

Part 1

Report

Biological Signal Processing

Ardalan Gerami 99102112

6/2/2024



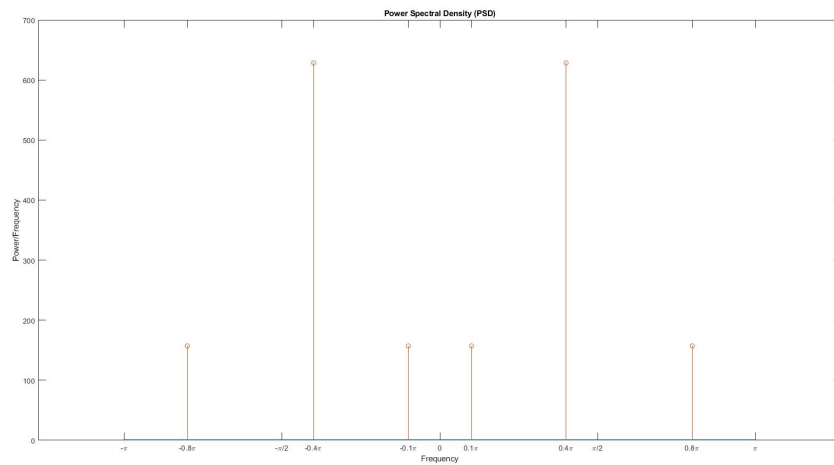
Contents

1	Question 1	3
1.1	part a	3
1.2	part b	3
1.3	part c	4
1.4	part d	4
1.5	part e	5
1.6	part f	7
2	Question 2	20
2.1	part a	20
2.2	part b	21
2.3	part c	23
2.4	part d	24
2.5	part e	25
2.6	part f	26
3	Question 3	36
3.1	part a	36
3.2	part b	36
3.3	part c	36
3.4	part d	37

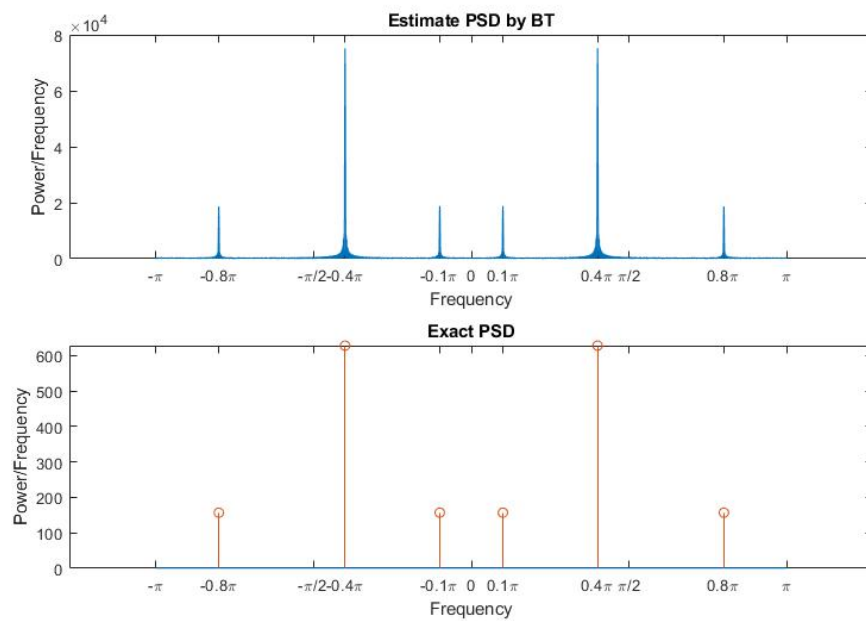
1 Question 1

1.1 part a

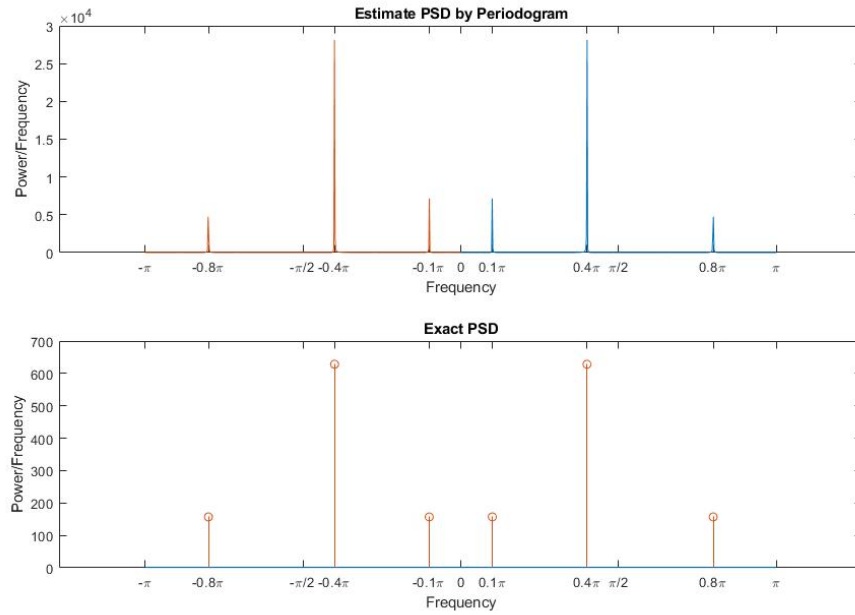
As we know for PSD of $A\cos(\omega_0 n + \phi)$ is $A^2/2 * \pi(\delta(\omega - \omega_0) + \delta(\omega + \omega_0))$ and the PSD of white noise is σ^2 .



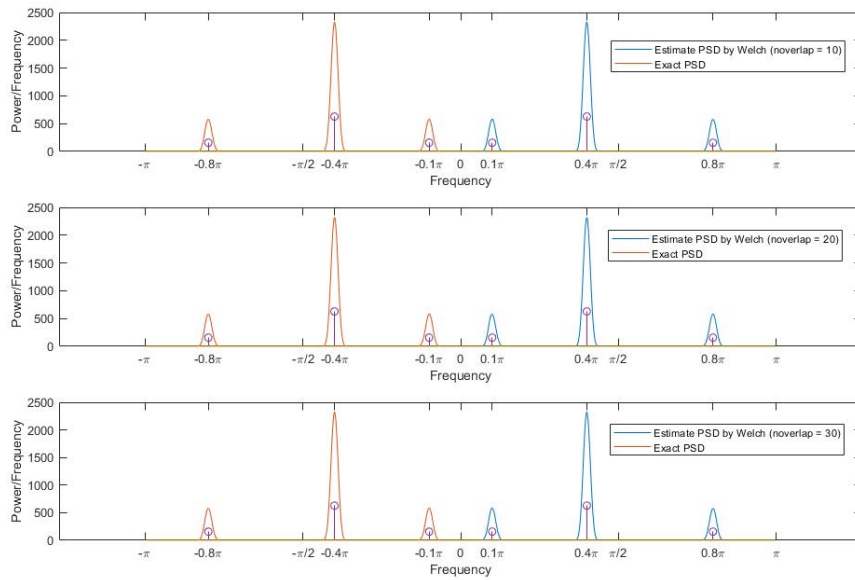
1.2 part b



1.3 part c



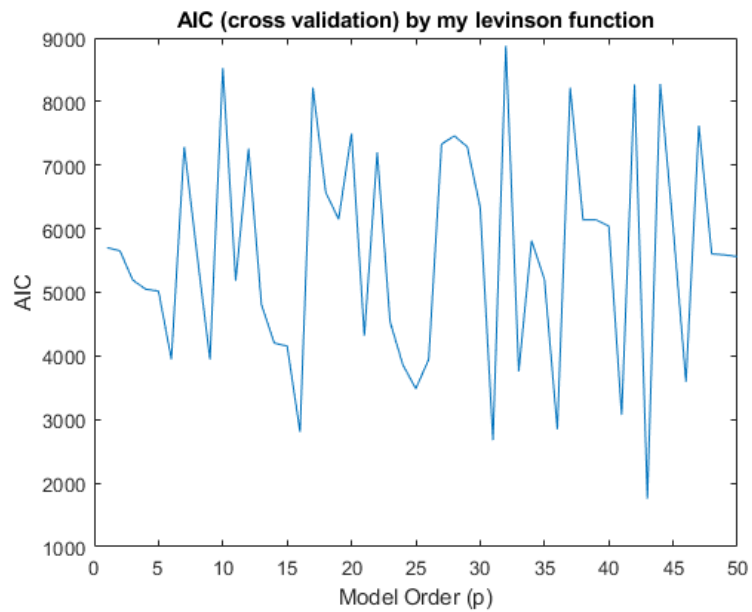
1.4 part d



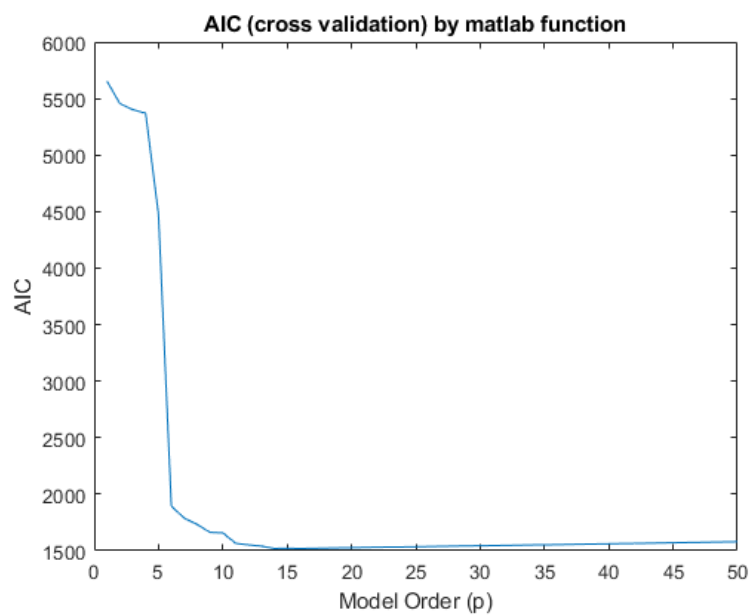
1.5 part e

- I have implemented my levinson function and matlab levin son and these are the results :

- By my function:



- By matlab function:



- optimal order:

Optimal AR Order (p) by Levinson-Durbin:

18

Optimal AR Order (p) by AIC:

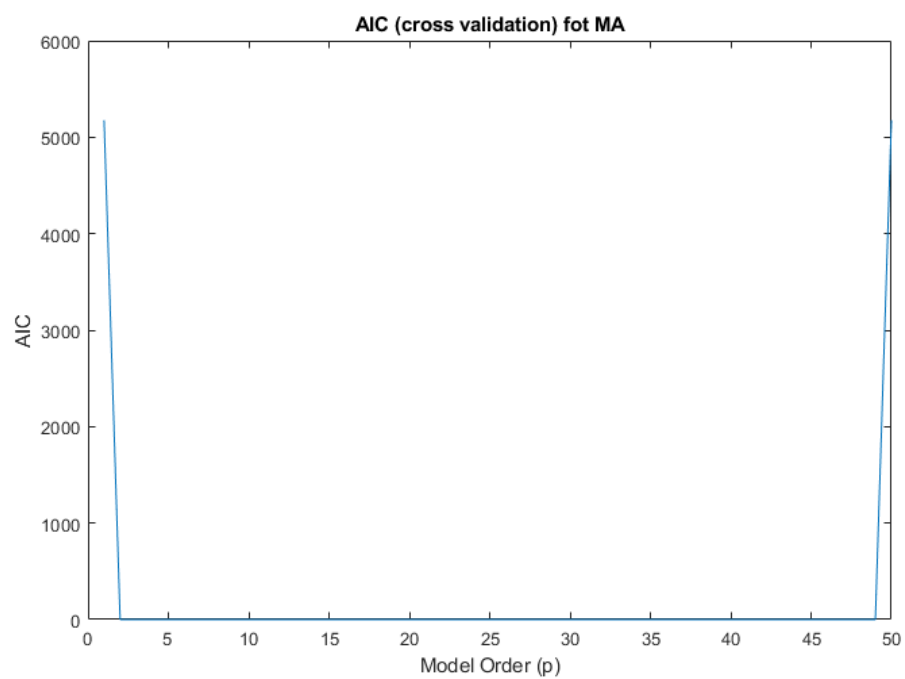
16

-coeffs for optimal order:

Optimal AR coeffs:

-0.1486
-0.0970
0.0584
-0.1814
-0.1533
0.0999
-0.0627
-0.1252
0.1621
0.1041
-0.0832
0.1649
0.1900
-0.0650
0.0728
-0.8610

1.6 part f



Best MA order:
40

Coeffs for best MA order

ans =

0.3391
-0.1467
-0.2139
-0.0768
0.8756
-0.0742
-0.6925
-0.5711
-0.3923
0.5877
-0.4059
-0.7806
-0.4434
-0.2575
0.8292
-0.0131
-0.1147
0.1644
0.0588
1.0000
0.3799
-0.0342
-0.0312
-0.1697
0.8246
0.0725
-0.5141
-0.5473
-0.4314
0.5733
-0.1124
-0.5549
-0.4261
-0.1507
0.5811
0.0476
-0.2711
-0.0392
0.0853
0.3704

Matlab Code

```
1 %% our information
2 samples = 1000;
3
4 alpha1 = 2*pi*rand;
5 alpha2 = 2*pi*rand;
6 alpha3 = 2*pi*rand;
7
8 n = 0:samples-1;
9 x = 10*cos(0.1*pi*n + alpha1) + 20*cos(0.4*pi*n +
    alpha2) + 10*cos(0.8*pi*n + alpha3) + randn(1,
    samples);
10
11
12 %% part a
13
14 f = linspace(-pi, pi, 1000);
15 PSD = ones(size(f));
16 add_delta = @(psd, freq, amp) psd + amp * (f == freq);
17 PSD = add_delta(PSD, 0.1*pi, pi * 50);
18 PSD = add_delta(PSD, -0.1*pi, pi * 50);
19 PSD = add_delta(PSD, 0.4 * pi, pi * 200);
20 PSD = add_delta(PSD, -0.4 * pi, pi * 200);
21 PSD = add_delta(PSD, 0.8 * pi, pi * 50);
22 PSD = add_delta(PSD, -0.8 * pi, pi * 50);
23 figure;
24 plot(f, PSD);
25 title('Power Spectral Density (PSD)');
26 xlabel('Frequency');
27 ylabel('Power / Frequency');
28 hold on;
```

```

29 stem([0.1*pi, -0.1*pi, 0.4 * pi, -0.4 * pi, 0.8 * pi,
      -0.8 * pi], ...
30      [pi * 50, pi * 50, pi * 200, pi * 200, pi * 50, pi
      * 50]);
31 xticks([-pi -0.8*pi -pi/2 -0.4*pi -0.1*pi 0 0.1*pi
      0.4*pi pi/2 0.8*pi pi]);
32 xticklabels({'-\pi', '-0.8\pi', '-\pi/2', '-0.4\pi',
      '-0.1\pi', '0', '0.1\pi', '0.4\pi', '\pi/2', '0.8\
      pi', '\pi'});
33
34
35
36 %% part b
37
38 estimated_corr = zeros(1, samples/2);
39 BT_PSD = zeros(1, 20000);
40 f1 = linspace(-pi, pi, 20000);
41
42 for i=1:samples/2
43     for j = 1:(samples-i)
44         estimated_corr(i) = x(j)*x(j+i) +
45             estimated_corr(i);
46     end
47     estimated_corr(i) = estimated_corr(i) / samples;
48 end
49 for w=1:20000
50     temp = 0;
51     for j=1:(samples/2)
52         temp = estimated_corr(j) * exp(-1i*f1(w)*j) +
53             temp;
54     end

```

```

53     BT_PSD(w) = estimated_corr(1) + 2 * real(temp);
54 end
55
56 figure;
57 subplot(2,1,1);
58 plot(f1,abs(BT_PSD))
59 title('Estimate PSD by BT');
60 xlabel('Frequency');
61 ylabel('Power/Frequency');
62 xticks([-pi -0.8*pi -pi/2 -0.4*pi -0.1*pi 0 0.1*pi
        0.4*pi pi/2 0.8*pi pi]);
63 xticklabels({'-\pi', '-0.8\pi', '-\pi/2', '-0.4\pi',
        '-0.1\pi', '0', '0.1\pi', '0.4\pi', '\pi/2', '0.8\pi', '\pi'});
64
65 subplot(2,1,2);
66 plot(f, PSD);
67 hold on;
68 title('Exact PSD');
69 xlabel('Frequency');
70 ylabel('Power/Frequency');
71 stem([0.1*pi, -0.1*pi, 0.4 * pi, -0.4 * pi, 0.8 * pi,
        -0.8 * pi], ...
72      [pi * 50, pi * 50, pi * 200, pi * 200, pi * 50, pi
        * 50]);
73 xticks([-pi -0.8*pi -pi/2 -0.4*pi -0.1*pi 0 0.1*pi
        0.4*pi pi/2 0.8*pi pi]);
74 xticklabels({'-\pi', '-0.8\pi', '-\pi/2', '-0.4\pi',
        '-0.1\pi', '0', '0.1\pi', '0.4\pi', '\pi/2', '0.8\pi', '\pi'});
75

```

```

76 %% part c
77
78 [pxx, f2] = periodogram(x,[],[],2*pi);
79
80 figure;
81 subplot(2,1,1);
82 plot(f2, pxx); % Adjust frequency range to [-pi, pi]
83 hold on;
84 plot(-f2, pxx)
85 title('Estimate PSD by Periodogram');
86 xlabel('Frequency');
87 ylabel('Power/Frequency');
88 xticks([-pi -0.8*pi -pi/2 -0.4*pi -0.1*pi 0 0.1*pi
          0.4*pi pi/2 0.8*pi pi]);
89 xticklabels({'-\pi', '-0.8\pi', '-\pi/2', '-0.4\pi',
              '-0.1\pi', '0', '0.1\pi', '0.4\pi', '\pi/2', '0.8\pi', '\pi'});
90
91 subplot(2,1,2);
92 plot(f, PSD);
93 hold on;
94 stem([0.1*pi, -0.1*pi, 0.4 * pi, -0.4 * pi, 0.8 * pi,
        -0.8 * pi], ...
95      [pi * 50, pi * 50, pi * 200, pi * 200, pi * 50, pi
        * 50]);
96 title('Exact PSD');
97 xlabel('Frequency');
98 ylabel('Power/Frequency');
99 xticks([-pi -0.8*pi -pi/2 -0.4*pi -0.1*pi 0 0.1*pi
          0.4*pi pi/2 0.8*pi pi]);
100 xticklabels({'-\pi', '-0.8\pi', '-\pi/2', '-0.4\pi',

```

```

    '-0.1\pi', '0', '0.1\pi', '0.4\pi', '\pi/2', '0.8\
    pi', '\pi'}));
101
102
103
104
105
106 %% part d
107
108
109 [pxx_welch_10, f3] = pwelch(x, 100, 10, samples, 2*pi)
    ;
110 [pxx_welch_20, f4] = pwelch(x, 100, 20, samples, 2*pi)
    ;
111 [pxx_welch_30, f5] = pwelch(x, 100, 30, samples, 2*pi)
    ;
112
113 figure;
114 subplot(3,1,1);
115 plot(f3, pxx_welch_10);
116 hold on;
117 plot(-f3, pxx_welch_10)
118 hold on;
119 plot(f, PSD);
120 hold on;
121 stem([0.1*pi, -0.1*pi, 0.4 * pi, -0.4 * pi, 0.8 * pi,
    -0.8 * pi], ...
122      [pi * 50, pi * 50, pi * 200, pi * 200, pi * 50, pi
    * 50]);
123 xlabel('Frequency');
124 ylabel('Power / Frequency');

```

```

125 legend('Estimate PSD by Welch (noverlap = 10)', 'Exact
      PSD')
126 xticks([-pi -0.8*pi -pi/2 -0.4*pi -0.1*pi 0 0.1*pi
      0.4*pi pi/2 0.8*pi pi]);
127 xticklabels({'-\pi', '-0.8\pi', '-\pi/2', '-0.4\pi',
      '-0.1\pi', '0', '0.1\pi', '0.4\pi', '\pi/2', '0.8\
      pi', '\pi'});

128
129 %%%%%%%%%%
130
131 subplot(3,1,2);
132 plot(f4, pxx_welch_20);
133 hold on;
134 plot(-f4, pxx_welch_20)
135 hold on;
136 plot(f, PSD);
137 hold on;
138 stem([0.1*pi, -0.1*pi, 0.4 * pi, -0.4 * pi, 0.8 * pi,
      -0.8 * pi], ...
139      [pi * 50, pi * 50, pi * 200, pi * 200, pi * 50, pi
      * 50]);
140 xlabel('Frequency');
141 ylabel('Power/Frequency');
142 legend('Estimate PSD by Welch (noverlap = 20)', 'Exact
      PSD')
143 xticks([-pi -0.8*pi -pi/2 -0.4*pi -0.1*pi 0 0.1*pi
      0.4*pi pi/2 0.8*pi pi]);
144 xticklabels({'-\pi', '-0.8\pi', '-\pi/2', '-0.4\pi',
      '-0.1\pi', '0', '0.1\pi', '0.4\pi', '\pi/2', '0.8\
      pi', '\pi'});

145

```

```

146 %%%
147
148
149 subplot(3,1,3);
150 plot(f5, pxx_welch_30);
151 hold on;
152 plot(-f5, pxx_welch_30)
153 hold on;
154 plot(f, PSD);
155 hold on;
156 stem([0.1*pi, -0.1*pi, 0.4 * pi, -0.4 * pi, 0.8 * pi,
      -0.8 * pi], ...
157      [pi * 50, pi * 50, pi * 200, pi * 200, pi * 50, pi
      * 50]);
158 xlabel('Frequency');
159 ylabel('Power/Frequency');
160 legend('Estimate PSD by Welch (noverlap = 30)', 'Exact
      PSD')
161 xticks([-pi -0.8*pi -pi/2 -0.4*pi -0.1*pi 0 0.1*pi
      0.4*pi pi/2 0.8*pi pi]);
162 xticklabels({'-\pi', '-0.8\pi', '-\pi/2', '-0.4\pi',
      '-0.1\pi', '0', '0.1\pi', '0.4\pi', '\pi/2', '0.8\
      pi', '\pi'});
163
164
165
166
167
168 %% part e
169
170

```

```

171 R_x = xcorr(x, 'biased');
172
173 % implementing levinson by myself and AIC cross
    valiadtation
174 E = [];
175 K=[];
176 AIC = zeros(50,1);
177 temp = 0;
178 R_xx = R_x(1000:1999);
179 for i = 1:50
180     if i == 1
181         E(i) = R_xx(i);
182     elseif i == 2
183         k(i) = -R_xx(i) / E(i-1);
184         a(i,i) = k(i);
185         E(i) = (1 - k(i)^2) * E(i-1);
186     else
187         for j = 1:i-1
188             temp = temp + a(j,i-1) * R_xx(i-j);
189         end
190         k(i) = -(R_xx(i) + temp) / E(i-1);
191         a(i,i) = k(i);
192         for j = 1:i-1
193             a(j,i) = a(j,i-1) + k(i) * a(i-j,i-1);
194         end
195         E(i) = (1 - k(i)^2) * E(i-1);
196     end
197     AIC(i) = samples * log(E(i)) + 2 * (i);
198     temp = 0;
199 end
200 [~, p_opt_AIC] = min(AIC);

```



```

201 p_opt_LD = 1;
202 disp('Optimal AR Order (p) by Levinson-Durbin:');
203 disp(p_opt_aic);
204 disp('Optimal AR Order (p) by AIC:');
205 disp(p_opt_AIC);
206
207 figure;
208 plot(1:50, AIC);
209 title('AIC (cross validation) by my levinson function'
        );
210 xlabel('Model Order (p)');
211 ylabel('AIC');
212 disp('Optimal AR coeffs:');
213 disp(a(:,p_opt_LD));
214 disp('Optimal AR coeffs:');
215 disp(a(:,p_opt_AIC));
216 % implementing levinson by matlab function and AIC
        cross valiadtion
217
218 e = zeros(51,1);
219 aic = zeros(50,1);
220 [a1, e(1)] = levinson(R_xx, 0); % Order 0 model
221 for p = 1:50
222     [a1, e(p+1)] = levinson(R_xx, p);
223     aic(p) = samples*log(e(p+1))+2*p;
224 end
225 [~, p_opt_aic] = min(aic);
226 p_opt_LD_m = 1;
227 disp('Optimal AR Order (p) by Levinson-Durbin:');
228 disp(p_opt_aic);
229 disp('Optimal AR Order (p) by AIC:');

```

```

230 disp(p_opt_aic);
231
232
233 disp('Optimal AR coeffs:');
234 disp(a(:,p_opt_LD_m));
235 disp('Optimal AR coeffs:');
236 disp(a(:,p_opt_aic));
237
238 figure;
239 plot(aic);
240 title('AIC (cross validation) by matlab function');
241 xlabel('Model Order (p)');
242 ylabel('AIC');
243
244
245 %% part f
246
247 bestOrder = 0;
248 bestAIC = Inf;
249 for q = 1:50
250     try
251         model = arima('MALags', 1:q, 'Constant', 0);
252         fit = estimate(model, x', 'Display', 'off');
253         [~,~,logL] = infer(fit, x');
254         numParams = q;
255         aic(i) = -2 * logL + 2 * numParams;
256         if aic(i) < bestAIC
257             bestOrder = q;
258             bestAIC = aic(i);
259             bestFit = fit;
260         end

```

```

261         catch
262             continue;
263         end
264     end
265
266     disp('Best MA order: ')
267     disp(num2str(bestOrder));
268     disp('MA coefficients:');
269     disp(cell2mat(bestFit.MA)');
270
271     figure;
272     plot(aic);
273     title('AIC (cross validation) fot MA');
274     xlabel('Model Order (p)');
275     ylabel('AIC');

```

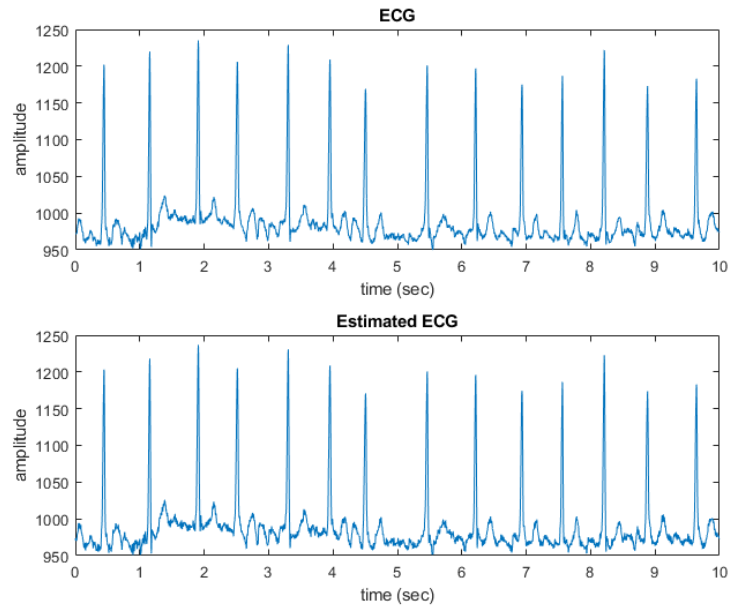
2 Question 2

2.1 part a

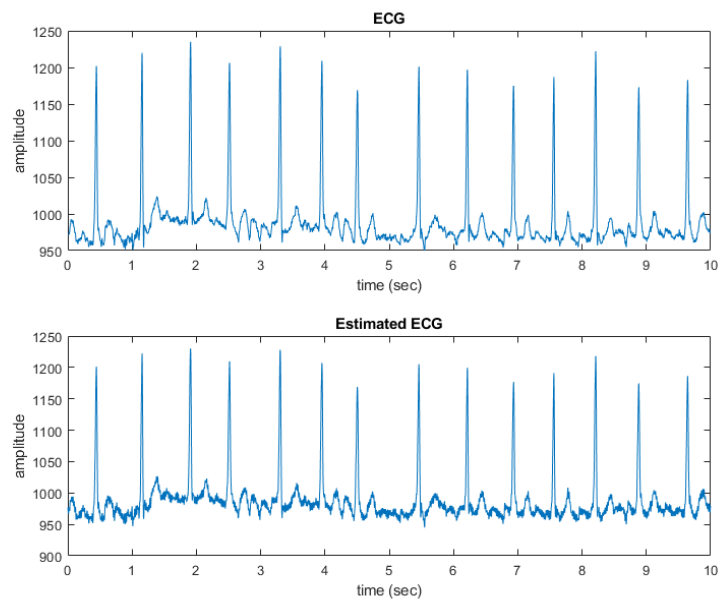
```
1 %% a
2 ECG_dataset = load('test.mat');
3 fs = 360;
4 s = ECG_dataset.val;
5 t = 0:1/fs:(length(s)-1)/fs;
6 phi1 = 2*pi*rand;
7 phi2 = 2*pi*rand;
8 N1 = 2*cos(100*pi*t+phi1);
9 N2 = 2*cos(100*pi*t+phi2);
10 reference_signal = N1;
11 primary_signal = s + N2;
12 w = zeros(length(s),1);
13 y = zeros(length(s),1);
14 e = zeros(length(s),1);
15 mu = 0.000001;
16 p = y-w;
17 [e,y] = adaptivefilter(w,reference_signal,
    primary_signal,mu);
18 %% adaptive filter function :
19
20 function [e,y] = adaptivefilter(w,reference_signal,
    primary_signal,mu)
21     for i=1:length(reference_signal)
22         y = reference_signal*w;
23         e = primary_signal - y;
24         w = w + 2*mu*e*reference_signal';
25     end
26 end
```

2.2 part b

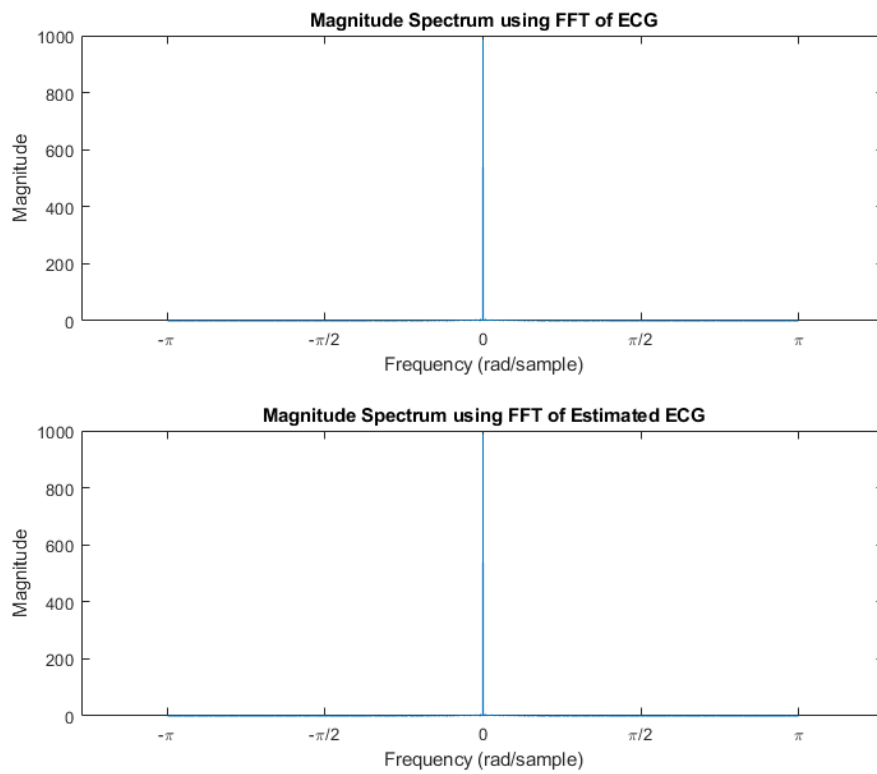
with Amplitude = 5 for noises :



with Amplitude = 20 for noises :



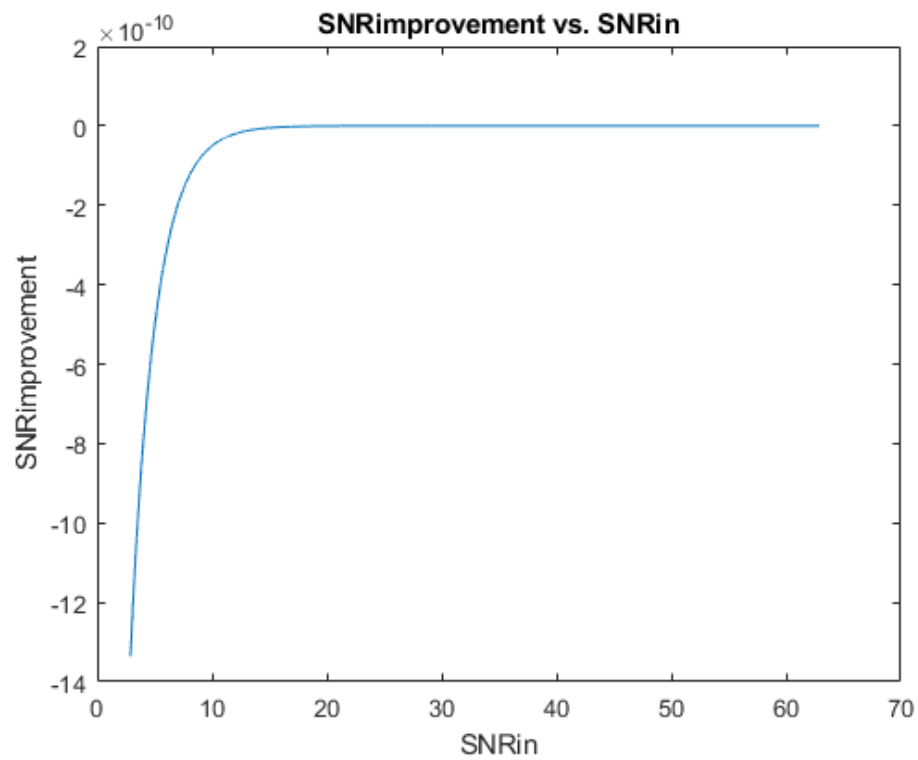
* As we can see still we have a little amount of effect from the noise , which added to the ECG signal, but we have estimated the ECG signal properly. Also we have to say that the effect of noise relate to the amplitude directly.



* As we can see there is almost no difference in spectrum of the ECG and estimated ECG.

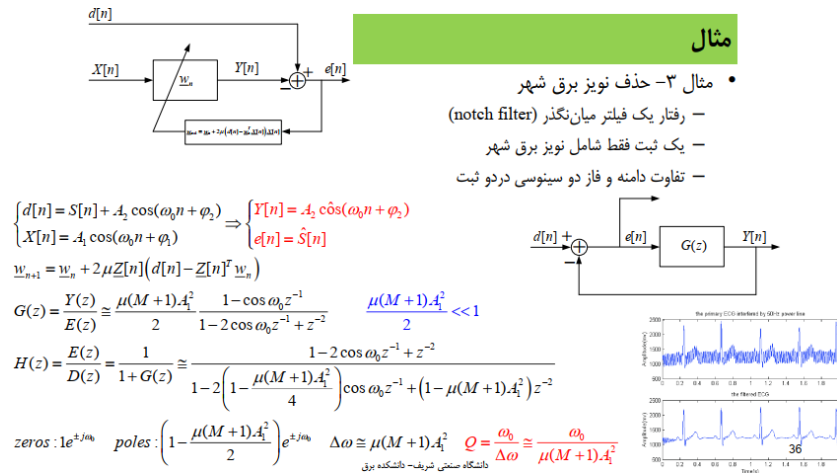
2.3 part c

- For different values of amplitude we have below figure, As we can see after some amplitudes the SNRimprovement will be constant even by increasing SNRin and converge.

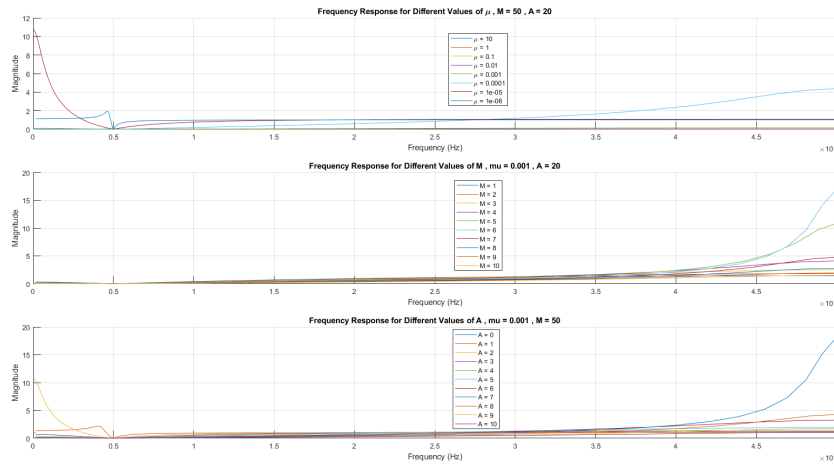


2.4 part d

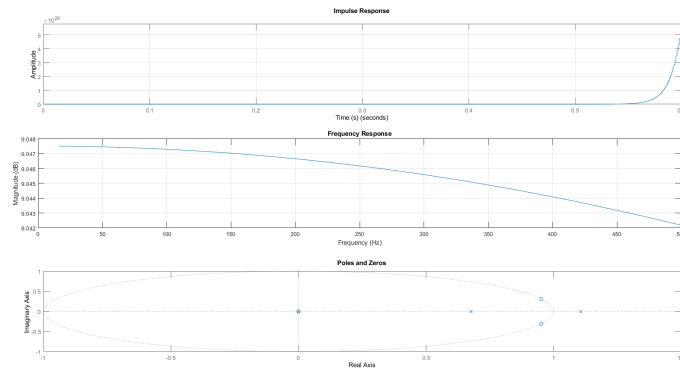
* As we had in slides :



- Now for different values of mu,A and M I've plotted frequency responses:

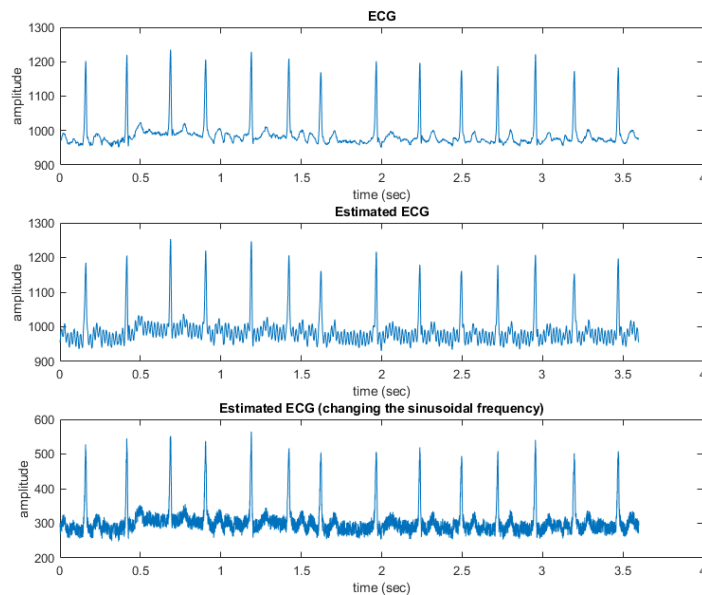


* Now for $\mu = 0.0001$, $A = 5$, $M = 100$:



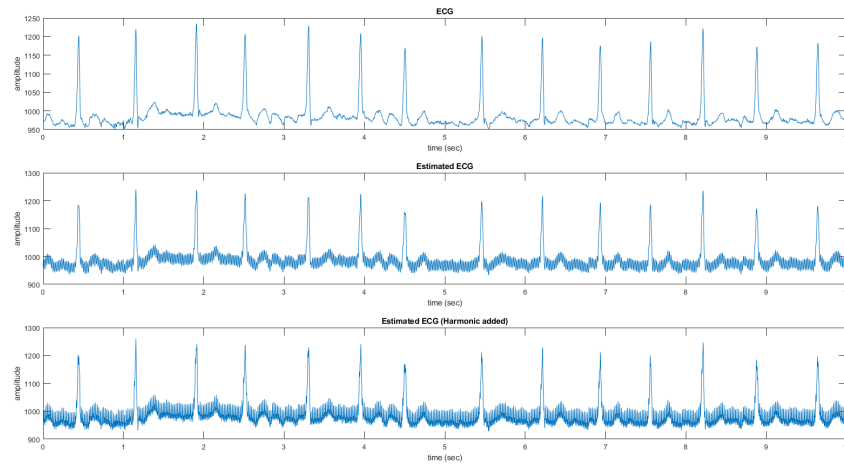
2.5 part e

- For analysis the effect of a little change in sinusoidal frequencies, I have added noise to the frequency and we will have :



* As we can see the output of Adaptive filter would be noisier than previous estimated ECG.

2.6 part f



* As we can see by adding the first harmonic component Adaptive filter can remove less noise from primary signal but still estimation works almost properly.

Matlab Code

```
1 %% a
2 ECG_dataset = load('test.mat');
3 fs = 360;
4 s = ECG_dataset.val;
5 t = 0:1/fs:(length(s)-1)/fs;
6 phi1 = 2*pi*rand;
7 phi2 = 2*pi*rand;
8 N1 = 20*cos(100*pi*t+phi1);
9 N2 = 20*cos(100*pi*t+phi2);
10 reference_signal = N1;
11 primary_signal = s + N2;
12 w = zeros(length(s),1);
13 y = zeros(length(s),1);
14 e = zeros(length(s),1);
```

```

15 mu = 0.000001;
16 p = y-w;
17 [e,y] = adaptivefilter(w,reference_signal ,
    primary_signal ,mu);
18
19 %% b
20
21 figure ;
22
23 subplot(2,1,1)
24 plot(t,s)
25 title('ECG');
26 xlabel('time (sec)');
27 ylabel('amplitude');
28
29 subplot(2,1,2)
30 plot(t,e)
31 title('Estimated ECG');
32 xlabel('time (sec)');
33 ylabel('amplitude');
34
35 f = (-length(s)/2:length(s)/2-1)*(2*pi/length(s));
36 figure ;
37 subplot(2,1,1)
38 plot(f, abs(fftshift(fft(s)))/length(s));
39 title('Magnitude Spectrum using FFT of ECG');
40 xlabel('Frequency (rad/sample)');
41 ylabel('Magnitude');
42 xticks([-pi -pi/2 0 pi/2 pi]);
43 xticklabels({'-\pi', '-\pi/2', '0', '\pi/2', '\pi'})
    ;

```

```

44
45 subplot(2,1,2)
46 plot(f, abs(fftshift(fft(e)))/length(e))
47 title('Magnitude Spectrum using FFT of Estimated ECG')
48 ;
49 xlabel('Frequency (rad/sample)');
50 ylabel('Magnitude');
51 xticks([-pi -pi/2 0 pi/2 pi]);
52 xticklabels({'-\pi', '-\pi/2', '0', '\pi/2', '\pi'})
53 ;
54
55 %% c
56
57 for i = 0:1000
58     N1 = i*cos(100*pi*t+phi1);
59     N2 = i*cos(100*pi*t+phi2);
60     reference_signal = N1;
61     primary_signal = s + N2;
62     w = zeros(length(s),1);
63     y = zeros(length(s),1);
64     e = zeros(length(s),1);
65     mu = 0.000001;
66     [e,y] = adaptivefilter(w,reference_signal ,
67         primary_signal ,mu);
68     SNRin(i+1) = 10*log10(norm(s)^2/norm(N2)^2);
69     SNRout(i+1) = 10*log10(norm(s)^2/(norm(e-s)^2));
70     SNRimprovement(i+1) = SNRout(i+1) - SNRin(i+1);
71 end
72
73 figure;
74 plot(SNRin,SNRimprovement)

```

```

72 title('SNRimprovement vs. SNRin');
73 xlabel('SNRin');
74 ylabel('SNRimprovement');
75
76
77 %% part d
78
79
80 f = 50;
81 z = tf('z', 1/fs);
82 figure;
83 mu_values = [10 1 0.1 0.01 0.001 0.0001 0.00001
84             0.000001];
85 M = 50;
86 A = 20;
87 for i = 1:length(mu_values)
88     mu = mu_values(i);
89     H = (1 - 2*cos(2*pi*f/fs)*z^(-1)+z^(-2)) / ...
          (1 - 2*(1 - (mu*(M+1)*A^2)/4)*cos(2*pi*f/fs)*z
          ^(-1) + (1 - mu*(M+1)*A^2)*z^(-2));
90     subplot(3, 1, 1);
91     hold on;
92     [mag, phase, w] = bode(H);
93     plot(w*fs/(2*pi), mag(:));
94 end
95 title('Frequency Response for Different Values of \mu'
96       ');
97 xlabel('Frequency (Hz)');
98 ylabel('Magnitude');
99 legend(arrayfun(@(x) ['\mu = ' num2str(x)], mu_values,
100                'UniformOutput', false));

```

```

99  grid on;
100
101  mu = 0.001;
102  M_values = 1:10;
103  A = 20;
104
105  for M = M_values
106      H = (1 - 2*cos(2*pi*f/fs)*z^(-1)+z^(-2)) / ...
107          (1 - 2*(1 - (mu*(M+1)*A^2)/4)*cos(2*pi*f/fs)*z
              ^(-1) + (1 - mu*(M+1)*A^2)*z^(-2));
108      subplot(3, 1, 2);
109      hold on;
110      [mag, phase, w] = bode(H);
111      plot(w*fs/(2*pi), mag(:));
112  end
113  title('Frequency Response for Different Values of M');
114  xlabel('Frequency (Hz)');
115  ylabel('Magnitude');
116  legend(arrayfun(@(x) ['M = ' num2str(x)], M_values, '
      UniformOutput', false));
117  grid on;
118  mu = 0.001;
119  M = 50;
120  A_values = 0:10;
121  for A = A_values
122      H = (1 - 2*cos(2*pi*f/fs)*z^(-1)+z^(-2)) / ...
123          (1 - 2*(1 - (mu*(M+1)*A^2)/4)*cos(2*pi*f/fs)*z
              ^(-1) + (1 - mu*(M+1)*A^2)*z^(-2));
124      subplot(3, 1, 3);
125      hold on;
126      [mag, phase, w] = bode(H);

```

```

127     plot(w*fs/(2*pi), mag(:));
128 end
129 title('Frequency Response for Different Values of A');
130 xlabel('Frequency (Hz)');
131 ylabel('Magnitude');
132 legend(arrayfun(@(x) ['A = ' num2str(x)], A_values, '
    UniformOutput', false));
133 grid on;
134 hold off;
135 %%%%%%%%%%%
136 mu = 0.0001;
137 A = 5;
138 M = 100;
139 figure;
140 subplot(3, 1, 1);
141 impulse(H);
142 title('Impulse Response');
143 xlabel('Time (s)');
144 ylabel('Amplitude');
145 grid on;
146 subplot(3, 1, 2);
147 [mag, phase, w] = bode(H, {0, pi});
148 mag = squeeze(mag);
149 w = squeeze(w);
150 plot(w*fs/(2*pi), 20*log10(mag));
151 title('Frequency Response');
152 xlabel('Frequency (Hz)');
153 ylabel('Magnitude (dB)');
154 grid on;
155 subplot(3, 1, 3);
156 pzmap(H);

```

```

157 title('Poles and Zeros');
158
159
160 %% part e
161
162
163 t = 0:1/fs:(length(s)-1)/fs;
164 phi1 = 2*pi*rand;
165 phi2 = 2*pi*rand;
166 noise = rand(1,length(t));
167 N1 = 20*cos(100*pi*(t+noise)+phi1);
168 N2 = 20*cos(100*pi*(t+noise)+phi2);
169 reference_signal = N1;
170 primary_signal = s + N2;
171 w = zeros(length(s),1);
172 y = zeros(length(s),1);
173 e = zeros(length(s),1);
174 mu = 0.000001;
175 [e,y] = adaptivefilter(w,reference_signal,
    primary_signal,mu);
176
177 figure;
178
179 subplot(2,1,1)
180 plot(t,s)
181 title('ECG');
182 xlabel('time (sec)');
183 ylabel('amplitude');
184
185 subplot(2,1,2)
186 plot(t,e)

```



```

187 title('Estimated ECG');
188 xlabel('time (sec)');
189 ylabel('amplitude');
190
191
192 %% part f
193
194 phi1 = 2*pi*rand;
195 phi2 = 2*pi*rand;
196 phi3 = 2*pi*rand;
197 N1 = 20*cos(100*pi*t+phi1);
198 N2 = 20*cos(100*pi*t+phi2);
199 reference_signal = N1;
200 primary_signal = s + N2;
201 w = zeros(length(s),1);
202 y = zeros(length(s),1);
203 e = zeros(length(s),1);
204 mu = 0.000001;
205 [eh1,yh1] = adaptivefilter(w,reference_signal,
    primary_signal,mu);
206
207 noise = rand(1,length(t));
208 N1 = 20*cos(100*pi*(t+noise)+phi1);
209 N2 = 20*cos(100*pi*(t+noise)+phi2);
210 reference_signal = N1;
211 primary_signal = s + N2;
212 w = zeros(length(s),1);
213 y = zeros(length(s),1);
214 e = zeros(length(s),1);
215 mu = 0.000001;
216 [eh,yh] = adaptivefilter(w,reference_signal,

```

```

        primary_signal,mu);
217
218 figure;
219 subplot(3,1,1)
220 plot(t,s)
221 title('ECG');
222 xlabel('time (sec)');
223 ylabel('amplitude');
224
225 subplot(3,1,2)
226 plot(t,eh1)
227 title('Estimated ECG');
228 xlabel('time (sec)');
229 ylabel('amplitude');
230
231 subplot(3,1,3)
232 plot(t,eh)
233 title('Estimated ECG (changing the sinusoidal
        frequency)');
234 xlabel('time (sec)');
235 ylabel('amplitude');
236
237
238
239 %% part g
240
241 phi1 = 2*pi*rand;
242 N2 = 20*cos(100*pi*t+phi2);
243 primary_signal = s + N2;
244 w = zeros(length(s),1);
245 mu = 0.000001;

```

```

246 for i = 1:15
247     [eal(i), yal(i)] = ALEfilter(w, primary_signal ,
        primary_signal , mu, i);
248     SNRinal(i) = 10*log10(norm(s)^2/norm(N2)^2);
249     SNRoutal(i) = 10*log10(norm(s)^2/(norm(e(i)-s)^2))
        ;
250     SNRimprovemental(i) = SNRoutal(i) - SNRinal(i);
251 end
252 figure ;
253 plot(SNRin, SNRimprovement)
254 title('SNRimprovement');
255 xlabel('sample');
256 ylabel('SNRimprovement');
257
258 %% Adaptive filter function :
259
260 function [e,y] = adaptivefilter(w,reference_signal ,
    primary_signal ,mu)
261     for i=1:length(reference_signal)
262         y = reference_signal*w;
263         e = primary_signal - y;
264         w = w + 2*mu*e*reference_signal';
265     end
266 end

```

3 Question 3

3.1 part a

Article: "Model-based Prediction of Heart Rate Variability"

Summary: This study focuses on the use of parametric models for predicting heart rate variability (HRV) in clinical settings. The authors utilize autoregressive models to forecast HRV based on historical data, helping in the early detection of cardiac conditions. The approach allows for real-time monitoring and prediction of heart anomalies, providing valuable insights for preventive healthcare. By leveraging parametric modeling, the system adapts to individual patient data, enhancing the accuracy and reliability of predictions.

3.2 part b

Article: "Super-resolution spectral estimation in short-time

non-contact vital sign measurement"

Summary: This study applies non-parametric spectral estimation methods to non-contact vital sign measurement using Doppler radar. Techniques such as short-time Fourier transform are employed to analyze the spectral content of radar signals reflected from the body. This allows for accurate extraction of vital signs like heart rate and respiratory rate in real-time, proving beneficial for remote patient monitoring and emergency medical applications.

3.3 part c

Article: "Parametric Spectral Estimation for Sleep Apnea Detection"

Summary: The research discusses the use of parametric spectral estimation techniques, specifically autoregressive (AR) modeling, to detect sleep apnea events from respiratory signals. By estimating the power spectral density of the respiratory signal, the method identifies characteristic patterns associated with apnea episodes. The parametric approach allows for high-resolution spectral analysis, making it possible to detect subtle changes in the signal that are indicative of apnea, thus aiding in the accurate diagnosis and monitoring of sleep disorders.

3.4 part d

Article: "Adaptive Filtering by Non-Invasive Vital Signals Monitoring and Diseases Diagnosis" Summary: The paper examines the use of adaptive filters, particularly LMS and RLS algorithms, in processing ECG and PPG signals. These filters dynamically adjust to varying noise conditions, effectively removing artifacts and enhancing signal quality. This facilitates accurate measurement of vital parameters such as heart rate and oxygen saturation, improving the reliability of non-invasive monitoring systems in clinical and home settings.