

Part 2

Report

Biological Signal Processing

Ardalan Gerami 99102112

6/26/2024



Contents

1	Question 1	3
1.1	part a,d,e	3
1.2	part b,d,e	4
1.3	part c,d,e	6
1.4	Explanation	7
2	Question 2	17
2.1	part a	17
2.2	part b	19
2.3	part c	20
2.4	part d	21
2.5	part e	23
2.6	part f	24
3	Question 3	46
3.1	part a	46
3.2	part b	47
4	Question 4	54
4.1	part d	54
4.2	part e	54
4.3	part f	54
4.4	part g	55

1 Question 1

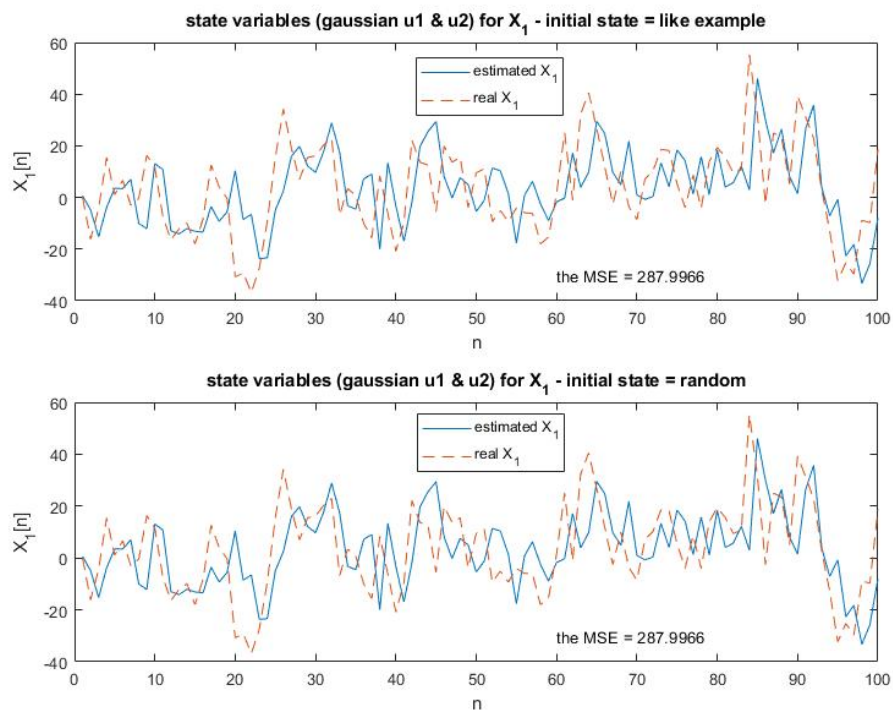
-Above all I should mention that I've answered the questions of part d and e in earlier three parts.

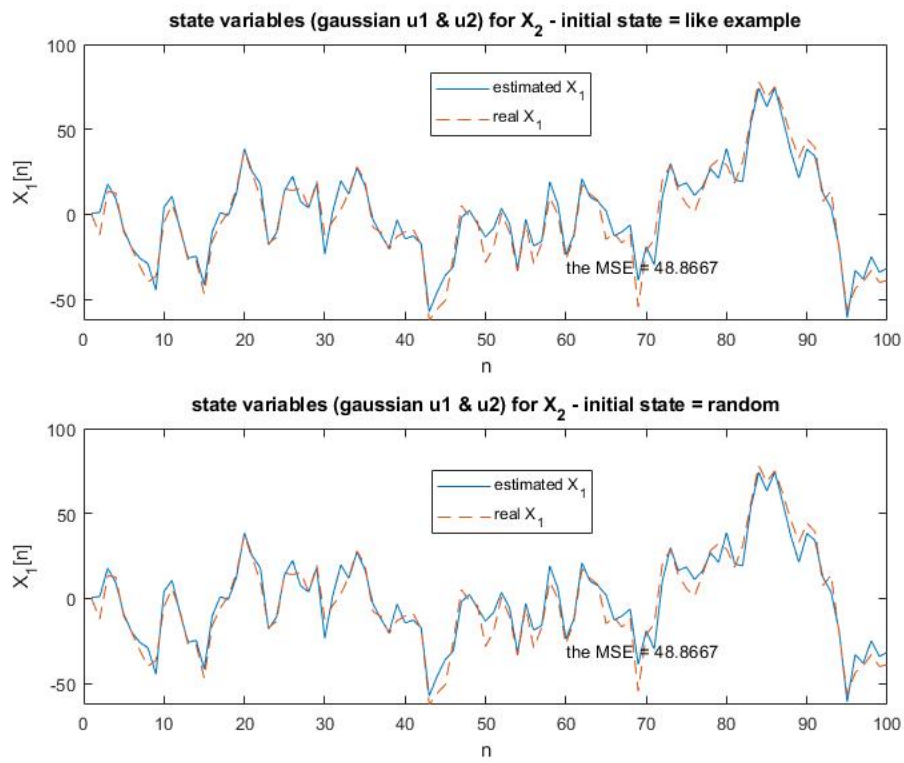
-Also the evaluation criteria is shown on the plot.

-At the end of the question I've explained the answers.

1.1 part a,d,e

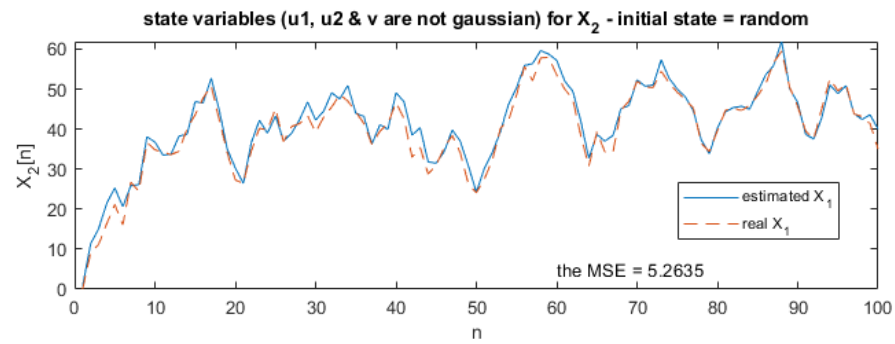
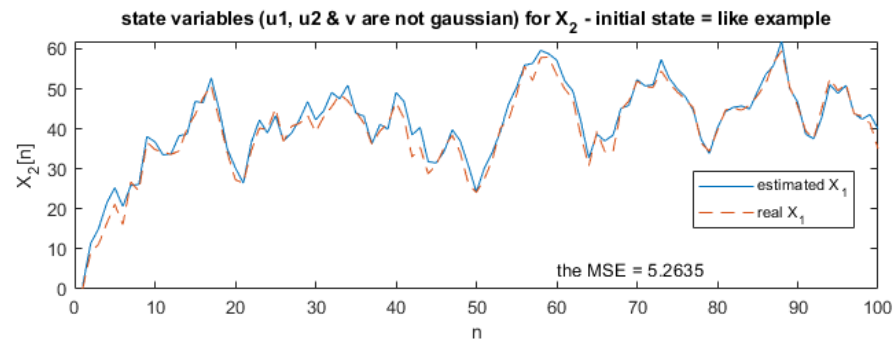
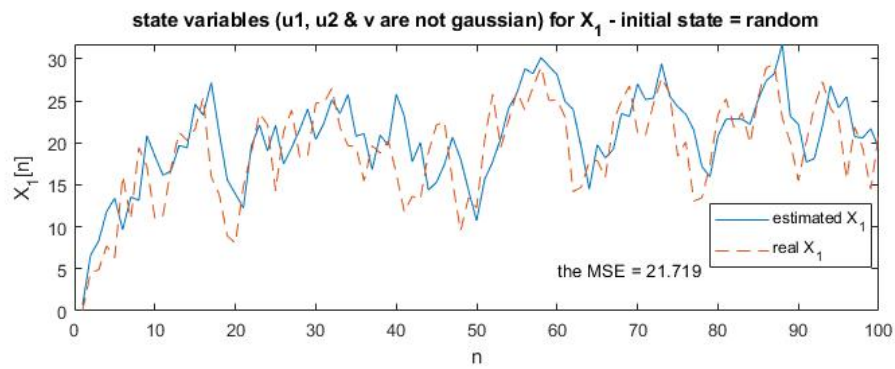
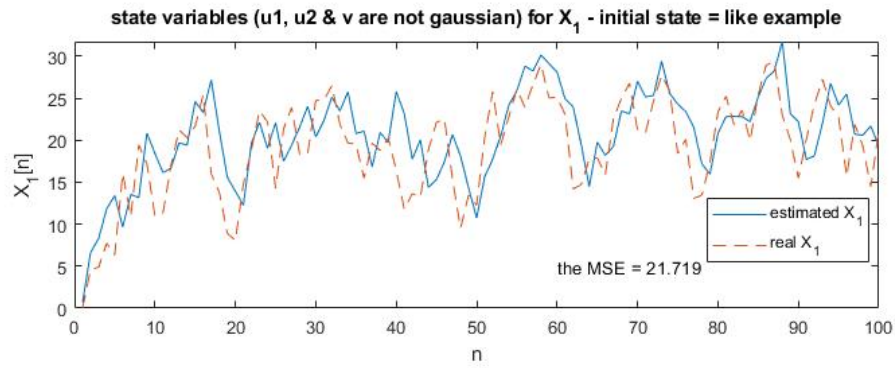
- The estimation of state variables by initial states like "example 2" and random initial states = $[0.65, 0.78]$, MSE as Evaluation criteria:





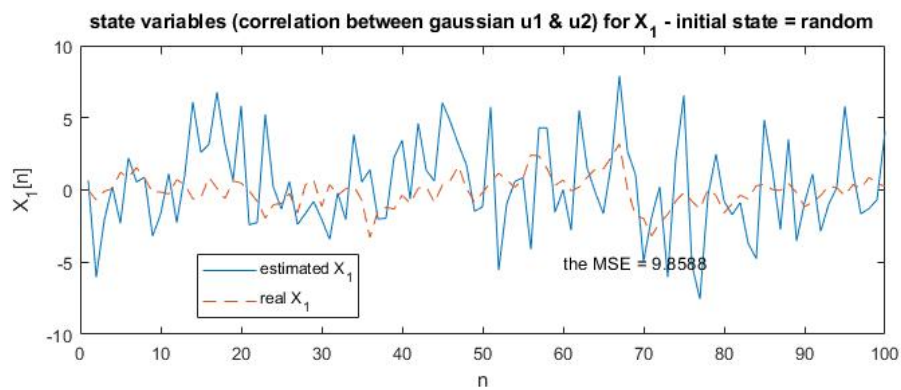
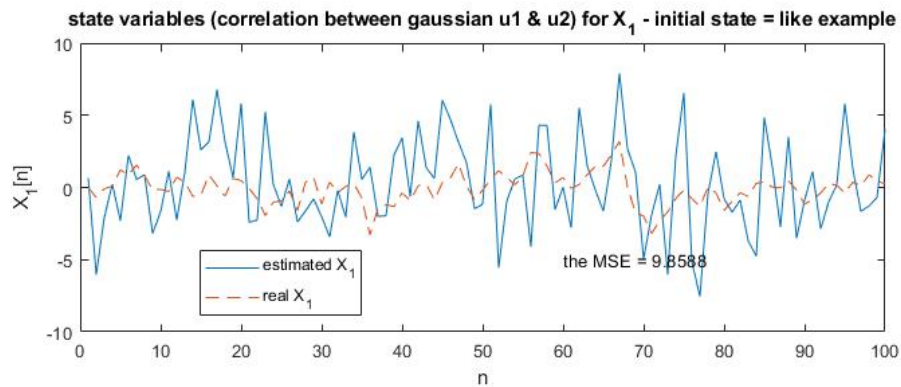
1.2 part b,d,e

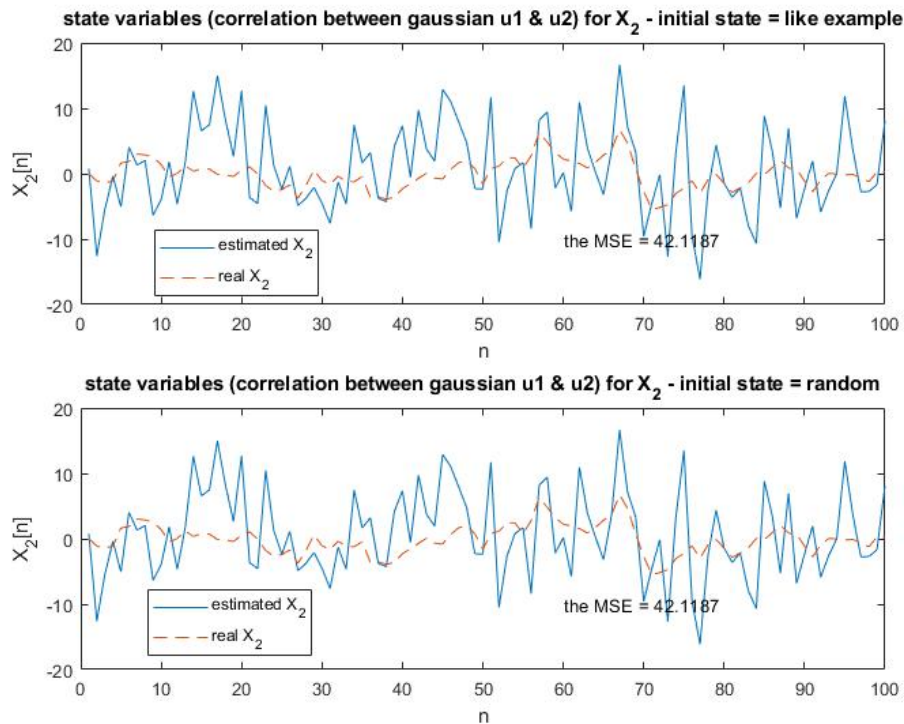
- The estimation of state variables by initial states like "example 2" and random initial states = $[0.65, 0.78]$, MSE as Evaluation criteria when noises are not gaussian :



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 u_1, u_2 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

```

1  clc; clear
2  %% inputs
3  a = 0.6;
4  b = 0.4;
5  u1_std = sqrt(12)*4;
6  u2_std = sqrt(12)*3;

```

```

7  v_std = sqrt(12)*2;
8  k = 100;
9  u1_a = u1_std * randn(1, k);
10 u2_a = u2_std * randn(1, k);
11 v_a = v_std * randn(1, k);
12 u1_b = u1_std * rand(1, k);
13 u2_b = u2_std * rand(1, k);
14 v_b = v_std * rand(1, k);
15 rho = 0.8;
16 x1 = randn(k, 1);
17 x2 = randn(k, 1);
18 R = [1 rho; rho 1];
19 L = chol(R, 'lower');
20 correlated_noises = L * [x1 x2]';
21 u1_c = correlated_noises(1, :);
22 u2_c = correlated_noises(2, :);
23 v_c = v_std * randn(1, k);
24 est_x0_handout = [0,0];
25 est_x0_random = [0.65,0.78];
26 %% part a
27 [~,~,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);
28 [~,~,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);
29 figure;
30 subplot(2,1,1)
31 plot(est_x_a(1,:))
32 hold on
33 plot(x1_a,'--')

```



```

34 legend('estimated X_1','real X_1')
35 title('state variables (gaussian u1 & u2) for X_1 -
      initial state = like example')
36 ylabel('X_1[n]')
37 xlabel('n')
38 MSE = mean((est_x_a(1,:)-x1_a).^2,2);
39 text(60,-30,['the MSE = ', num2str(MSE)])
40 subplot(2,1,2)
41 plot(est_x_da(1,:))
42 hold on
43 plot(x1_da,'--')
44 legend('estimated X_1','real X_1')
45 title('state variables (gaussian u1 & u2) for X_1 -
      initial state = random')
46 ylabel('X_1[n]')
47 xlabel('n')
48 MSE = mean((est_x_da(1,:)-x1_da).^2,2);
49 text(60,-30,['the MSE = ', num2str(MSE)])
50 % part d
51 figure;
52 subplot(2,1,1)
53 plot(est_x_a(2,:))
54 hold on
55 plot(x2_a,'--')
56 legend('estimated X_1','real X_1')
57 title('state variables (gaussian u1 & u2) for X_2 -
      initial state = like example')
58 ylabel('X_1[n]')
59 xlabel('n')
60 MSE = mean((est_x_a(2,:)-x2_a).^2,2);
61 text(60,-30,['the MSE = ', num2str(MSE)])

```

```

62 subplot(2,1,2)
63 plot(est_x_da(2,:))
64 hold on
65 plot(x2_da, '--')
66 legend('estimated X_1', 'real X_1')
67 title('state variables (gaussian u1 & u2) for X_2 -
        initial state = random')
68 ylabel('X_1[n]')
69 xlabel('n')
70 MSE = mean((est_x_da(2,:)-x2_da).^2,2);
71 text(60,-30,['the MSE = ', num2str(MSE)])
72 %% part b
73 [~,~,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);
74 [~,~,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);
75 figure;
76 subplot(2,1,1)
77 plot(est_x_b(1,:))
78 hold on
79 plot(x1_b, '--')
80 legend('estimated X_1', 'real X_1')
81 title('state variables (u1, u2 & v are not gaussian)
        for X_1 - initial state = like example')
82 ylabel('X_1[n]')
83 xlabel('n')
84 MSE = mean((est_x_b(1,:)-x1_b).^2,2);
85 text(60,5,['the MSE = ', num2str(MSE)])
86 subplot(2,1,2)

```

```

87 plot(est_x_db(1,:))
88 hold on
89 plot(x1_db,'--')
90 legend('estimated X_1','real X_1')
91 title('state variables (u1, u2 & v are not gaussian)
        for X_1 - initial state = random')
92 ylabel('X_1[n]')
93 xlabel('n')
94 MSE = mean((est_x_db(1,:)-x1_db).^2,2);
95 text(60,5,['the MSE = ', num2str(MSE)])
96 %
97 figure;
98 subplot(2,1,1)
99 plot(est_x_b(2,:))
100 hold on
101 plot(x2_b,'--')
102 legend('estimated X_1','real X_1')
103 title('state variables (u1, u2 & v are not gaussian)
        for X_2 - initial state = like example')
104 ylabel('X_2[n]')
105 xlabel('n')
106 MSE = mean((est_x_b(2,:)-x2_b).^2,2);
107 text(60,5,['the MSE = ', num2str(MSE)])
108 subplot(2,1,2)
109 plot(est_x_db(2,:))
110 hold on
111 plot(x2_db,'--')
112 legend('estimated X_1','real X_1')
113 title('state variables (u1, u2 & v are not gaussian)
        for X_2 - initial state = random')
114 ylabel('X_2[n]')

```

```

115 xlabel('n')
116 MSE = mean((est_x_db(2,:) - x2_db).^2,2);
117 text(60,5,['the MSE = ', num2str(MSE)])
118 %% part c
119 [~,~,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);
120 [~,~,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);
121 figure;
122 subplot(2,1,1)
123 plot(est_x_c(1,:))
124 hold on
125 plot(x1_c,'--')
126 legend('estimated X_1','real X_1')
127 title('state variables (correlation between gaussian
    u1 & u2) for X_1 - initial state = like example')
128 ylabel('X_1[n]')
129 xlabel('n')
130 MSE = mean((est_x_c(1,:) - x1_c).^2,2);
131 text(60,-5,['the MSE = ', num2str(MSE)])
132 subplot(2,1,2)
133 plot(est_x_dc(1,:))
134 hold on
135 plot(x1_dc,'--')
136 legend('estimated X_1','real X_1')
137 title('state variables (correlation between gaussian
    u1 & u2) for X_1 - initial state = random')
138 ylabel('X_1[n]')
139 xlabel('n')

```

```

140 MSE = mean((est_x_dc(1,:) - x1_dc).^2,2);
141 text(60,-5,['the MSE = ', num2str(MSE)])
142 %
143 figure;
144 subplot(2,1,1)
145 plot(est_x_c(2,:))
146 hold on
147 plot(x2_c, '--')
148 legend('estimated X_2', 'real X_2')
149 title('state variables (correlation between gaussian
        u1 & u2) for X_2 - initial state = like example')
150 ylabel('X_2[n]')
151 xlabel('n')
152 MSE = mean((est_x_c(2,:) - x2_c).^2,2);
153 text(60,-5*2,['the MSE = ', num2str(MSE)])
154 subplot(2,1,2)
155 plot(est_x_dc(2,:))
156 hold on
157 plot(x2_dc, '--')
158 legend('estimated X_2', 'real X_2')
159 title('state variables (correlation between gaussian
        u1 & u2) for X_2 - initial state = random')
160 ylabel('X_2[n]')
161 xlabel('n')
162 MSE = mean((est_x_dc(2,:) - x2_dc).^2,2);
163 text(60,-5*2,['the MSE = ', num2str(MSE)])
164 %% the function for kalman filter
165 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)

```

```

166     k = 100;
167     H1 = tf([1], [1 -a], 1);
168     H2 = tf([1], [1 -b], 1);
169     x1 = filter(H1.Numerator{1}, H1.Denominator{1}, u1
170               );
171     x2 = filter(H2.Numerator{1}, H2.Denominator{1}, x1
172               + u2);
173     real_output = x2 + v;
174     est_x = zeros(2, k);
175     est_xminus = zeros(2, k);
176     est_p = zeros(2, 2, k);
177     est_pminus = zeros(2, 2, k);
178     est_G = zeros(2, 1, k);
179     est_p(:, :, 1) = [std1^2/(1-a^2), std1^2/((1-a^2)
180                   *(1-a*b)); std1^2/((1-a^2)*(1-a*b)), (std1
181                   ^2+(1-a^2)*std2^2)/((1-a^2)*(1-b^2))];
182     est_x(:, 1) = est_x0;
183     est_x(:, 1) = [0.65; 0.78];
184     F = [a,0;a,b];
185     Q = [std1^2, std1^2; std1^2, std1^2 + std2^2];
186     R = stdv^2;
187     H = [0,1];
188     estimated_z = zeros(1,k);
189     estimated_z(1,1) = H*est_x(:,1) + v(1);
190     for i = 2:k
191         est_xminus(:, i) = F * est_x(:, i-1);
192         est_pminus(:, :, i) = F * est_p(:, :, i-1) * F
193             ' + Q;
194         Gk = est_pminus(:, :, i) * H' / (H *
195             est_pminus(:, :, i) * H' + R);

```

```

190     est_x(:, i) = est_xminus(:, i) + Gk * (
        real_output(i) - H * est_xminus(:, i));
191     est_p(:, :, i) = est_pminus(:, :, i) - Gk * H
        * est_pminus(:, :, i);
192     est_G(:, :, i) = Gk;
193     estimated_z(1, i) = H*est_x(:, i) + v(i);
194     end
195 end
196
197 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)
198     k = 100;
199     H1 = tf([1], [1 -a], 1);
200     H2 = tf([1], [1 -b], 1);
201     x1 = filter(H1.Numerator{1}, H1.Denominator{1}, u1
    );
202     x2 = filter(H2.Numerator{1}, H2.Denominator{1}, x1
    + u2);
203     real_output = x2 + v;
204     est_x = zeros(2, k);
205     est_xminus = zeros(2, k);
206     est_p = zeros(2, 2, k);
207     est_pminus = zeros(2, 2, k);
208     est_G = zeros(2, 1, k);
209     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))];
210     est_x(:, 1) = est_x0;
211     est_x(:, 1) = [0.65; 0.78];

```

```

212 F = [a,0;a,b];
213 Q = [std1^2, std1*std2*rho; std1*std2*rho, std1^2
      + std2^2];
214 R = stdv^2;
215 H = [0,1];
216 estimated_z = zeros(1,k);
217 estimated_z(1,1) = H*est_x(:,1) + v(1);
218 for i = 2:k
219     est_xminus(:, i) = F * est_x(:, i-1);
220     est_pminus(:, :, i) = F * est_p(:, :, i-1) * F
        ' + Q;
221     Gk = est_pminus(:, :, i) * H' / (H *
        est_pminus(:, :, i) * H' + R);
222     est_x(:, i) = est_xminus(:, i) + Gk * (
        real_output(i) - H * est_xminus(:, i));
223     est_p(:, :, i) = est_pminus(:, :, i) - Gk * H
        * est_pminus(:, :, i);
224     est_G(:, :, i) = Gk;
225     estimated_z(1,i) = H*est_x(:,i) + v(i);
226 end
227 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

```
1 feature1 = zeros(1,train_num);
2 feature2 = zeros(1,train_num);
3 feature3 = zeros(1,train_num);
4 mean_train = zeros(1,train_num);
5 for i = 1:train_num
6     if i <= 9
7         field_name = [ 'ECG0' num2str(i) ];
8     else
9         field_name = [ 'ECG' num2str(i) ];
10    end
11    mean_train(i) = mean(ecg.(field_name));
12    lower_than_m = [];
13    higher_than_m = [];
14    k = 0;
15    l = 0;
16    for j = 1:length(ecg.(field_name))
17        if ecg.(field_name)(j) < mean_train(i)
18            k = k+1;
19            lower_than_m(k) = ecg.(field_name)(j);
20        else
21            l = l+1;
22            higher_than_m(l) = ecg.(field_name)(j);
23        end
24    end
```

```

25     feature1(i) = mean(lower_than_m) / min(ecg.(
        field_name));
26     feature2(i) = mean(lower_than_m)/var(lower_than_m)
        ;
27     feature3(i) = (mean(higher_than_m)-mean(
        lower_than_m))/(max(ecg.(field_name))-min(ecg.(
        field_name)));
28 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   -0.0194   -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

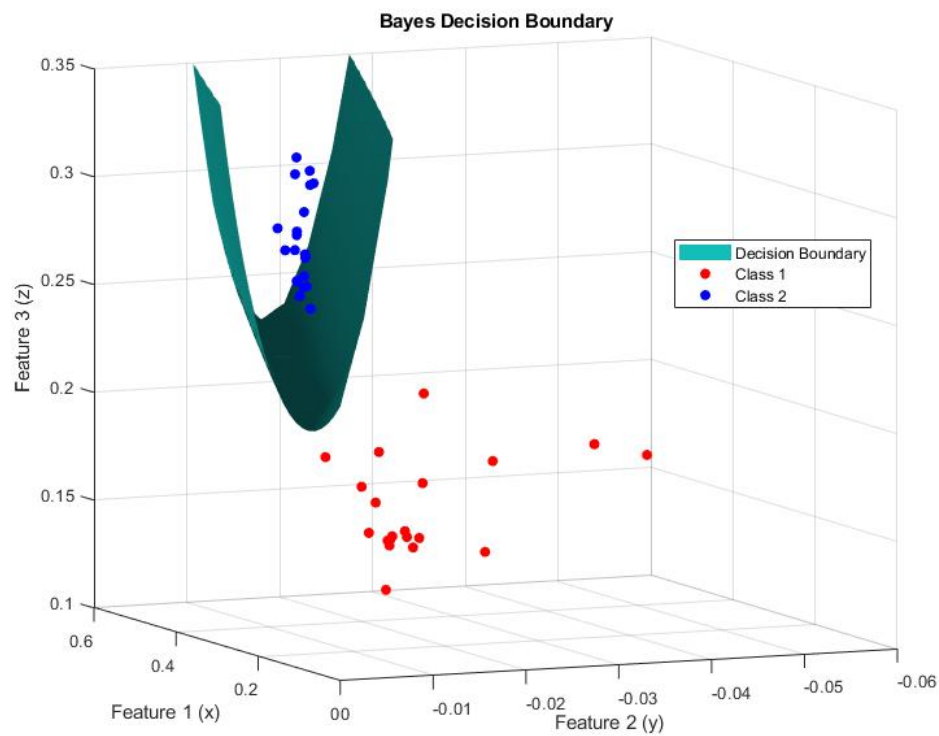
```

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

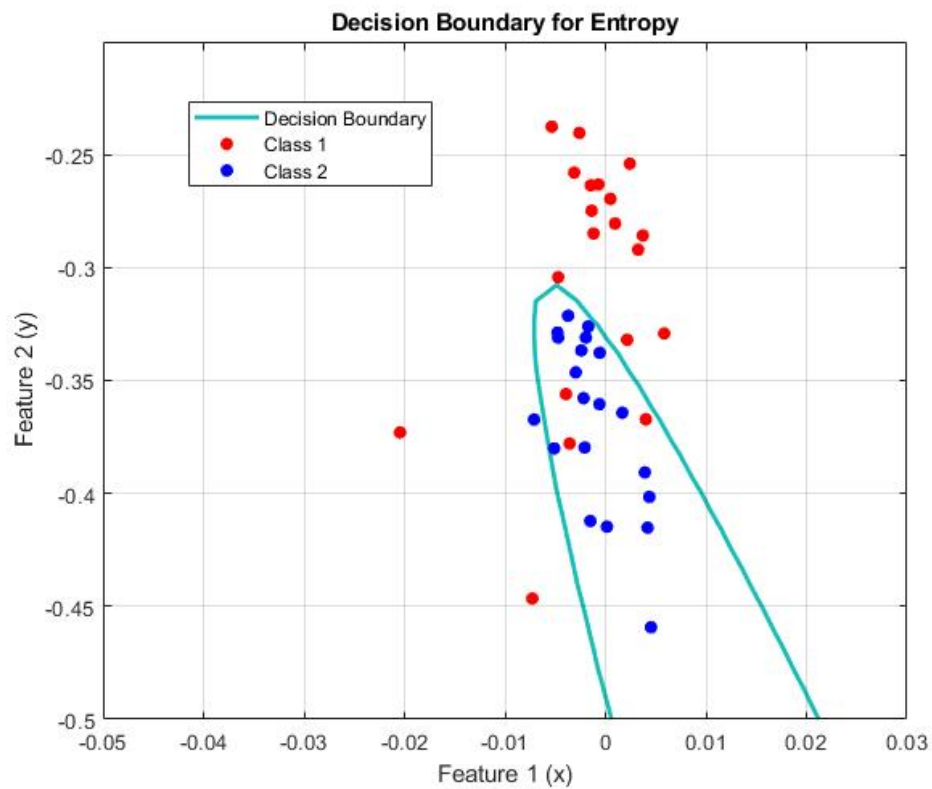
I have done classification by bayes decision boundary and using the equation from handout:



As we can see they are classified correctly.

2.3 part c

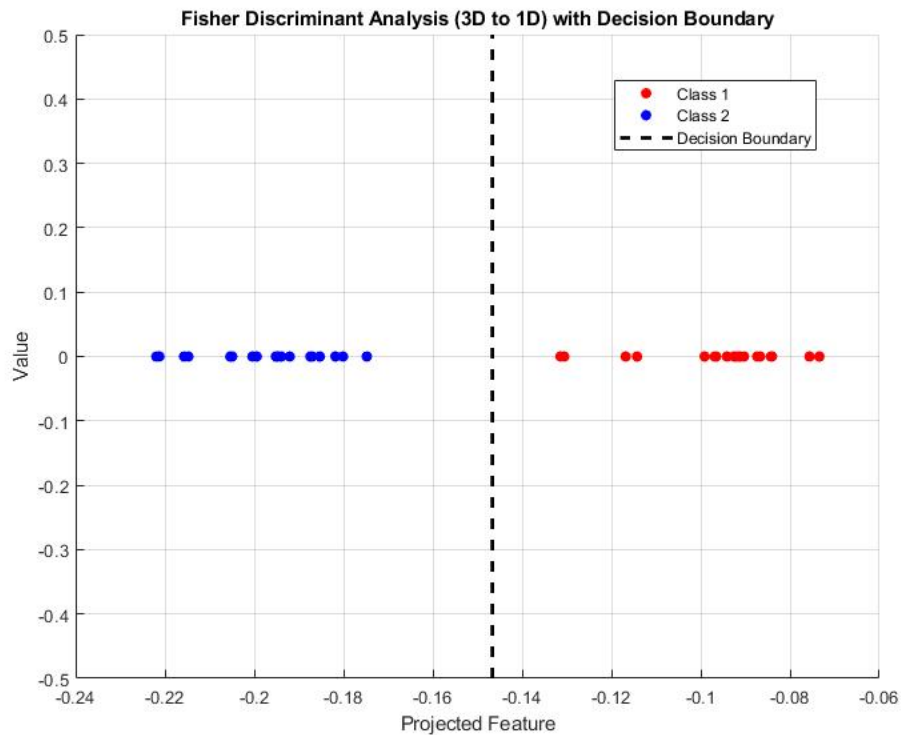
I have done classification by entropy reduction and using the equation from hand-out:



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

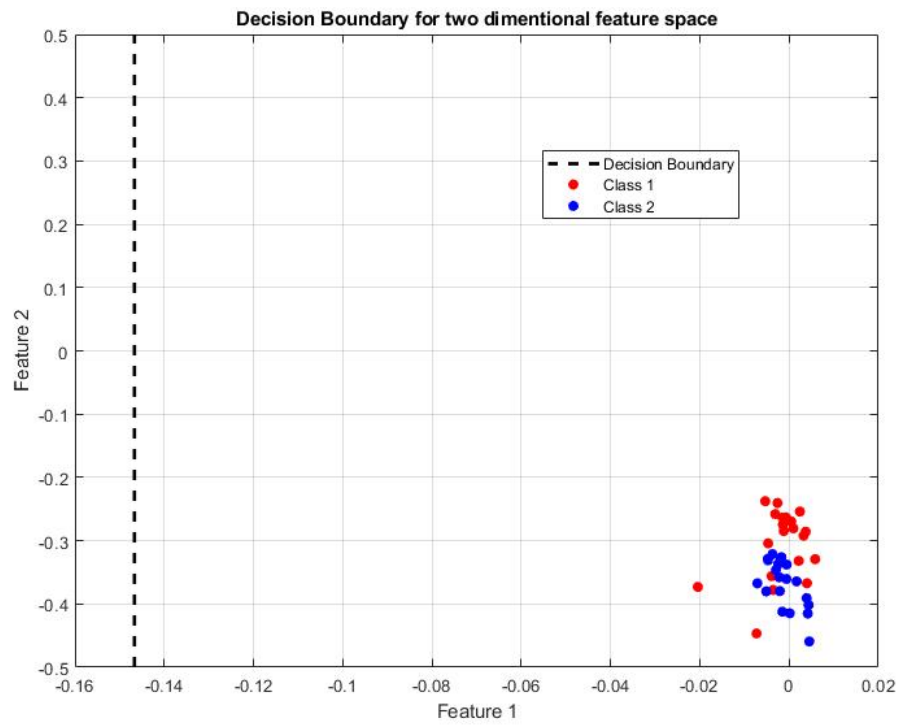
2.4 part d

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:



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

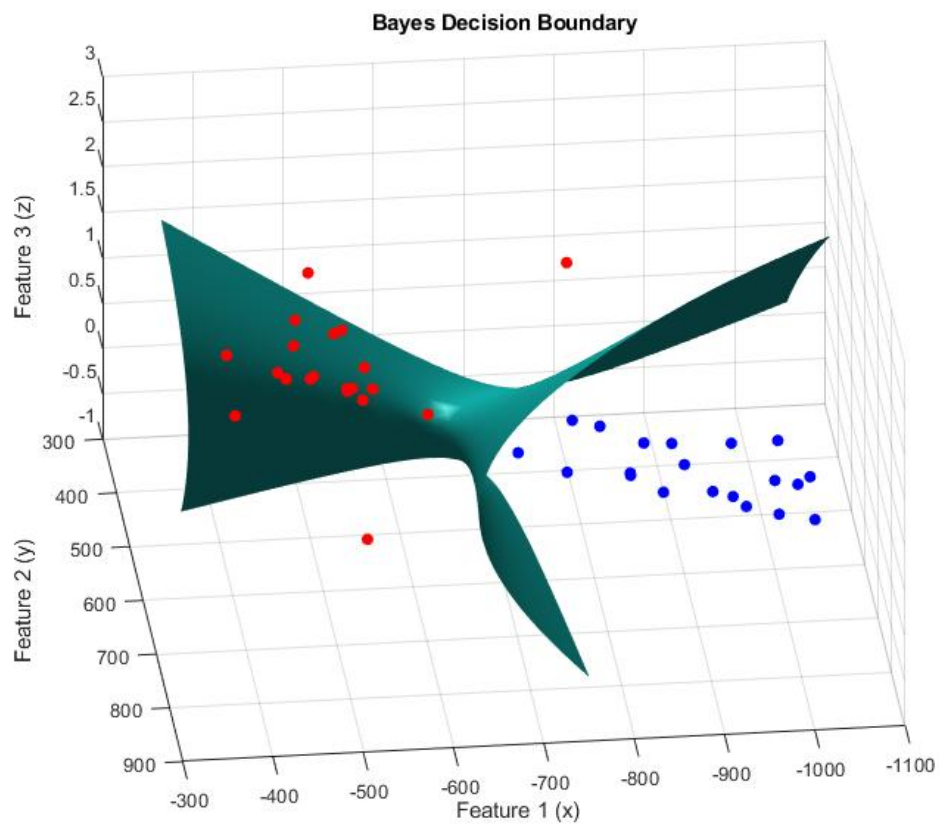
Now I repeat the classification for three new features:

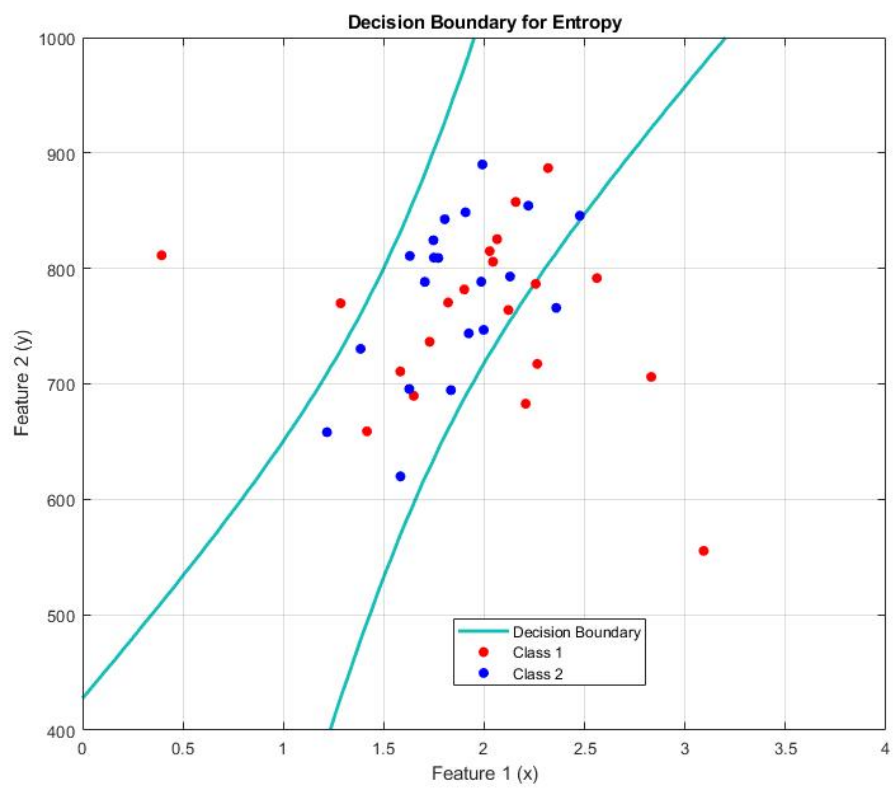
feature1: T amp wave

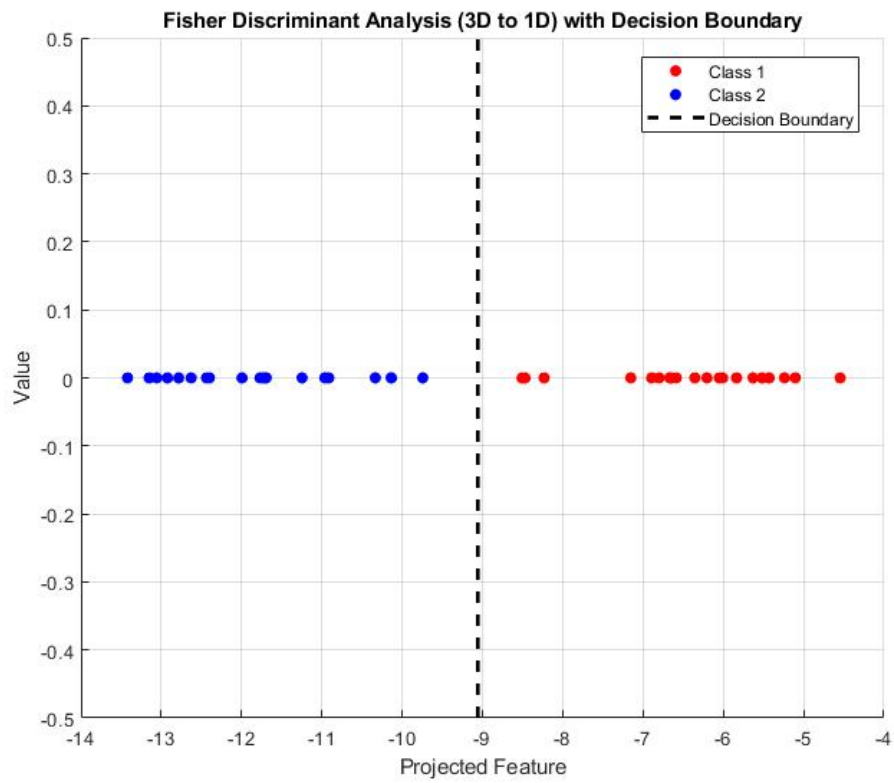
feature1: P amp wave

feature1: std

Now I have repeated all parts ass below :







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

```
1 clc; clear
2 ecg = load('ECG.mat');
3 train_num = 40;
4 test_num = 40;
5 %% part a
6 feature1 = zeros(1,train_num);
7 feature2 = zeros(1,train_num);
8 feature3 = zeros(1,train_num);
```

```

9  mean_train = zeros(1,train_num);
10 for i = 1:train_num
11     if i<=9
12         field_name = ['ECG0' num2str(i)];
13     else
14         field_name = ['ECG' num2str(i)];
15     end
16     mean_train(i) = mean(ecg.(field_name));
17     lower_than_m= [];
18     higher_than_m = [];
19     k = 0;
20     l = 0;
21     for j = 1:length(ecg.(field_name))
22         if ecg.(field_name)(j)<mean_train(i)
23             k = k+1;
24             lower_than_m(k) = ecg.(field_name)(j);
25         else
26             l = l+1;
27             higher_than_m(l) = ecg.(field_name)(j);
28         end
29     end
30     feature1(i) = mean(lower_than_m) / min(ecg.(
        field_name));
31     feature2(i) = mean(lower_than_m)/var(lower_than_m)
        ;
32     feature3(i) = (mean(higher_than_m)-mean(
        lower_than_m))/(max(ecg.(field_name))-min(ecg.(
        field_name)));
33 end
34 %% part b
35 class1_featurematrix = [feature1(1:20)' feature2(1:20)

```

```

        ' feature3(1:20) '];
36 mean1_mat = mean(class1_featurematrix);
37 cov1_mat = cov(class1_featurematrix);
38 class2_featurematrix = [feature1(21:40)' feature2
    (21:40)' feature3(21:40) '];
39 mean2_mat = mean(class2_featurematrix);
40 cov2_mat = cov(class2_featurematrix);
41 cov_mat_train = cov([feature1(1:40)' feature2(1:40)'
    feature3(1:40) ']);
42 x_range = linspace(0, 0.6, 40);
43 y_range = linspace(0, -0.1, 40);
44 z_range = linspace(0, 0.35, 40);
45 [X, Y, Z] = meshgrid(x_range, y_range, z_range);
46 D1 = zeros(size(X));
47 D2 = zeros(size(X));
48 x_flat = X(:);
49 y_flat = Y(:);
50 z_flat = Z(:);
51 for i = 1:length(x_flat)
52     beta = [x_flat(i); y_flat(i); z_flat(i)];
53     D1(i) = -0.5 * log(det(cov1_mat)) - 0.5 * (beta -
        mean1_mat')' * inv(cov1_mat) * (beta -
        mean1_mat');
54     D2(i) = -0.5 * log(det(cov2_mat)) - 0.5 * (beta -
        mean2_mat')' * inv(cov2_mat) * (beta -
        mean2_mat');
55 end
56 D1 = reshape(D1, size(X));
57 D2 = reshape(D2, size(X));
58 Difference = D1 - D2;
59 figure;

```

```

60  isosurface(X, Y, Z, Difference , 0);
61  hold on;
62  scatter3(class1_featurematrix(:,1),
            class1_featurematrix(:,2), class1_featurematrix
           (:,3), 'r', 'filled');
63  hold on;
64  scatter3(class2_featurematrix(:,1),
            class2_featurematrix(:,2), class2_featurematrix
           (:,3), 'b', 'filled');
65  xlabel('Feature 1 (x)');
66  ylabel('Feature 2 (y)');
67  zlabel('Feature 3 (z)');
68  title('Bayes Decision Boundary');
69  grid on;
70  legend({'Decision Boundary', 'Class 1', 'Class 2'}, '
        Location', 'best')
71  %% part c
72  [evc1, eval] = eig(cov1_mat);
73  [evc2, eva2] = eig(cov2_mat);
74  [evc, eva] = eig(cov_mat_train);
75  u = evc(:,1:2);
76  u1 = evc1(:,1:2); % Assuming these are the principal
        components
77  u2 = evc2(:,1:2);
78  class1_featurematrix_entropy = class1_featurematrix *
        u;
79  class2_featurematrix_entropy = class2_featurematrix *
        u;
80  mean1_mat_entropy = mean(class1_featurematrix_entropy)
        ';
81  mean2_mat_entropy = mean(class2_featurematrix_entropy)

```

```

    ',';
82 cov1_mat_entropy = cov(class1_featurematrix_entropy);
83 cov2_mat_entropy = cov(class2_featurematrix_entropy);
84 x_range = linspace(-0.05, 0.03, 40);
85 y_range = linspace(-0.5, -0.2, 40);
86 [Xent, Yent] = meshgrid(x_range, y_range);
87 D1 = zeros(size(Xent));
88 D2 = zeros(size(Xent));
89 x_flat = Xent(:);
90 y_flat = Yent(:);
91 for i = 1:length(x_flat)
92     beta = [x_flat(i); y_flat(i)];
93     D1(i) = -0.5 * log(det(cov1_mat_entropy)) - 0.5 *
        (beta - mean1_mat_entropy)' * inv(
            cov1_mat_entropy) * (beta - mean1_mat_entropy);
94     D2(i) = -0.5 * log(det(cov2_mat_entropy)) - 0.5 *
        (beta - mean2_mat_entropy)' * inv(
            cov2_mat_entropy) * (beta - mean2_mat_entropy);
95 end
96 D1 = reshape(D1, size(Xent));
97 D2 = reshape(D2, size(Xent));
98 Difference_ent = D1 - D2;
99 figure;
100 contour(Xent, Yent, Difference_ent, [0, 0], 'LineWidth
    ', 2);
101 hold on;
102 scatter(class1_featurematrix_entropy(:, 1),
    class1_featurematrix_entropy(:, 2), 'r', 'filled');
103 hold on;
104 scatter(class2_featurematrix_entropy(:, 1),
    class2_featurematrix_entropy(:, 2), 'b', 'filled');

```

```

105 xlabel('Feature 1 (x)');
106 ylabel('Feature 2 (y)');
107 title('Decision Boundary for Entropy');
108 grid on;
109 legend({'Decision Boundary', 'Class 1', 'Class 2'}, '
        Location', 'best');
110 %% part d
111
112 class1_featurematrix = [feature1(1:20)' feature2(1:20)
        ' feature3(1:20)'];
113 class2_featurematrix = [feature1(21:40)' feature2
        (21:40)' feature3(21:40)'];
114 mean1 = mean(class1_featurematrix)';
115 mean2 = mean(class2_featurematrix)';
116 S_W = cov(class1_featurematrix) + cov(
        class2_featurematrix);
117 mean_diff = mean1 - mean2;
118 S_B = mean_diff * mean_diff';
119 [V, D] = eig(S_W \ S_B);
120 [~, idx] = max(diag(D));
121 w = V(:, idx);
122 projected_class1 = class1_featurematrix * w;
123 projected_class2 = class2_featurematrix * w;
124 mean_proj1 = mean(projected_class1);
125 mean_proj2 = mean(projected_class2);
126 var_proj1 = var(projected_class1);
127 var_proj2 = var(projected_class2);
128 pooled_variance = ((length(projected_class1) - 1) *
        var_proj1 + (length(projected_class2) - 1) *
        var_proj2) / (length(projected_class1) + length(
        projected_class2) - 2);

```



```

129 decision_boundary = (mean_proj1 + mean_proj2) / 2;
130 figure;
131 hold on;
132 scatter(projected_class1 , zeros(size(projected_class1)
    ), 'r', 'filled');
133 scatter(projected_class2 , zeros(size(projected_class2)
    ), 'b', 'filled');
134 plot([decision_boundary decision_boundary], [-0.5
    0.5], 'k--', 'LineWidth', 2);
135 xlabel('Projected Feature');
136 ylabel('Value');
137 title('Fisher Discriminant Analysis (3D to 1D) with
    Decision Boundary');
138 legend({'Class 1', 'Class 2', 'Decision Boundary'}, '
    Location', 'best');
139 grid on;
140 figure;
141 plot([decision_boundary decision_boundary], [-0.5
    0.5], 'k--', 'LineWidth', 2);
142 hold on;
143 scatter(class1_featurematrix_entropy(:, 1),
    class1_featurematrix_entropy(:, 2), 'r', 'filled');
144 hold on;
145 scatter(class2_featurematrix_entropy(:, 1),
    class2_featurematrix_entropy(:, 2), 'b', 'filled');
146 xlabel('Feature 1');
147 ylabel('Feature 2');
148 title('Decision Boundary for two dimensional feature
    space');
149 grid on;
150 legend({'Decision Boundary', 'Class 1', 'Class 2'}, '

```

```

        'Location', 'best')
151 %% part e
152 feature1test = zeros(1,test_num);
153 feature2test  = zeros(1,test_num);
154 feature3test  = zeros(1,test_num);
155 mean_test = zeros(1,test_num);
156 for i = 41:80
157     field_name = ['ECG' num2str(i)];
158     mean_test(i-40) = mean(ecg.(field_name));
159     lower_than_mtest= [];
160     higher_than_mtest = [];
161     k = 0;
162     l = 0;
163     for j = 1:length(ecg.(field_name))
164         if ecg.(field_name)(j)<mean_test(i-40)
165             k = k+1;
166             lower_than_mtest(k) = ecg.(field_name)(j);
167         else
168             l = l+1;
169             higher_than_mtest(l) = ecg.(field_name)(j);
170         end
171     end
172     feature1test (i-40) = mean(lower_than_mtest) / min
        (ecg.(field_name));
173     feature2test (i-40) = mean(lower_than_mtest)/var(
        lower_than_mtest);
174     feature3test (i-40) = (mean(higher_than_mtest)-
        mean(lower_than_mtest))/(max(ecg.(field_name))-
        min(ecg.(field_name)));
175 end
176 class1test_featurematrix = [feature1test(1:20)']

```

```

        feature2test(1:20)' feature3test(1:20) '];
177 class2test_featurematrix = [feature1test(21:40)'
        feature2test(21:40)' feature3test(21:40) '];
178 cov1_mat_test = cov(class1test_featurematrix);
179 cov2_mat_test = cov(class2test_featurematrix);
180 cov_mat_test = cov([feature1test(1:40)' feature2test
        (1:40)' feature3test(1:40) ']);
181 % for part b
182 labels_class1_mah = zeros(1,20);
183 labels_class2_mah = zeros(1,20);
184 labels_class1_o = zeros(1,20);
185 labels_class2_o = zeros(1,20);
186 labels_class1_bayes = zeros(1,20);
187 labels_class2_bayes = zeros(1,20);
188 temp_o = 0;
189 temp_mah = 0;
190 temp_bayes = 0;
191 for i = 1:20
192     d1_mah = (class1test_featurematrix(i,1:3)-
        mean1_mat)*inv(cov1_mat)*(
        class1test_featurematrix(i,1:3)-mean1_mat)';
193     d2_mah = (class1test_featurematrix(i,1:3)-
        mean2_mat)*inv(cov2_mat)*(
        class1test_featurematrix(i,1:3)-mean2_mat)';
194     if d1_mah>d2_mah
195         labels_class1_mah(1,i) = 2;
196         temp_mah = temp_mah +1;
197     else
198         labels_class1_mah(1,i) = 1;
199     end
200     d1_mah = (class2test_featurematrix(i,1:3)-

```

```

        mean1_mat)*inv(cov1_mat)*(
        class2test_featurematrix(i,1:3)-mean1_mat)';
201 d2_mah = (class2test_featurematrix(i,1:3)-
        mean2_mat)*inv(cov2_mat)*(
        class2test_featurematrix(i,1:3)-mean2_mat)';
202 if d1_mah>d2_mah
203     labels_class2_mah(1,i) = 2;
204 else
205     labels_class2_mah(1,i) = 1;
206     temp_mah = temp_mah +1;
207 end
208 end
209 for i = 1:20
210     d1_o = (class1test_featurematrix(i,1:3)-mean1_mat)
        *(class1test_featurematrix(i,1:3)-mean1_mat)';
211     d2_o = (class1test_featurematrix(i,1:3)-mean2_mat)
        *(class1test_featurematrix(i,1:3)-mean2_mat)';
212 if d1_o>d2_o
213     labels_class1_o(1,i) = 2;
214     temp_o = temp_o +1;
215 else
216     labels_class1_o(1,i) = 1;
217 end
218 d1_o = (class2test_featurematrix(i,1:3)-mean1_mat)
        *(class2test_featurematrix(i,1:3)-mean1_mat)';
219 d2_o = (class2test_featurematrix(i,1:3)-mean2_mat)
        *(class2test_featurematrix(i,1:3)-mean2_mat)';
220 if d1_o>d2_o
221     labels_class2_o(1,i) = 2;
222 else
223     labels_class2_o(1,i) = 1;

```

```

224         temp_o = temp_o +1;
225     end
226 end
227 for i = 1:20
228     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)';
229     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)';
230     if d1_o>d2_o
231         labels_class1_bayes(1,i) = 2;
232         temp_bayes = temp_bayes +1;
233     else
234         labels_class1_bayes(1,i) = 1;
235     end
236     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)';
237     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)';
238     if d1_o>d2_o
239         labels_class2_bayes(1,i) = 2;
240     else
241         labels_class2_bayes(1,i) = 1;
242         temp_bayes = temp_bayes +1;

```

```

243     end
244 end
245 disp('accuracy by mahlanobis (part b) :')
246 disp((40-temp_o)/40*100)
247 disp('accuracy by Euclidean (part b) :')
248 disp((40-temp_mah)/40*100)
249 disp('accuracy by bayes decision boundary (part b) :')
250 disp((40-temp_bayes)/40*100)
251 disp('
-----
    ')
252 %for part c
253 [evc1_test_b , eva1_test_b] = eig(cov1_mat_test);
254 [evc2_test_b , eva2_test_b] = eig(cov2_mat_test);
255 [evc_test_b , eva_test_b] = eig(cov_mat_test);
256 u1_test_b = evc1_test_b(:,1:2);
257 u2_test_b = evc2_test_b(:,1:2);
258 u_test_b = evc_test_b(:,1:2);
259 class1_featurematrix_test_entropy =
    class1_featurematrix * u_test_b;
260 class2_featurematrix_test_entropy =
    class2_featurematrix * u_test_b;
261 labels_class1_mah_ent = zeros(1,20);
262 labels_class2_mah_ent = zeros(1,20);
263 labels_class1_o_ent = zeros(1,20);
264 labels_class2_o_ent = zeros(1,20);
265 labels_class1_bayes_ent = zeros(1,20);
266 labels_class2_bayes_ent = zeros(1,20);
267 temp_o = 0;
268 temp_mah = 0;
269 temp_bayes = 0;

```

```

270 for i = 1:20
271     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)';
272     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)';
273     if d1_mah>d2_mah
274         labels_class1_mah_ent(1,i) = 2;
275         temp_mah = temp_mah +1;
276     else
277         labels_class2_mah_ent(1,i) = 1;
278     end
279     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)';
280     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)';
281     if d1_mah>d2_mah
282         labels_class1_mah_ent(1,i) = 2;
283     else
284         labels_class2_mah_ent(1,i) = 1;
285         temp_mah = temp_mah +1;
286     end
287 end
288 for i = 1:20

```

```

289     d1_o = (class1_featurematrix_test_entropy(i,1:2)-
              mean1_mat_entropy)*(
              class1_featurematrix_test_entropy(i,1:2)-
              mean1_mat_entropy)';
290     d2_o = (class1_featurematrix_test_entropy(i,1:2)-
              mean2_mat_entropy)*(
              class1_featurematrix_test_entropy(i,1:2)-
              mean2_mat_entropy)';
291     if d1_o>d2_o
292         labels_class1_o_ent(1,i) = 2;
293         temp_o = temp_o +1;
294     else
295         labels_class1_o_ent(1,i) = 1;
296     end
297     d1_o = (class2_featurematrix_test_entropy(i,1:2)-
              mean1_mat_entropy)*(
              class2_featurematrix_test_entropy(i,1:2)-
              mean1_mat_entropy)';
298     d2_o = (class2_featurematrix_test_entropy(i,1:2)-
              mean2_mat_entropy)*(
              class2_featurematrix_test_entropy(i,1:2)-
              mean2_mat_entropy)';
299     if d1_o>d2_o
300         labels_class2_o_ent(1,i) = 2;
301     else
302         labels_class2_o_ent(1,i) = 1;
303         temp_o = temp_o +1;
304     end
305 end
306 for i = 1:20
307     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)';
308 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)';
309 if d1_mah>d2_mah
310     labels_class1_bayes_ent(1,i) = 2;
311     temp_bayes = temp_bayes +1;
312 else
313     labels_class1_bayes_ent(1,i) = 1;
314 end
315 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)';
316 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)';
317 if d1_mah>d2_mah
318     labels_class2_bayes_ent(1,i) = 2;
319 else
320     labels_class2_mah_ent(1,i) = 1;
321     temp_bayes = temp_bayes +1;
322 end

```

```

323 end
324 disp('accuracy by mahlanobis (part c) :')
325 disp((40-temp_mah)/40*100)
326 disp('accuracy by Euclidean (part c) :')
327 disp((40-temp_o)/40*100)
328 disp('accuracy by bayes decision boundary (part c) :')
329 disp((40-temp_bayes)/40*100)
330 disp('
-----
    ')
331 % for part d
332 mean1_test_fisher = mean(class1test_featurematrix)';
333 mean2_test_fisher = mean(class2test_featurematrix)';
334 S_W_test = cov(class1test_featurematrix) + cov(
    class2test_featurematrix);
335 mean_diff = mean1_test_fisher - mean2_test_fisher;
336 S_B = mean_diff * mean_diff';
337 [V_test, D_test] = eig(S_W_test \ S_B);
338 [~, idx_test] = max(diag(D_test));
339 w_test = V_test(:, idx_test);
340 projected_class1_test = class1test_featurematrix *
    w_test;
341 projected_class2_test = class2test_featurematrix *
    w_test;
342 mean_proj1_test = mean(projected_class1_test);
343 mean_proj2_test = mean(projected_class2_test);
344 var_proj1_test = var(projected_class1_test);
345 var_proj2_test = var(projected_class2_test);
346 labels_class1_mah_fisher = zeros(1,20);
347 labels_class2_mah_fisher = zeros(1,20);
348 labels_class1_o_fisher = zeros(1,20);

```

```

349 labels_class2_o_fisher = zeros(1,20);
350 labels_class1_o_bayes = zeros(1,20);
351 labels_class2_o_bayes = zeros(1,20);
352 temp_o = 0;
353 temp_mah = 0;
354 temp_bayes = 0;
355 for i = 1:20
356     d1_mah = (projected_class1_test(i)-mean_proj1_test
               )*inv(var_proj1_test)*(projected_class1_test(i)
               -mean_proj1_test)';
357     d2_mah = (projected_class1_test(i)-mean_proj2_test
               )*inv(var_proj2_test)*(projected_class1_test(i)
               -mean_proj2_test)';
358     if d1_mah>d2_mah
359         labels_class1_mah_fisher(1,i) = 2;
360         temp_mah = temp_mah +1;
361     else
362         labels_class2_mah_fisher(1,i) = 1;
363     end
364     d1_mah = (projected_class2_test(i)-mean_proj1_test
               )*inv(var_proj1_test)*(projected_class2_test(i)
               -mean_proj1_test)';
365     d2_mah = (projected_class2_test(i)-mean_proj2_test
               )*inv(var_proj2_test)*(projected_class2_test(i)
               -mean_proj2_test)';
366     if d1_mah>d2_mah
367         labels_class1_mah_fisher(1,i) = 2;
368     else
369         labels_class2_mah_fisher(1,i) = 1;
370         temp_mah = temp_mah +1;
371     end

```

```

372 end
373 for i = 1:20
374     d1_o = (projected_class1_test(i)-mean_proj1_test)
            *(projected_class1_test(i)-mean_proj1_test)';
375     d2_o = (projected_class1_test(i)-mean_proj2_test)
            *(projected_class1_test(i)-mean_proj2_test)';
376     if d1_o>d2_o
377         labels_class1_o_fisher(1,i) = 2;
378         temp_o = temp_o +1;
379     else
380         labels_class1_o_fisher(1,i) = 1;
381     end
382     d1_o = (projected_class2_test(i)-mean_proj1_test)
            *(projected_class2_test(i)-mean_proj1_test)';
383     d2_o = (projected_class2_test(i)-mean_proj2_test)
            *(projected_class2_test(i)-mean_proj2_test)';
384     if d1_o>d2_o
385         labels_class2_o_fisher(1,i) = 2;
386     else
387         labels_class2_o_fisher(1,i) = 1;
388         temp_o = temp_o +1;
389     end
390 end
391 for i = 1:20
392     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)';
393     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)-

```

```

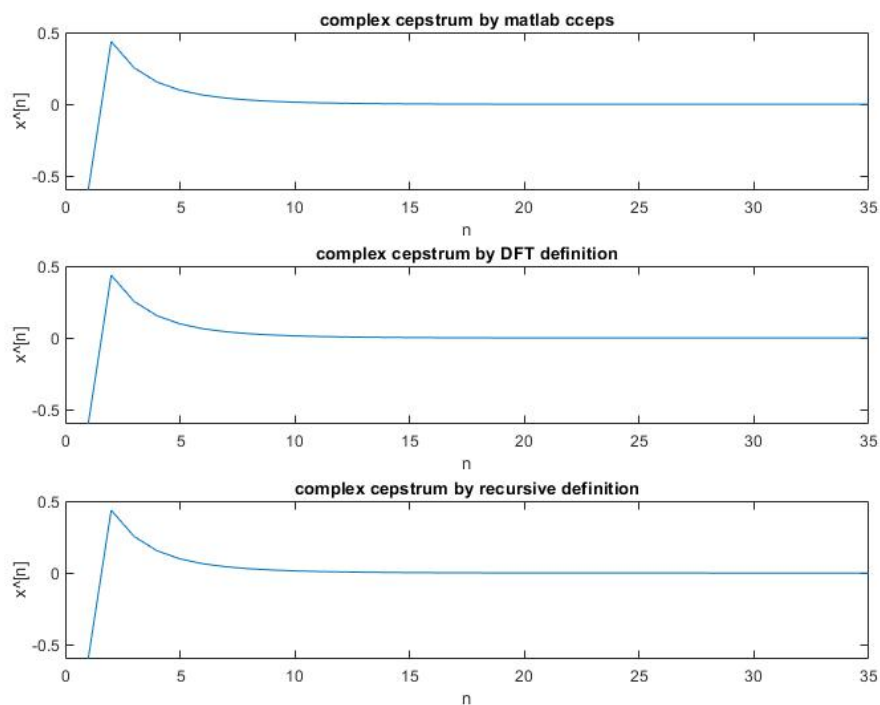
        mean_proj2_test) ' ;
394 if d1_mah>d2_mah
395     labels_class1_mah_fisher(1,i) = 2;
396     temp_bayes = temp_bayes +1;
397 else
398     labels_class2_mah_fisher(1,i) = 1;
399 end
400 d1_mah = 0.5*log(det(var_proj1_test))+0.5*(
        projected_class2_test(i)-mean_proj1_test)*inv(
        var_proj1_test)*(projected_class2_test(i)-
        mean_proj1_test) ' ;
401 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) ' ;
402 if d1_mah>d2_mah
403     labels_class1_mah_fisher(1,i) = 2;
404 else
405     labels_class2_mah_fisher(1,i) = 1;
406     temp_bayes = temp_bayes +1;
407 end
408 end
409 disp('accuracy by mahlanobis (part d) :')
410 disp((40-temp_o)/40*100)
411 disp('accuracy by Euclidean (part d) :')
412 disp((40-temp_mah)/40*100)
413 disp('accuracy by bayes decision boundary (part d) :')
414 disp((40-temp_bayes)/40*100)
415 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.

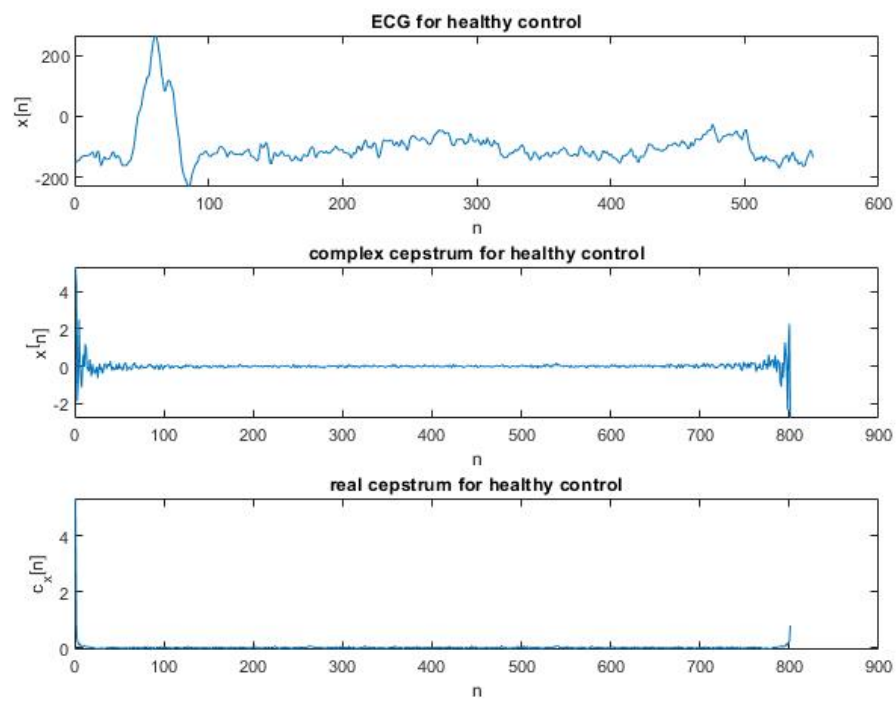
3.1 part a



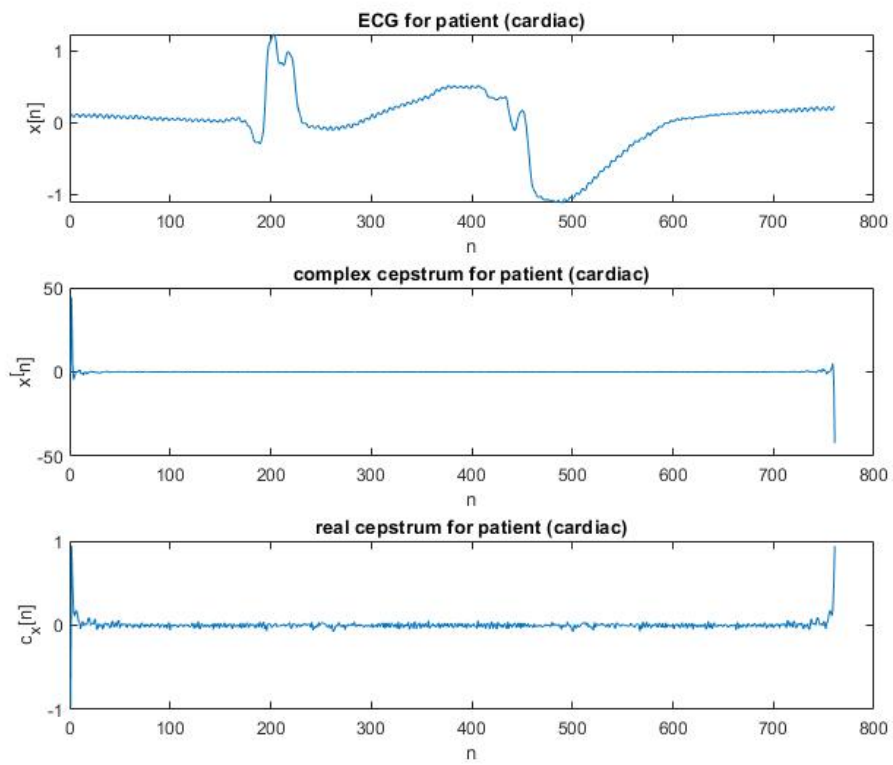
* **Explanation:** As we can see there is no difference between three methods and they have same answers.

3.2 part b

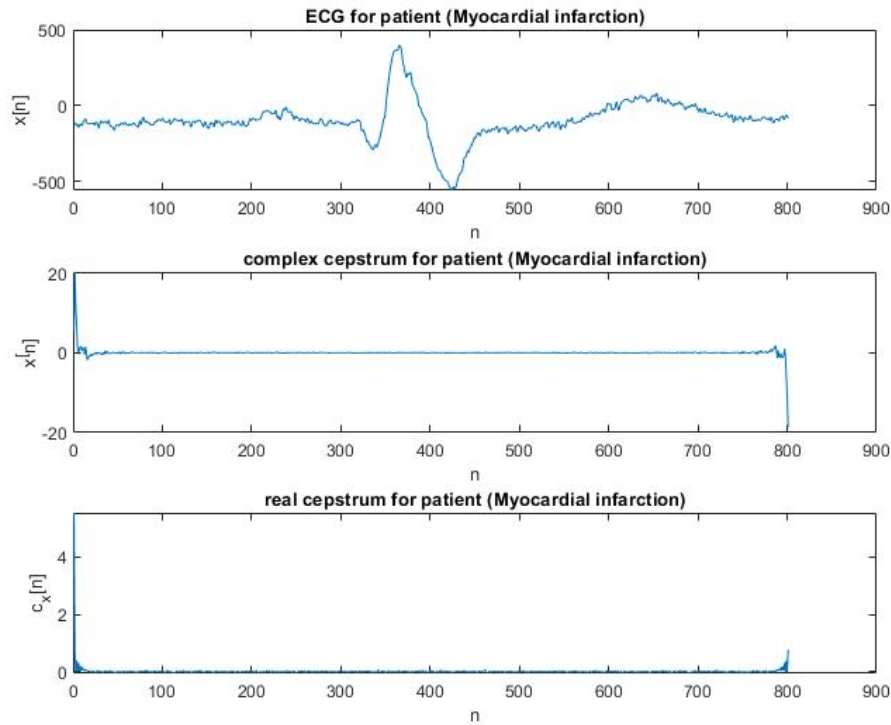
- For healthy person :



- 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

```
1 %% part a
2 length = 35;
3 n = 1:length;
4 x(n) = 0.8.^n.*heaviside(n)-0.5.*0.8.^(n-1).*heaviside
    (n-1);
5 fft_x = fft(x);
6 complex_cepstrum_recursive = [];
7 complex_cepstrum_definition = ifft(log(fft_x));
8 complex_cepstrum = cceps(x);
9 complex_cepstrum_recursive(1) = log(x(1));
10 for i=2:length
11     a = 0;
12     for j=1:i-1
13         a = a + (j-1)/(i-1)*complex_cepstrum_recursive
            (j)*x(i-(j-1))/x(1);
14     end
15     complex_cepstrum_recursive(i) = x(i)/x(1) - a;
16 end
17 figure;
18 subplot(3,1,1)
19 plot(complex_cepstrum)
```

```

20 title('complex cepstrum by matlab cceps')
21 ylabel('x\^[n]')
22 xlabel('n')
23 subplot(3,1,2)
24 plot(complex_cepstrum_definition)
25 title('complex cepstrum by DFT definition')
26 ylabel('x\^[n]')
27 xlabel('n')
28 subplot(3,1,3)
29 plot(complex_cepstrum_recursive)
30 title('complex cepstrum by recursive definition')
31 ylabel('x\^[n]')
32 xlabel('n')
33 %% part b
34 % disease : cardiac
35 ecg_p1 = readtable("207-try.csv");
36 % disease : Myocardial infarction
37 ecg_p2 = load('s0001.mat');
38 % healthy control
39 ecg_h = load('s0285.mat');
40 figure;
41 subplot(3,1,1)
42 plot(ecg_p1.heart(640:1400))
43 title('ECG for patient (cardiac)')
44 ylabel('x[n]')
45 xlabel('n')
46 subplot(3,1,2)
47 plot(cceps(ecg_p1.heart(640:1400)))
48 title('complex cepstrum for patient (cardiac)')
49 ylabel('x^[n]')
50 xlabel('n')

```

```

51 subplot(3,1,3)
52 plot(rceps(ecg_p1.heart(640:1400)))
53 title('real cepstrum for patient (cardiac)')
54 ylabel('c_x[n]')
55 xlabel('n')
56 figure;
57 subplot(3,1,1)
58 plot(ecg_p2.val(1,1750:2550))
59 title('ECG for patient (Myocardial infarction)')
60 ylabel('x[n]')
61 xlabel('n')
62 subplot(3,1,2)
63 plot(cceps(ecg_p2.val(1,1750:2550)))
64 title('complex cepstrum for patient (Myocardial
        infarction)')
65 ylabel('x^[n]')
66 xlabel('n')
67 subplot(3,1,3)
68 plot(rceps(ecg_p2.val(1,1750:2550)))
69 title('real cepstrum for patient (Myocardial
        infarction)')
70 ylabel('c_x[n]')
71 xlabel('n')
72 figure;
73 subplot(3,1,1)
74 plot(ecg_h.val(1,1750:550+1750))
75 title('ECG for healthy control')
76 ylabel('x[n]')
77 xlabel('n')
78 subplot(3,1,2)
79 plot(cceps(ecg_h.val(1,1750:2550)))

```

```
80 title('complex cepstrum for healthy control')
81 ylabel('x^[n]')
82 xlabel('n')
83 subplot(3,1,3)
84 plot(rceps(ecg_h.val(1,1750:2550)))
85 title('real cepstrum for healthy control')
86 ylabel('c_x[n]')
87 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.