باسمه تعالی

گزارش تمرین کامپیوتری سری ۲

پردازش علائم بیولوژیک

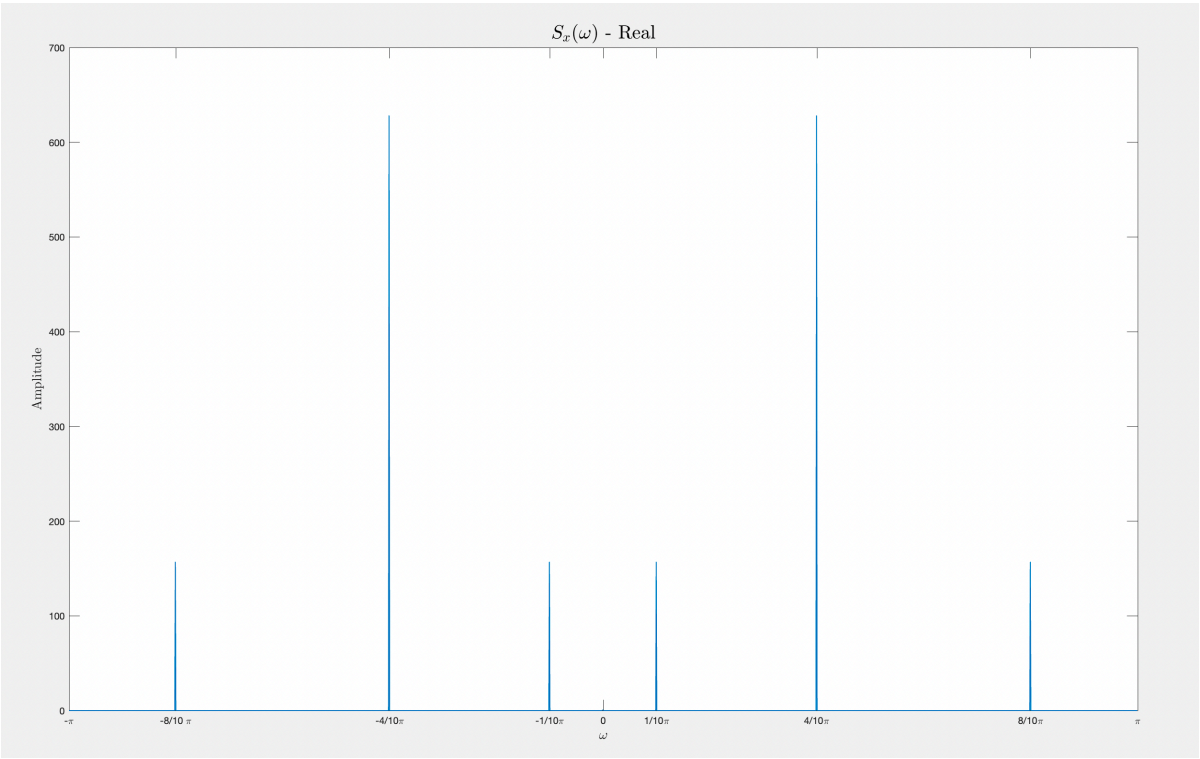
دانشکده مهندسی برق

استاد: محمدباقر شمس اللهی

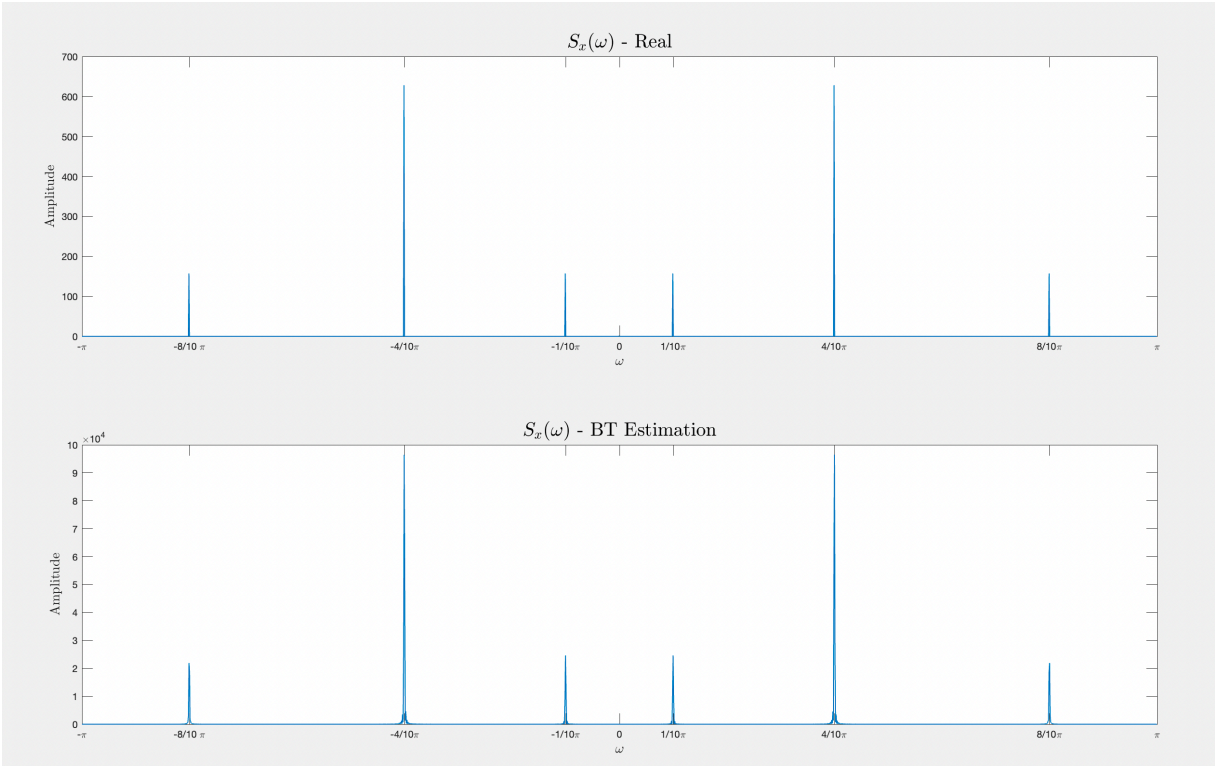
گردآورنده: رادین خیّام

سوال ۱-

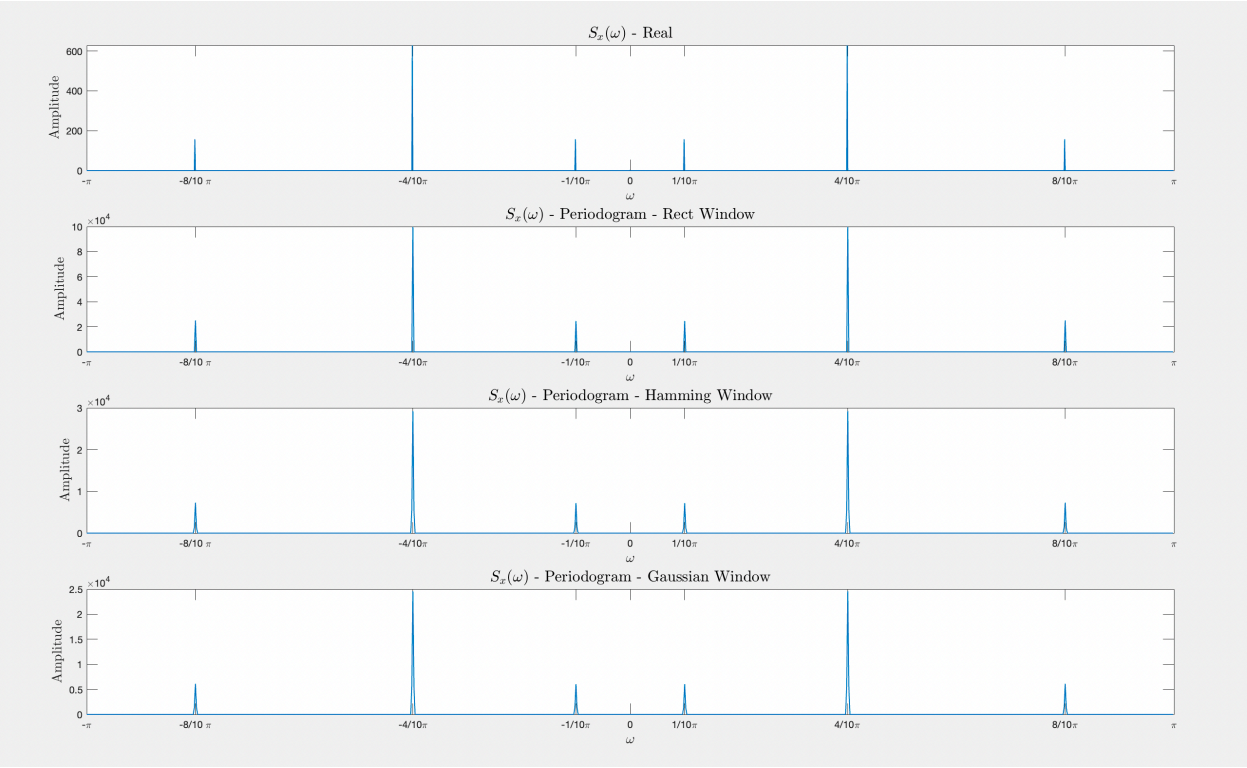
الف)


$S_x(\omega)$ - Real
Amplitude
700
600
500
400
300
200
100
0
$-\pi$
-8/10 π
$-4/10\pi$
$-1/10\pi$
0
$1/10\pi$
$4/10\pi$
$8/10\pi$
π
ω

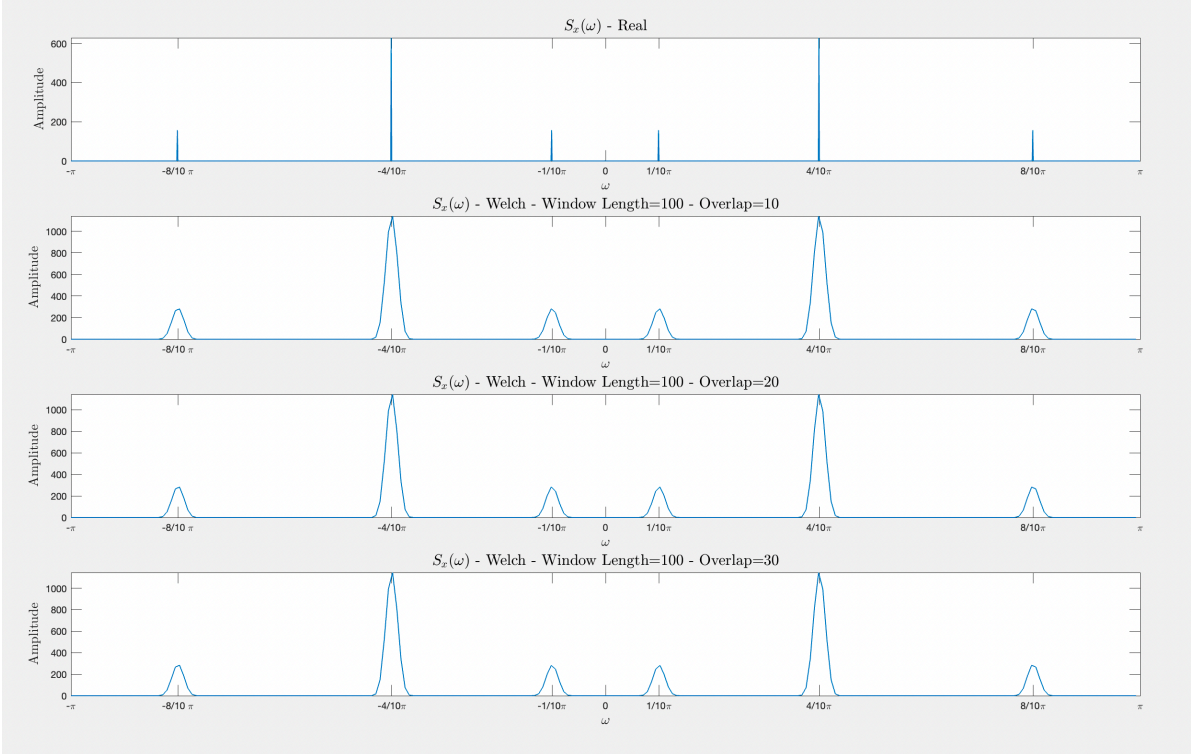
ب)

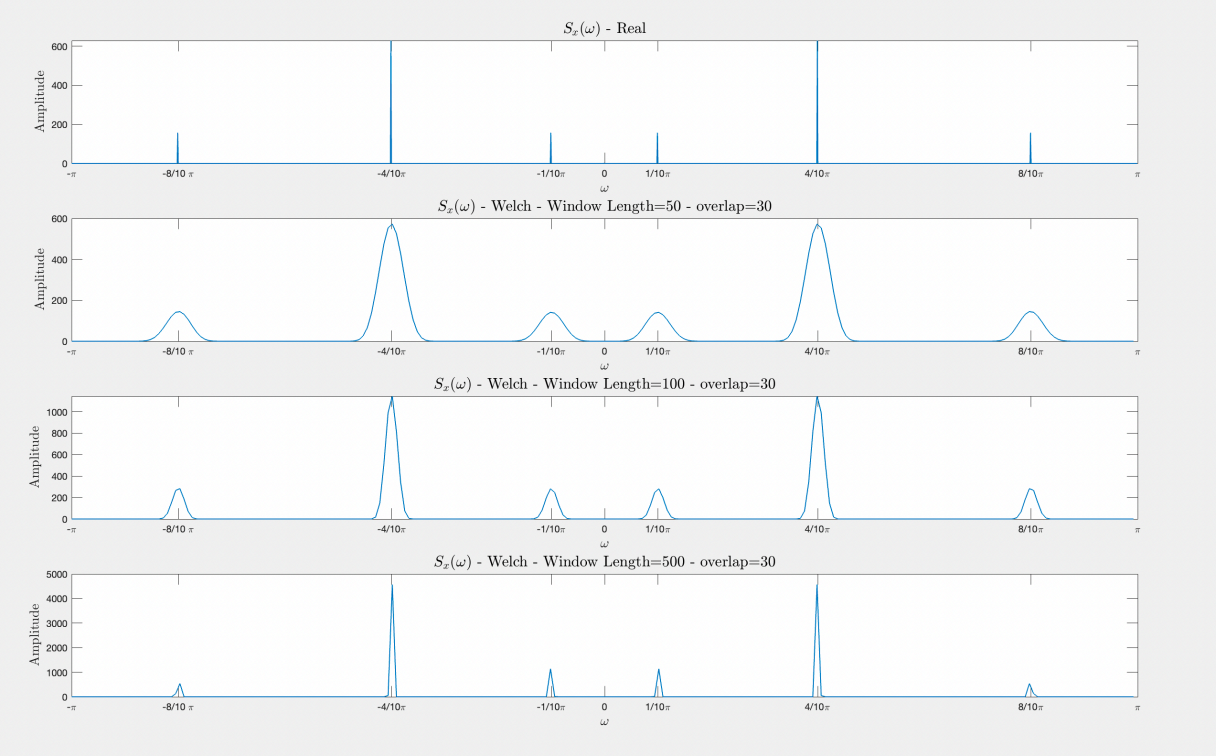

$S_x(\omega)$ - Real
Amplitude
700
600
500
400
300
200
100
0
$-\pi$
-8/10 π
$-4/10\pi$
$-1/10\pi$
0
$1/10\pi$
$4/10\pi$
$8/10\pi$
π
ω
$S_x(\omega)$ - BT Estimation
$\times 10^4$
Amplitude
10
9
8
7
6
5
4
3
2
1
0
$-\pi$
-8/10 π
$-4/10\pi$
$-1/10\pi$
0
$1/10\pi$
$4/10\pi$
$8/10\pi$
π
ω

پ)

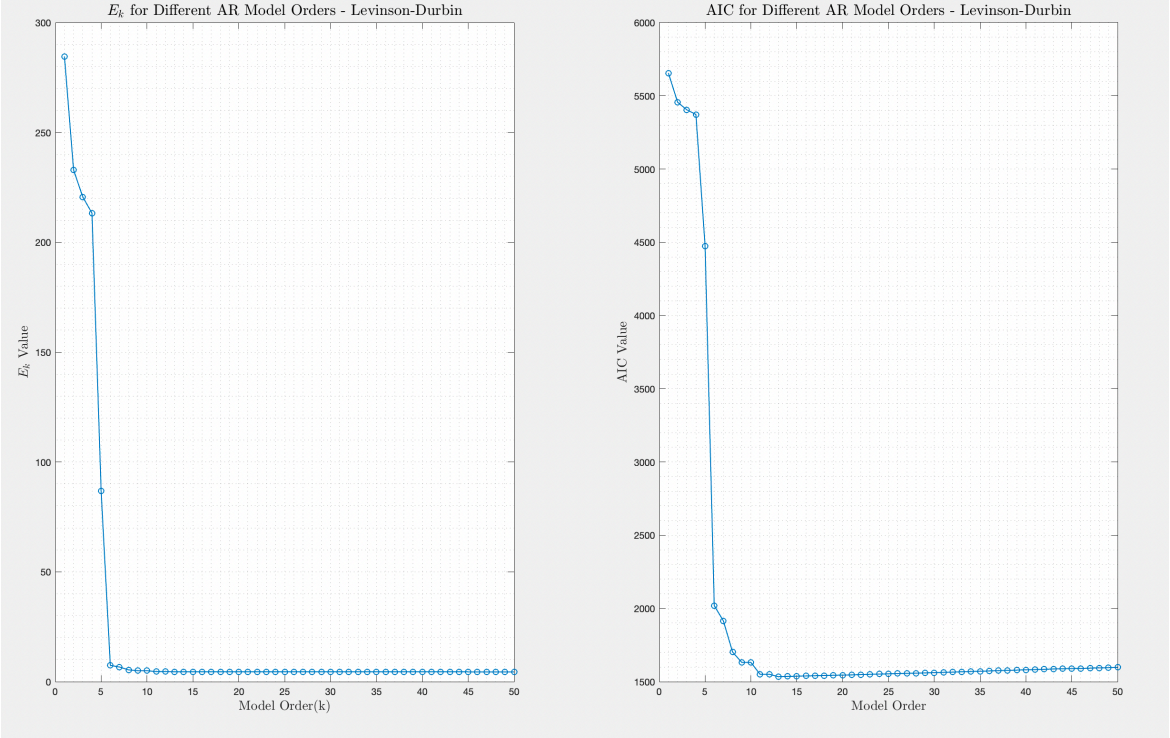

$S_x(\omega)$ - Real
$S_x(\omega)$ - Periodogram - Rect Window
$S_x(\omega)$ - Periodogram - Hamming Window
$S_x(\omega)$ - Periodogram - Gaussian Window
Amplitude
ω
-π
-8/10 π
-4/10π
-1/10π
0
1/10π
4/10π
8/10π
π

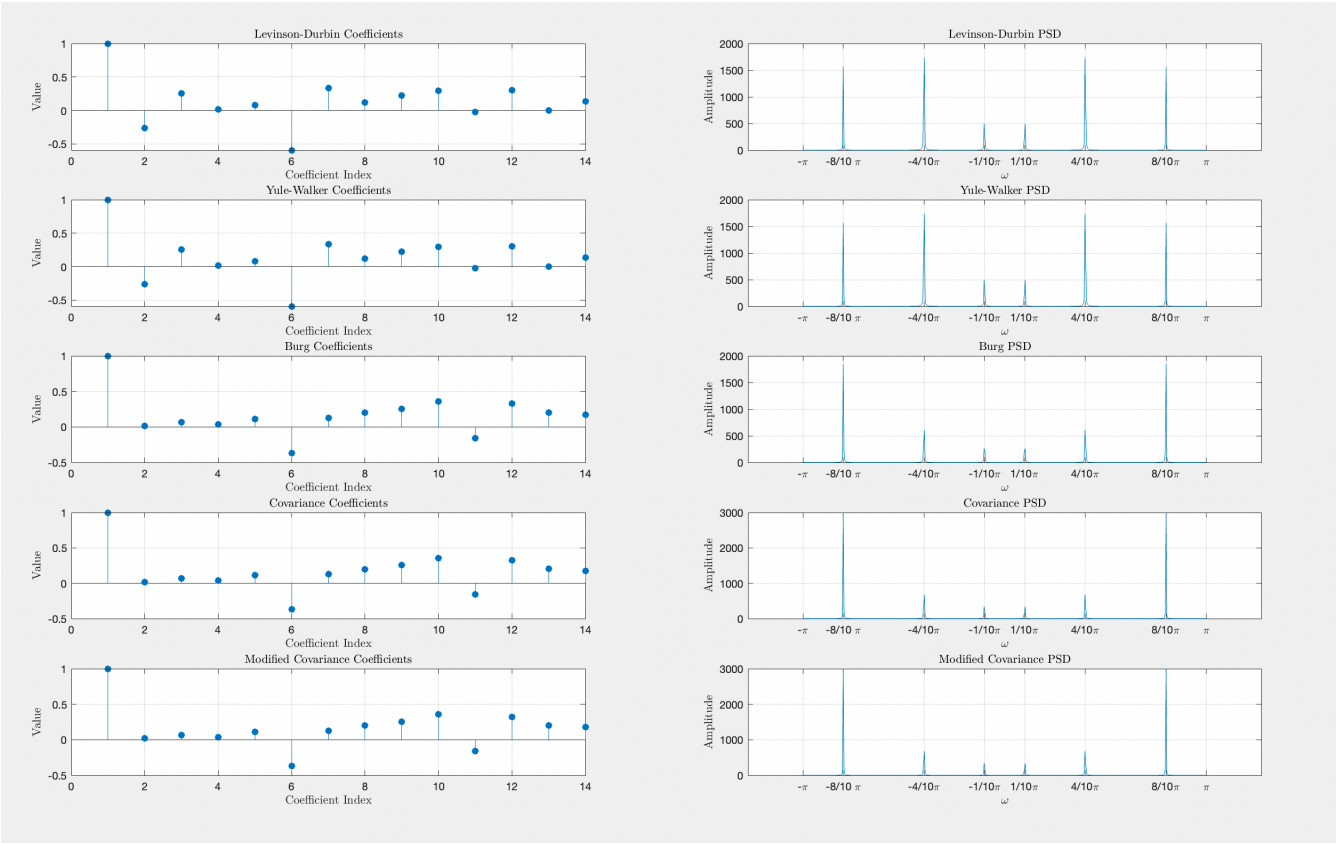
ت)


$S_x(\omega)$ - Real
$S_x(\omega)$ - Welch - Window Length=100 - Overlap=10
$S_x(\omega)$ - Welch - Window Length=100 - Overlap=20
$S_x(\omega)$ - Welch - Window Length=100 - Overlap=30
Amplitude
ω
-π
-8/10 π
-4/10π
-1/10π
0
1/10π
4/10π
8/10π
π


$S_x(\omega)$ - Real
$S_x(\omega)$ - Welch - Window Length=50 - overlap=30
$S_x(\omega)$ - Welch - Window Length=100 - overlap=30
$S_x(\omega)$ - Welch - Window Length=500 - overlap=30
Amplitude
ω
-π
-8/10 π
-4/10π
-1/10π
0
1/10π
4/10π
8/10π
π

ث)


E_k for Different AR Model Orders - Levinson-Durbin
E_k Value
Model Order(k)
AIC for Different AR Model Orders - Levinson-Durbin
AIC Value
Model Order




```
%% BSP - CHW2 - 99101579

%% Q1

clc; clear; close all;



alpha_1 = 2*pi*rand;

alpha_2 = 2*pi*rand;

alpha_3 = 2*pi*rand;

n = 0:999;

x = 10*cos(0.1*pi*n + alpha_1) + 20*cos(0.4*pi*n + alpha_2) + 10*cos(0.8*pi*n + alpha_3)
+ randn(1, 1000);

%% Q1 - part a

clc;


omega_real = -pi:2*pi/2000:pi-(2*pi/2000);
```

```
S_x_real = zeros(1,2000);

S_x_real(1100) = 50*pi;

S_x_real(900) = 50*pi;

S_x_real(1400) = 200*pi;

S_x_real(600) = 200*pi;

S_x_real(1800) = 50*pi;

S_x_real(200) = 50*pi;

S_x_real(1000) = 1;


plot(omega_real, S_x_real,'LineWidth',1);

title('$$S_{x}(\omega)$$ - Real','Interpreter','latex','FontSize',20);

ylabel('Amplitude','Interpreter','latex','FontSize',14);

xlabel('$$\omega$$','Interpreter','latex','FontSize',14);

xlim([-pi pi]);

xticks([-pi -(8/10)*pi -(4/10)*pi -(1/10)*pi 0 (1/10)*pi (4/10)*pi  (8/10)*pi pi]);

xticklabels({'-\pi','-8/10 \pi','-4/10\pi','-1/10\pi','0',
'1/10\pi','4/10\pi','8/10\pi','\pi','interpreter','latex','FontSize',10});

%% Q1 - part b

clc;

M = 1000;

Rx_est = zeros(1,2*M-1);

for m=0:M-1

    tmp = 0;

    for n=0:M-m-1

        tmp = tmp + x(n+1)*x(n+m+1);

    end

    Rx_est(m+1)=1/M * tmp;

    if(m~=0)

            Rx_est(2*M-m) = Rx_est(m+1);

    end

end
```

```
s_x_BT = fftshift(fft(Rx_est));

L = length(s_x_BT);

omega_BT = -pi:2*pi/L:pi-(2*pi/L);


subplot(2,1,1);

plot(omega_real, S_x_real,'LineWidth',1);

title('$$S_{x}(\omega)$$ - Real','Interpreter','latex','FontSize',20);

ylabel('Amplitude','Interpreter','latex','FontSize',14);

xlabel('$$\omega$$','Interpreter','latex','FontSize',14);

xlim([-pi pi]);

xticks([-pi -(8/10)*pi -(4/10)*pi -(1/10)*pi 0 (1/10)*pi (4/10)*pi  (8/10)*pi pi]);

xticklabels({'-\pi','-8/10 \pi','-4/10\pi','-1/10\pi','0',
'1/10\pi','4/10\pi','8/10\pi','\pi','interpreter','latex','FontSize',10});


subplot(2,1,2);

plot(omega_BT, s_x_BT,'Linewidth',1);

title('$$S_{x}(\omega)$$ - BT Estimation','Interpreter','latex','FontSize',20);

ylabel('Amplitude','Interpreter','latex','FontSize',14);

xlabel('$$\omega$$','Interpreter','latex','FontSize',14);

xlim([-pi pi]);

xticks([-pi -(8/10)*pi -(4/10)*pi -(1/10)*pi 0 (1/10)*pi (4/10)*pi  (8/10)*pi pi]);

xticklabels({'-\pi','-8/10 \pi','-4/10\pi','-1/10\pi','0',
'1/10\pi','4/10\pi','8/10\pi','\pi','interpreter','latex','FontSize',10});

%% Q1 - part c

clc;

W_1 = rectwin(M)';

W_2 = hamming(M)';

W_3 = gausswin(M)';


Xw_1 = W_1.*x;

Xw_2 = W_2.*x;
```

```
Xw_3 = W_3.*x;


Xw_1_fourier = fftshift(fft(Xw_1));

Xw_2_fourier = fftshift(fft(Xw_2));

Xw_3_fourier = fftshift(fft(Xw_3));


Sx_1_peridogram = 1/M * abs(Xw_1_fourier).^2;

Sx_2_peridogram = 1/M * abs(Xw_2_fourier).^2;

Sx_3_peridogram = 1/M * abs(Xw_3_fourier).^2;


omega_periodogram = -pi:2*pi/M:pi-(2*pi/M);


subplot(4,1,1);

plot(omega_real, S_x_real,'LineWidth',1);

title('$$S_{x}(\omega)$$ - Real','Interpreter','latex','FontSize',16);

ylabel('Amplitude','Interpreter','latex','FontSize',14);

xlabel('$$\omega$$','Interpreter','latex','FontSize',14);

xlim([-pi pi]);

xticks([-pi -(8/10)*pi -(4/10)*pi -(1/10)*pi 0 (1/10)*pi (4/10)*pi  (8/10)*pi pi]);

xticklabels({'-\pi','-8/10 \pi','-4/10\pi','-1/10\pi','0',
'1/10\pi','4/10\pi','8/10\pi','\pi','interpreter','latex','FontSize',10});


subplot(4,1,2);

plot(omega_periodogram, Sx_1_peridogram,'LineWidth',1);

title('$$S_{x}(\omega)$$ - Periodogram - Rect
Window','Interpreter','latex','FontSize',16);

ylabel('Amplitude','Interpreter','latex','FontSize',14);

xlabel('$$\omega$$','Interpreter','latex','FontSize',14);

xlim([-pi pi]);

xticks([-pi -(8/10)*pi -(4/10)*pi -(1/10)*pi 0 (1/10)*pi (4/10)*pi  (8/10)*pi pi]);

xticklabels({'-\pi','-8/10 \pi','-4/10\pi','-1/10\pi','0',
'1/10\pi','4/10\pi','8/10\pi','\pi','interpreter','latex','FontSize',10});
```

```
subplot(4,1,3);

plot(omega_periodogram, Sx_2_peridogram,'LineWidth',1);

title('$$S_{x}(\omega)$$ - Periodogram - Hamming
Window','Interpreter','latex','FontSize',16);

ylabel('Amplitude','Interpreter','latex','FontSize',14);

xlabel('$$\omega$$','Interpreter','latex','FontSize',14);

xlim([-pi pi]);

xticks([-pi -(8/10)*pi -(4/10)*pi -(1/10)*pi 0 (1/10)*pi (4/10)*pi  (8/10)*pi pi]);

xticklabels({'-\pi','-8/10 \pi','-4/10\pi','-1/10\pi','0',
'1/10\pi','4/10\pi','8/10\pi','\pi','interpreter','latex','FontSize',10});


subplot(4,1,4);

plot(omega_periodogram, Sx_3_peridogram,'LineWidth',1);

title('$$S_{x}(\omega)$$ - Periodogram - Gaussian
Window','Interpreter','latex','FontSize',16);

ylabel('Amplitude','Interpreter','latex','FontSize',14);

xlabel('$$\omega$$','Interpreter','latex','FontSize',14);

xlim([-pi pi]);

xticks([-pi -(8/10)*pi -(4/10)*pi -(1/10)*pi 0 (1/10)*pi (4/10)*pi  (8/10)*pi pi]);

xticklabels({'-\pi','-8/10 \pi','-4/10\pi','-1/10\pi','0',
'1/10\pi','4/10\pi','8/10\pi','\pi','interpreter','latex','FontSize',10});


%% Q1 - Part d

clc;

window_lengths = [50 100 500];

nfft = 256;

overlaps = [10, 20, 30];

psds_fix_wl = [];

psds_fix_ol = [];


for i = 1:3
```

```
    psds_fix_wl = [psds_fix_wl,
fftshift(pwelch(x,window_lengths(2),overlaps(i),nfft,'twosided'))];
end
for i = 1:3
    psds_fix_ol = [psds_fix_ol,
fftshift(pwelch(x,window_lengths(i),overlaps(3),nfft,'twosided'))];
end
L = nfft;
omega_welch = -pi:2*pi/L:pi-2*pi/L;


subplot(4,1,1);
plot(omega_real, S_x_real,'LineWidth',1);
title('$$S_{x}(\omega)$$ - Real','Interpreter','latex','FontSize',16);
ylabel('Amplitude','Interpreter','latex','FontSize',14);
xlabel('$$\omega$$','Interpreter','latex','FontSize',14);
xlim([-pi pi]);
xticks([-pi -(8/10)*pi -(4/10)*pi -(1/10)*pi 0 (1/10)*pi (4/10)*pi  (8/10)*pi pi]);
xticklabels({'-\pi','-8/10 \pi','-4/10\pi','-1/10\pi','0',
'1/10\pi','4/10\pi','8/10\pi','\pi','interpreter','latex','FontSize',10});


for i=1:3
    subplot(4,1,i+1)
    plot(omega_welch,psds_fix_wl(:,i),'LineWidth',1);
    title(['$$S_{x}(\omega)$$ - Welch - Window Length=100 -
Overlap=',num2str(overlaps(i))],'Interpreter','latex','FontSize',16);
    ylabel('Amplitude','Interpreter','latex','FontSize',14);
    xlabel('$$\omega$$','Interpreter','latex','FontSize',14);
    xlim([-pi pi]);
    xticks([-pi -(8/10)*pi -(4/10)*pi -(1/10)*pi 0 (1/10)*pi (4/10)*pi  (8/10)*pi pi]);
    xticklabels({'-\pi','-8/10 \pi','-4/10\pi','-1/10\pi','0',
'1/10\pi','4/10\pi','8/10\pi','\pi','interpreter','latex','FontSize',10});
    hold on
```

```
end


figure;
subplot(4,1,1);
plot(omega_real, S_x_real,'LineWidth',1);
title('$$S_{x}(\omega)$$ - Real','Interpreter','latex','FontSize',16);
ylabel('Amplitude','Interpreter','latex','FontSize',14);
xlabel('$$\omega$$','Interpreter','latex','FontSize',14);
xlim([-pi pi]);
xticks([-pi -(8/10)*pi -(4/10)*pi -(1/10)*pi 0 (1/10)*pi (4/10)*pi  (8/10)*pi pi]);
xticklabels({'-\pi','-8/10 \pi','-4/10\pi','-1/10\pi','0',
'1/10\pi','4/10\pi','8/10\pi','\pi','interpreter','latex','FontSize',10});


for i=1:3
    subplot(4,1,i+1)
    plot(omega_welch,psds_fix_ol(:,i),'LineWidth',1);
    title(['$$S_{x}(\omega)$$ - Welch - Window Length=',num2str(window_lengths(i)),' -
overlap=30'],'Interpreter','latex','FontSize',16);
    ylabel('Amplitude','Interpreter','latex','FontSize',14);
    xlabel('$$\omega$$','Interpreter','latex','FontSize',14);
    xlim([-pi pi]);
    xticks([-pi -(8/10)*pi -(4/10)*pi -(1/10)*pi 0 (1/10)*pi (4/10)*pi  (8/10)*pi pi]);
    xticklabels({'-\pi','-8/10 \pi','-4/10\pi','-1/10\pi','0',
'1/10\pi','4/10\pi','8/10\pi','\pi','interpreter','latex','FontSize',10});
    hold on
end


%% Q1 - Part e
clc;
M = 1000;
Rx = zeros(1,M);
for m=0:M-1
```

```
    tmp = 0;
    for n=0:M-m-1
        tmp = tmp + x(n+1)*x(n+m+1);
    end
    Rx(m+1)=1/M * tmp;
end

max_order = 50;
AIC = zeros(1, max_order);
E = zeros(1, max_order);

% Compute AR coefficients and AIC for each model order
for p = 1:max_order
    [a, e] = levinson(Rx(1:p+1), p);
    AIC(p) = M * log(e) + 2 * (p + 1);
    E(p)= e;
end

% Plot AIC values
figure;
subplot(1,2,1);
plot(1:max_order, E, '-o','LineWidth',1);
title('$$E_k$$ for Different AR Model Orders - Levinson-
Durbin','Interpreter','latex','FontSize',16);
xlabel('Model Order(k)','Interpreter','latex','FontSize',14);
ylabel('$$E_k$$ Value','Interpreter','latex','FontSize',14);
grid minor;
subplot(1,2,2);
plot(1:max_order, AIC, '-o','LineWidth',1);
title('AIC for Different AR Model Orders - Levinson-
Durbin','Interpreter','latex','FontSize',16);
```

```
xlabel('Model Order','Interpreter','latex','FontSize',14);
ylabel('AIC Value','Interpreter','latex','FontSize',14);
grid minor;

[~, optimal_order] = min(AIC);
fprintf('Optimal AR Model Order: %d\n', optimal_order);



methods = {'Levinson-Durbin', 'Yule-Walker', 'Burg', 'Covariance', 'Modified
Covariance'};
coeffs = cell(length(methods), 1);
psd = cell(length(methods), 1);



omega = -pi:2*pi/512:pi-2*pi/512;
figure;
for i = 1:5
    switch methods{i}
        case 'Levinson-Durbin'
            [a_est, ~] = levinson(Rx, optimal_order);
        case 'Yule-Walker'
            a_est = aryule(x, optimal_order);
        case 'Burg'
            a_est = arburg(x, optimal_order);
        case 'Covariance'
            a_est = arcov(x, optimal_order);
        case 'Modified Covariance'
            a_est = armcov(x, optimal_order);
    end

    coeffs{i} = a_est;
```

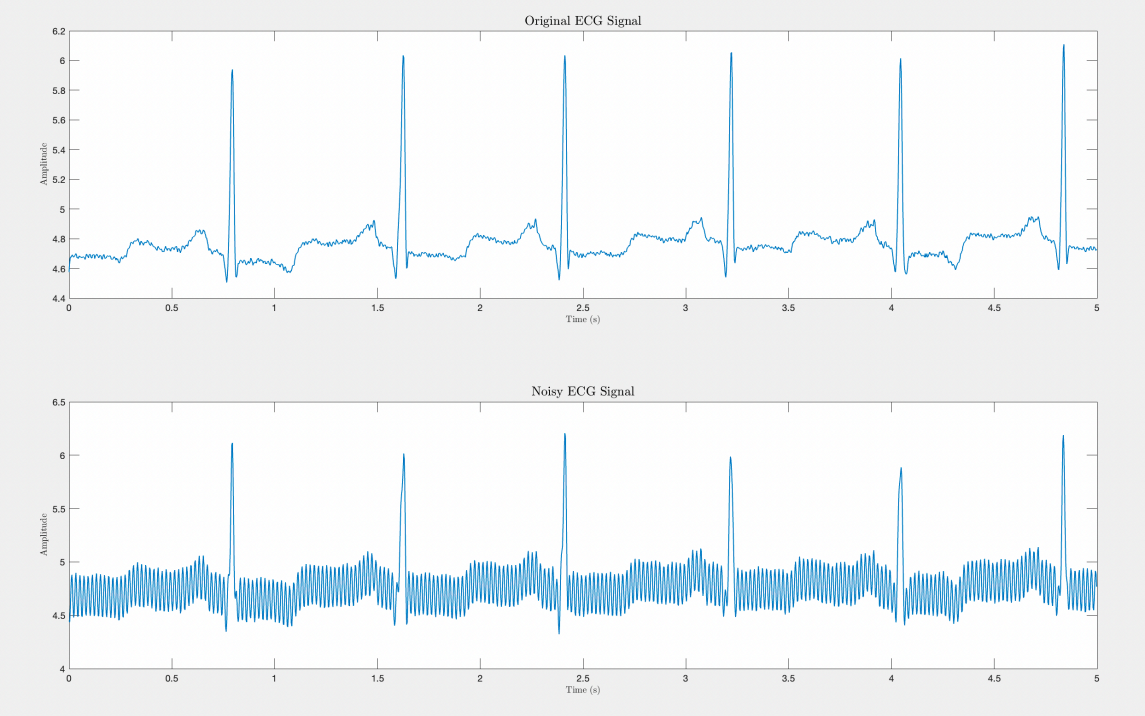
```
    [h, w] = freqz(1, a_est, 512, 'whole');
    psd{i} = fftshift(abs(h).^2);

    subplot(5, 2, 2*i-1);
    stem(a_est, 'filled');
    title([methods{i} ' Coefficients'],'Interpreter','latex');
    xlabel('Coefficient Index','Interpreter','latex');
    ylabel('Value','Interpreter','latex');
    grid on;

    subplot(5, 2, 2*i);
    plot(omega, psd{i});
    title([methods{i} ' PSD'],'Interpreter','latex');
    xlabel('$$\omega$$','Interpreter','latex');
    ylabel('Amplitude','Interpreter','latex');
    xticks([-pi -(8/10)*pi -(4/10)*pi -(1/10)*pi (1/10)*pi (4/10)*pi  (8/10)*pi pi]);
    xticklabels({'-\pi','-8/10 \pi','-4/10\pi','-1/10\pi',
'1/10\pi','4/10\pi','8/10\pi','\pi','interpreter','latex','FontSize',10});
    grid on;
end

%% Q1 - Part f
clc;



max_order = 10;

AIC = zeros(maxOrder, 1);
coeffsArray = cell(max_order, 1);
```

```
for q = 1:max_order
    M = zeros(q+1, q+1);
    b = zeros(q+1, 1);

    for m = 0:q
        b(m+1, 1) = Rx(m+1);
        for k = 0:q
            if m-k >= 0
                M(m+1, k+1) = Rx(abs(m-k)+1);
            else
                M(m+1, k+1) = Rx(abs(k-m)+1);
            end
        end
    end


    coeffs = M \ b;


    coeffsArray{q} = coeffs(2:end) / coeffs(1);

    e = filter([1; -coeffsArray{q}], 1, x);

    sigma2 = var(e);
    N = length(x);
    AIC(q) = N * log(sigma2) + 2 * (q + 1);
end

[~, optimal_order] = min(AIC);

fprintf('Optimal MA Model Order: %d\n', optimal_order);
```

```
fprintf('Minimum AIC: %.2f\n', minAIC);

fprintf('Optimal MA Coefficients:');

disp(coeffsArray{optimal_order});


plot(1:max_order, AIC, '-o','LineWidth',1);

title('AIC for Different MA Model Orders - Levinson-
Durbin','Interpreter','latex','FontSize',16);

xlabel('Model Order','Interpreter','latex','FontSize',14);

ylabel('AIC Value','Interpreter','latex','FontSize',14);

grid minor;
```

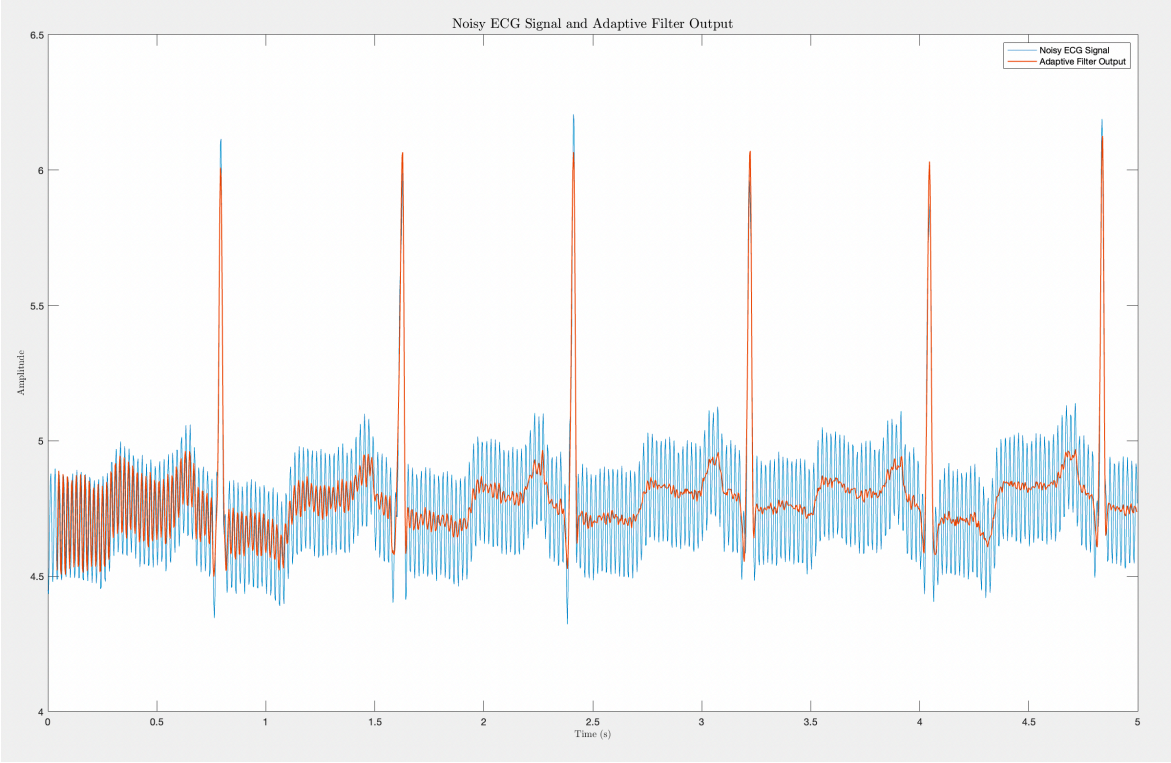
سوال ۲-

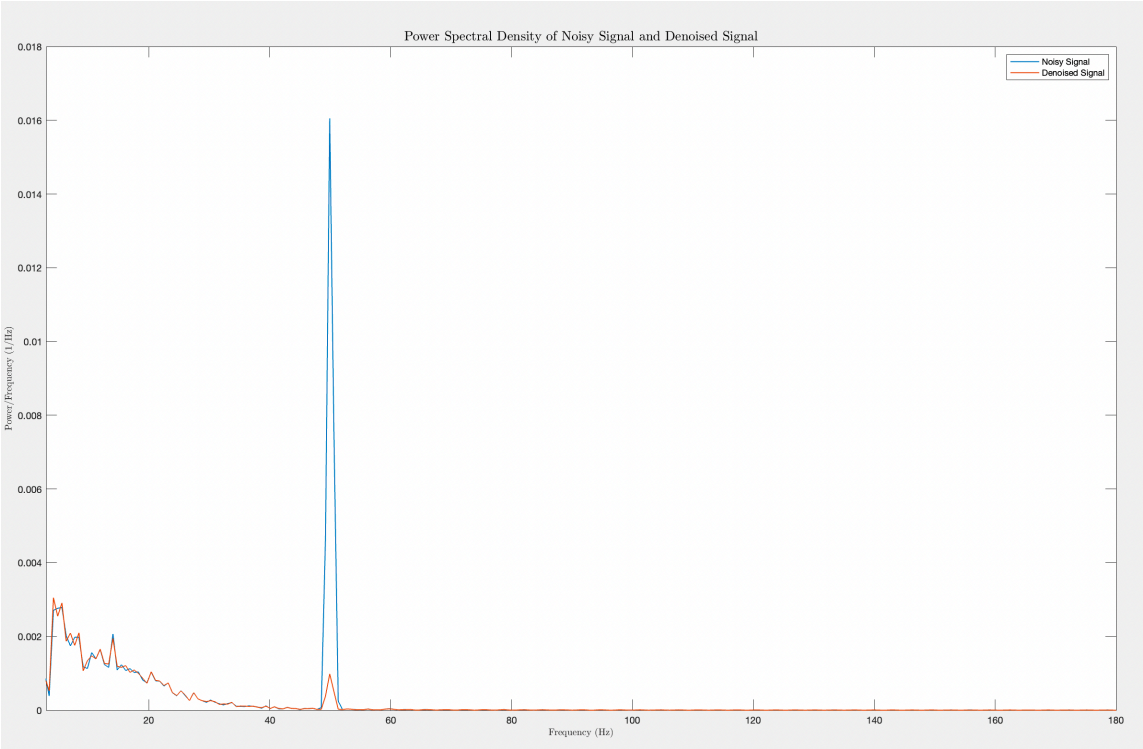
الف)



ب)

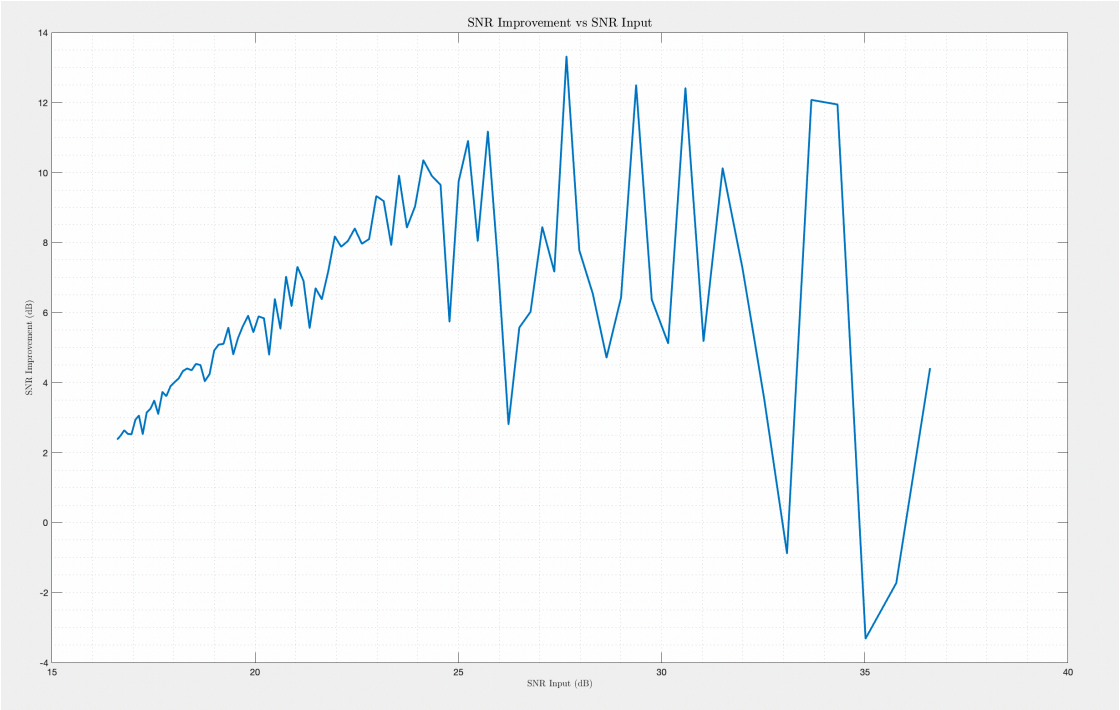
فیلتر وفقی را به صورت دستی پیاده سازی کرده‌ام نه با استفاده از توابع آماده.





میبینیم که مولفه فرکانسی ۵۰ هرتز به خوبی کاهش پیدا کرده است.

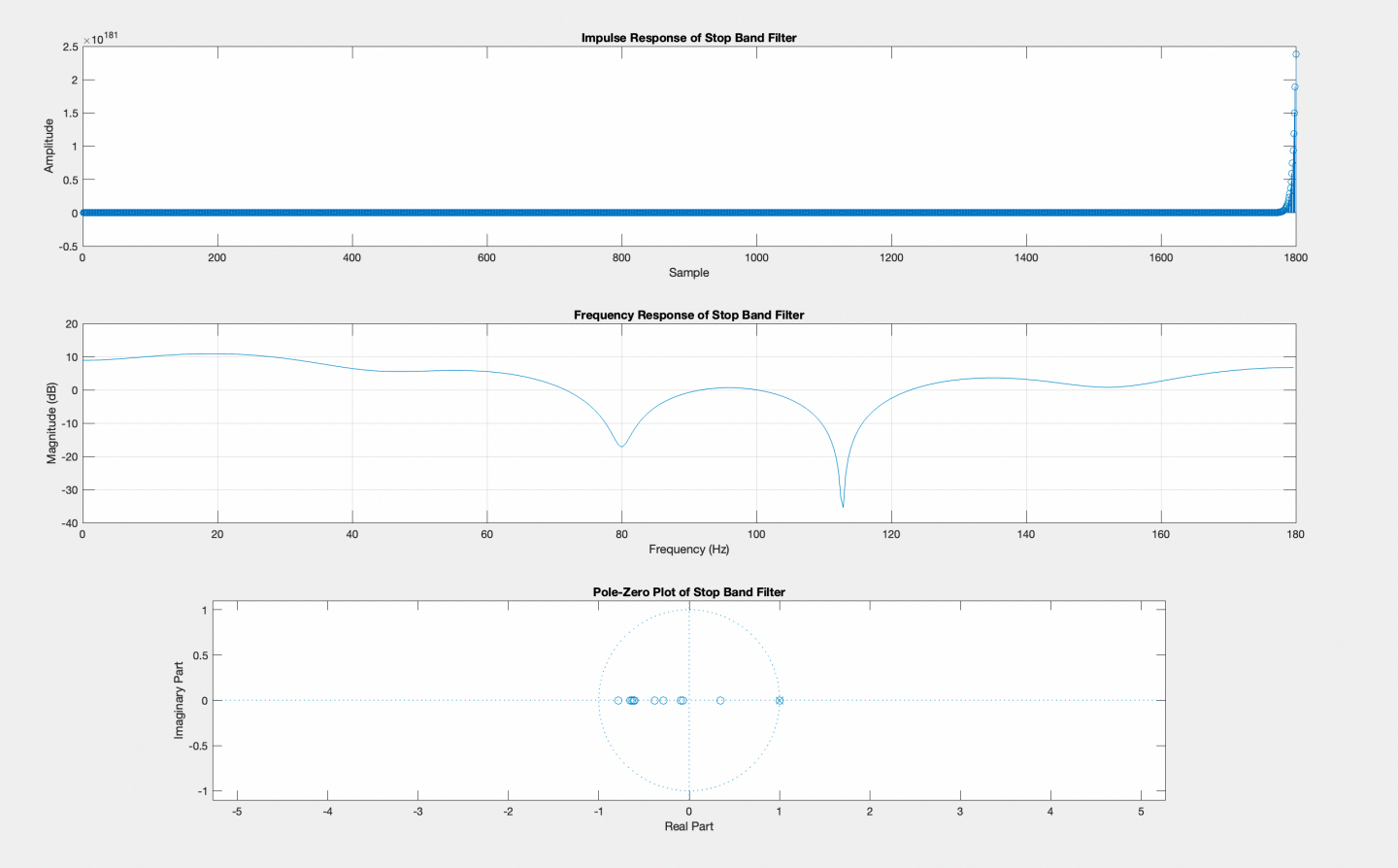
پ)



این نمودار به شدت وابسته به میو و مرتبه فیلتر و طول سیگنال می‌باشد، چنانچه میو از یه حدی بزرگتر انتخاب بشود در خیلی از موارد الگوریتم واگرا می‌شود و همین باعث می‌شود سیگنال ساخته شده خیلی خراب بشود، میو هم اگر کوچک باشد ممکن است در

طول سیگنال (که محدود است) نتوانیم نویز را حذف کنیم. به صورت کلی وقتی که نویز زیاد هست یعنی SNR ورودی زیاده، اگر این پارامترها خوب تنظیم شده باشد، مقدار بهبود SNR بیشتر می‌باشد.

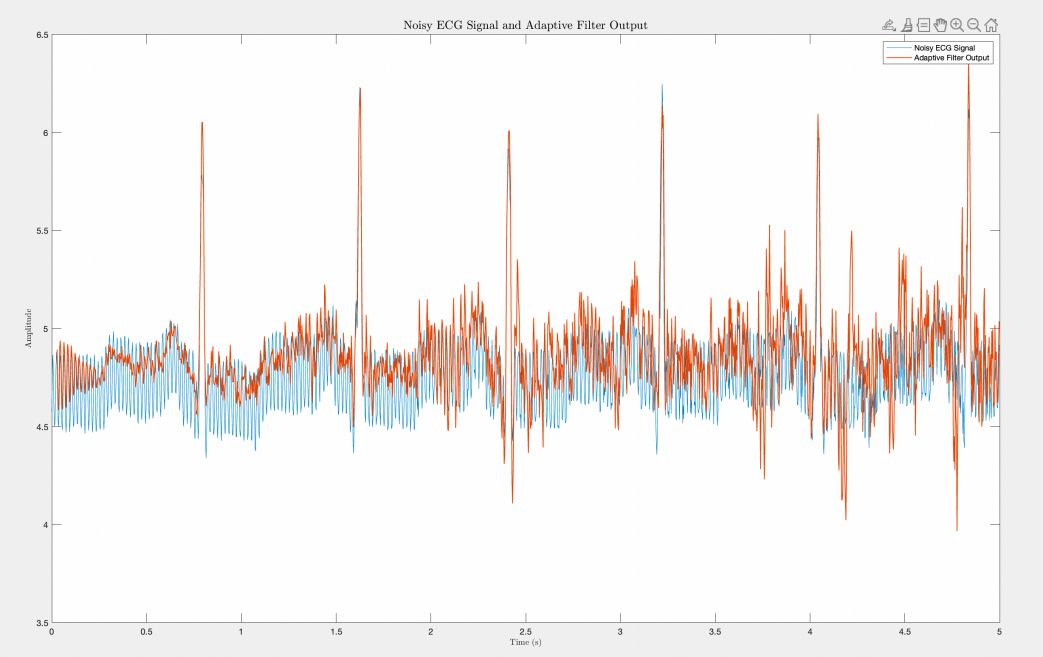
ت)

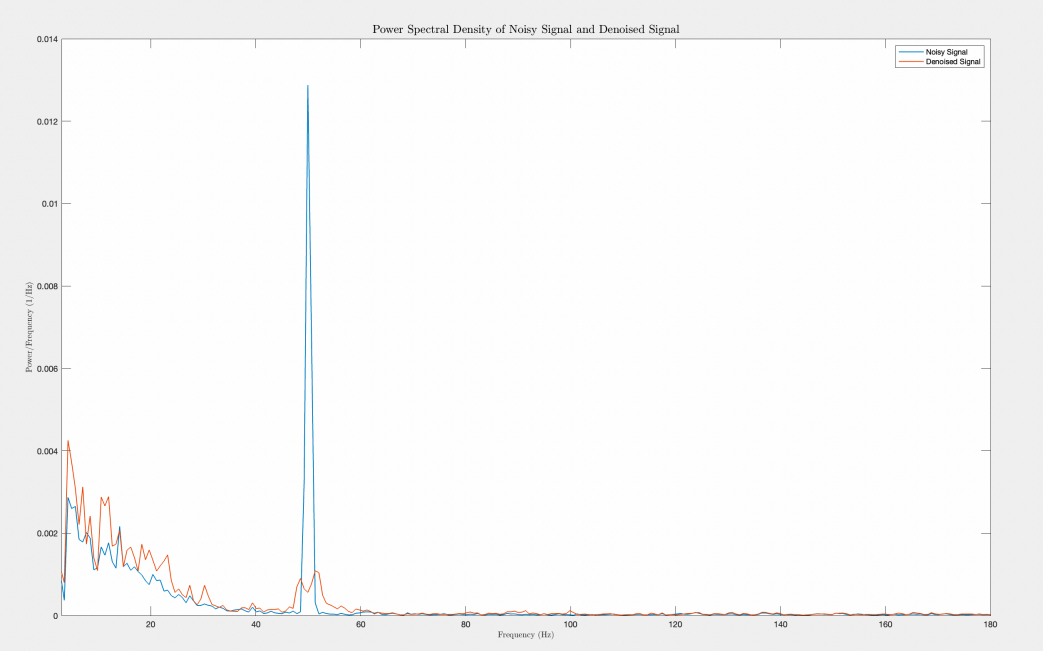


ث)

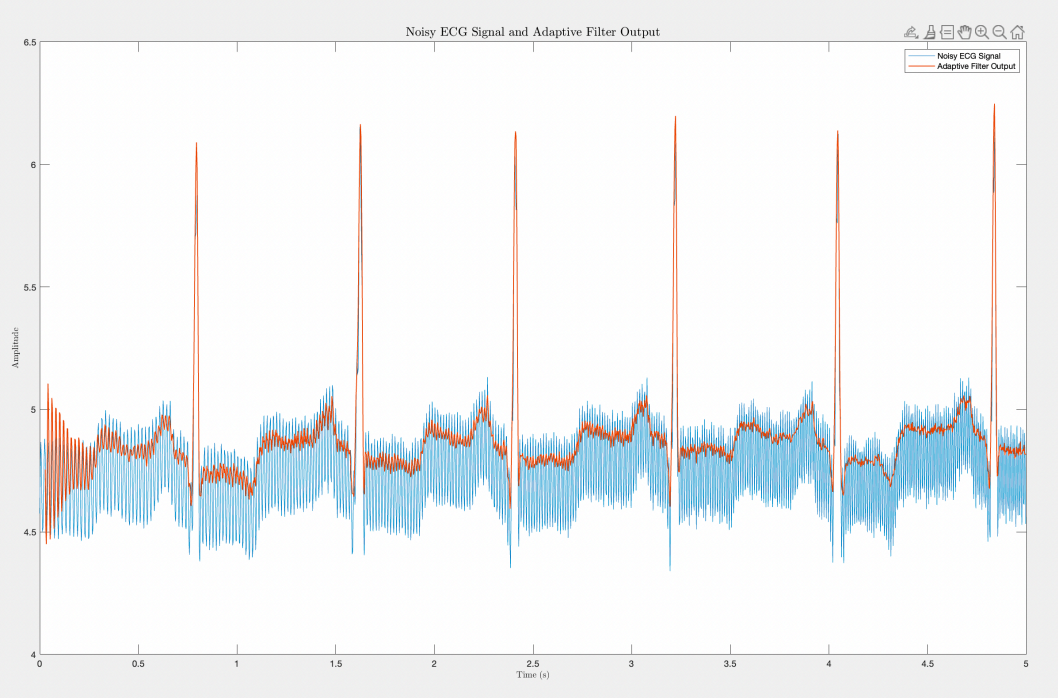
چیزی که من مشاهده کردم این بود که زمانی که این نوسان فرکانسی به صورت خطی بود، یعنی مثلا میگفتیم در طول زمان فرکانس از ۴۸ هرتز به ۵۲ هرتز برسد، فیلتر به خوبی می‌توانست عملیات حذف نویز را انجام دهد. اما زمانی که این تغییرات فرکانسی را به صورت رندوم انجام می‌دادیم (مثلا میگفتیم فرکانس در هر لحظه ۵۰ مثبت منفی ۰.۱ هرتز است و این نوسان رو به صورت رندوم صورت می‌گرفت) فیلتر اصلا نمی‌توانست عملیات حذف نویز را به خوبی انجام دهد.

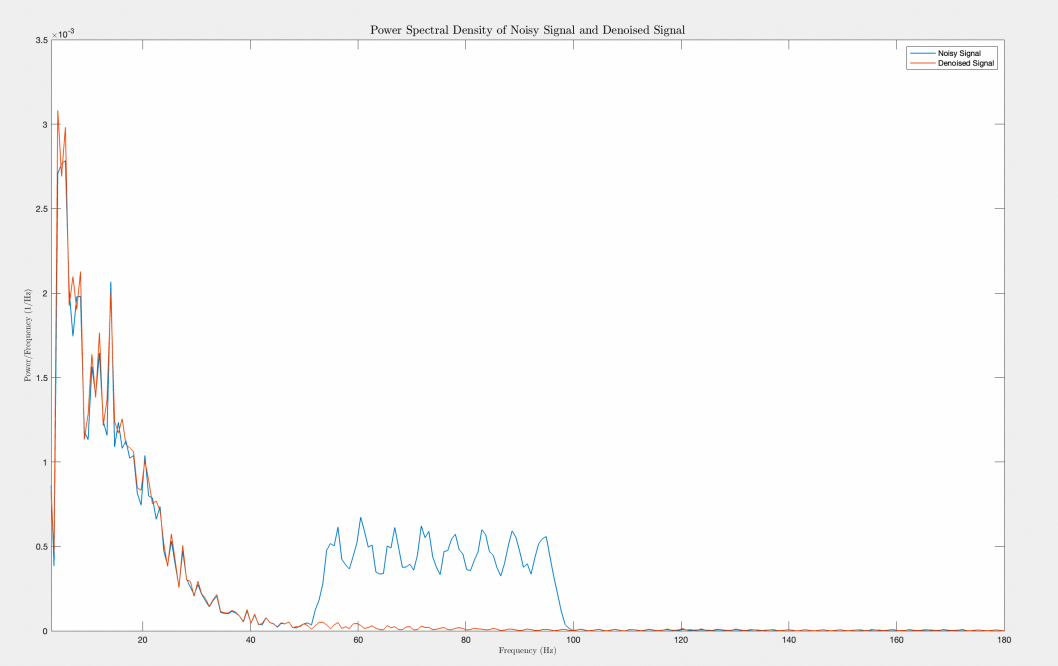
نوسان ۰.۱ به صورت رندوم:





نوسان ۵ هرتز به صورت خطی:



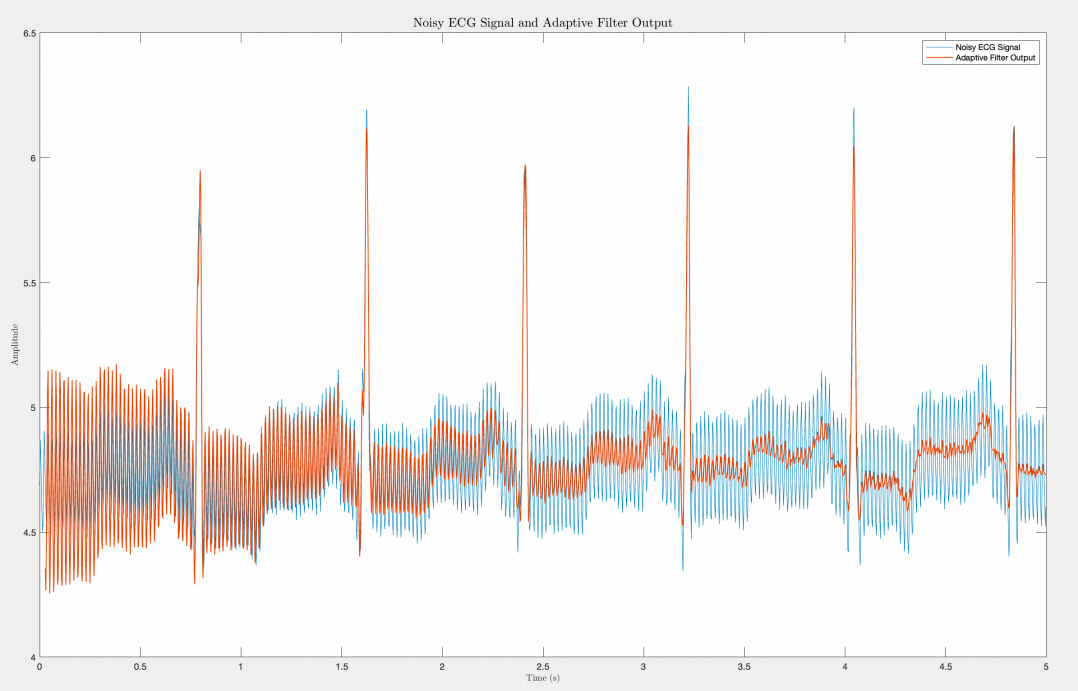


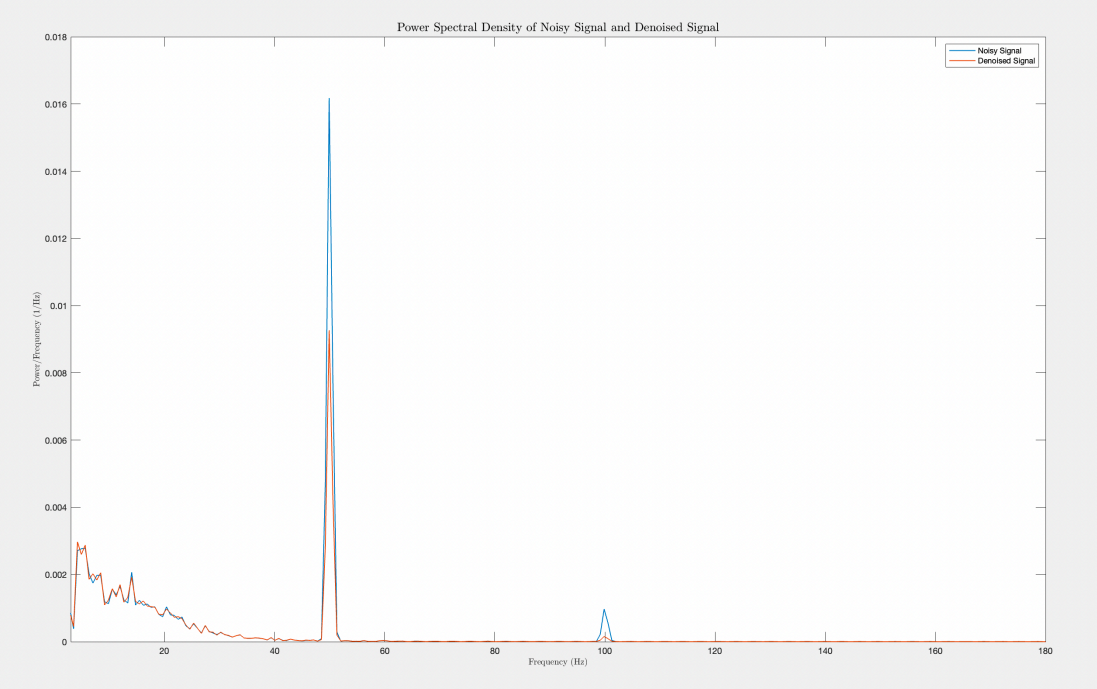
ج)

مشاهده کردیم که در اکثر موارد سیگنال بدون هارمونیک راحت‌تر دی نویز شده است و میزان بهبود سیگنال به نویز در آن بیشتر بوده است ولی خب بعضی دفعات هم میزان بهبود در سیگنال هارمونیک دار بیشتر بوده است.

```
SNR improvement for signal with first harmonic = 1.4671
SNR improvement for harmonic free signal = 4.9233
```

هارمونیک دار:





چ)
متاسفانه به نتیجه مطلوبی نرسید ولی کد زده شد در قسمت کدها موجود می‌باشد.

```
%% BSP - CHW2 - 99101579
%% Q2 - Part a
clc; clear; close all;

fs = 360;
length_sig = 5; % 5 seconds signal
gain = 200;
ecg_signal = load("ECG.mat").val(1:length_sig*fs)/gain;

t = 0:1/fs:length_sig-1/fs;
frequency = 50;
amplitude = 0.2;

phase1 = rand() * 2 * pi;
phase2 = rand() * 2 * pi;

noise1 = amplitude * sin(2 * pi * frequency * t + phase1);
noise2 = amplitude * sin(2 * pi * frequency * t + phase2); % Reference signal

noisy_ecg_signal = ecg_signal + noise1; % Primary signal

figure;
subplot(2, 1, 1);
plot(t, ecg_signal,'LineWidth',1);
title('Original ECG Signal','Interpreter','latex','FontSize',14);
xlabel('Time (s)','Interpreter','latex','FontSize',10);
ylabel('Amplitude','Interpreter','latex','FontSize',10);

subplot(2, 1, 2);
plot(t, noisy_ecg_signal,'LineWidth',1);
title('Noisy ECG Signal','Interpreter','latex','FontSize',14);
```

```
xlabel('Time (s)','Interpreter','latex','FontSize',10);
ylabel('Amplitude','Interpreter','latex','FontSize',10);

M = 15; %Filter Order
mu = 0.01;

[output_signal, error_signal, w] = myAdaptiveFilter(M, mu, noisy_ecg_signal, noise2);

%% Q2 - Part b
clc;
figure;
plot(t, noisy_ecg_signal);
hold on;
plot(t(M+1:end), error_signal(M+1:end),'LineWidth',1);
title('Noisy ECG Signal and Adaptive Filter Output','Interpreter','latex','FontSize',14);
xlabel('Time (s)','Interpreter','latex','FontSize',10);
ylabel('Amplitude','Interpreter','latex','FontSize',10);
legend('Noisy ECG Signal', 'Adaptive Filter Output');
hold off;

% Spectral analysis using pwelch
[pxx_noisy, f] = pwelch(noisy_ecg_signal, [], [], [], fs);
[pxx_denoised, f] = pwelch(error_signal, [], [], [], fs);
[pxx_ecg, f] = pwelch(ecg_signal, [], [], [], fs);

figure;
plot(f, pxx_noisy,'LineWidth',1);
hold on;
plot(f, pxx_denoised,'LineWidth',1);
title('Power Spectral Density of Noisy Signal and Denoised
Signal','Interpreter','latex','FontSize',14);
```

```
xlabel('Frequency (Hz)','Interpreter','latex','FontSize',10);
ylabel('Power/Frequency (1/Hz)','Interpreter','latex','FontSize',10);
legend('Noisy Signal', 'Denoised Signal');
xlim([3 fs/2]);
hold off;



%% Q2 - Part c
clc;
noise_amplitudes = 0.1:0.01:1; % Range of noise amplitudes
snr_improvement = zeros(size(noise_amplitudes));
snr_input = zeros(size(noise_amplitudes));
snr_output = zeros(size(noise_amplitudes));

M = 20;
mu = 0.01;
for i = 1:length(noise_amplitudes)

    amplitude = noise_amplitudes(i);
    phase1 = rand() * 2 * pi;
    phase2 = rand() * 2 * pi;
    noise1 = amplitude * sin(2 * pi * frequency * t + phase1);
    noise2 = amplitude * sin(2 * pi * frequency * t + phase2);

    noisy_ecg_signal = ecg_signal + noise1;

    [output_signal, error_signal, w] = myAdaptiveFilter(M, mu, noisy_ecg_signal, noise2);

    snr_input(i) = calculate_snr(ecg_signal(M+1:end), noise1(M+1:end));
    snr_output(i) = calculate_snr(ecg_signal(M+1:end), error_signal(M+1:end) -
ecg_signal(M+1:end));
```

```
    snr_improvement(i) = snr_output(i) - snr_input(i);

end

figure;
plot(snr_input, snr_improvement,'LineWidth',2);
title('SNR Improvement vs SNR Input','Interpreter','latex','FontSize',14);
xlabel('SNR Input (dB)','Interpreter','latex','FontSize',10);
ylabel('SNR Improvement (dB)','Interpreter','latex','FontSize',10);
grid minor;

%% Q2 - Part d
clc;

fs = 360;
length_sig = 5; % 5 seconds signal
gain = 200;
ecg_signal = load("ECG.mat").val(1:length_sig*fs)/gain;

t = 0:1/fs:length_sig-1/fs;
frequency = 50;
amplitude = 0.2;

phase1 = rand() * 2 * pi;
phase2 = rand() * 2 * pi;

noise1 = amplitude * sin(2 * pi * frequency * t + phase1);
noise2 = amplitude * sin(2 * pi * frequency * t + phase2); % Reference signal

noisy_ecg_signal = ecg_signal + noise1; % Primary signal
```

```
M = 10; %Filter Order
mu = 0.01;

[output_signal, error_signal, w] = myAdaptiveFilter(M, mu, noisy_ecg_signal, noise2);

impulse_response = filter(1, [1; -w], [1; zeros(length(t)-1, 1)]);

[h, f] = freqz([1; -w], 1, 512, fs);

figure;
subplot(3, 1, 1);
stem(impulse_response);
title('Impulse Response of Stop Band Filter');
xlabel('Sample');
ylabel('Amplitude');

subplot(3, 1, 2);
plot(f, 20*log10(abs(h)));
title('Frequency Response of Stop Band Filter');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
grid on;

subplot(3, 1, 3);
zplane([1; -w], 1);
title('Pole-Zero Plot of Stop Band Filter');



%% Q2 - Part e
clc;
```

```
amplitude = 0.2;
center_frequency = 50;
frequency_fluctuation = 5;
% noise_frequency = center_frequency + frequency_fluctuation * rand(size(t));
noise_frequency = center_frequency + frequency_fluctuation * t;

phase1 = rand() * 2 * pi;
phase2 = rand() * 2 * pi;

noise1 = amplitude * sin(2 * pi * noise_frequency .* t + phase1);
noise2 = amplitude * sin(2 * pi * noise_frequency .* t + phase2);

noisy_ecg_signal = ecg_signal + noise1;

M = 10;
mu = 0.1;

[output_signal, error_signal, w] = myAdaptiveFilter(M, mu, noisy_ecg_signal, noise2);



figure;
plot(t, noisy_ecg_signal);
hold on;
plot(t(M+1:end), error_signal(M+1:end),'LineWidth',1);
title('Noisy ECG Signal and Adaptive Filter Output','Interpreter','latex','FontSize',14);
xlabel('Time (s)','Interpreter','latex','FontSize',10);
ylabel('Amplitude','Interpreter','latex','FontSize',10);
legend('Noisy ECG Signal', 'Adaptive Filter Output');
hold off;

[pxx_noisy_harmonic, f] = pwelch(noisy_ecg_signal, [], [], [], fs);
```

```
[pxx_denoised_harmonic, f] = pwelch(error_signal, [], [], [], fs);

[pxx_ecg, f] = pwelch(ecg_signal, [], [], [], fs);


figure;

plot(f, pxx_noisy_harmonic,'LineWidth',1);

hold on;

plot(f, pxx_denoised_harmonic,'LineWidth',1);

title('Power Spectral Density of Noisy Signal and Denoised
Signal','Interpreter','latex','FontSize',14);

xlabel('Frequency (Hz)','Interpreter','latex','FontSize',10);

ylabel('Power/Frequency (1/Hz)','Interpreter','latex','FontSize',10);

legend('Noisy Signal', 'Denoised Signal');

xlim([3 fs/2]);

hold off;



%% Q2 - Part f

clc;

amplitude = 0.2;

noise_frequency = 50;


phase1 = rand() * 2 * pi;

phase2 = rand() * 2 * pi;


noise1_harmonic = amplitude * sin(2 * pi * noise_frequency * t + phase1) + 0.25 *
amplitude * sin(2 * pi * 2 * noise_frequency * t + phase1);

noise2_harmonic = amplitude * sin(2 * pi * noise_frequency * t + phase2) + 0.25 *
amplitude * sin(2 * pi * 2 * noise_frequency * t + phase2);


noise1 = amplitude * sin(2 * pi * noise_frequency * t + phase1);

noise2 = amplitude * sin(2 * pi * noise_frequency * t + phase2);
```

```
noisy_ecg_signal_harmonic = ecg_signal + noise1_harmonic;

noisy_ecg_signal = ecg_signal + noise1;


M = 10;

mu = 0.01;


[output_signal_harmonic, error_signal_harmonic, w_harmonic] = myAdaptiveFilter(M, mu,
noisy_ecg_signal_harmonic, noise2_harmonic);

[output_signal, error_signal, w] = myAdaptiveFilter(M, mu, noisy_ecg_signal, noise2);


snr_input_harmonic = calculate_snr(ecg_signal(M+1:end), noise1_harmonic(M+1:end));

snr_output_harmonic = calculate_snr(ecg_signal(M+1:end), error_signal_harmonic(M+1:end) -
ecg_signal(M+1:end));

snr_improvement_harmonic = snr_output_harmonic - snr_input_harmonic;


snr_input = calculate_snr(ecg_signal(M+1:end), noise1(M+1:end));

snr_output_harmonic = calculate_snr(ecg_signal(M+1:end), error_signal(M+1:end) -
ecg_signal(M+1:end));

snr_improvement = snr_output_harmonic - snr_input_harmonic;


disp(['SNR improvement for signal with first harmonic =
',num2str(snr_improvement_harmonic)]);

disp(['SNR improvement for harmonic free signal = ',num2str(snr_improvement)]);


figure;

plot(t, noisy_ecg_signal);

hold on;

plot(t(M+1:end), error_signal(M+1:end),'LineWidth',1);

title('Noisy ECG Signal and Adaptive Filter Output','Interpreter','latex','FontSize',14);

xlabel('Time (s)','Interpreter','latex','FontSize',10);

ylabel('Amplitude','Interpreter','latex','FontSize',10);

legend('Noisy ECG Signal', 'Adaptive Filter Output');
```

```
hold off;

%% Q2 - Part g
clc;



amplitude = 0.2;
noise_frequency = 50;
noise1 = amplitude * sin(2 * pi * noise_frequency .* t);

noisy_ecg_signal = ecg_signal + noise1;

M = 10;
mu = 0.01;

delays = [1, 5, 10, 20, 50];
snr_improvements = zeros(length(delays), 1);

for i = 1:length(delays)
    delay = delays(i);
    delayed_signal = [zeros(1, delay), noisy_ecg_signal(1:end-delay)];

    [~, error_signal, ~] = myAdaptiveFilter(M, mu, noisy_ecg_signal, delayed_signal);

    snr_input = calculate_snr(ecg_signal(M+1:end), noise1(M+1:end));
    snr_output_harmonic = calculate_snr(ecg_signal(M+1:end), error_signal(M+1:end) -
ecg_signal(M+1:end));
    snr_improvement = snr_output_harmonic - snr_input_harmonic;

    fprintf('SNR Improvement for delay = %d: %.2f dB\n', delay, snr_improvement);
end
```

```
figure;
plot(delays, snr_improvements, '-o');
title('SNR Improvement vs. Delay');
xlabel('Delay (samples)');
ylabel('SNR Improvement (dB)');
grid on;
```

سوال ۳-

(a) The Use of Parametric Models in Processing Vital Signals

Parametric models are used in vital signal processing to provide a detailed mathematical representation of physiological signals. An example is the application of autoregressive models to heart rate variability (HRV) analysis, which allows for the quantification and assessment of autonomic nervous system activity. This approach can detect subtle changes in heart rate dynamics that might indicate underlying health issues. By fitting a model to the observed data, these techniques can enhance signal interpretation, offering insights into physiological conditions and improving diagnostic accuracy (Source: IEEE Xplore).

(b) The Use of Non-Parametric Methods of Spectrum Estimation in Processing Vital Signals

Non-parametric spectrum estimation methods, such as the Welch method, are widely used in the analysis of vital signals to assess power spectral density without assuming an underlying model. For instance, they are employed in EEG signal analysis to identify and monitor brain wave patterns associated with different mental states and pathologies. These methods provide robust frequency domain representations that help in diagnosing neurological conditions by highlighting abnormal spectral features in the EEG signals (Source: SpringerLink).

(c) The Use of Parametric Methods of Spectrum Estimation in Processing Vital Signals

Parametric methods, like the autoregressive (AR) model, are used for spectral estimation in the processing of vital signals such as ECG and EEG. These methods provide high-resolution estimates of the power spectral density, which is essential in identifying specific frequency components associated with physiological and pathological conditions. For example, AR modeling can enhance the detection of arrhythmic events in ECG signals, improving the accuracy of cardiac health assessments (Source: PubMed).

(d) Application of Adaptive Filters in Vital Signal Processing

Adaptive filters are crucial in the processing of vital signals due to their ability to dynamically adjust their parameters for optimal noise cancellation and signal enhancement. Applications include the removal of maternal ECG artifacts from fetal ECG recordings during labor and the reduction of ocular artifacts from EEG signals. These filters improve the clarity and interpretability of the signals by minimizing interference and preserving the desired signal components, thereby enhancing diagnostic accuracy (Source: IntechOpen, Springer).