# Part 1

Report

Biological Signal Processing

Ardalan Gerami 99102112

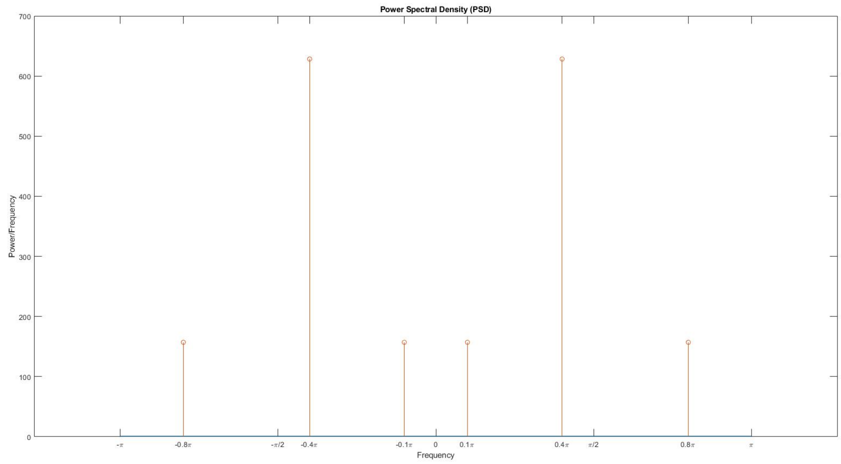
6/2/2024

# Contents

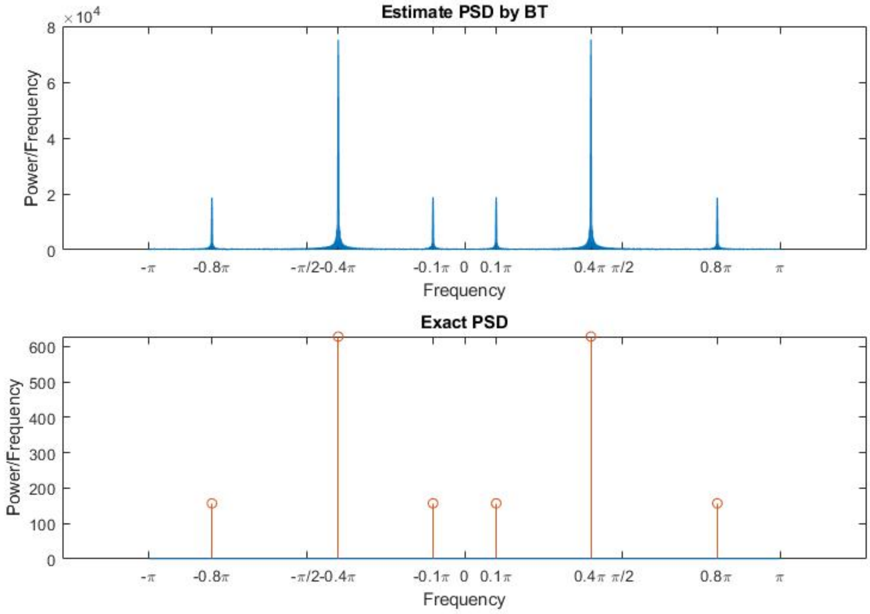
# 1 Question 1

## 1.1 part a

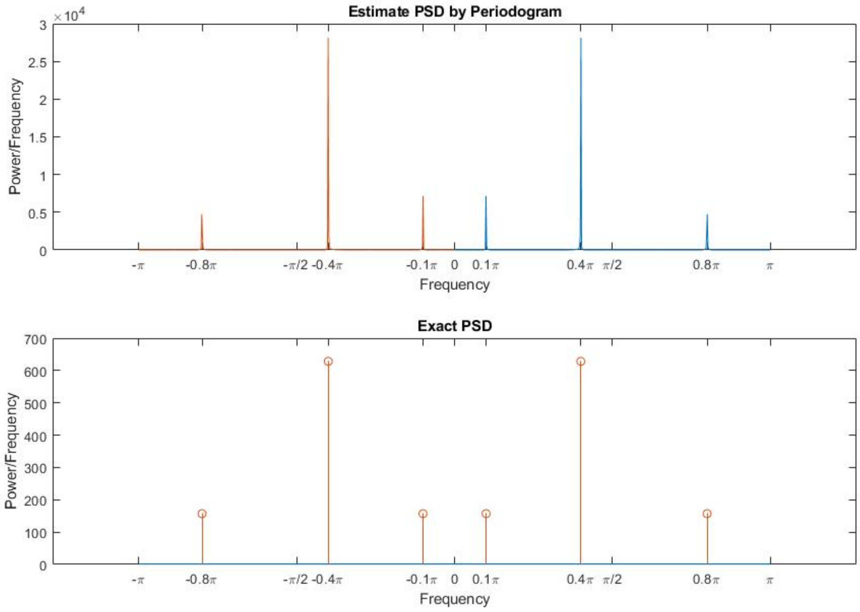
As we know for PSD of $Acos(\omega 0n + phi)$ is $A^2/2 * \pi(\delta(\omega - \omega 0) + \delta(\omega + \omega 0))$ and the PSD of white noise is $\sigma^2$.



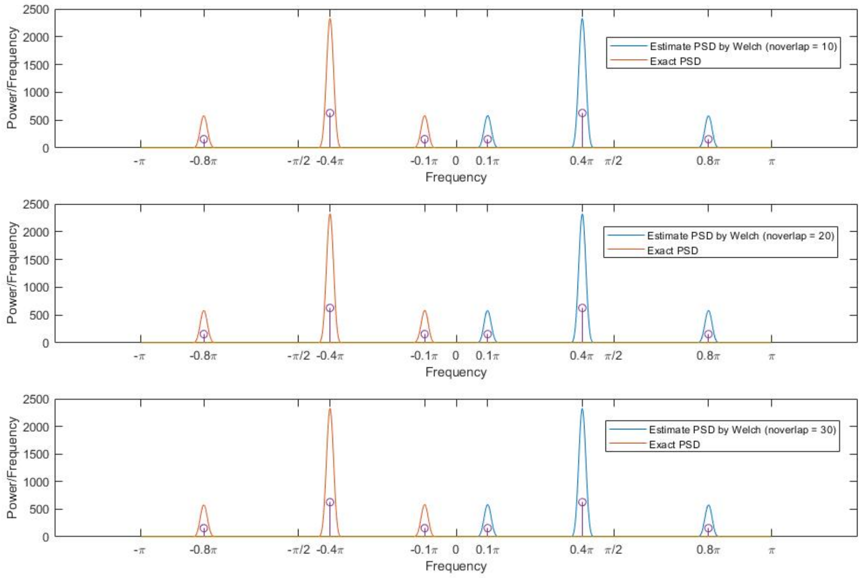

## 1.2 part b



## 1.3 part c

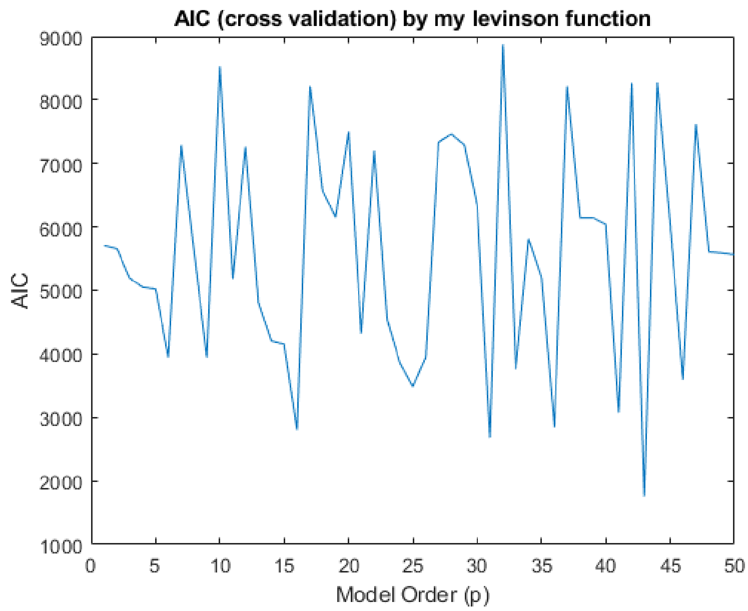



## 1.4 part d

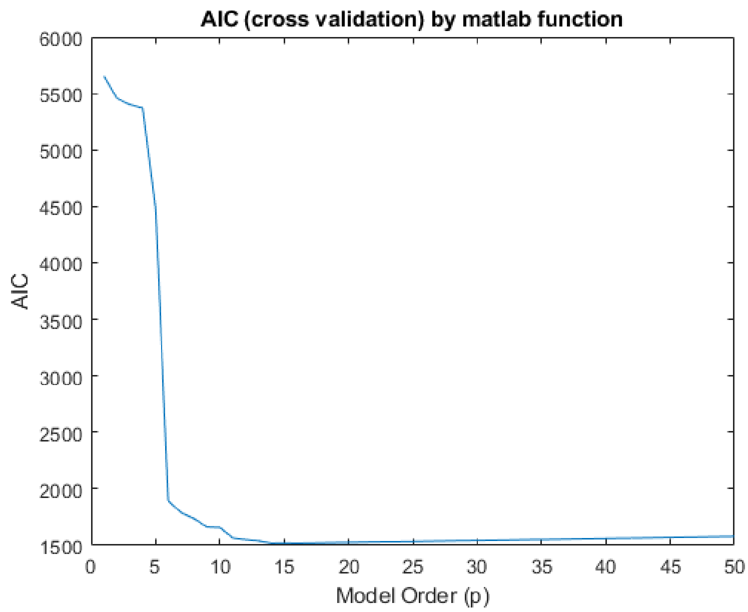


## 1.5 part e

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




- By matlab function:



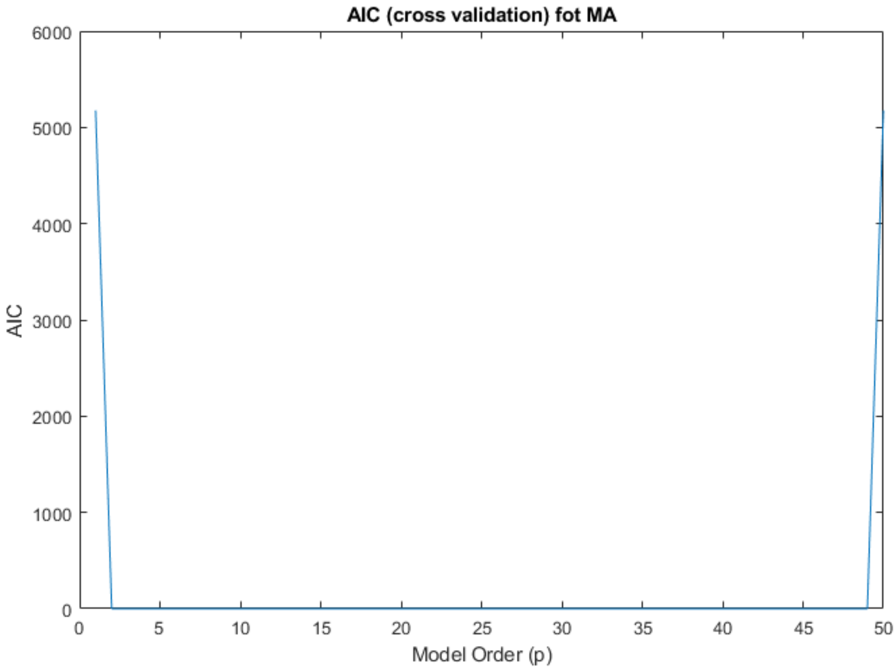
- optimal order:

```
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

```
%% our information
samples = 1000;

alpha1 = 2*pi*rand;
alpha2 = 2*pi*rand;
alpha3 = 2*pi*rand;

n = 0:samples-1;
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);


%% part a

f = linspace(-pi, pi, 1000);
PSD = ones(size(f));
add_delta = @(psd, freq, amp) psd + amp * (f == freq);
PSD = add_delta(PSD, 0.1*pi, pi * 50);
PSD = add_delta(PSD, -0.1*pi, pi * 50);
PSD = add_delta(PSD, 0.4 * pi, pi * 200);
PSD = add_delta(PSD, -0.4 * pi, pi * 200);
PSD = add_delta(PSD, 0.8 * pi, pi * 50);
PSD = add_delta(PSD, -0.8 * pi, pi * 50);
figure;
plot(f, PSD);
title('Power Spectral Density (PSD)');
xlabel('Frequency');
ylabel('Power/Frequency');
hold on;
```

```matlab
stem([0.1*pi, -0.1*pi, 0.4 * pi, -0.4 * pi, 0.8 * pi, -0.8 * pi], ...
    [pi * 50, pi * 50, pi * 200, pi * 200, pi * 50, pi * 50]);
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]);
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'});



%% part b

estimated_corr = zeros(1,samples/2);
BT_PSD = zeros(1, 20000);
f1 = linspace(-pi, pi, 20000);

for i=1:samples/2
    for j = 1:(samples-i)
        estimated_corr(i) = x(j)*x(j+i) + estimated_corr(i);
    end
    estimated_corr(i) = estimated_corr(i) / samples;
end
for w=1:20000
    temp = 0;
    for j=1:(samples/2)
        temp = estimated_corr(j) * exp(-1i*f1(w)*j) + temp;
    end
```

```
    BT_PSD(w) = estimated_corr(1) + 2 * real(temp);
end

figure;
subplot(2,1,1);
plot(f1,abs(BT_PSD))
title('Estimate PSD by BT');
xlabel('Frequency');
ylabel('Power/Frequency');
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]);
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'});

subplot(2,1,2);
plot(f, PSD);
hold on;
title('Exact PSD');
xlabel('Frequency');
ylabel('Power/Frequency');
stem([0.1*pi, -0.1*pi, 0.4 * pi, -0.4 * pi, 0.8 * pi,
    -0.8 * pi], ...
     [pi * 50, pi * 50, pi * 200, pi * 200, pi * 50, pi
         * 50]);
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]);
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'});

```

```
%% part c

[pxx,f2] = periodogram(x,[],[],2*pi);

figure;
subplot(2,1,1);
plot(f2 , pxx); % Adjust frequency range to [-pi, pi]
hold on;
plot(-f2,pxx)
title('Estimate PSD by Periodogram');
xlabel('Frequency');
ylabel('Power/Frequency');
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]);
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'});

subplot(2,1,2);
plot(f, PSD);
hold on;
stem([0.1*pi, -0.1*pi, 0.4 * pi, -0.4 * pi, 0.8 * pi, -0.8 * pi], ...
    [pi * 50, pi * 50, pi * 200, pi * 200, pi * 50, pi * 50]);
title('Exact PSD');
xlabel('Frequency');
ylabel('Power/Frequency');
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]);
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'});





%% part d


[pxx_welch_10, f3] = pwelch(x, 100, 10, samples, 2*pi);
[pxx_welch_20, f4] = pwelch(x, 100, 20, samples, 2*pi);
[pxx_welch_30, f5] = pwelch(x, 100, 30, samples, 2*pi);

figure;
subplot(3,1,1);
plot(f3, pxx_welch_10);
hold on;
plot(-f3,pxx_welch_10)
hold on;
plot(f, PSD);
hold on;
stem([0.1*pi, -0.1*pi, 0.4 * pi, -0.4 * pi, 0.8 * pi, -0.8 * pi], ...
    [pi * 50, pi * 50, pi * 200, pi * 200, pi * 50, pi * 50]);
xlabel('Frequency');
ylabel('Power/Frequency');
```

```
legend('Estimate PSD by Welch (noverlap = 10)','Exact PSD')
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]);
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'});

%%%%

subplot(3,1,2);
plot(f4 , pxx_welch_20);
hold on;
plot(-f4,pxx_welch_20)
hold on;
plot(f, PSD);
hold on;
stem([0.1*pi, -0.1*pi, 0.4 * pi, -0.4 * pi, 0.8 * pi, -0.8 * pi], ...
    [pi * 50, pi * 50, pi * 200, pi * 200, pi * 50, pi * 50]);
xlabel('Frequency');
ylabel('Power/Frequency');
legend('Estimate PSD by Welch (noverlap = 20)','Exact PSD')
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]);
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'});

```

```
%%%%%

subplot(3,1,3);
plot(f5, pxx_welch_30);
hold on;
plot(-f5,pxx_welch_30)
hold on;
plot(f, PSD);
hold on;
stem([0.1*pi, -0.1*pi, 0.4 * pi, -0.4 * pi, 0.8 * pi, -0.8 * pi], ...
    [pi * 50, pi * 50, pi * 200, pi * 200, pi * 50, pi * 50]);
xlabel('Frequency');
ylabel('Power/Frequency');
legend('Estimate PSD by Welch (noverlap = 30)','Exact PSD')
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]);
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'});

%% part e
```

```
R_x = xcorr(x,'biased');

% implementing levinson by myself and AIC cross
    valiadation
E = [];
K=[];
AIC = zeros(50,1);
temp = 0;
R_xx = R_x(1000:1999);
for i = 1:50
    if i == 1
        E(i) = R_xx(i);
    elseif i == 2
        k(i) = -R_xx(i) / E(i-1);
        a(i,i) = k(i);
        E(i) = (1 - k(i)^2) * E(i-1);
    else
        for j = 1:i-1
            temp = temp + a(j,i-1) * R_xx(i-j);
        end
        k(i) = -(R_xx(i) + temp) / E(i-1);
        a(i,i) = k(i);
        for j = 1:i-1
            a(j,i) = a(j,i-1) + k(i) * a(i-j,i-1);
        end
        E(i) = (1 - k(i)^2) * E(i-1);
    end
    AIC(i) = samples * log(E(i)) + 2 * (i);
    temp = 0;
end
[~, p_opt_AIC] = min(AIC);
```

```
p_opt_LD = 1;
disp('Optimal AR Order (p) by Levinson-Durbin:');
disp(p_opt_aic);
disp('Optimal AR Order (p) by AIC:');
disp(p_opt_AIC);

figure;
plot(1:50, AIC);
title('AIC (cross validation) by my levinson function');
xlabel('Model Order (p)');
ylabel('AIC');
disp('Optimal AR coeffs:');
disp(a(:,p_opt_LD));
disp('Optimal AR coeffs:');
disp(a(:,p_opt_AIC));
% implementing levinson by matlab function and AIC cross valiadation

e = zeros(51,1);
aic = zeros(50,1);
[al, e(1)] = levinson(R_xx, 0); % Order 0 model
for p = 1:50
    [al, e(p+1)] = levinson(R_xx, p);
    aic(p) = samples*log(e(p+1))+2*p;
end
[~, p_opt_aic] = min(aic);
p_opt_LD_m = 1;
disp('Optimal AR Order (p) by Levinson-Durbin:');
disp(p_opt_aic);
disp('Optimal AR Order (p) by AIC:');
```

```
disp(p_opt_aic);


disp('Optimal AR coeffs:');
disp(a(:,p_opt_LD_m));
disp('Optimal AR coeffs:');
disp(a(:,p_opt_aic));

figure;
plot(aic);
title('AIC (cross validation) by matlab function');
xlabel('Model Order (p)');
ylabel('AIC');


%% part f

bestOrder = 0;
bestAIC = Inf;
for q = 1:50
    try
        model = arima('MALags', 1:q, 'Constant', 0);
        fit = estimate(model, x', 'Display', 'off');
        [~,~,logL] = infer(fit, x');
        numParams = q;
        aic(i) = -2 * logL + 2 * numParams;
        if aic(i) < bestAIC
            bestOrder = q;
            bestAIC = aic(i);
            bestFit = fit;
        end
```

```
        catch
            continue;
        end
end

disp('Best MA order: ')
disp(num2str(bestOrder));
disp('MA coefficients:');
disp(cell2mat(bestFit.MA)');

figure;
plot(aic);
title('AIC (cross validation) fot MA');
xlabel('Model Order (p)');
ylabel('AIC');
```

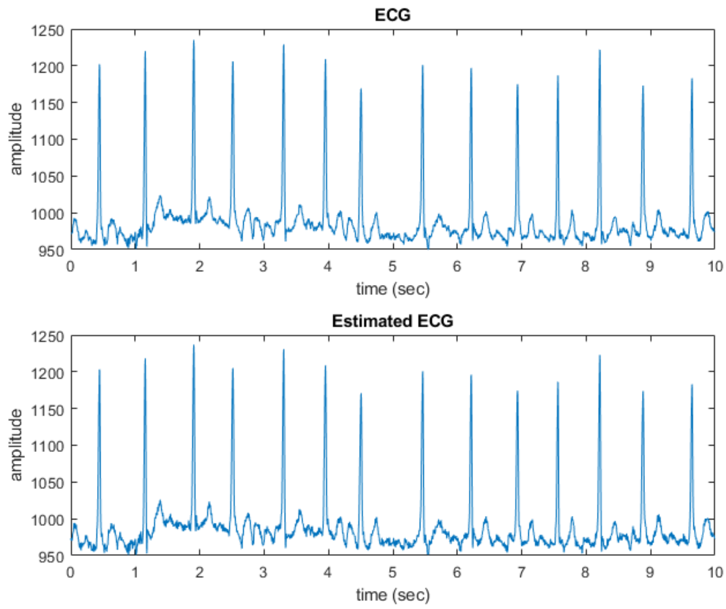
# 2 Question 2

## 2.1 part a

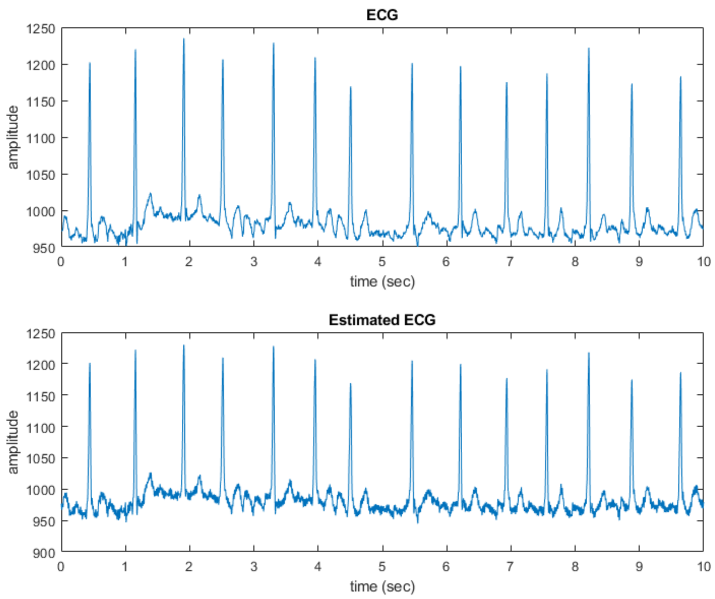
```
%% a
ECG_dataset = load('test.mat');
fs = 360;
s = ECG_dataset.val;
t = 0:1/fs:(length(s)-1)/fs;
phi1 = 2*pi*rand;
phi2 = 2*pi*rand;
N1 = 2*cos(100*pi*t+phi1);
N2 = 2*cos(100*pi*t+phi2);
reference_signal = N1;
primary_signal = s + N2;
w = zeros(length(s),1);
y = zeros(length(s),1);
e = zeros(length(s),1);
mu = 0.000001;
p = y-w;
[e,y] = adaptivefilter(w,reference_signal,primary_signal,mu);
%% adaptive filter function :

function [e,y] = adaptivefilter(w,reference_signal,primary_signal,mu)
    for i=1:length(reference_signal)
        y = reference_signal*w;
        e = primary_signal - y;
        w = w + 2*mu*e*reference_signal';
    end
end
```

## 2.2 part b

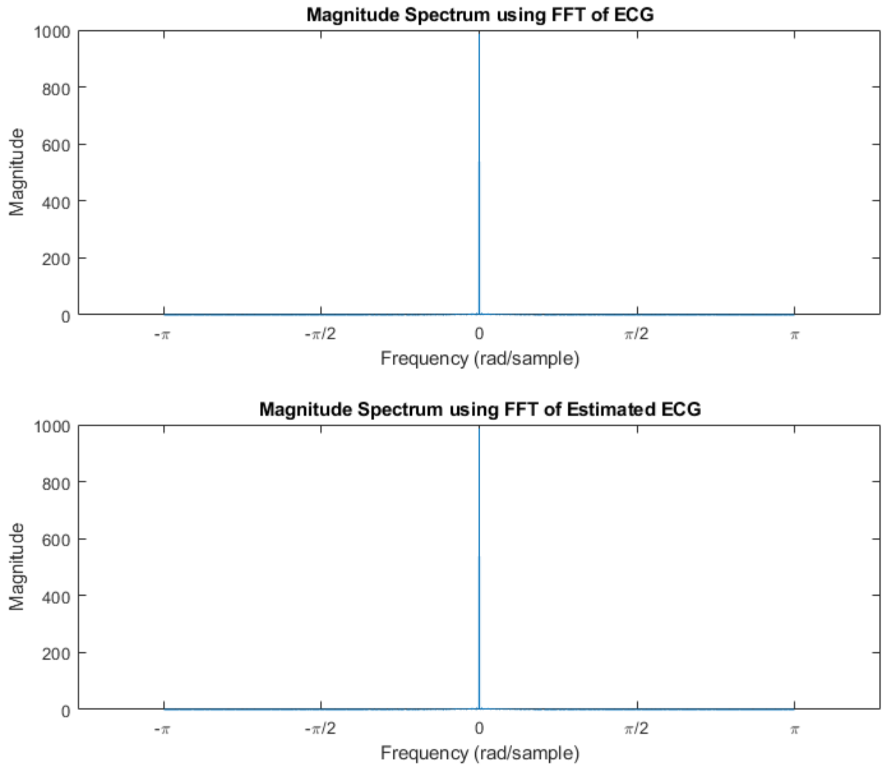
with Amplitude = 5 for noises :




with Amplitude = 20 for noises :



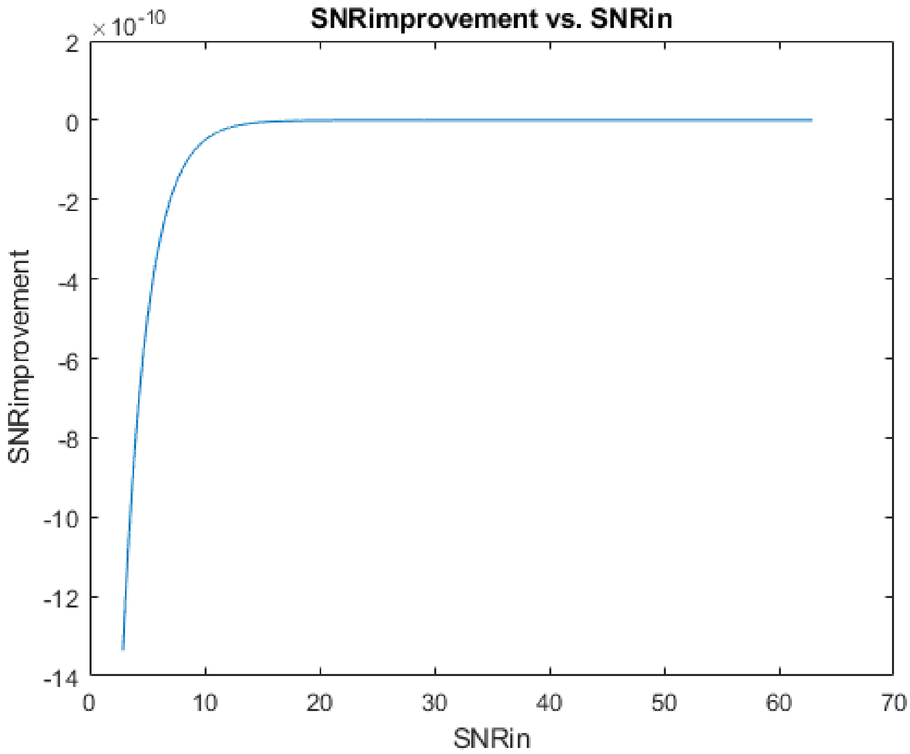
* As we can see still we have a little amount of effect from the noise , which added to the ECG signal, but we have estimated the ECG signal properly.  Also we have to say that the effect of noise relate to the amplitude directly.




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

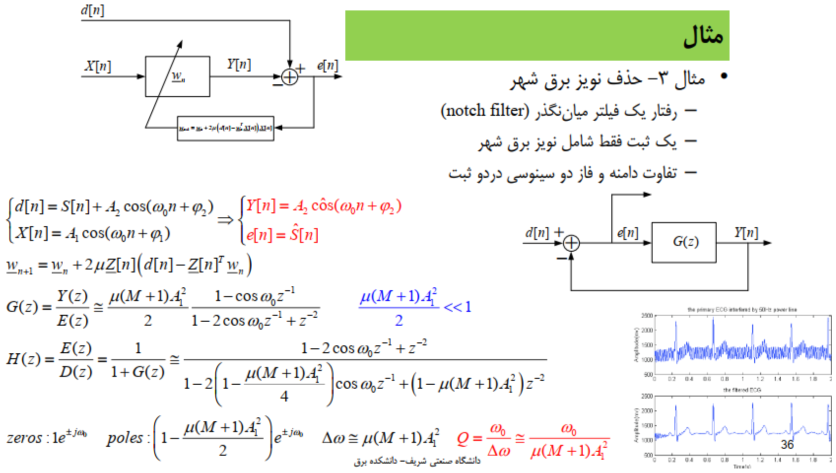
## 2.3 part c

- 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.

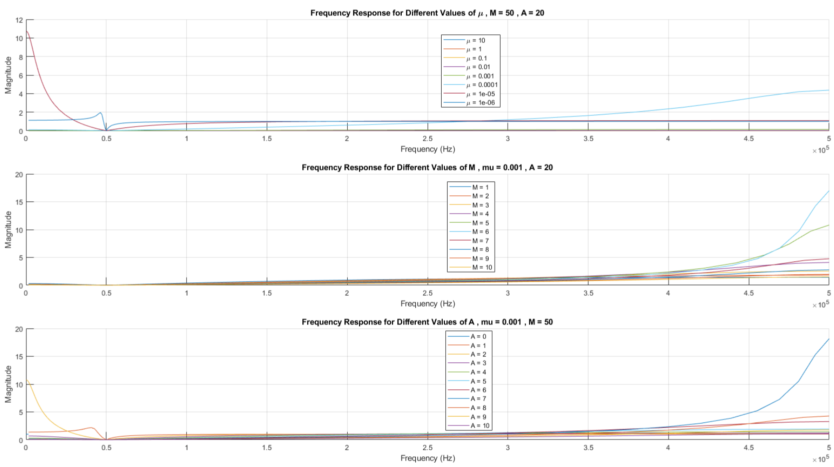


## 2.4 part d

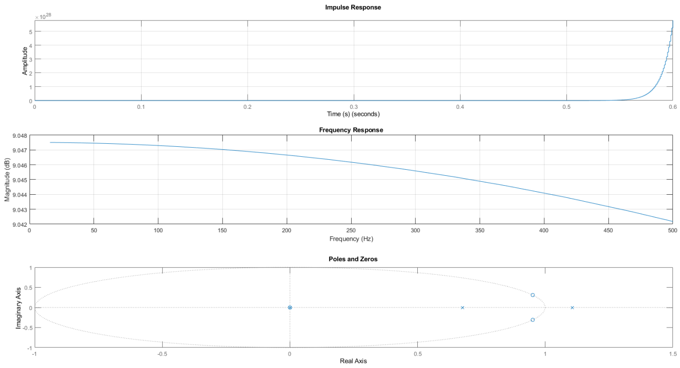
* As we had in slides :




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

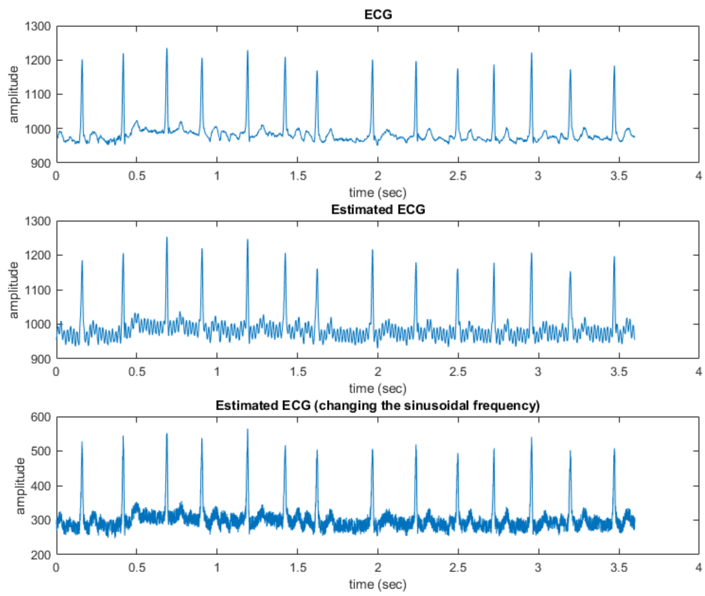


* Now for mu = 0.0001 , A = 5 , M = 100 :



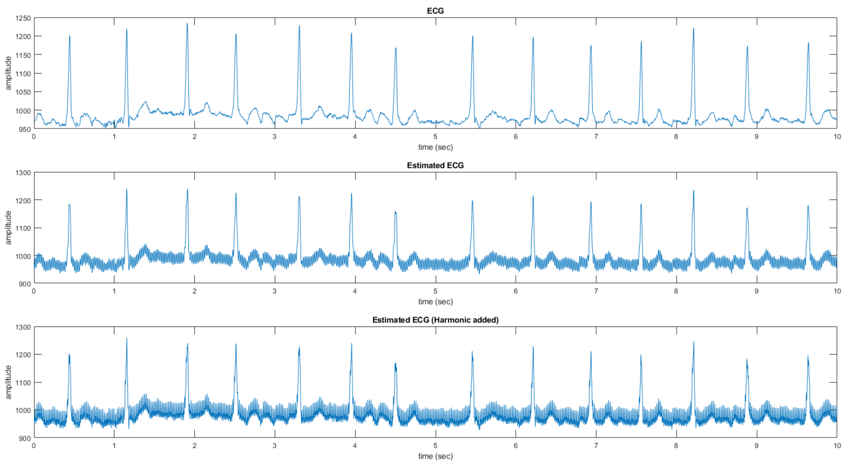

## 2.5 part e

- 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

```
%% a
ECG_dataset = load('test.mat');
fs = 360;
s = ECG_dataset.val;
t = 0:1/fs:(length(s)-1)/fs;
phi1 = 2*pi*rand;
phi2 = 2*pi*rand;
N1 = 20*cos(100*pi*t+phi1);
N2 = 20*cos(100*pi*t+phi2);
reference_signal = N1;
primary_signal = s + N2;
w = zeros(length(s),1);
y = zeros(length(s),1);
e = zeros(length(s),1);
```

```
mu = 0.000001;
p = y-w;
[e,y] = adaptivefilter(w,reference_signal,primary_signal,mu);

%% b

figure;

subplot(2,1,1)
plot(t,s)
title('ECG');
xlabel('time (sec)');
ylabel('amplitude');

subplot(2,1,2)
plot(t,e)
title('Estimated ECG');
xlabel('time (sec)');
ylabel('amplitude');

f = (-length(s)/2:length(s)/2-1)*(2*pi/length(s));
figure;
subplot(2,1,1)
plot(f, abs(fftshift(fft(s)))/length(s));
title('Magnitude Spectrum using FFT of ECG');
xlabel('Frequency (rad/sample)');
ylabel('Magnitude');
xticks([-pi -pi/2 0 pi/2 pi]);
xticklabels({'-\pi', '-\pi/2' , '0', '\pi/2', '\pi'});
```

```
subplot(2,1,2)
plot(f, abs(fftshift(fft(e)))/length(e))
title('Magnitude Spectrum using FFT of Estimated ECG');
xlabel('Frequency (rad/sample)');
ylabel('Magnitude');
xticks([-pi -pi/2 0 pi/2 pi]);
xticklabels({'-\pi', '-\pi/2' , '0', '\pi/2', '\pi'});

%% c

for i = 0:1000
    N1 = i*cos(100*pi*t+phi1);
    N2 = i*cos(100*pi*t+phi2);
    reference_signal = N1;
    primary_signal = s + N2;
    w = zeros(length(s),1);
    y = zeros(length(s),1);
    e = zeros(length(s),1);
    mu = 0.000001;
    [e,y] = adaptivefilter(w,reference_signal, primary_signal,mu);
    SNRin(i+1) = 10*log10(norm(s)^2/norm(N2)^2);
    SNRout(i+1) = 10*log10(norm(s)^2/(norm(e-s)^2));
    SNRimprovement(i+1) = SNRout(i+1) - SNRin(i+1);
end

figure;
plot(SNRin,SNRimprovement)
```

```
title('SNRimprovement vs. SNRin');
xlabel('SNRin');
ylabel('SNRimprovement');


%% part d


f = 50;
z = tf('z', 1/fs);
figure;
mu_values = [10 1 0.1 0.01 0.001 0.0001 0.00001
    0.000001];
M = 50;
A = 20;
for i = 1:length(mu_values)
    mu = mu_values(i);
    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));
    subplot(3, 1, 1);
    hold on;
    [mag, phase, w] = bode(H);
    plot(w*fs/(2*pi), mag(:));
end
title('Frequency Response for Different Values of \mu'
    );
xlabel('Frequency (Hz)');
ylabel('Magnitude');
legend(arrayfun(@(x) ['\mu = ' num2str(x)], mu_values,
    'UniformOutput', false));
```

```
grid on;

mu = 0.001;
M_values = 1:10;
A = 20;

for M = M_values
    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));
    subplot(3, 1, 2);
    hold on;
    [mag, phase, w] = bode(H);
    plot(w*fs/(2*pi), mag(:));
end
title('Frequency Response for Different Values of M');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
legend(arrayfun(@(x) ['M = ' num2str(x)], M_values, '
    UniformOutput', false));
grid on;
mu = 0.001;
M = 50;
A_values = 0:10;
for A = A_values
    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));
    subplot(3, 1, 3);
    hold on;
    [mag, phase, w] = bode(H);
```

```
    plot(w*fs/(2*pi), mag(:));
end
title('Frequency Response for Different Values of A');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
legend(arrayfun(@(x) ['A = ' num2str(x)], A_values, 'UniformOutput', false));
grid on;
hold off;
%%%%%%%%%%%%%%%%
mu = 0.0001;
A = 5;
M = 100;
figure;
subplot(3, 1, 1);
impulse(H);
title('Impulse Response');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
subplot(3, 1, 2);
[mag, phase, w] = bode(H, {0, pi});
mag = squeeze(mag);
w = squeeze(w);
plot(w*fs/(2*pi), 20*log10(mag));
title('Frequency Response');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
grid on;
subplot(3, 1, 3);
pzmap(H);
```

```
title('Poles and Zeros');


%% part e


t = 0:1/fs:(length(s)-1)/fs;
phi1 = 2*pi*rand;
phi2 = 2*pi*rand;
noise = rand(1,length(t));
N1 = 20*cos(100*pi*(t+noise)+phi1);
N2 = 20*cos(100*pi*(t+noise)+phi2);
reference_signal = N1;
primary_signal = s + N2;
w = zeros(length(s),1);
y = zeros(length(s),1);
e = zeros(length(s),1);
mu = 0.000001;
[e,y] = adaptivefilter(w,reference_signal,
    primary_signal,mu);

figure;

subplot(2,1,1)
plot(t,s)
title('ECG');
xlabel('time (sec)');
ylabel('amplitude');

subplot(2,1,2)
plot(t,e)
```

```
title('Estimated ECG');
xlabel('time (sec)');
ylabel('amplitude');


%% part f

phi1 = 2*pi*rand;
phi2 = 2*pi*rand;
phi3 = 2*pi*rand;
N1 = 20*cos(100*pi*t+phi1);
N2 = 20*cos(100*pi*t+phi2);
reference_signal = N1;
primary_signal = s + N2;
w = zeros(length(s),1);
y = zeros(length(s),1);
e = zeros(length(s),1);
mu = 0.000001;
[eh1,yh1] = adaptivefilter(w,reference_signal,
    primary_signal,mu);

noise = rand(1,length(t));
N1 = 20*cos(100*pi*(t+noise)+phi1);
N2 = 20*cos(100*pi*(t+noise)+phi2);
reference_signal = N1;
primary_signal = s + N2;
w = zeros(length(s),1);
y = zeros(length(s),1);
e = zeros(length(s),1);
mu = 0.000001;
[eh,yh] = adaptivefilter(w,reference_signal,
```

```
    primary_signal,mu);

figure;
subplot(3,1,1)
plot(t,s)
title('ECG');
xlabel('time (sec)');
ylabel('amplitude');

subplot(3,1,2)
plot(t,eh1)
title('Estimated ECG');
xlabel('time (sec)');
ylabel('amplitude');

subplot(3,1,3)
plot(t,eh)
title('Estimated ECG (changing the sinusoidal frequency)');
xlabel('time (sec)');
ylabel('amplitude');



%% part g

phi1 = 2*pi*rand;
N2 = 20*cos(100*pi*t+phi2);
primary_signal = s + N2;
w = zeros(length(s),1);
mu = 0.000001;
```

```
for i = 1:15
    [eal(i), yal(i)] = ALEfilter(w, primary_signal, primary_signal, mu, i);
    SNRinal(i) = 10*log10(norm(s)^2/norm(N2)^2);
    SNRoutal(i) = 10*log10(norm(s)^2/(norm(e(i)-s)^2));
    SNRimprovemental(i) = SNRoutal(i) - SNRinal(i);
end
figure;
plot(SNRin,SNRimprovement)
title('SNRimprovement');
xlabel('sample');
ylabel('SNRimprovement');

%% Adaptive filter function :

function [e,y] = adaptivefilter(w,reference_signal,primary_signal,mu)
    for i=1:length(reference_signal)
        y = reference_signal*w;
        e = primary_signal - y;
        w = w + 2*mu*e*reference_signal';
    end
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.