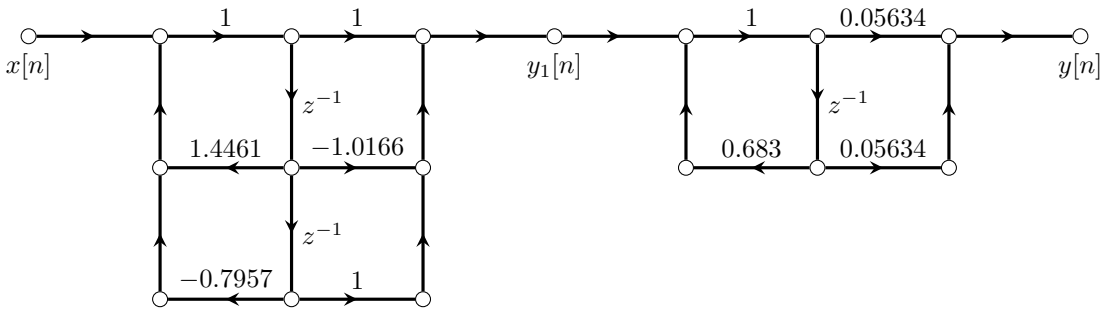
Problem 1

(a)

$$\begin{aligned} H(z) &= 0.05634 \frac{(1+z^{-1})(1-1.0166z^{-1}+z^{-2})}{(1-0.683z^{-1})(1-1.4461z^{-1}+0.7957z^{-2})} \\ &= 0.05634 \frac{1+z^{-1}}{1-0.683z^{-1}} \cdot \frac{1-1.0166z^{-1}+z^{-2}}{1-1.4461z^{-1}+0.7957z^{-2}} \end{aligned} \tag{1}$$

To implement each subsystem you could choose whatever form. The picture below shows the implementation assuming a direct form II.

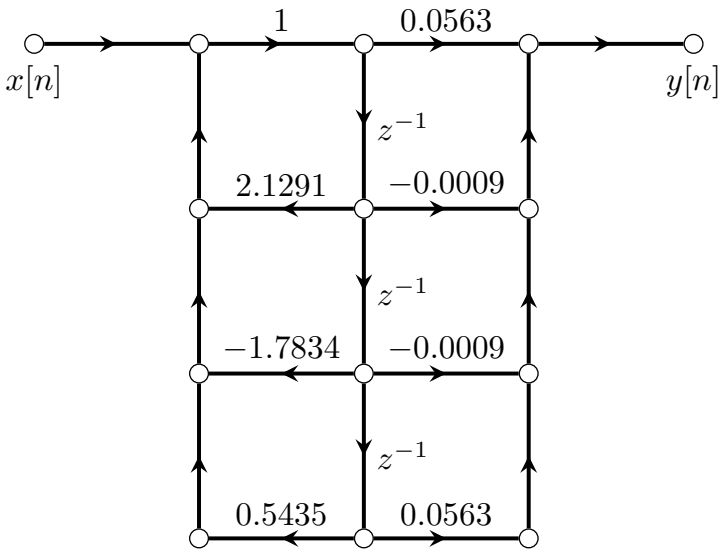


Notation warning: the coefficients corresponding to the poles appear with a different sign, since the diagram assumes a transfer function of the form:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}} \tag{2}$$

(b)

$$\begin{aligned} H(z) &= 0.05634 \frac{(1+z^{-1})(1-1.0166z^{-1}+z^{-2})}{(1-0.683z^{-1})(1-1.4461z^{-1}+0.7957z^{-2})} \\ &= \frac{0.0563 - 0.0009z^{-1} - 0.0009z^{-2} + 0.0563z^{-3}}{1.0000 - 2.1291z^{-1} + 1.7834z^{-2} - 0.5435z^{-3}} \end{aligned} \tag{3}$$



Notation warning: the coefficients corresponding to the poles appear with a different sign.

(c)

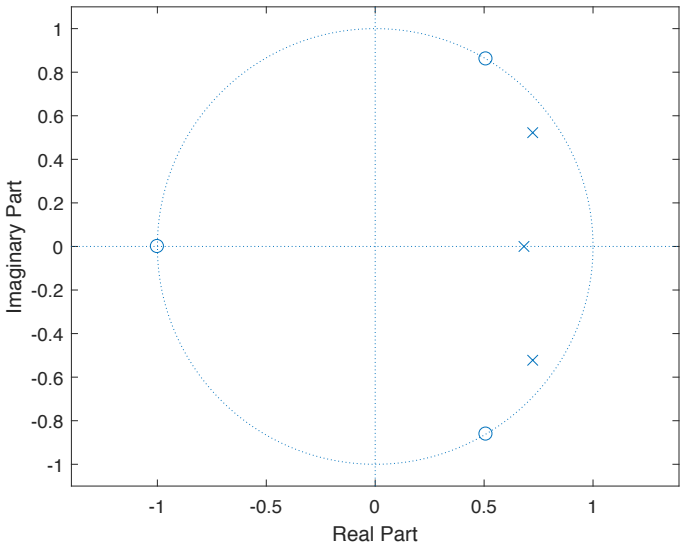


Figure 1: Pole-zero diagram of $H(z)$.

(d)

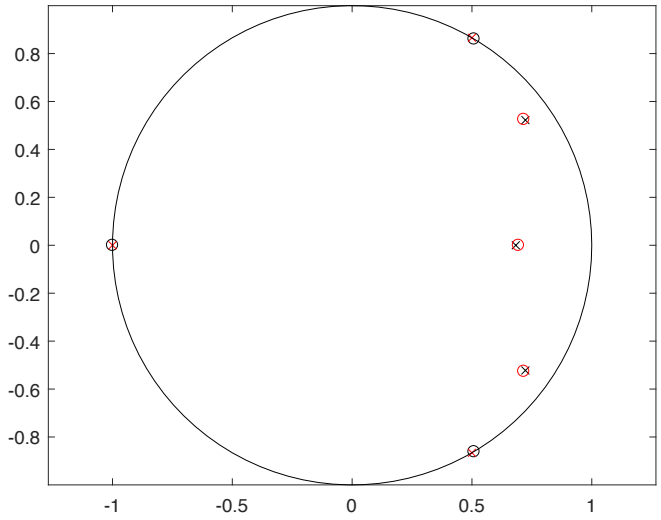


Figure 2: Comparison of pole-zero diagram of $H(z)$ before and after coefficient quantization.

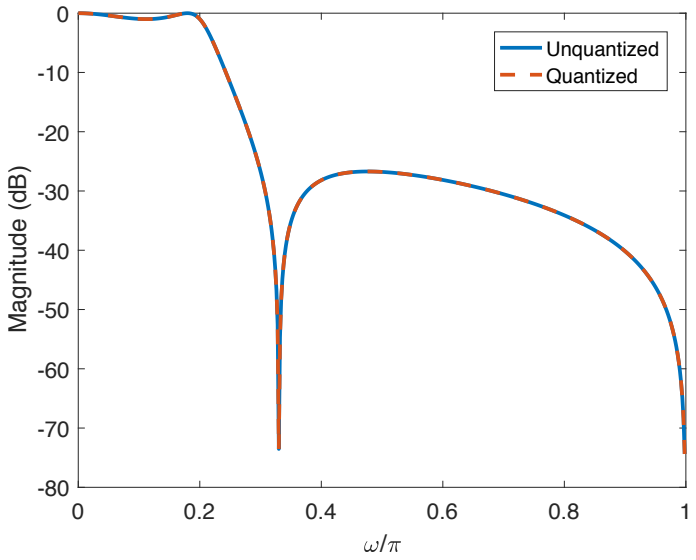


Figure 3: Comparison of magnitude response of $H(z)$ before and after coefficient quantization.

The pole-zero diagram shows that the poles and zeros did not change much, and consequently all other properties of the system should remain roughly the same.

Problem 2

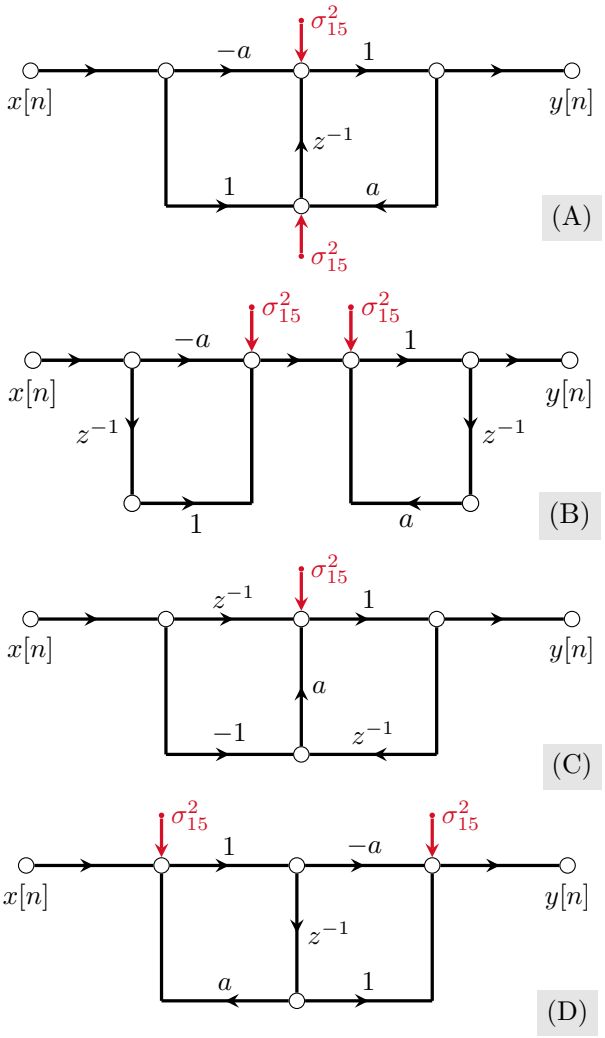
(a)

By inspection

$$H(z) = \frac{z^{-1} - a}{1 - az^{-1}} \tag{4}$$

(b)

$$\sigma_{15}^2 = \frac{\Delta^2}{12} = \frac{2^{-30}}{12} \tag{5}$$



(c)

$$\Phi_{ff}(e^{j\omega}) = 2\sigma_{15}^2 \left| \frac{1}{1 - ae^{-j\omega}} \right|^2 = \frac{2\sigma_{15}^2}{1 + a^2 - 2a\cos\omega} \tag{(A)}$$

$$\Phi_{ff}(e^{j\omega}) = 2\sigma_{15}^2 \left| \frac{1}{1 - ae^{-j\omega}} \right|^2 = \frac{2\sigma_{15}^2}{1 + a^2 - 2a\cos\omega} \tag{(B)}$$

$$\Phi_{ff}(e^{j\omega}) = \sigma_{15}^2 \left| \frac{1}{1 - ae^{-j\omega}} \right|^2 = \frac{\sigma_{15}^2}{1 + a^2 - 2a\cos\omega} \tag{(C)}$$

$$\Phi_{ff}(e^{j\omega}) = \sigma_{15}^2 + \sigma_{15}^2 |H(e^{j\omega})|^2 = 2\sigma_{15}^2 \tag{(D)}$$

(d)

(A) and (B) will have the largest output noise average power.

$$\sigma_f^2 = 2\sigma_{15}^2 \sum_{k=0}^{\infty} (a^2)^k = \frac{2\sigma_{15}^2}{1 - a^2} \tag{6}$$

The above equation comes from computing the inverse DTFT of the PSD found in part (c), which is a a typical second-order transform.

Note: It can be shown that (A) and (B) will have the largest output noise by showing that the integral of the PSD is larger in those systems.

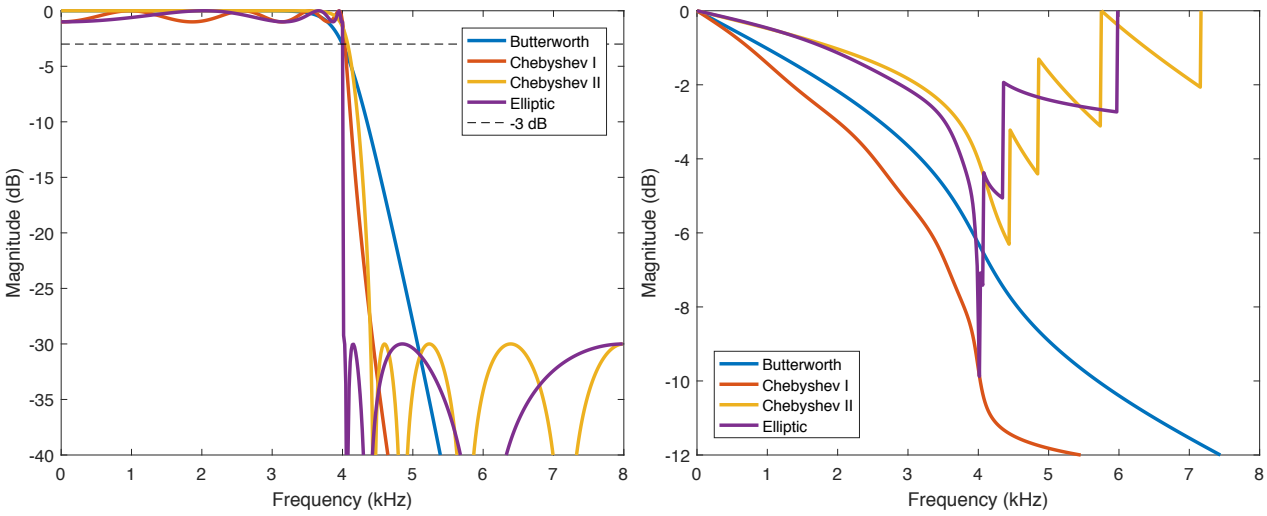
Problem 3

Matlab code for this problem is attached at the end of this question.

(a)

The parameters of the filters are listed below

- 3-dB bandwidth or cut-off frequency: 4 kHz, which for $1/T = 16$ kHz corresponds to $\omega_c = \pi/2$
- Order: 8
- Ripple (only used in Chebyshev I and elliptic filters): 1 dB, any other answer is acceptable. However, the ripple in the passband cannot be very large.
- Stopband edge frequency (only used in Chebyshev II and elliptic filters): $1.1\omega_c$. This value led to a cutoff frequency of nearly 4 kHz. Other answer are accepted, as long as the cutoff frequency is close to 4 kHz
- Stopband attenuation (only used in Chebyshev II and elliptic filters): 30 dB. Other answers are accepted.



(a) Magnitude (b) Phase

Figure 4: (a) Magnitude and (b) phase responses of the filters designed.

(b)

See Matlab code for solution to this question.

To achieve the desired non-integer sampling rate change, we must upsample by 320 and decimate by 441.

One comment regarding implementation is that the lowpass filter between the upsampling and downsampling stages must have gain L. However, the lowpass filter prior to downsampling in decimation (and in the `decimate` function) has gain 1. Therefore, we must scale the result by L. For more details, refer to slide 30 of lecture notes 4.

(c)

This problem is equivalent to interchanging decimation and filtering. Therefore, it follows that

$$H_{32}(z) = H_{16}(z^2) \tag{7}$$

Matlab code for Problem 3

```
%% Comparison of famous filters
clear, clc, close all

%% a) Design filters
N = 8;                          % filter order
wc = pi/2;                      % cutoff frequency wc = 2pi*4/16 = pi/2
Rp = 1;                         % pass-band ripple
ws = 1.1*wc;                    % stopband edge frequency to achieve wc = 4 kHz
Rs = 30;                        % stopband attenuation
T = 1/16e3;                     % sampling period

% Design filters
[num_butter, den_butter] = butter(N, wc/pi);
[num_cheby1, den_cheby1] = cheby1(N, Rp, wc/pi);
[num_cheby2, den_cheby2] = cheby2(N, Rs, ws/pi);
[num_ellip, den_ellip] = ellip(N, Rp, Rs, wc/pi);

% Frequency response
[H_butter, w] = freqz(num_butter, den_butter);
H_cheby1 = freqz(num_cheby1, den_cheby1, w);
H_cheby2 = freqz(num_cheby2, den_cheby2, w);
H_ellip = freqz(num_ellip, den_ellip, w);

% Pole-zero
figure, zplane(num_butter, den_butter)
figure, zplane(num_cheby1, den_cheby1)
figure, zplane(num_cheby2, den_cheby2)
figure, zplane(num_ellip, den_ellip)

% Plot results
figure, hold on, box on
plot(w/(2*pi*T)*1e-3, 20*log10(abs(H_butter)),...
    'LineWidth', 2, 'displayName', 'Butterworth')
plot(w/(2*pi*T)*1e-3, 20*log10(abs(H_cheby1)),...
    'LineWidth', 2, 'displayName', 'Chebyshev I')
plot(w/(2*pi*T)*1e-3, 20*log10(abs(H_cheby2)),...
    'LineWidth', 2, 'displayName', 'Chebyshev II')
plot(w/(2*pi*T)*1e-3, 20*log10(abs(H_ellip)),...
```

```
'LineWidth', 2, 'displayName', 'Elliptic')
plot([0 1e-3/T], [-3 -3], '--k', 'DisplayName', '-3 dB')
xlabel('Frequency (kHz)')
ylabel('Magnitude (dB)')
axis([0 0.5e-3/T -40 0])
legend('-dynamiclegend')
saveas(gca, '../figs/classic_filters_mag', 'epsc')

figure, hold on, box on
plot(w/(2*pi*T)*1e-3, unwrap(angle(H_butter)),...
    'LineWidth', 2, 'displayName', 'Butterworth')
plot(w/(2*pi*T)*1e-3, unwrap(angle(H_cheby1)),...
    'LineWidth', 2, 'displayName', 'Chebyshev I')
plot(w/(2*pi*T)*1e-3, unwrap(angle(H_cheby2)),...
    'LineWidth', 2, 'displayName', 'Chebyshev II')
plot(w/(2*pi*T)*1e-3, unwrap(angle(H_ellip)),...
    'LineWidth', 2, 'displayName', 'Elliptic')
legend('-dynamiclegend', 'Location', 'SouthWest')
xlabel('Frequency (kHz)')
ylabel('Magnitude (dB)')
axis([0 0.5e-3/T -12 0])
saveas(gca, '../figs/classic_filters_phase', 'epsc')

%% b) Filter speech
% Load speech signal
[x_orig, Fs] = audioread('speech_dft.wav');          % Fs is sampling frequency
T = 1/Fs;                                            % sampling period

% Express 16/22.05 as fraction L/M
[L, M] = rat(16/22.05); % L is the umpsampling factor and M is the downsampling factor

%% Upsampling followed by decimation.
% Note that the lowpass filter between the upsampling and downsampling
% stages must have gain L. However, the lowpass filter prior to
% downsampling in decimation (and in the decimate function) has gain 1.
% Therefore, we must scale the result by L
% For more details, refer to slide 30 of lecture notes 4.
x = L*decimate(upsample(x_orig, L), M);
Fs = 16e3;  % Sampling rate after upsampling/decimation

y_butter = filter(num_butter, den_butter, x);
y_cheby1 = filter(num_cheby1, den_cheby1, x);
y_cheby2 = filter(num_cheby2, den_cheby2, x);
y_ellip = filter(num_ellip, den_ellip, x);

```

```
84 sound(y_ellip, Fs) % play sound after filtering
85
86
```

Problem 4

Matlab code for this problem is attached at the end of this question.

(a)

For λ there's only one choice: $\lambda = 2\pi 60 = 120\pi$.

The parameter b controls the sharpness of the notch (loosely the so-called Q factor). We would like to make b smaller as possible, but this would reduce the attenuation of the notch. A reasonable choice is $b = 2\pi 5 = 10\pi$. This allows reasonable sharpness with enough attenuation. Other reasonable answers are accepted.

(b)

For this we just can use the `bilinear` function from Matlab. The frequency pre-warping frequency must be $\Omega_p = \lambda$ so that the location of the notch is preserved when converting the filter to discrete time.

```
>> [bzw, azw] = bilinear(bs, as, 1/T);              (without frequency pre-warping)
>> [bz, az] = bilinear(bs, as, 1/T, lamb/(2*pi));   (with frequency pre-warping)
```

See code for context.

(c)

From Figure 5 we clearly see that we must use frequency pre-warping, otherwise the frequency of the notch would shift, and consequently the filter would no longer be suitable to remove the 60 Hz interference.

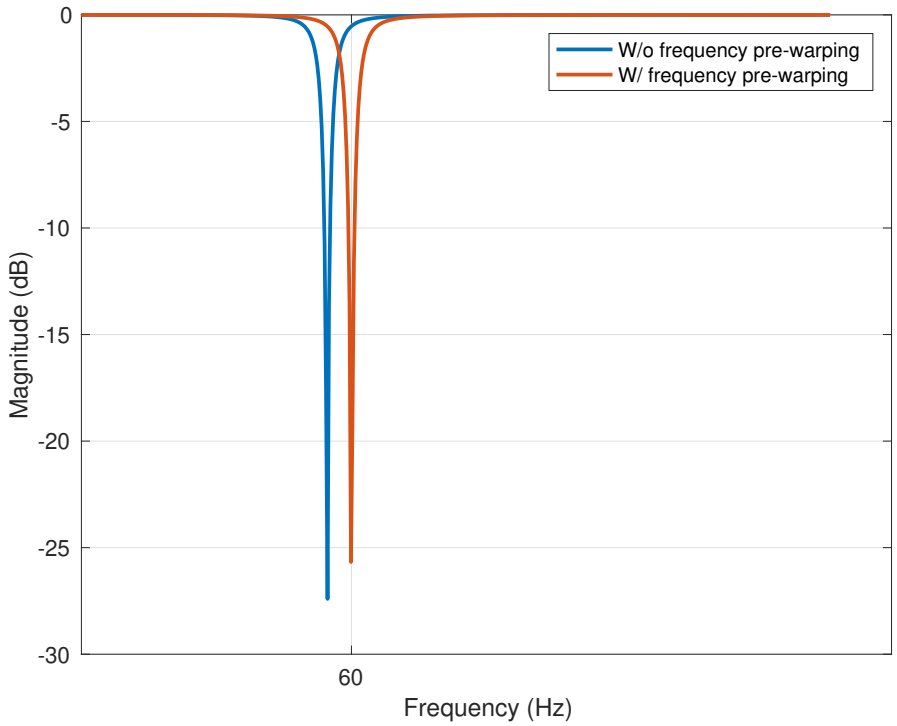


Figure 5: Magnitude response of digital filter obtained by bilinear transformation with and without frequency pre-warping.

(d)

From Figure 5 we see that $H(z)$ obtained with bilinear transformation has zeros at $e^{\pm j2\pi 60T}$. These zeros are responsible for the notch in the magnitude response. The pole close to the zero guarantees that away from $60Hz$ the magnitude response looks flat.

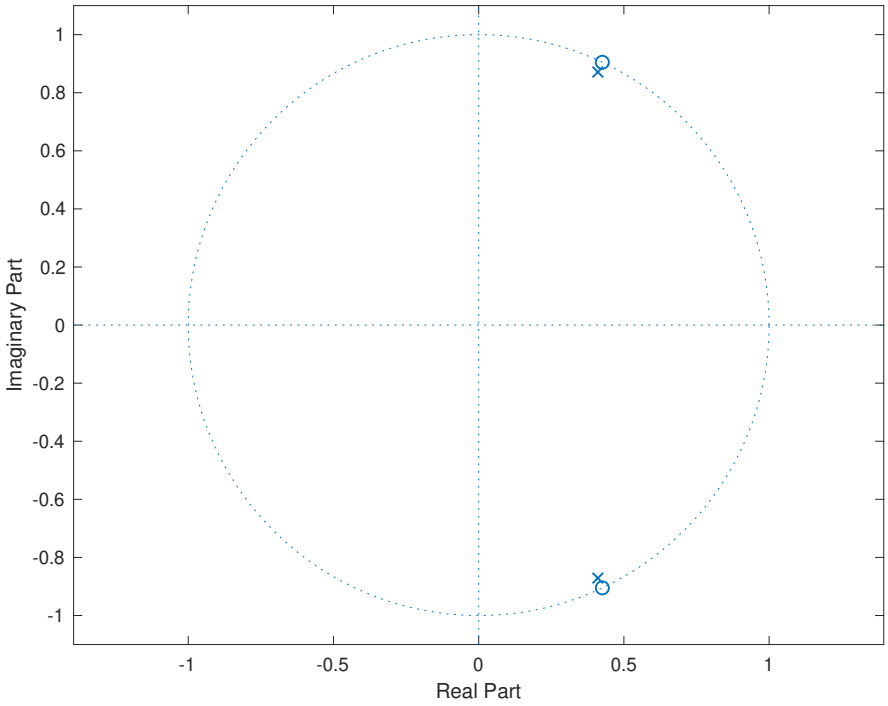


Figure 6: Pole-zero diagram of $H(z)$ obtained by bilinear transformation with frequency pre-warping.

(e)

As shown in Figures 7 and 8 the digital notch filter designed effectively eliminates the 60-Hz interference.

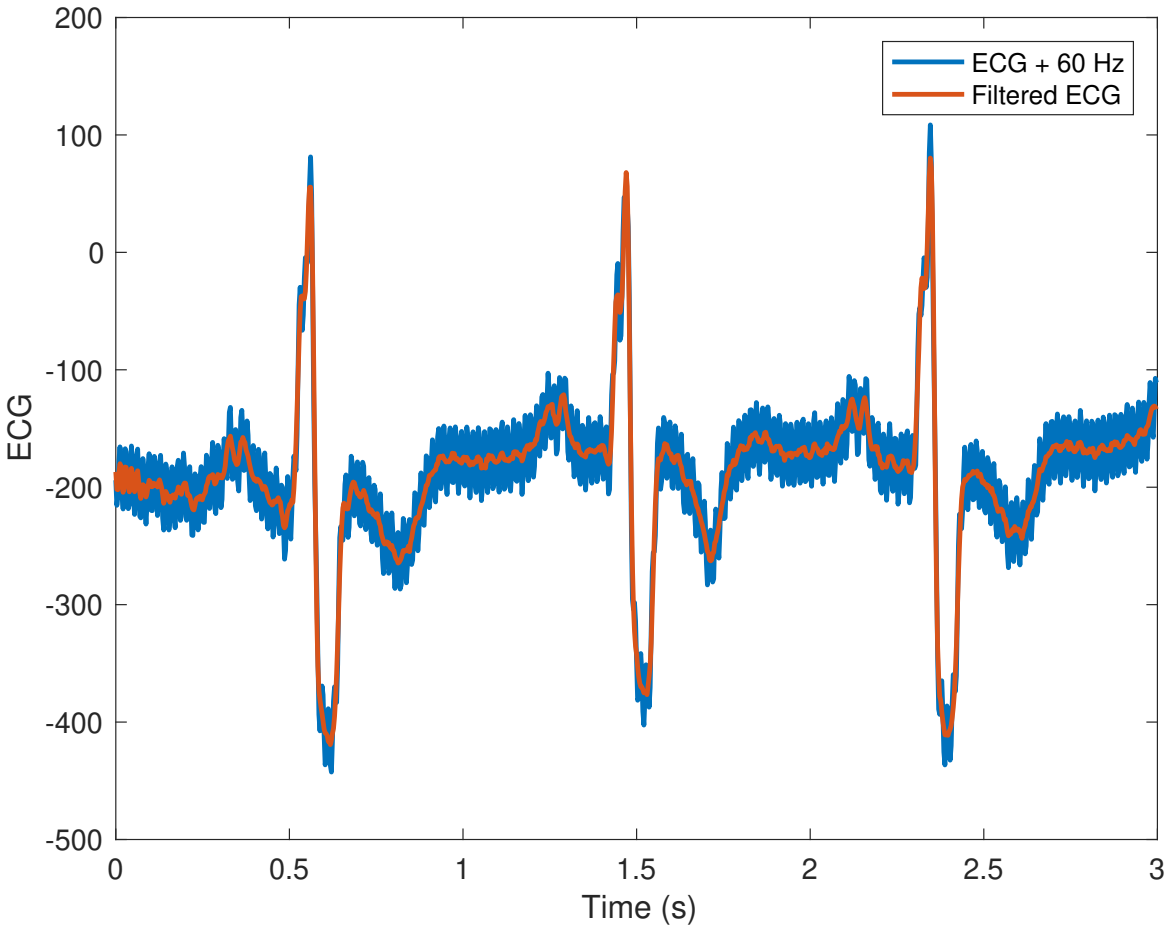


Figure 7: Comparison between digital notch filter input (ECG with 60 interference), and digital notch filter output.

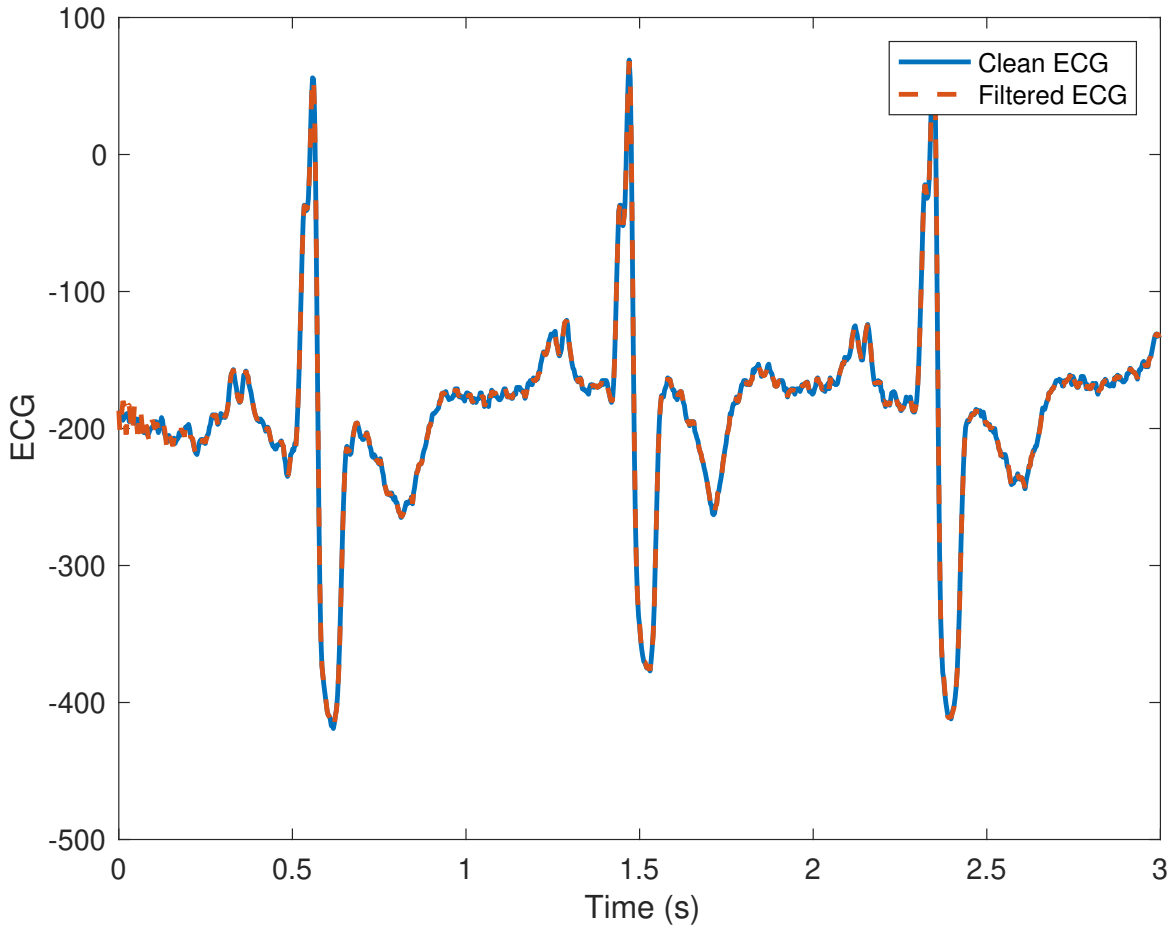


Figure 8: Comparison between digital "clean" ECG signal (ECG without 60 interference), and the digital notch filter output.

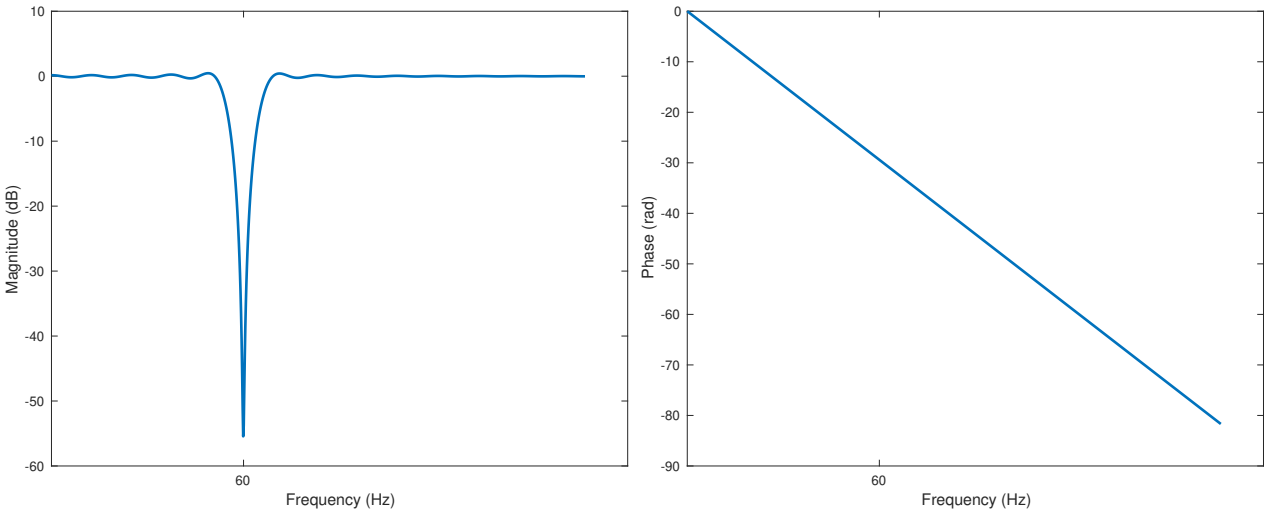
(f)

Figure 9a shows the magnitude response of the designed FIR filter. The filter was optimized using the least-squares algorithm with the following design parameters:

- Order: 52 the highest value allowed in the problem statement
- $\omega_c T = 2\pi 60 = 120\pi$. So that the notch falls at 60 Hz.
- $\Delta\omega_1 = 0.018\pi$. Other values are accepted.
- $\Delta\omega_2 = 0.04\pi$. Other values are accepted.

This filter has nice properties. It has a high attenuation of 55 dB in the notch, which is higher than the filter from part (c). Moreover, the notch is not too wide. Furthermore, the ripples in the passband are not significant.

The filter from the 2004 paper has higher attenuation at the cost of a wider notch and more ripples in the passband. Moreover, their filter was designed for a higher oversampling rate, which facilitates filtering. Their sampling rate was 1 kHz, while the sample rate of our signal was ~ 333 Hz.



(a) Magnitude response of FIR notch filter.

(b) Phase response of FIR notch filter.

Figure 9: (a) magnitude and (b) phase response of FIR notch filter.

Due to the linear phase property, the FIR filter introduces a delay of $M/2 = 26$ samples. The plots below show the comparison of the FIR filter with the "clean" ECG with delay and without delay.

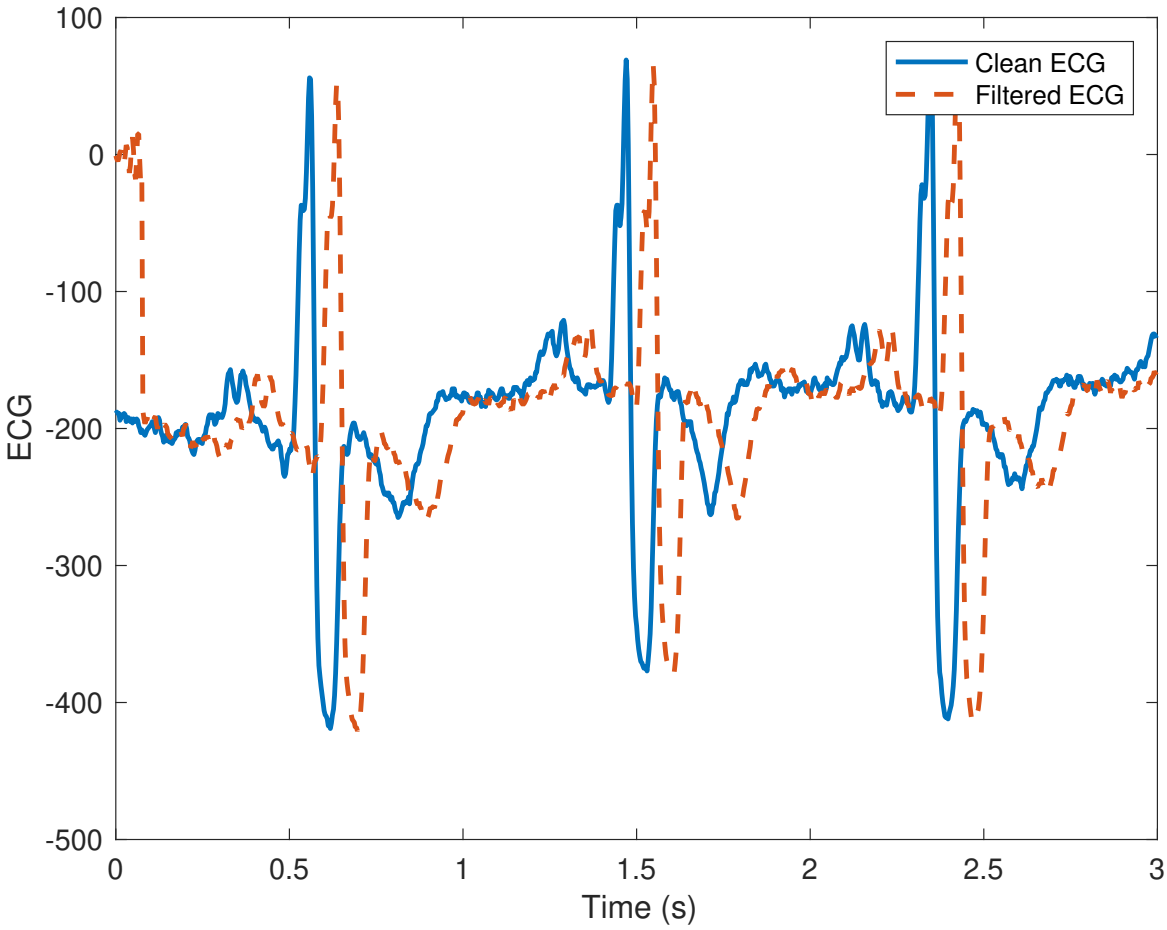


Figure 10: Comparison between digital "clean" ECG signal (ECG without 60 interference), and the digital notch FIR filter output. In this case the delay introduced by the filter was not removed.

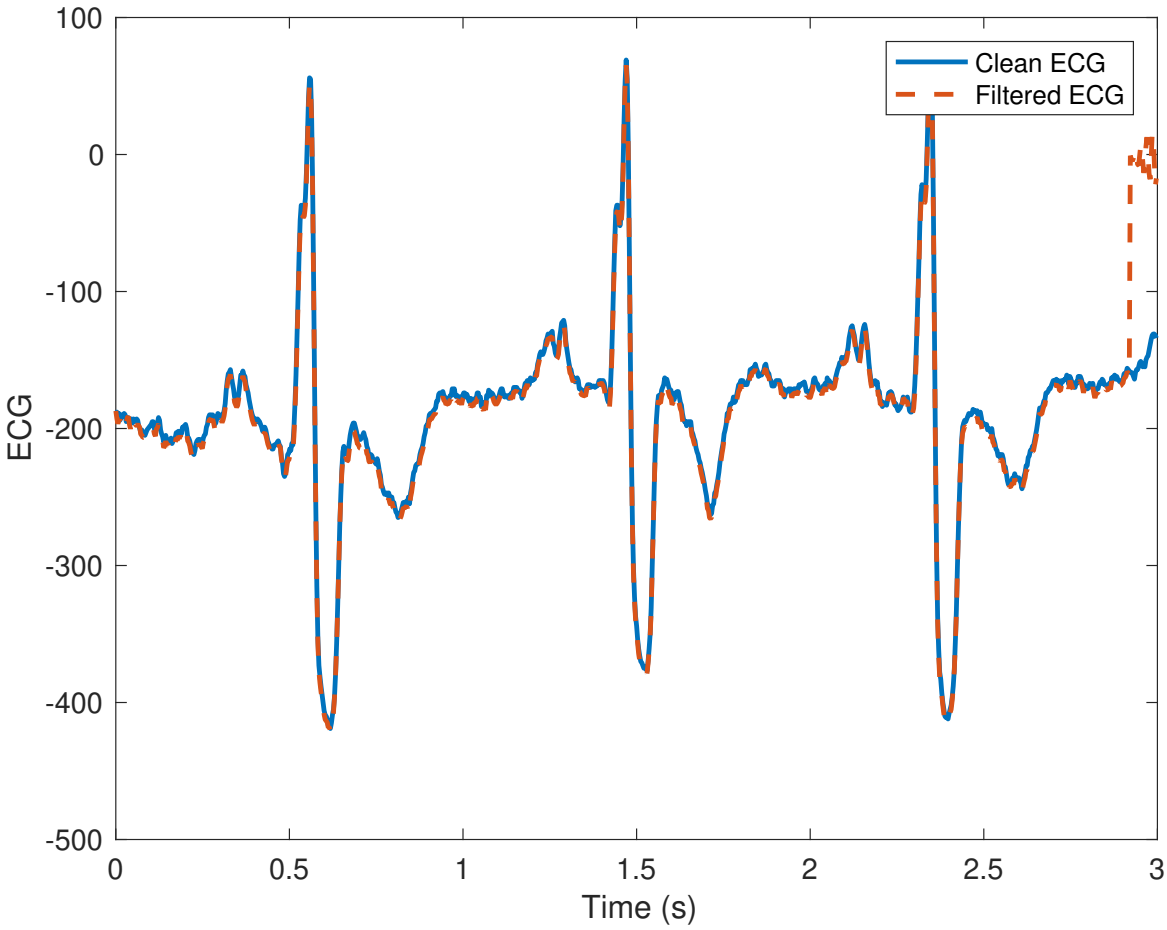


Figure 11: Comparison between digital "clean" ECG signal (ECG without 60 interference), and the digital notch FIR filter output. Delay introduced by the FIR filter was removed.

Matlab code for Problem 4

```
%% Template for notch filtering of ECG signals
clear, clc, close all

%% Loads ECG signal from file
% This loads the following variables:
% - ecg_clean: ecg_signal without 60 Hz interference. Use this only for
% comparison
% - ecg_60Hz: ECG signal corrupted by 60 Hz interference
% - N: length of the vectors ecg_clean and ecg_60Hz
% - T (in seconds): sampling period of ECG recordings. T = 3 ms.
load('ecg_recording.mat')
% Note: these signals are real ECG recordings made with standard ECG recorders,
% leads, and electrodes. The quality is typical of ambulatory ECG recordings.
% The 60 Hz component was introduced artificially for this exercise.
% The complete database can be found online on PhysioNet [1]:
% http://www.physionet.org/physiobank/database/nstdb/
% This database is discrebed in [2].
% References:
% [1] Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG,
% Mietus JE, Moody GB, Peng C-K, Stanley HE. PhysioBank, PhysioToolkit, and
% PhysioNet: Components of a New Research Resource for Complex Physiologic
% Signals. Circulation 101(23):e215-e220 [Circulation Electronic Pages;
% http://circ.ahajournals.org/content/101/23/e215.full]; 2000 (June 13).
% [2] Moody GB, Muldrow WE, Mark RG. A noise stress test for arrhythmia detectors.
% Computers in Cardiology 1984; 11:381-384.

t = 0:T:(N-1)*T; % time vector

% Plot signals
figure, box on, hold on
plot(t, ecg_clean, 'k')
plot(t, ecg_60Hz)
xlabel('Time (s)')
ylabel('ECG')
legend('Clean ECG', 'ECG corrupted by 60 Hz')

%% Your code goes here
% a)
lamb = 2*pi*60;
b = 2*pi*5; % there's more than one right answer for this.

% Analog notch filter coefficients
bs = [1 0 lamb^2];
as = [1 b lamb^2];
```

```
45
46 % b)
47 % Bilinear transformation with and without frequency pre-warping
48 [bzw, azw] = bilinear(bs, as, 1/T); % without
49 [bz, az] = bilinear(bs, as, 1/T, lamb/(2*pi)); % with
50
51 % c)
52 [Hw, w] = freqz(bzw, azw);
53 H = freqz(bz, az, w);
54
55 figure, hold on, box on
56 plot(w/(2*pi*T), 20*log10(abs(Hw)), 'Linewidth', 2, 'DisplayName', 'W/o frequency pre-warp
57 plot(w/(2*pi*T), 20*log10(abs(H)), 'Linewidth', 2, 'DisplayName', 'W/ frequency pre-warpi
58 xlabel('Frequency (Hz)', 'FontSize', 12)
59 ylabel('Magnitude (dB)', 'FontSize', 12)
60 set(gca, 'xtick', [60])
61 grid on
62 set(gca, 'FontSize', 12)
63 legend('-dynamiclegend')
64 saveas(gca, '../figs/bilinear_ecg', 'epsc')
65
66 % d)
67 figure, zplane(bz, az)
68 saveas(gca, '../figs/zplane_bilinear_ecg', 'epsc')
69
70 % e)
71 ecg_filtered = filter(bz, az, ecg_60Hz);
72
73 figure, hold on, box on
74 plot(t, ecg_60Hz, 'LineWidth', 2, 'DisplayName', 'ECG + 60 Hz')
75 plot(t, ecg_filtered, 'LineWidth', 2, 'DisplayName', 'Filtered ECG')
76 legend('-dynamiclegend')
77 xlabel('Time (s)', 'FontSize', 12)
78 ylabel('ECG', 'FontSize', 12)
79 set(gca, 'FontSize', 12)
80 saveas(gca, '../figs/ecg_comparison1', 'epsc')
81
82 figure, hold on, box on
83 plot(t, ecg_clean, 'LineWidth', 2, 'DisplayName', 'Clean ECG')
84 plot(t, ecg_filtered, '--', 'LineWidth', 2, 'DisplayName', 'Filtered ECG')
85 legend('-dynamiclegend')
86 xlabel('Time (s)', 'FontSize', 12)
87 ylabel('ECG', 'FontSize', 12)
88 set(gca, 'FontSize', 12)
89 saveas(gca, '../figs/ecg_comparison2', 'epsc')
```

```
90
91  % f)
92  % Define filter parameters
93  L = 53;
94  M = L-1;
95  Deltaw1 = 0.018*pi;
96  Deltaw2 = 0.04*pi;
97  N = 1000;
98
99  % Frequnecy vector
100 wc = 2*pi*60*T;
101 w = linspace(0, pi, N);
102
103 % Desired response
104 d = ones(size(w));
105 d(w >= wc-Deltaw1/2 & w <= wc+Deltaw1/2) = 0;
106
107 % Weight function
108 wv = ones(size(w));
109 wv(w >= wc-Deltaw1/2-Deltaw2 & w < wc-Deltaw1/2) = 0;
110 wv(w >= wc+Deltaw1/2 & w < wc+Deltaw1/2+Deltaw2) = 0;
111
112 % Plot desired response and window function
113 figure, hold on
114 plot(w, d)
115 plot(w, wv, 'r')
116 title('desired response and window function')
117
118 % Design the filter
119 hls = firls(M, w(wv ~= 0)/pi, d(wv ~= 0));
120
121 % Plot results
122 Hls = freqz(hls, 1, w);
123
124 ecg_fir =filter(hls, 1, ecg_60Hz);
125
126 figure, box on
127 plot(w/(2*pi*T), 20*log10(abs(Hls)), 'LineWidth', 2)
128 xlabel('Frequency (Hz)', 'FontSize', 12)
129 ylabel('Magnitude (dB)', 'FontSize', 12)
130 set(gca, 'xtick', [60])
131 saveas(gca, '../figs/ecg_fir_notch', 'epsc')
132
133 figure, box on
134 plot(w/(2*pi*T), unwrap(angle(Hls)), 'LineWidth', 2)
```

```
135 xlabel('Frequency (Hz)', 'FontSize', 12)
136 ylabel('Phase (rad)', 'FontSize', 12)
137 set(gca, 'xtick', [60])
138 saveas(gca, '../figs/ecg_fir_notch_phase', 'epsc')
139
140 figure, hold on, box on
141 plot(t, ecg_clean, 'LineWidth', 2, 'DisplayName', 'Clean ECG')
142 plot(t, ecg_fir, '--', 'LineWidth', 2, 'DisplayName', 'Filtered ECG')
143 legend('-dynamiclegend')
144 xlabel('Time (s)', 'FontSize', 12)
145 ylabel('ECG', 'FontSize', 12)
146 set(gca, 'FontSize', 12)
147 saveas(gca, '../figs/ecg_fir_comparison1', 'epsc')
148
149 figure, hold on, box on
150 plot(t, ecg_clean, 'LineWidth', 2, 'DisplayName', 'Clean ECG')
151 plot(t, circshift(ecg_fir, [0, -M/2]), '--', 'LineWidth', 2, 'DisplayName', 'Filtered ECG')
152 legend('-dynamiclegend')
153 xlabel('Time (s)', 'FontSize', 12)
154 ylabel('ECG', 'FontSize', 12)
155 set(gca, 'FontSize', 12)
156 saveas(gca, '../figs/ecg_fir_comparison2', 'epsc')
```

Problem 5

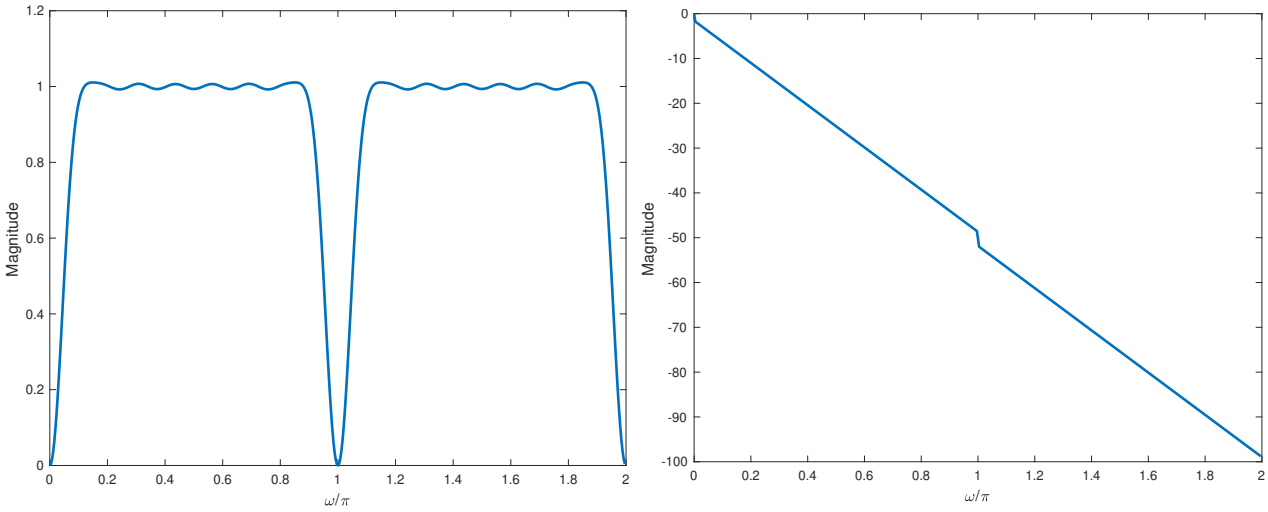
(a)

For the window method we simply need to make

$$h[n] = h_d[n - M/2]w[n] = \begin{cases} w[n]\frac{2}{\pi}\frac{\sin^2((n-M/2)\pi/2)}{n}, & n = 0, \ldots, M-1 \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

where $w[n]$ is any window of your choice, and the time shift by $M/2$ is necessary to make $h[n]$ causal.

The magnitude response of the FIR Hilbert transformer using the window method with Hamming window is shown below. Other windows are also accepted.



(a) Magnitude (b) Phase

Figure 12: (a) magnitude and (b) phase response of Hilbert transformer designed by windowing with Hamming window.

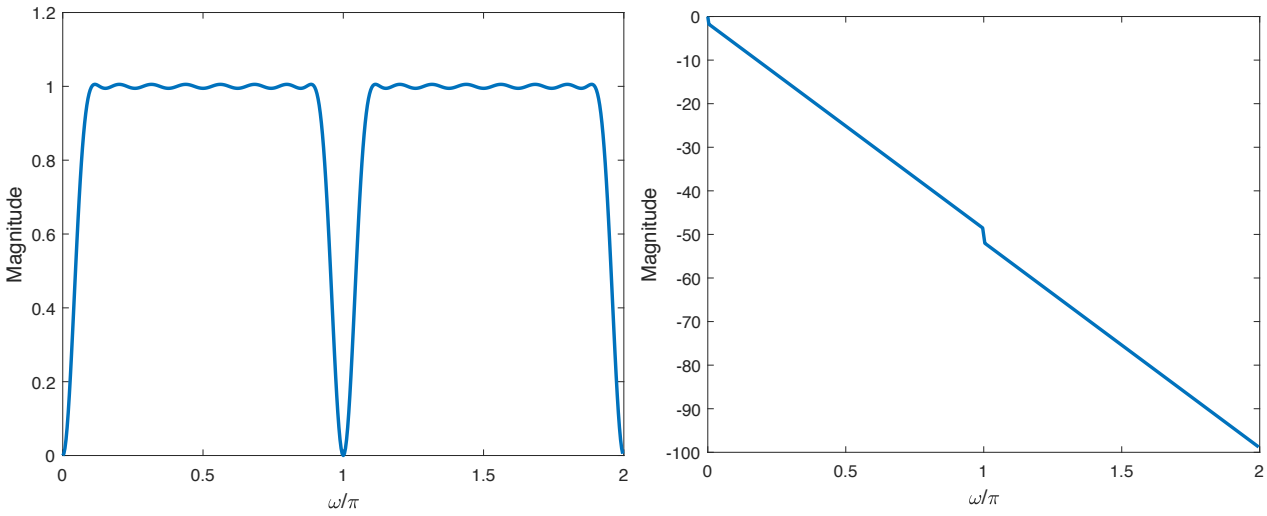
See the code for details.

(b)

To design the Hilbert transformer by Parks-McClellan we can simply do

```
>> hpm = firpm(M, [0.1 0.9], [1 1], 'hilbert');
```

This filter will have constant magnitude between 0.1π and 0.9π. The parameter `'hilbert'` guarantees that the filter has odd symmetry.



(a) Magnitude (b) Phase

Figure 13: (a) magnitude and (b) phase response of Hilbert transformer designed by Parks-McClellan method.

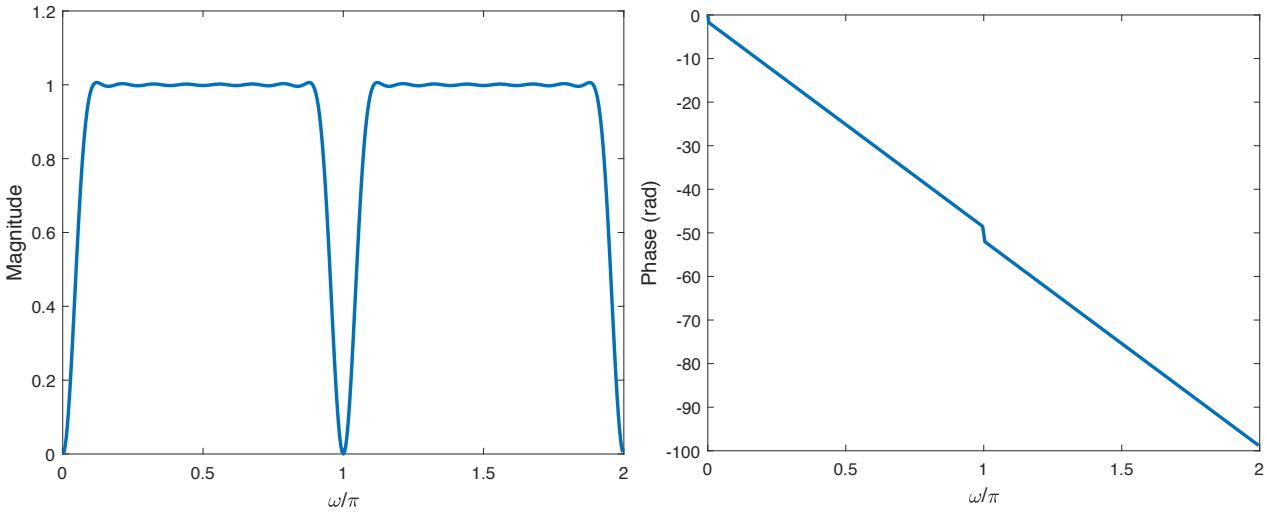
See the code for details.

(c)

Similarly to part (c), to design the Hilbert transformer by leasts-squares algorithm we can simply do

```
>> hls = firls(M, [0.1 0.9], [1 1], 'hilbert');
```

This filter will have constant magnitude between 0.1π and 0.9π. The parameter `'hilbert'` guarantees that the filter has odd symmetry.



(a) Magnitude (b) Phase

Figure 14: (a) magnitude and (b) phase response of Hilbert transformer designed by leasts-squares method.

See the code for details.

Matlab code for Problem 5

```
%% Hilbert transformer
clear, clc, close all

% Define parameters
L = 31;
M = L -1;

%% a) Window method
n = -M/2:M/2;
hd = 2/pi*((sin(pi*n/2)).^2)./n;
hd(n == 0) = 0; % enforce value at 0

% window by Hamming window
hwin = hd.*hamming(M+1).';

[Hwin, w] = freqz(hwin, 1, 'whole');

figure, box on
plot(w/pi, abs(Hwin).^2, 'LineWidth', 2)
xlabel('\omega/\pi', 'FontSize', 12)
ylabel('Magnitude', 'FontSize', 12)
saveas(gca, '../figs/hilbert_win_mag', 'epsc')

figure, box on
plot(w/pi, unwrap(angle(Hwin)), 'LineWidth', 2)
xlabel('\omega/\pi', 'FontSize', 12)
ylabel('Magnitude', 'FontSize', 12)
saveas(gca, '../figs/hilbert_win_phase', 'epsc')

%% b) Parks-McClellan method
hpm = firpm(M, [0.1 0.9], [1 1], 'hilbert');
figure, freqz(hpm, 1)

[Hpm, w] = freqz(hpm, 1, 'whole');

figure, box on
plot(w/pi, abs(Hpm).^2, 'LineWidth', 2)
xlabel('\omega/\pi', 'FontSize', 12)
ylabel('Magnitude', 'FontSize', 12)
saveas(gca, '../figs/hilbert_pm_mag', 'epsc')

figure, box on
plot(w/pi, unwrap(angle(Hpm)), 'LineWidth', 2)
xlabel('\omega/\pi', 'FontSize', 12)
```

```
ylabel('Magnitude', 'FontSize', 12)
saveas(gca, '../figs/hilbert_pm_phase', 'epsc')

%% c) Least-squares method
hls = firls(M, [0.1 0.9], [1 1], 'hilbert');

[Hls, w] = freqz(hls, 1, 'whole');

figure, box on
plot(w/pi, abs(Hls).^2, 'LineWidth', 2)
xlabel('\omega/\pi', 'FontSize', 12)
ylabel('Magnitude', 'FontSize', 12)
saveas(gca, '../figs/hilbert_ls_mag', 'epsc')

figure, box on
plot(w/pi, unwrap(angle(Hls)), 'LineWidth', 2)
xlabel('\omega/\pi', 'FontSize', 12)
ylabel('Phase (rad)', 'FontSize', 12)
saveas(gca, '../figs/hilbert_ls_phase', 'epsc')
```

Problem 6

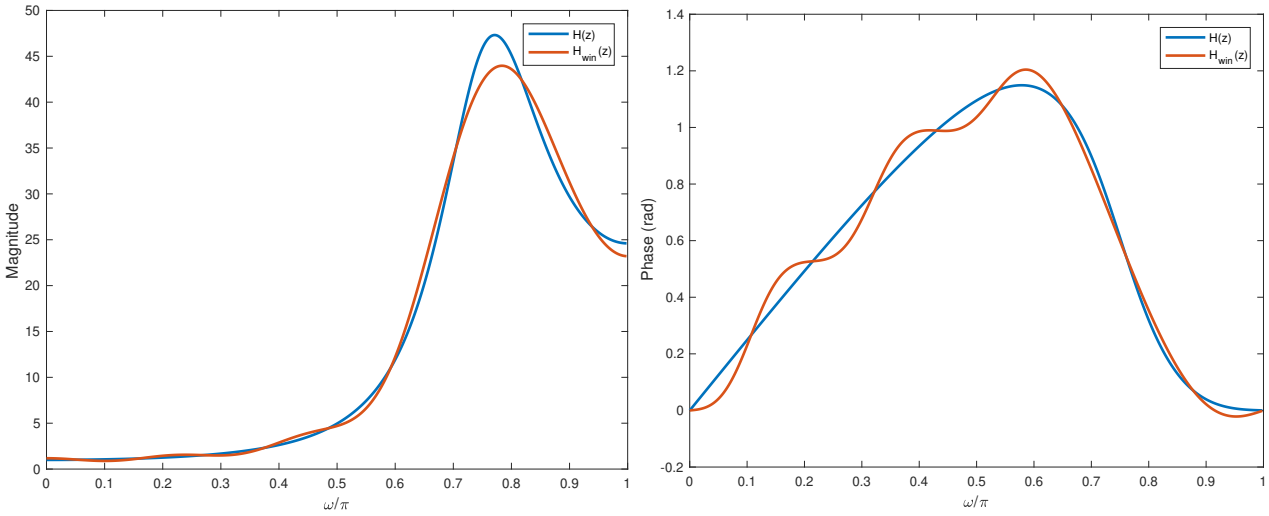
(a)

For the window method we simply need to make

$$h_{win}[n] = \begin{cases} h[n], n = 0, \ldots, M-1 \\ 0, \text{otherwise} \end{cases} \tag{9}$$

We can obtain $h[n]$ analytically or by using the `impz` function from Matlab.

The magnitude and phase plots of $H(z)$ and the filter designed by windowing $H_{win}(z)$ are shown below. The magnitude is plotted in linear units. Note that the phase is nonlinear.



(a) Magnitude (b) Phase

Figure 15: (a) Magnitude and (b) phase response of the filter designed by windowing compared to the desired $H(z)$.

(b)

```
>> BER = simple_channel(1) = 0.2442        (without equalization)
>> BER = simple_channel(hwin) = 0          (with equalization)
```

Different (but close) values are possible depending on the random number generator seed.

(c)

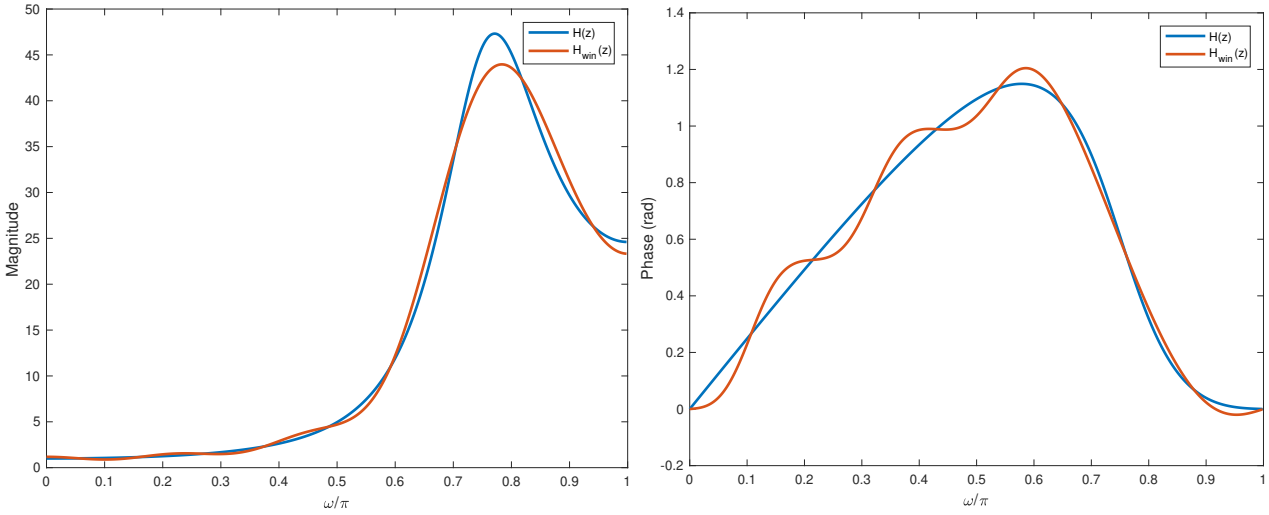
To design a non-linear phase FIR filter using the least-squares algorithm, we need to define the matrix Q as done in lecture notes 9, slide 49. The code to generate this matrix was available on Canvas/Files/Matlab: `predicing_bandlimited_signals.m`.

Once the matrix Q, we can use least-squares as usual:

$$h = Q^{\dagger} d \tag{10}$$

since the weight matrix W is equal to the identity.

This results in the following magnitude and phase responses:



(a) Magnitude (b) Phase

Figure 16: (a) Magnitude and (b) phase response of the filter designed by leasts-squares compared to the desired $H(z)$.

As expected, this is identical to the result obtained by windoing. This only happens because in this case the weight function was one everywhere i.e., the matrix Q was the identity.

(d)

The noise is filtered by $H_{win}(z) = H_{ls}(z)$. Therefore, the noise PSD is shaped by

$$\Phi_{ff}(e^{j\omega}) = \sigma^2 |H_{win}(e^{j\omega})|^2 \tag{11}$$

Hence, the total power after filtering is

$$\sigma_f^2 = \frac{\sigma^2}{2\pi} \int_{-\pi}^{\pi} |H_{win}(e^{j\omega})|^2 d\omega \tag{12}$$

$$= \sum_{m=0}^{M} |h_{win}[m]|^2 \tag{13}$$

$$= 0.0144 \tag{14}$$

This value is 14 times larger than σ^2.

Matlab code for problem 6

```
%% Linear equalization
clear, clc, close all

M = 8; % Filter order

```

```
% Filter coefficients
c = [0.4032    0.3992    0.1976];

% Impulse response of 1/C(z)
hc = impz(1, c);

%% a) Window method with rectangular window
hwin = hc(1:M+1);

BER = simple_channel(hwin)

% Plot results
[H, w] = freqz(1, c);
Hwin = freqz(hwin, 1, w);

figure, hold on, box on
plot(w/pi, abs(H).^2, 'LineWidth', 2, 'displayname', 'H(z)')
plot(w/pi, abs(Hwin).^2, 'LineWidth', 2, 'displayname', 'H_{win}(z)')
xlabel('\omega/\pi', 'FontSize', 12)
ylabel('Magnitude', 'FontSize', 12)
legend('-dynamiclegend')
saveas(gca, '../figs/eq_rect_win_mag', 'epsc')

figure, hold on, box on
plot(w/pi, unwrap(angle(H)), 'LineWidth', 2, 'displayname', 'H(z)')
plot(w/pi, unwrap(angle(Hwin)), 'LineWidth', 2, 'displayname', 'H_{win}(z)')
xlabel('\omega/\pi', 'FontSize', 12)
ylabel('Phase (rad)', 'FontSize', 12)
legend('-dynamiclegend')
saveas(gca, '../figs/eq_rect_win_phase', 'epsc')

%% c) Least-squares method
% Calculate matrix Q
% This assumes even symmetry
N = 100;
w = linspace(-pi, pi).';
d = freqz(1, c, w);

Q = zeros(N, M+1); % initialized
n = 0:M; % time vector
for i = 1:N % for every frequency wi
    Q(i, :) = exp(-1j*w(i)*n);
end

% Least-squares filter
```

```
hls = pinv(Q)*d;

BER = simple_channel(hls)

% Plot results
[H, w] = freqz(1, c);
Hls = freqz(hls, 1, w);

figure, hold on, box on
plot(w/pi, abs(H).^2, 'LineWidth', 2, 'displayname', 'H(z)')
plot(w/pi, abs(Hls).^2, 'LineWidth', 2, 'displayname', 'H_{win}(z)')
xlabel('\omega/\pi', 'FontSize', 12)
ylabel('Magnitude', 'FontSize', 12)
legend('-dynamiclegend')
saveas(gca, '../figs/eq_ls_mag', 'epsc')

figure, hold on, box on
plot(w/pi, unwrap(angle(H)), 'LineWidth', 2, 'displayname', 'H(z)')
plot(w/pi, unwrap(angle(Hls)), 'LineWidth', 2, 'displayname', 'H_{win}(z)')
xlabel('\omega/\pi', 'FontSize', 12)
ylabel('Phase (rad)', 'FontSize', 12)
legend('-dynamiclegend')
saveas(gca, '../figs/eq_ls_phase', 'epsc')

% d)
sigma2 = 0.001;
P = sigma2/(pi)*trapz(w, abs(Hls).^2)
% Note division by pi instead of 2pi since w is defined in the interval (0,
% pi)

% Another way to calculate P
P = sigma2*sum(abs(hls).^2) % from Parseval's identity
```