 part d

I have done classification by FLD and using the equation from handout:



As we can see they are classified correctly.

-Now if i classify the 2D feature space we can see they are not classified correctly at all:


Decision Boundary for two dimentional feature space
Decision Boundary
Class 1
Class 2
Feature 2
Feature 1
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.16
-0.14
-0.12
-0.1
-0.08
-0.06
-0.04
-0.02
0
0.02

2.5 part e

-I have done classification by features ,which are described in question, for test dataset and these are the accuracy of them:

```
accuracy by mahlanobis (part b) :
   97.5000

accuracy by Euclidean (part b) :
   97.5000

accuracy by bayes decision boundary (part b) :
   97.5000

-------------------------------------------------------------------
accuracy by mahlanobis (part c) :
   82.5000

accuracy by Euclidean (part c) :
    85

accuracy by bayes decision boundary (part c) :
   92.5000

-------------------------------------------------------------------
accuracy by mahlanobis (part d) :
   97.5000

accuracy by Euclidean (part d) :
   97.5000

accuracy by bayes decision boundary (part d) :
   97.5000

-------------------------------------------------------------------
```

2.6 part f

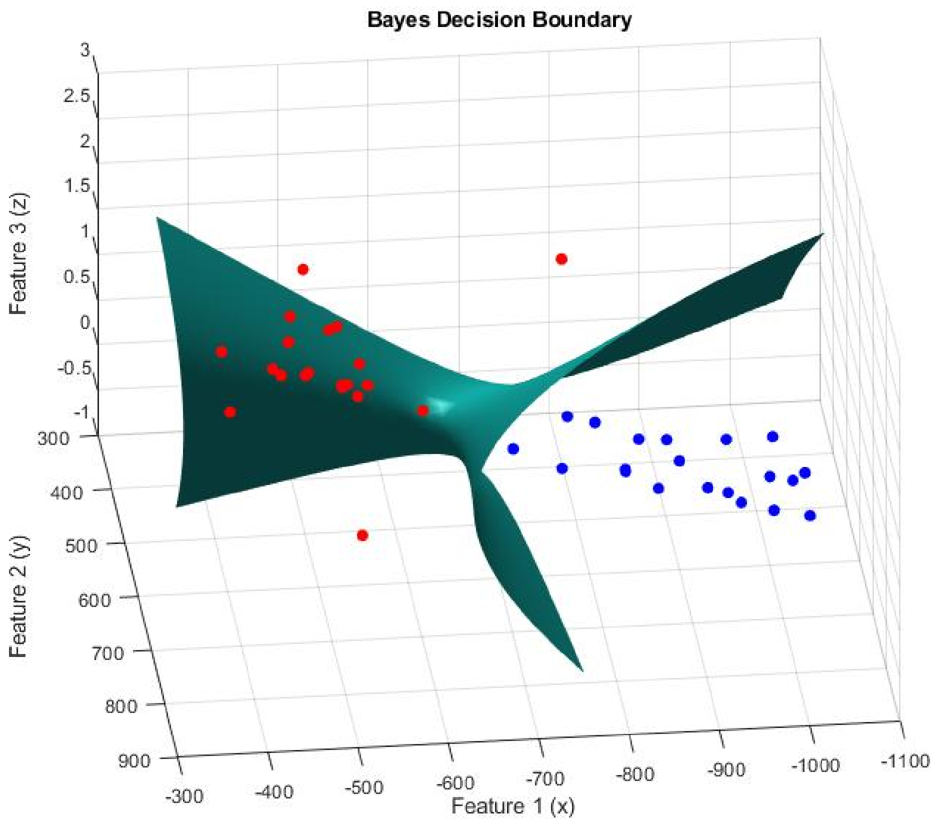
Now I repeat the classification for three new features:

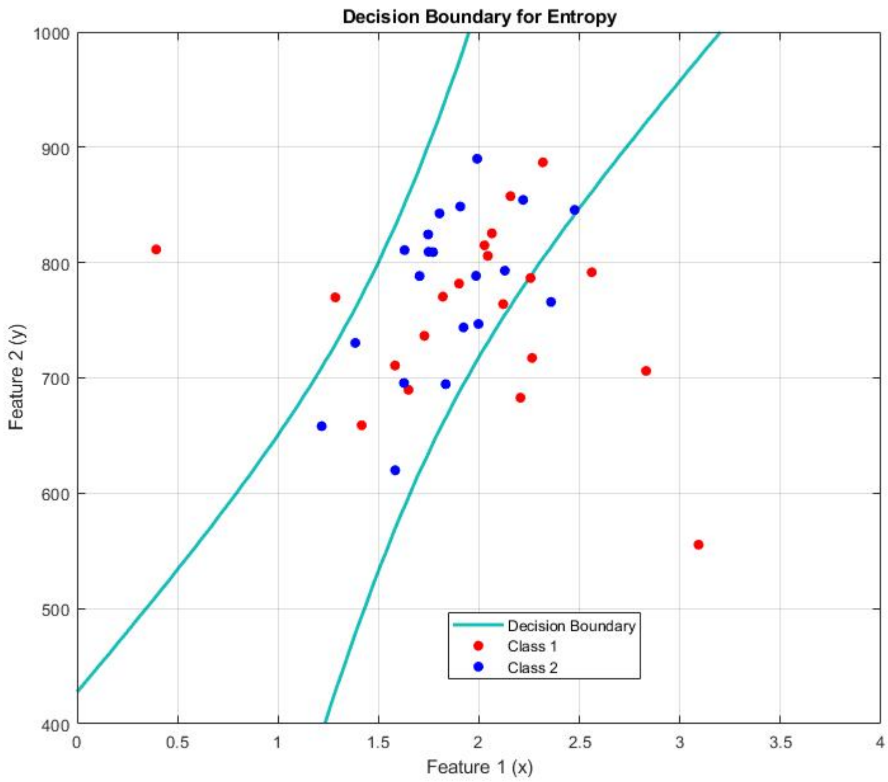
feature1: T amp wave

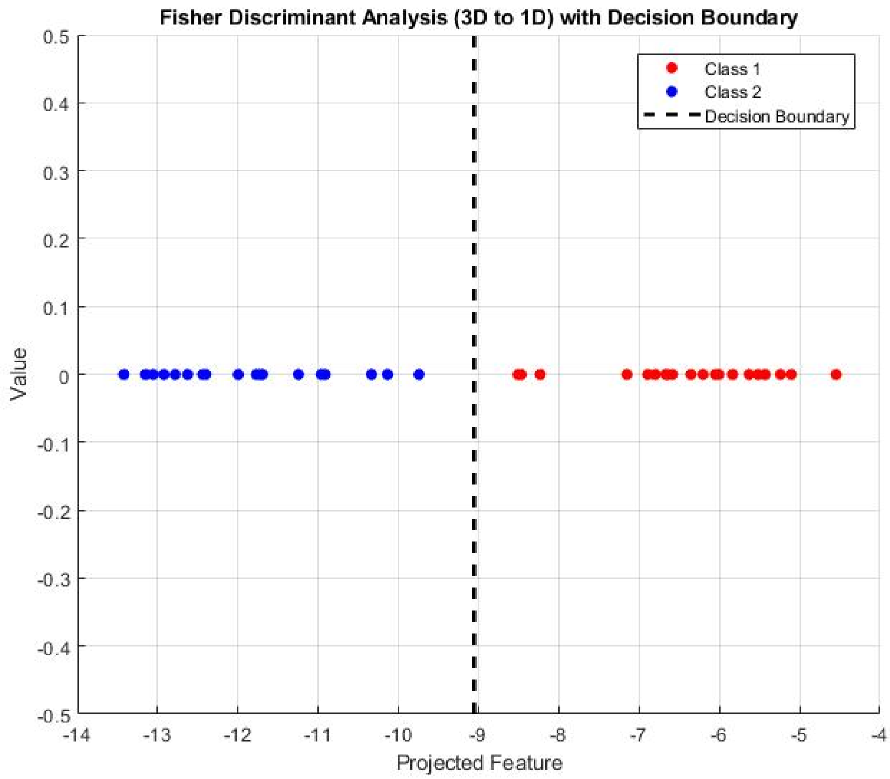
feature1: P amp wave

feature1: std

Now I have repeated all parts ass below :




Decision Boundary for Entropy
1000
900
800
700
600
500
400
0
0.5
1
1.5
2
2.5
3
3.5
4
Feature 2 (y)
Feature 1 (x)
Decision Boundary
Class 1
Class 2


Fisher Discriminant Analysis (3D to 1D) with Decision Boundary
Class 1
Class 2
Decision Boundary
Value
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-14
-13
-12
-11
-10
-9
-8
-7
-6
-5
-4
Projected Feature

```
accuracy by mahlanobis (part b) :
    95

accuracy by Euclidean (part b) :
    95

accuracy by bayes decision boundary (part b) :
   97.5000

-------------------------------------------------------------------
accuracy by mahlanobis (part c) :
   72.5000

accuracy by Euclidean (part c) :
    95

accuracy by bayes decision boundary (part c) :
   82.5000

-------------------------------------------------------------------
accuracy by mahlanobis (part d) :
   100

accuracy by Euclidean (part d) :
   100

accuracy by bayes decision boundary (part d) :
   100

-------------------------------------------------------------------
```

Matlab Code

```
clc; clear
ecg = load('ECG.mat');
train_num = 40;
test_num = 40;
%% part a
feature1 = zeros(1,train_num);
feature2 = zeros(1,train_num);
feature3 = zeros(1,train_num);
```

```
mean_train = zeros(1,train_num);
for i = 1:train_num
    if i<=9
        field_name = ['ECG0' num2str(i)];
    else
        field_name = ['ECG' num2str(i)];
    end
    mean_train(i) = mean(ecg.(field_name));
    lower_than_m= [];
    higher_than_m = [];
    k = 0;
    l = 0;
    for j = 1:length(ecg.(field_name))
        if ecg.(field_name)(j)<mean_train(i)
            k = k+1;
            lower_than_m(k) = ecg.(field_name)(j);
        else
            l = l+1;
            higher_than_m(l) = ecg.(field_name)(j);
        end
    end
    feature1(i) = mean(lower_than_m) / min(ecg.(field_name));
    feature2(i) = mean(lower_than_m)/var(lower_than_m);
    feature3(i) = (mean(higher_than_m)-mean(lower_than_m))/(max(ecg.(field_name))-min(ecg.(field_name)));
end
%% part b
class1_featurematrix = [feature1(1:20)' feature2(1:20)
```

```
' feature3(1:20)'];
mean1_mat = mean(class1_featurematrix);
cov1_mat = cov(class1_featurematrix);
class2_featurematrix = [feature1(21:40)' feature2(21:40)' feature3(21:40)'];
mean2_mat = mean(class2_featurematrix);
cov2_mat = cov(class2_featurematrix);
cov_mat_train = cov([feature1(1:40)' feature2(1:40)' feature3(1:40)']);
x_range = linspace(0, 0.6, 40);
y_range = linspace(0, -0.1, 40);
z_range = linspace(0, 0.35, 40);
[X, Y, Z] = meshgrid(x_range, y_range, z_range);
D1 = zeros(size(X));
D2 = zeros(size(X));
x_flat = X(:);
y_flat = Y(:);
z_flat = Z(:);
for i = 1:length(x_flat)
    beta = [x_flat(i); y_flat(i); z_flat(i)];
    D1(i) = -0.5 * log(det(cov1_mat)) - 0.5 * (beta - mean1_mat')' * inv(cov1_mat) * (beta - mean1_mat');
    D2(i) = -0.5 * log(det(cov2_mat)) - 0.5 * (beta - mean2_mat')' * inv(cov2_mat) * (beta - mean2_mat');
end
D1 = reshape(D1, size(X));
D2 = reshape(D2, size(X));
Difference = D1 - D2;
figure;
```

```
isosurface(X, Y, Z, Difference, 0);
hold on;
scatter3(class1_featurematrix(:,1), class1_featurematrix(:,2), class1_featurematrix(:,3), 'r', 'filled');
hold on;
scatter3(class2_featurematrix(:,1), class2_featurematrix(:,2), class2_featurematrix(:,3), 'b', 'filled');
xlabel('Feature 1 (x)');
ylabel('Feature 2 (y)');
zlabel('Feature 3 (z)');
title('Bayes Decision Boundary');
grid on;
legend({'Decision Boundary', 'Class 1', 'Class 2'}, 'Location', 'best')
%% part c
[evc1, eva1] = eig(cov1_mat);
[evc2, eva2] = eig(cov2_mat);
[evc, eva] = eig(cov_mat_train);
u = evc(:,1:2);
u1 = evc1(:,1:2);  % Assuming these are the principal components
u2 = evc2(:,1:2);
class1_featurematrix_entropy = class1_featurematrix * u;
class2_featurematrix_entropy = class2_featurematrix * u;
mean1_mat_entropy = mean(class1_featurematrix_entropy)';
mean2_mat_entropy = mean(class2_featurematrix_entropy)
```

```
';
cov1_mat_entropy = cov(class1_featurematrix_entropy);
cov2_mat_entropy = cov(class2_featurematrix_entropy);
x_range = linspace(-0.05, 0.03, 40);
y_range = linspace(-0.5, -0.2, 40);
[Xent, Yent] = meshgrid(x_range, y_range);
D1 = zeros(size(Xent));
D2 = zeros(size(Xent));
x_flat = Xent(:);
y_flat = Yent(:);
for i = 1:length(x_flat)
    beta = [x_flat(i); y_flat(i)];
    D1(i) = -0.5 * log(det(cov1_mat_entropy)) - 0.5 * (beta - mean1_mat_entropy)' * inv(cov1_mat_entropy) * (beta - mean1_mat_entropy);
    D2(i) = -0.5 * log(det(cov2_mat_entropy)) - 0.5 * (beta - mean2_mat_entropy)' * inv(cov2_mat_entropy) * (beta - mean2_mat_entropy);
end
D1 = reshape(D1, size(Xent));
D2 = reshape(D2, size(Xent));
Difference_ent = D1 - D2;
figure;
contour(Xent, Yent, Difference_ent, [0, 0], 'LineWidth', 2);
hold on;
scatter(class1_featurematrix_entropy(:, 1), class1_featurematrix_entropy(:, 2), 'r', 'filled');
hold on;
scatter(class2_featurematrix_entropy(:, 1), class2_featurematrix_entropy(:, 2), 'b', 'filled');
```

```
xlabel('Feature 1 (x)');
ylabel('Feature 2 (y)');
title('Decision Boundary for Entropy');
grid on;
legend({'Decision Boundary', 'Class 1', 'Class 2'}, 'Location', 'best');
%% part d

class1_featurematrix = [feature1(1:20)' feature2(1:20)' feature3(1:20)'];
class2_featurematrix = [feature1(21:40)' feature2(21:40)' feature3(21:40)'];
mean1 = mean(class1_featurematrix)';
mean2 = mean(class2_featurematrix)';
S_W = cov(class1_featurematrix) + cov(class2_featurematrix);
mean_diff = mean1 - mean2;
S_B = mean_diff * mean_diff';
[V, D] = eig(S_W \ S_B);
[~, idx] = max(diag(D));
w = V(:, idx);
projected_class1 = class1_featurematrix * w;
projected_class2 = class2_featurematrix * w;
mean_proj1 = mean(projected_class1);
mean_proj2 = mean(projected_class2);
var_proj1 = var(projected_class1);
var_proj2 = var(projected_class2);
pooled_variance = ((length(projected_class1) - 1) * var_proj1 + (length(projected_class2) - 1) * var_proj2) / (length(projected_class1) + length(projected_class2) - 2);
```

```
decision_boundary = (mean_proj1 + mean_proj2) / 2;
figure;
hold on;
scatter(projected_class1, zeros(size(projected_class1)), 'r', 'filled');
scatter(projected_class2, zeros(size(projected_class2)), 'b', 'filled');
plot([decision_boundary decision_boundary], [-0.5 0.5], 'k--', 'LineWidth', 2);
xlabel('Projected Feature');
ylabel('Value');
title('Fisher Discriminant Analysis (3D to 1D) with Decision Boundary');
legend({'Class 1', 'Class 2', 'Decision Boundary'}, 'Location', 'best');
grid on;
figure;
plot([decision_boundary decision_boundary], [-0.5 0.5], 'k--', 'LineWidth', 2);
hold on;
scatter(class1_featurematrix_entropy(:, 1), class1_featurematrix_entropy(:, 2), 'r', 'filled');
hold on;
scatter(class2_featurematrix_entropy(:, 1), class2_featurematrix_entropy(:, 2), 'b', 'filled');
xlabel('Feature 1');
ylabel('Feature 2');
title('Decision Boundary for two dimentional feature space');
grid on;
legend({'Decision Boundary', 'Class 1', 'Class 2'}, '
```

```
    Location', 'best')
%% part e
feature1test = zeros(1,test_num);
feature2test  = zeros(1,test_num);
feature3test  = zeros(1,test_num);
mean_test = zeros(1,test_num);
for i = 41:80
    field_name = ['ECG' num2str(i)];
    mean_test(i-40) = mean(ecg.(field_name));
    lower_than_mtest= [];
    higher_than_mtest = [];
    k = 0;
    l = 0;
    for j = 1:length(ecg.(field_name))
        if ecg.(field_name)(j)<mean_test(i-40)
            k = k+1;
            lower_than_mtest(k) = ecg.(field_name)(j);
        else
            l = l+1;
            higher_than_mtest(l) = ecg.(field_name)(j);
        end
    end
    feature1test (i-40) = mean(lower_than_mtest) / min
       (ecg.(field_name));
    feature2test (i-40) = mean(lower_than_mtest)/var(
       lower_than_mtest);
    feature3test (i-40) = (mean(higher_than_mtest)-
       mean(lower_than_mtest))/(max(ecg.(field_name))-
       min(ecg.(field_name)));
end
class1test_featurematrix = [feature1test(1:20)'
```

```
feature2test(1:20)' feature3test(1:20)'];
class2test_featurematrix = [feature1test(21:40)' feature2test(21:40)' feature3test(21:40)'];
cov1_mat_test = cov(class1test_featurematrix);
cov2_mat_test = cov(class2test_featurematrix);
cov_mat_test = cov([feature1test(1:40)' feature2test(1:40)' feature3test(1:40)']);
% for part b
labels_class1_mah = zeros(1,20);
labels_class2_mah = zeros(1,20);
labels_class1_o = zeros(1,20);
labels_class2_o = zeros(1,20);
labels_class1_bayes = zeros(1,20);
labels_class2_bayes = zeros(1,20);
temp_o = 0;
temp_mah = 0;
temp_bayes = 0;
for i = 1:20
    d1_mah = (class1test_featurematrix(i,1:3)-mean1_mat)*inv(cov1_mat)*(class1test_featurematrix(i,1:3)-mean1_mat)';
    d2_mah = (class1test_featurematrix(i,1:3)-mean2_mat)*inv(cov2_mat)*(class1test_featurematrix(i,1:3)-mean2_mat)';
    if d1_mah>d2_mah
        labels_class1_mah(1,i) = 2;
        temp_mah = temp_mah +1;
    else
        labels_class1_mah(1,i) = 1;
    end
    d1_mah = (class2test_featurematrix(i,1:3)-
```

```
        mean1_mat)*inv(cov1_mat)*(
        class2test_featurematrix(i,1:3)-mean1_mat)';
    d2_mah = (class2test_featurematrix(i,1:3)-
        mean2_mat)*inv(cov2_mat)*(
        class2test_featurematrix(i,1:3)-mean2_mat)';
    if d1_mah>d2_mah
        labels_class2_mah(1,i) = 2;
    else
        labels_class2_mah(1,i) = 1;
        temp_mah = temp_mah +1;
    end
end
for i = 1:20
    d1_o = (class1test_featurematrix(i,1:3)-mean1_mat)
        *(class1test_featurematrix(i,1:3)-mean1_mat)';
    d2_o = (class1test_featurematrix(i,1:3)-mean2_mat)
        *(class1test_featurematrix(i,1:3)-mean2_mat)';
    if d1_o>d2_o
        labels_class1_o(1,i) = 2;
        temp_o = temp_o +1;
    else
        labels_class1_o(1,i) = 1;
    end
    d1_o = (class2test_featurematrix(i,1:3)-mean1_mat)
        *(class2test_featurematrix(i,1:3)-mean1_mat)';
    d2_o = (class2test_featurematrix(i,1:3)-mean2_mat)
        *(class2test_featurematrix(i,1:3)-mean2_mat)';
    if d1_o>d2_o
        labels_class2_o(1,i) = 2;
    else
        labels_class2_o(1,i) = 1;
```

```
            temp_o = temp_o +1;
        end
end
for i = 1:20
        d1_o = +0.5*log(det(cov1_mat))+0.5*(
            class1test_featurematrix(i,1:3)-mean1_mat)*inv(
            cov1_mat)*(class1test_featurematrix(i,1:3)-
            mean1_mat)';
        d2_o = +0.5*log(det(cov2_mat))+0.5*(
            class1test_featurematrix(i,1:3)-mean2_mat)*inv(
            cov2_mat)*(class1test_featurematrix(i,1:3)-
            mean2_mat)';
        if d1_o>d2_o
            labels_class1_bayes(1,i) = 2;
            temp_bayes = temp_bayes +1;
        else
            labels_class1_bayes(1,i) = 1;
        end
        d1_o = +0.5*log(det(cov1_mat))+0.5*(
            class2test_featurematrix(i,1:3)-mean1_mat)*inv(
            cov1_mat)*(class2test_featurematrix(i,1:3)-
            mean1_mat)';
        d2_o = +0.5*log(det(cov2_mat))+0.5*(
            class2test_featurematrix(i,1:3)-mean2_mat)*inv(
            cov2_mat)*(class2test_featurematrix(i,1:3)-
            mean2_mat)';
        if d1_o>d2_o
            labels_class2_bayes(1,i) = 2;
        else
            labels_class2_bayes(1,i) = 1;
            temp_bayes = temp_bayes +1;
```

```
    end
end
disp('accuracy by mahlanobis (part b) :')
disp((40-temp_o)/40*100)
disp('accuracy by Euclidean (part b) :')
disp((40-temp_mah)/40*100)
disp('accuracy by bayes decision boundary (part b) :')
disp((40-temp_bayes)/40*100)
disp('
    ---------------------------------------------------------------------------
    ')
% for part c
[evc1_test_b, eva1_test_b] = eig(cov1_mat_test);
[evc2_test_b, eva2_test_b] = eig(cov2_mat_test);
[evc_test_b, eva_test_b] = eig(cov_mat_test);
u1_test_b = evc1_test_b(:,1:2);
u2_test_b = evc2_test_b(:,1:2);
u_test_b = evc_test_b(:,1:2);
class1_featurematrix_test_entropy =
    class1_featurematrix * u_test_b;
class2_featurematrix_test_entropy =
    class2_featurematrix * u_test_b;
labels_class1_mah_ent = zeros(1,20);
labels_class2_mah_ent = zeros(1,20);
labels_class1_o_ent = zeros(1,20);
labels_class2_o_ent = zeros(1,20);
labels_class1_bayes_ent = zeros(1,20);
labels_class2_bayes_ent = zeros(1,20);
temp_o = 0;
temp_mah = 0;
temp_bayes = 0;
```

```
for i = 1:20
    d1_mah = (class1_featurematrix_test_entropy(i,1:2)
       -mean1_mat_entropy)*inv(cov1_mat_entropy)*(
       class1_featurematrix_test_entropy(i,1:2)-
       mean1_mat_entropy)';
    d2_mah = (class1_featurematrix_test_entropy(i,1:2)
       -mean2_mat_entropy)*inv(cov2_mat_entropy)*(
       class1_featurematrix_test_entropy(i,1:2)-
       mean2_mat_entropy)';
    if d1_mah>d2_mah
        labels_class1_mah_ent(1,i) = 2;
        temp_mah = temp_mah +1;
    else
        labels_class2_mah_ent(1,i) = 1;
    end
    d1_mah = (class2_featurematrix_test_entropy(i,1:2)
       -mean1_mat_entropy)*inv(cov1_mat_entropy)*(
       class2_featurematrix_test_entropy(i,1:2)-
       mean1_mat_entropy)';
    d2_mah = (class2_featurematrix_test_entropy(i,1:2)
       -mean2_mat_entropy)*inv(cov2_mat_entropy)*(
       class2_featurematrix_test_entropy(i,1:2)-
       mean2_mat_entropy)';
    if d1_mah>d2_mah
        labels_class1_mah_ent(1,i) = 2;
    else
        labels_class2_mah_ent(1,i) = 1;
        temp_mah = temp_mah +1;
    end
end
for i = 1:20
```

```
        d1_o = (class1_featurematrix_test_entropy(i,1:2)-
            mean1_mat_entropy)*(
            class1_featurematrix_test_entropy(i,1:2)-
            mean1_mat_entropy)';
        d2_o = (class1_featurematrix_test_entropy(i,1:2)-
            mean2_mat_entropy)*(
            class1_featurematrix_test_entropy(i,1:2)-
            mean2_mat_entropy)';
        if d1_o>d2_o
            labels_class1_o_ent(1,i) = 2;
            temp_o = temp_o +1;
        else
            labels_class1_o_ent(1,i) = 1;
        end
        d1_o = (class2_featurematrix_test_entropy(i,1:2)-
            mean1_mat_entropy)*(
            class2_featurematrix_test_entropy(i,1:2)-
            mean1_mat_entropy)';
        d2_o = (class2_featurematrix_test_entropy(i,1:2)-
            mean2_mat_entropy)*(
            class2_featurematrix_test_entropy(i,1:2)-
            mean2_mat_entropy)';
        if d1_o>d2_o
            labels_class2_o_ent(1,i) = 2;
        else
            labels_class2_o_ent(1,i) = 1;
            temp_o = temp_o +1;
        end
    end
    for i = 1:20
        d1_mah = 0.5*log(det(cov1_mat_entropy))+0.5*(
```

```
    class1_featurematrix_test_entropy(i,1:2)-
    mean1_mat_entropy)*inv(cov1_mat_entropy)*(
    class1_featurematrix_test_entropy(i,1:2)-
    mean1_mat_entropy)';
d2_mah = 0.5*log(det(cov2_mat_entropy))+0.5*(
    class1_featurematrix_test_entropy(i,1:2)-
    mean2_mat_entropy)*inv(cov2_mat_entropy)*(
    class1_featurematrix_test_entropy(i,1:2)-
    mean2_mat_entropy)';
if d1_mah>d2_mah
     labels_class1_bayes_ent(1,i) = 2;
     temp_bayes = temp_bayes +1;
else
     labels_class1_bayes_ent(1,i) = 1;
end
d1_mah = 0.5*log(det(cov1_mat_entropy))+0.5*(
    class2_featurematrix_test_entropy(i,1:2)-
    mean1_mat_entropy)*inv(cov1_mat_entropy)*(
    class2_featurematrix_test_entropy(i,1:2)-
    mean1_mat_entropy)';
d2_mah = 0.5*log(det(cov2_mat_entropy))+0.5*(
    class2_featurematrix_test_entropy(i,1:2)-
    mean2_mat_entropy)*inv(cov2_mat_entropy)*(
    class2_featurematrix_test_entropy(i,1:2)-
    mean2_mat_entropy)';
if d1_mah>d2_mah
     labels_class2_bayes_ent(1,i) = 2;
else
     labels_class2_mah_ent(1,i) = 1;
     temp_bayes = temp_bayes +1;
end
```

```
end
disp('accuracy by mahlanobis (part c) :')
disp((40-temp_mah)/40*100)
disp('accuracy by Euclidean (part c) :')
disp((40-temp_o)/40*100)
disp('accuracy by bayes decision boundary (part c) :')
disp((40-temp_bayes)/40*100)
disp('------------------------------------------------------------------------------------------------
')
% for part d
mean1_test_fisher = mean(class1test_featurematrix)';
mean2_test_fisher = mean(class2test_featurematrix)';
S_W_test = cov(class1test_featurematrix) + cov(
    class2test_featurematrix);
mean_diff = mean1_test_fisher - mean2_test_fisher;
S_B = mean_diff * mean_diff';
[V_test, D_test] = eig(S_W_test \ S_B);
[~, idx_test] = max(diag(D_test));
w_test = V_test(:, idx_test);
projected_class1_test = class1test_featurematrix *
    w_test;
projected_class2_test = class2test_featurematrix *
    w_test;
mean_proj1_test = mean(projected_class1_test);
mean_proj2_test = mean(projected_class2_test);
var_proj1_test = var(projected_class1_test);
var_proj2_test = var(projected_class2_test);
labels_class1_mah_fisher = zeros(1,20);
labels_class2_mah_fisher = zeros(1,20);
labels_class1_o_fisher = zeros(1,20);
```

```
labels_class2_o_fisher = zeros(1,20);
labels_class1_o_bayes = zeros(1,20);
labels_class2_o_bayes = zeros(1,20);
temp_o = 0;
temp_mah = 0;
temp_bayes = 0;
for i = 1:20
    d1_mah = (projected_class1_test(i)-mean_proj1_test
        )*inv(var_proj1_test)*(projected_class1_test(i)
        -mean_proj1_test)';
    d2_mah = (projected_class1_test(i)-mean_proj2_test
        )*inv(var_proj2_test)*(projected_class1_test(i)
        -mean_proj2_test)';
    if d1_mah>d2_mah
        labels_class1_mah_fisher(1,i) = 2;
        temp_mah = temp_mah +1;
    else
        labels_class2_mah_fisher(1,i) = 1;
    end
    d1_mah = (projected_class2_test(i)-mean_proj1_test
        )*inv(var_proj1_test)*(projected_class2_test(i)
        -mean_proj1_test)';
    d2_mah = (projected_class2_test(i)-mean_proj2_test
        )*inv(var_proj2_test)*(projected_class2_test(i)
        -mean_proj2_test)';
    if d1_mah>d2_mah
        labels_class1_mah_fisher(1,i) = 2;
    else
        labels_class2_mah_fisher(1,i) = 1;
        temp_mah = temp_mah +1;
    end
```

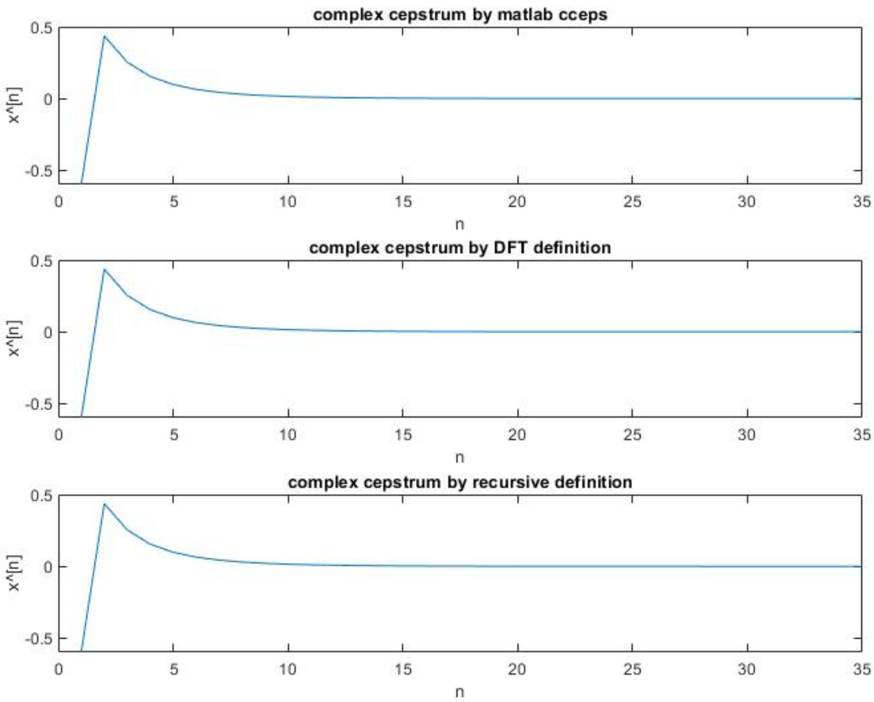
```
end
for i = 1:20
    d1_o = (projected_class1_test(i)-mean_proj1_test)
        *(projected_class1_test(i)-mean_proj1_test)';
    d2_o = (projected_class1_test(i)-mean_proj2_test)
        *(projected_class1_test(i)-mean_proj2_test)';
    if d1_o>d2_o
        labels_class1_o_fisher(1,i) = 2;
        temp_o = temp_o +1;
    else
        labels_class1_o_fisher(1,i) = 1;
    end
    d1_o = (projected_class2_test(i)-mean_proj1_test)
        *(projected_class2_test(i)-mean_proj1_test)';
    d2_o = (projected_class2_test(i)-mean_proj2_test)
        *(projected_class2_test(i)-mean_proj2_test)';
    if d1_o>d2_o
        labels_class2_o_fisher(1,i) = 2;
    else
        labels_class2_o_fisher(1,i) = 1;
        temp_o = temp_o +1;
    end
end
for i = 1:20
    d1_mah = 0.5*log(det(var_proj1_test))+0.5*(
        projected_class1_test(i)-mean_proj1_test)*inv(
        var_proj1_test)*(projected_class1_test(i)-
        mean_proj1_test)';
    d2_mah = 0.5*log(det(var_proj2_test))+0.5*(
        projected_class1_test(i)-mean_proj2_test)*inv(
        var_proj2_test)*(projected_class1_test(i)-
```

```
        mean_proj2_test)';
    if d1_mah>d2_mah
        labels_class1_mah_fisher(1,i) = 2;
        temp_bayes = temp_bayes +1;
    else
        labels_class2_mah_fisher(1,i) = 1;
    end
    d1_mah = 0.5*log(det(var_proj1_test))+0.5*(
        projected_class2_test(i)-mean_proj1_test)*inv(
        var_proj1_test)*(projected_class2_test(i)-
        mean_proj1_test)';
    d2_mah = 0.5*log(det(var_proj2_test))+0.5*(
        projected_class2_test(i)-mean_proj2_test)*inv(
        var_proj2_test)*(projected_class2_test(i)-
        mean_proj2_test)';
    if d1_mah>d2_mah
        labels_class1_mah_fisher(1,i) = 2;
    else
        labels_class2_mah_fisher(1,i) = 1;
        temp_bayes = temp_bayes +1;
    end
end
disp('accuracy by mahlanobis (part d) :')
disp((40-temp_o)/40*100)
disp('accuracy by Euclidean (part d) :')
disp((40-temp_mah)/40*100)
disp('accuracy by bayes decision boundary (part d) :')
disp((40-temp_bayes)/40*100)
disp('
    ----------------------------------------------------------------------------------------------------
    ')
```

3 Question 3

I have used an ECG dataset which contain healthy controls, cardiac patients and Myocardial infarction patients which these diseases are exactly Heart failure as question said. Also explanation of each part located at the end of them.

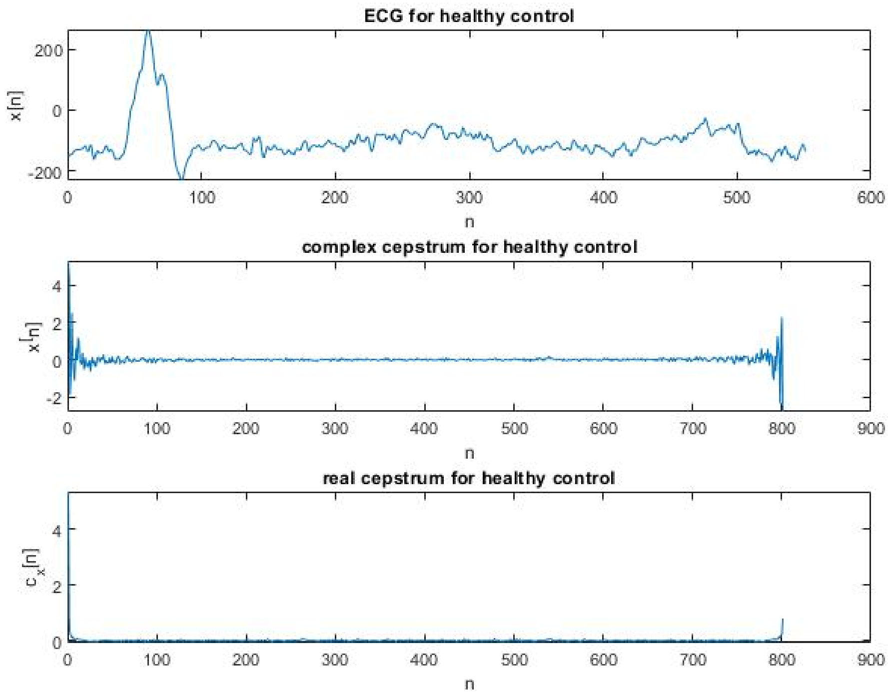
3.1 part a



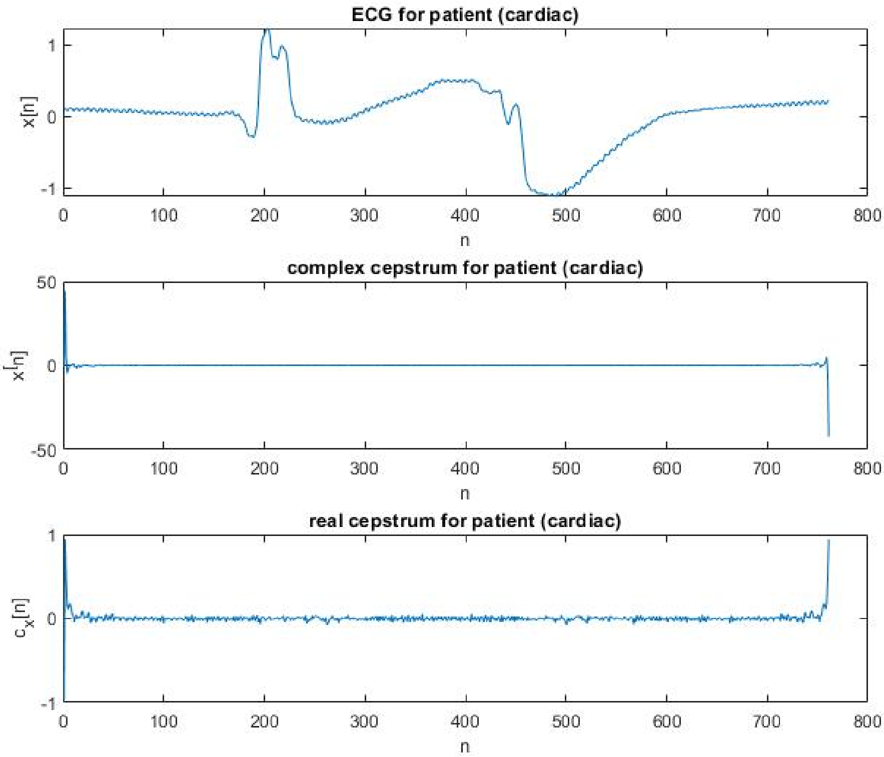
*** Explanation:** As we can see there is no difference between three methods and they have same answers.

3.2 part b

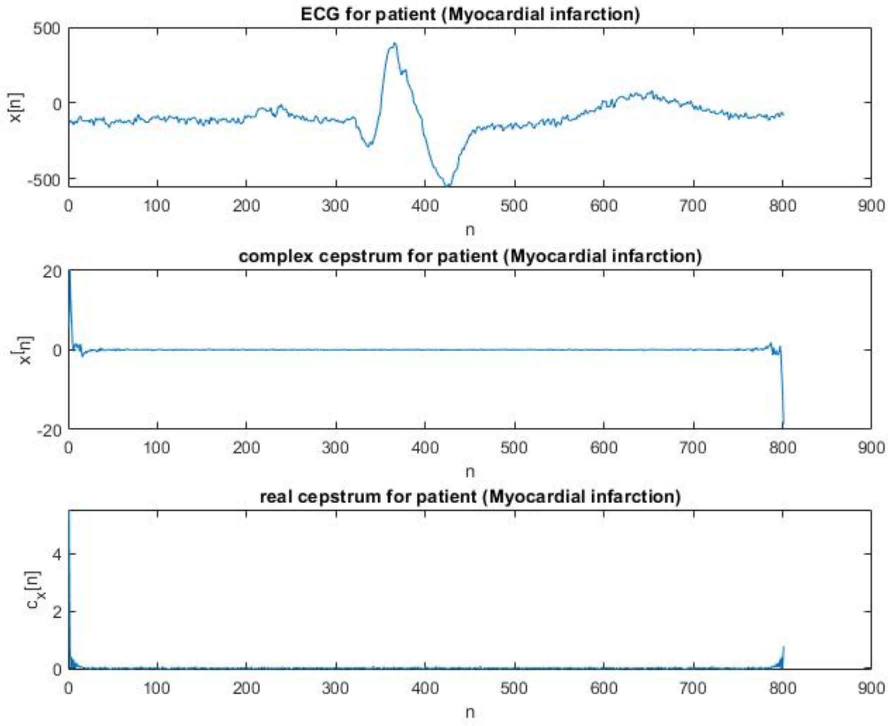
- For healthy person :



- For cardiac patient :



- For Myocardial infarction patient :



*** Explanation:**

-Distinct Features:

The differences in the complex and real cepstrum plots suggest that there are distinct features that can be used for classification. The cardiac and myocardial infarction patients show more pronounced peaks and variations compared to the healthy control.

-Cepstrum Analysis:

Cepstrum analysis captures the periodicity and echo-like features in the signals, which are reflective of the underlying physiological conditions. For the cardiac patient and myocardial infarction patient, the higher cepstrum values and distinct patterns can serve as markers for classification.

-Healthy Control:

The healthy control shows a relatively stable and less varying cepstrum, making

it distinguishable from the other two cases.
* Finally given the distinct differences in the cepstrum plots for the three cases, it is plausible to classify patients based on cepstrum analysis. The significant features and variations in the complex and real cepstrum values for cardiac patients and myocardial infarction patients compared to healthy controls provide a basis for differentiation. However, the exact effectiveness and accuracy of such classification would depend on the implementation of suitable machine learning algorithms and the quality of the feature extraction process.

Matlab Code

```
%% part a
length = 35;
n = 1:length;
x(n) = 0.8.^n.*heaviside(n)-0.5.*0.8.^(n-1).*heaviside
    (n-1);
fft_x = fft(x);
complex_cepstrum_recursive = [];
complex_cepstrum_definition = ifft(log(fft_x));
complex_cepstrum = cceps(x);
complex_cepstrum_recursive(1) = log(x(1));
for i=2:length
    a = 0;
    for j=1:i-1
        a = a + (j-1)/(i-1)*complex_cepstrum_recursive
            (j)*x(i-(j-1))/x(1);
    end
   complex_cepstrum_recursive(i) = x(i)/x(1) - a;
end
figure;
subplot(3,1,1)
plot(complex_cepstrum)
```

```
title('complex cepstrum by matlab cceps')
ylabel('x\^[n]')
xlabel('n')
subplot(3,1,2)
plot(complex_cepstrum_definition )
title('complex cepstrum by DFT definition')
ylabel('x\^[n]')
xlabel('n')
subplot(3,1,3)
plot(complex_cepstrum_recursive)
title('complex cepstrum by recursive definition')
ylabel('x\^[n]')
xlabel('n')
%% part b
% disease : cardiac
ecg_p1 = readtable("207-try.csv");
% disease : Myocardial infarction
ecg_p2 = load('s0001.mat');
% healthy control
ecg_h = load('s0285.mat');
figure;
subplot(3,1,1)
plot(ecg_p1.heart(640:1400))
title('ECG for patient (cardiac)')
ylabel('x[n]')
xlabel('n')
subplot(3,1,2)
plot(cceps(ecg_p1.heart(640:1400)))
title('complex cepstrum for patient (cardiac)')
ylabel('x^[n]')
xlabel('n')
```

```
subplot(3,1,3)
plot(rceps(ecg_p1.heart(640:1400)))
title('real cepstrum for patient (cardiac)')
ylabel('c_x[n]')
xlabel('n')
figure;
subplot(3,1,1)
plot(ecg_p2.val(1,1750:2550))
title('ECG for patient (Myocardial infarction)')
ylabel('x[n]')
xlabel('n')
subplot(3,1,2)
plot(cceps(ecg_p2.val(1,1750:2550)))
title('complex cepstrum for patient (Myocardial
    infarction)')
ylabel('x^[n]')
xlabel('n')
subplot(3,1,3)
plot(rceps(ecg_p2.val(1,1750:2550)))
title('real cepstrum for patient (Myocardial
    infarction)')
ylabel('c_x[n]')
xlabel('n')
figure;
subplot(3,1,1)
plot(ecg_h.val(1,1750:550+1750))
title('ECG for healthy control')
ylabel('x[n]')
xlabel('n')
subplot(3,1,2)
plot(cceps(ecg_h.val(1,1750:2550)))
```

```
title('complex cepstrum for healthy control')
ylabel('x^[n]')
xlabel('n')
subplot(3,1,3)
plot(rceps(ecg_h.val(1,1750:2550)))
title('real cepstrum for healthy control')
ylabel('c_x[n]')
xlabel('n')
```

4 Question 4

First 3 parts are same as the previous homework and as Dr.Shamsollahi said I avoid from repeating them.

4.1 part d

Article: ”Kalman Filtering for Improved Motion Artifact Reduction in Wearable ECG Monitors”
Summary: Published in IEEE Transactions on Biomedical Engineering in 2024, this article explored the application of Kalman filtering to reduce motion artifacts in wearable ECG monitors. The researchers developed a Kalman filter-based approach to separate true cardiac signals from motion-induced noise. The filtered signals showed significant improvements in clarity, making it easier to detect and monitor cardiac events accurately. This study emphasized the Kalman filter's role in enhancing the reliability of wearable health monitoring devices.

4.2 part e

Article: ”Fuzzy Logic-Based Decision Support System for Early Diagnosis of Alzheimer's Disease”
Summary: The article, published in Expert Systems with Applications in 2023, presented a fuzzy logic-based decision support system designed for the early diagnosis of Alzheimer's disease. The system integrated various clinical and cognitive parameters to evaluate the likelihood of Alzheimer's in patients. Using fuzzy logic, the system handled uncertainties and provided interpretable diagnostic results. The study found that the fuzzy logic approach enhanced diagnostic accuracy and could serve as a valuable tool in clinical settings.

4.3 part f

Article: ”Feature Extraction Using Cepstrum Analysis for Automated Speech Disorder Detection”

Summary: In a 2023 study published in Speech Communication, researchers utilized cepstrum analysis for feature extraction in automated detection of speech disorders. They analyzed the cepstral coefficients of speech recordings from patients with various speech impairments. These features were used to train a neural network classifier, which achieved high accuracy in distinguishing between different types of speech disorders. The findings highlighted the efficacy of cepstrum analysis in capturing distinctive speech characteristics, offering a valuable tool for early diagnosis and intervention.

4.4 part g

Article: ”Hidden Markov Models for Real-Time Sign Language Recognition”
Summary: The 2024 article in Pattern Recognition focused on using hidden Markov models (HMMs) for real-time sign language recognition. The researchers developed an HMM-based system that could accurately interpret sign language gestures from video data. The system was tested with a diverse set of sign language gestures, demonstrating high recognition accuracy and robustness to variations in signing speed and style. This study showcased the potential of HMMs in creating effective communication tools for the hearing impaired.