University of Brasília

Electrical Engineering Department

Topics in Biomedical Engineering

Exercise List 5

Authors:

Caio Luiz Candeias Flôres 190134283

Felipe Carneiro da Motta 200017616

João Pedro Daher Aranha 190109742

Brasília

September 23, 2022

Contents

1 Exercises

1.1 Exercise 4.6:

Find the spectrum (magnitude and phase) of the system represented by the Z-transform:

$$H(z) = \frac{0.42 \cdot 10^{-3} + 1.25 \cdot 10^{-3} z^{-1} + 1.25 \cdot 10^{-3} z^{-2} - 0.42 \cdot 10^{-3} z^{-3}}{1 - 2.686 z^{-1} + 2.42 z^{-2} - 0.73 z^{-3}}$$

The MATLAB's code:

```
%% data
close all; clear all; clc;

N = 512; % number of points
fs = 1e3; % sampling frequency
Ts = 1/fs;

%% transfer function
b = [0.42e-3 1.25e-3 1.25e-3 1.25e-3 -0.42e-3];
a = [1 -2.686 2.42 -0.73];
H = fft(b, N) ./ fft(a, N);

H_mag = 20*log10(abs(H)); % mag in dB
H_phase = unwrap(phase(H))*180/pi; % phase in deg

f = (0:N-1)*fs/N; % frequency axis
fnyq = 1:N/2;

%% plots

% magnitude
subplot(3,1,1);
semilogx(f(fnyq), H_mag(fnyq), 'b', 'linewidth', 1.3);
title ("Frequency Response - Magnitude");
ylabel ("Magnitude (dB)");
xlabel ("Frequency (Hz)");
xlim([0 4e2]);
```

```
ylim ([ -80 10]) ; % yscale from -80 to 10 dB

% phase
subplot(3,1,2);
semilogx(f(fnyq), H_phase(fnyq), 'b', 'linewidth', 1.3);
title ("Frequency Response - Phase");
ylabel ("Phase (degrees -    )");
xlabel ("Frequency (Hz)");
ylim ([ -80 10]) ; % yscale from -80 to 10 dB

% H[z] step response

t = (0:N-1)*Ts;
s = ones(1, N);
sresp = filter(b, a, s); % insert the signal to the H[z] filter
subplot(3,1,3);
plot(t, sresp, 'b', 'linewidth', 1.3);
title ("Step Response");
ylabel ("Amplitude");
xlabel ("Time (s)");
xlim([0 0.5]);
ylim([0 1.3])

saveas(gcf, "ex4_6.png"); % save image
```

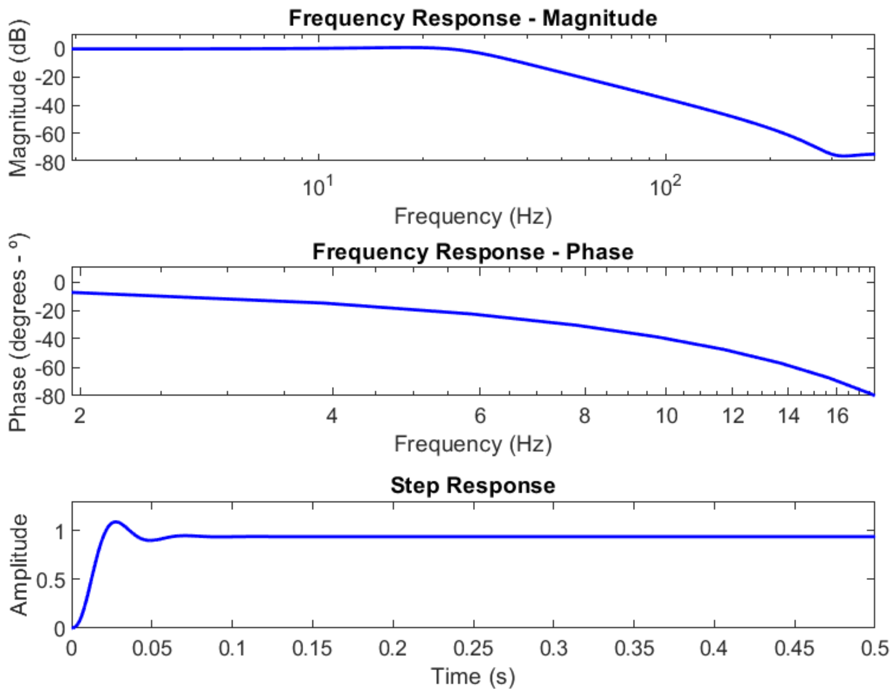

Figure 1: Spectra (Magnitude and Phase) and Step Response.

In this Figure 1, the magnitude and the phase spectra were calculated. Furthermore, the step response was also plotted to get the analysis richer. According to the the Magnitude Response, it is possible to visualize the low pass behavior of H[z].

1.2 Exercise 4.9:

Find the magnitude spectrum of an FIR filter with a weighting function of b = [.2 .2 .2 .2 .2] in two ways: (a) apply the fft with padding to the filter coefficients as in Problem 4.7 and plot the magnitude spectrum of the result; (b) pass white noise through the filter using conv and plot the magnitude spectra of the output. Since white noise has, theoretically, a flat spectrum, the spectrum of the filter's output to white noise should be the spectrum of the filter. In the second method, use a 20,000- point noise array; that is, y = conv(b, randn(20000,1)). Use the Welch averaging method described in Section 4.3 to smooth the spectrum. For the Welch method, use a segment size of 128 points and a 50% segment overlap. Since the pwelch routine produces the power spectrum, you need to take the square root to get the magnitude spectrum for comparison with the FT method. The two methods use different scaling, so the vertical axes are slightly different. Assume a sampling frequency of 200 Hz for plotting the spectra.

The MATLAB's code:

```
clear all; close all; clc;

%%First method:
N=256;
fs=200;
f1=(0:1/fs:N-1/fs);          %freq vector 1
b=[.2 .2 .2 .2 .2];          %given coefficients of b
B_mag=abs(fft(b,length(f1)));     %Magnitude spectrum
figure(1);
subplot(2,1,1);
plot(f1(1:end/2),B_mag(1:end/2),'b');
xlabel("Frequency (Hz)");
ylabel("Magnitude");
title("First Method");

%%Second method:
y = conv(b, randn(20000,1));
[PS,f] = pwelch(y,[] ,[] ,[],fs); % Apply the Welch
    method
subplot(2,1,2);
plot(f,sqrt(PS),'b')
xlabel("Frequency (Hz)");
ylabel("Magnitude");
title("Second Method");

saveas(gcf, "ex4_9.png")
```

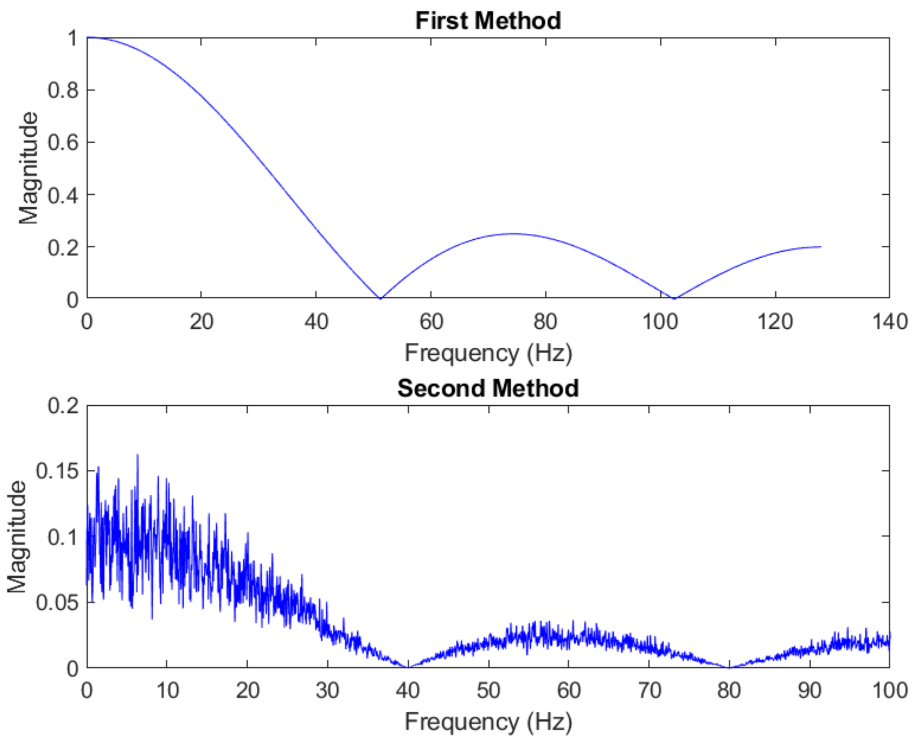

Figure 2: Comparison of the two methods - Exercise 4.9

The exercise requested the magnitude spectrum of a FIR filter and, as we can see, the two methods used, the FFT and the Welch, generated graphs that have different scales, but we can notice that one resembles the other.

1.3 Exercise 4.12:

This problem compares the Blackman–Harris and Hamming windows in the frequency domain. Write a program to construct the coefficients of a lowpass rectangular window filter with a cutoff frequency of 200 Hz. Make the filter length L = 33 coefficients. Assume fs = 1000 Hz. Generate and apply the appropriate length Hamming (Equation 4.24) and Blackman–Harris (Equation 4.25) windows to the filter coefficients, b[k]. Construct the two windows from the basic equations as in Example 4.5. Find and plot the spectra of the filter without a window and with the two windows. Plot the spectra superimposed to aid comparison and pad to 256 samples. As always, do not plot redundant points. Note that the Hamming window produces a somewhat steeper attenuation, but has just the slightest ripple before the cutoff.

The MATLAB's code:

```matlab
close all; clear all; clc;
fs=1000;
cutoff_frequency=200/(fs/2);

padding=256;
N=33;
b1=fir1(N,cutoff_frequency,'low',rectwin(N+1));

w_hamming=0.5-0.46*cos(2*pi*(0:N)/N);
b2=b1.*w_hamming;

a0 = 0.35875;
a1 = 0.48829;
a2 = 0.14128;
a3 = 0.01168;

n=-floor(N/2):floor(N/2)+1;
w_blackman=a0+a1*cos((2*pi/N)*n)+a2*cos((2*pi/N)*2*n)+
    a3*((2*pi/N)*3*n);
b3=b1.*w_blackman;

figure(1);
[h1,f]=freqz(b1,1,padding,fs);
[h2]=freqz(b2,1,padding,fs);
[h3]=freqz(b3,1,padding,fs);

plot(f,abs(h1),'DisplayName','Rectangular window');
hold on;
plot(f,abs(h2),'r','DisplayName','Hamming');
plot(f,abs(h3),'g','DisplayName','Blackman-Harris');
xlabel('Frequency');
ylabel('Frequency response');
legend;
```

In this exercise, we compared the behaviour of a rectangular window (Figure 3), a Hamming window and a Blackman-Harris window. Plotting the spectra of the three, it was comprovated indeed that the Hamming window produces a steeper attenuation and has a slight ripple just before the cutoff.

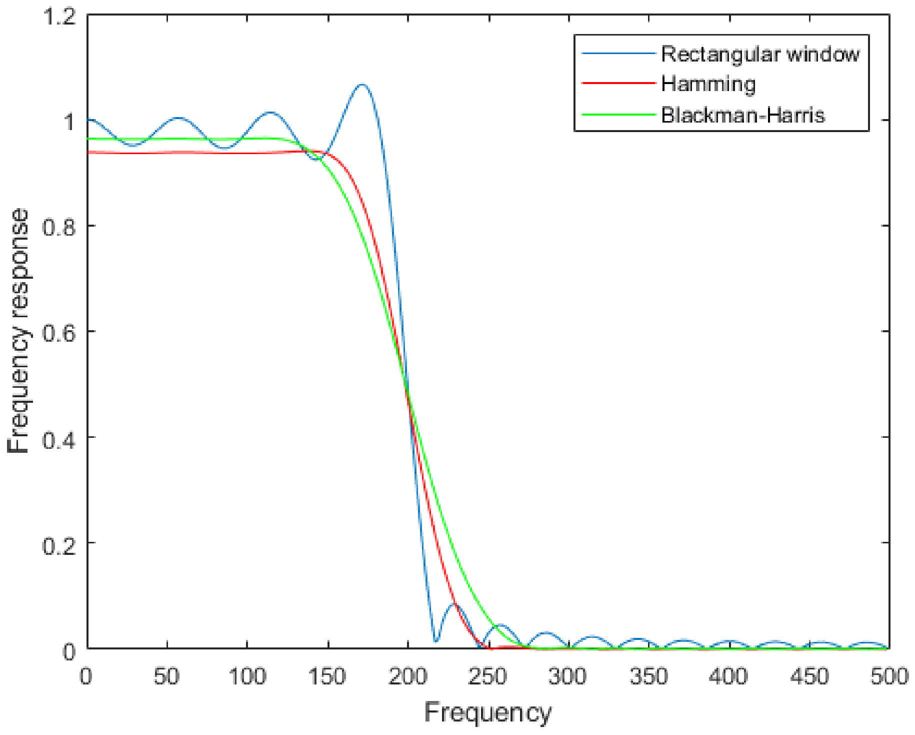

Figure 3: Output of the code of exercise 4.12

1.4 Exercise 4.13:

This problem compares the Blackman–Harris and Hamming windows in the time domain. Construct the rectangular window filter used in Problem 4.12, but make L = 129 coefficients and the cutoff frequency 100 Hz. As in Problem 4.12, generate and apply the appropriate length Hamming (Equation 4.24) and Blackman–Harris (Equation 4.25) windows to the filter coefficients, b[k]. Determine the impulse response of these two filters using the MATLAB filter routine with an impulse input. The impulse input should consist of a 1 followed by 255 zeros. Plot the impulse responses superimposed (in different colors) and you will find they look nearly identical. Limit the time axis to 0.1 s to better observe the responses. Subtract the impulse response of the Blackman–Harris-windowed filter from the impulse response of the Hamming-windowed filter and plot the difference. Note that, while there is a difference, it is several orders of magnitude less than the impulse responses.

The MATLAB's code:

```
%% data
clear all; close all; clc;

fs = 1e3; % sampling frequency
N = 256; % number of points | zpadding = N-1
zp = N-1;
cutoff = 100; % cutoff freq
fc = cutoff/fs; % normalized fc by fs
f = (0:N-1)*fs/N; % frequency axis
L = 129; % size of the filters

%% windowed filters
k = -floor(L/2):-1; % window vector
b = sin(2*pi*fc*k)./(pi*k); % construct negative b[k]
b = [b 2*fc , fliplr(b)]; % flip for positive b[k]

% Hamming
n_ham = 1:L; % hamming window vector
ham = 0.54 - 0.46* cos ((2* pi*n_ham)/L); % hamming
    window eq
b_ham = b .* ham; % apply hamming window

H_ham = fft(b_ham, N); % spectrum of hamming window
H_ham_mag = abs(H_ham); % magnitude of the spectrum of
    hamming window

% Blackman-Harris
n_black = -floor(L/2):floor(L/2); % bharris window
    vector
a_window = [0.35875 0.48829 0.14128 0.01168]; % bharris
     window coeffs
black = a_window (1) + a_window (2)*cos ((2* pi*n_black
    )/L) + ...
        a_window (3)*cos (2*(2* pi*n_black)/L) + ...
        a_window (4)*cos (3*(2* pi*n_black)/L); %
            bharris window equation

```

```
b_black = b .* black; % apply bharris window

H_black = fft(b_black , N); % spectrum of bharris
    window
H_black_mag = abs(H_black); % magnitude of the spectrum
     of bharris window

% impulse reponse
impulse = [1 zeros(1,zp)]; % impulse vector (zero
    padding = 255)
y_ham = filter(b_ham , 1, impulse); % impulse response
    in the hamming window
y_black = filter(b_black , 1, impulse); % impulse
    response in the bharris window
t = (0:N-1)/fs; % time axis

%% plots
subplot(2,1,1);
plot(t, y_ham, 'r', 'linewidth', 1.1, 'DisplayName', '
    Hamming'); % impulse response hamming window filter
hold on;
plot(t, y_black, 'k', 'linewidth', 1.1, 'DisplayName',
    'Blackman-Harris'); % impulse response bharris
    window filter
title ('Impulse responses in different windowed filters
    ');
legend();
xlabel ('Time (s)');
ylabel ('Amplitude');
xlim ([0 0.1]); % time axis limited in 0.1s

subplot(2,1,2);
diff = y_black - y_ham;
plot(t, diff, 'b', 'linewidth', 1.3); % diff of impulse
     responses
title ("Difference between Blackman-Harris and Hamming
    ");
xlabel ("Time (s)");
ylabel ("Amplitude");
xlim ([0 0.1]);

```

```
saveas(gcf, "ex4_13.png"); % save image
```

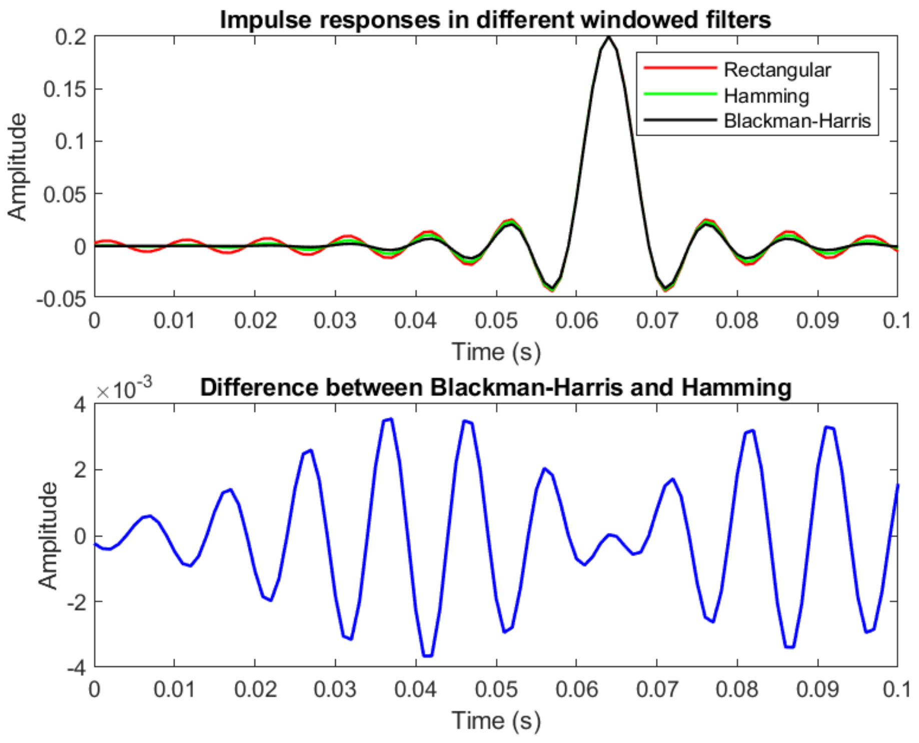

Figure 4

In this exercise, three different windows were compared: Rectangular, Hamming and Blackman-Harris. A low pass filter with cutoff frequency 100 and order 29 was also implemented (used in the Problem 4.12). In sequence, each window was applied to the respective coefficients b[k] and the impulse response was created, with the function filter() and considering the coeficient a[k] as 1, and plotted.

As it is possible to visualize in the Figure 4, the impulse responses are very close to each other and they almost overlap the color to three windows to turn in a unique color. The difference between the impulse reponse of the Blackman-Harris window and the Hamming window was also plotted, and the result confirmed the slightly difference between the impulse responses. This difference is in the order of 10^-3.

1.5 Exercise 4.14:

Load file ECG_9.mat, which contains 9 s of ECG data in variable x. These ECG data have been sampled at 250 Hz. The data have a low-frequency signal superimposed over the ECG signal, possibly due to respiration artifact. Filter the data with a highpass filter having 65 coefficients and a cutoff frequency of 8 Hz. Use the window of your choice to taper the filter weights. Plot the magnitude spectrum of the filter to confirm the correct type and cutoff frequency. Also plot the filtered and unfiltered ECG data. [Hint: You can modify a section of the code in Example 4.4 to generate the highpass filter.]

The MATLAB's code:

```
clear all; close all; clc;
load ECG_9.mat
T = 9; % Number of samples for plotting
fs = 250; % Sampling frequency
N=fs*T;
f1 = (1:N)*fs/N; % Frequency vector for plotting
t = (0:N-1)*fs/N; %Time vector for plotting
fc = 8/fs; % Cutoff frequency (normalized to fs)
L = 65; % Requested filter length

ffiltro=[0,8/(fs/2),8/(fs/2),1];
G=[0,0,1,1];
b=fir2(L,ffiltro,G);

[H,f] = freqz(b,1,fs);   %The impulse response
Xh = filter(b,1,x);             %

 % Plot magnitude spectrum
figure(1);
subplot(3,1,1)
plot(f,abs(H),'b','linewidth',1.2);
title('Magnitude spectrum');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

 %Plot unfiltered signal
```

```
subplot(3,1,2);
plot(t,abs(x),'b','linewidth',1.2);
title('Unfiltered signal');
xlabel('Time (s)');
ylabel('Amplitude');

%Plot filtered signal
subplot(3,1,3);
plot(t,abs(Xh),'b','linewidth',1.2);
title('Filtered signal');
xlabel('Time (s)');
ylabel('Amplitude');

saveas(gcf, "ex4_14.png");
```

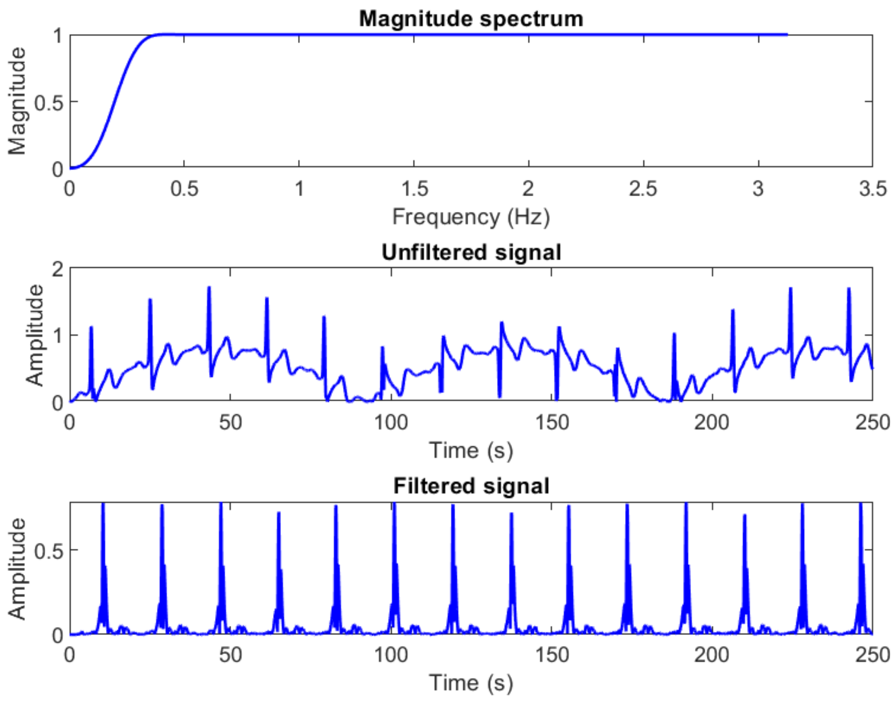

Figure 5: Graph 1 - Exercise 4.14

The exercise wanted us to filter the given signal with a highpass filter with a cutoff frequency of 8 Hz to remove the unwanted sinusoidal wave clearly seen in the unfiltered signal figure. As we can see from the figures, filtering the signal was successful. The filtered signal will be easier to analyze.

1.6 Exercise 4.15:

ECG data are often used to determine the heart rate by measuring the time interval between the peaks that occur in each cycle known as the R wave. To determine the position of this peak accurately, ECG data are first prefiltered with a bandpass filter that enhances the R wave and the two small negative peaks on either side, the Q and S wave. The three waves together are termed the QRS complex. Load file ECG_noise.mat, which contains 10 s of noisy ECG data in variable ecg. Filter the data with a 65-coefficient FIR bandpass filter to best enhance the R-wave peaks. Use the tapering window of your choice. (Note that it is not possible to eliminate all the noise, just improve the identification of the peaks.) Determine the low and high cutoff frequencies empirically, but they will be in the range of 4–24 Hz. The sampling frequency is 250 Hz.

The MATLAB's code:

```
clc;close all;clear all;
load ECG_noise.mat;
fs=250; %sample frequency
N=length(ecg); %number of points
t=(1:N)/fs; %time vector

%plot settings
subplot(2,1,1);
plot(t,ecg);
title('Unfiltered ECG');
xlabel('Time (s)');
ylabel('Amplitude');

%Signal filtering
N_coefficients=65; %number of FIR coefficients
frequencies = [7 16]/(fs/2); %frequencies normalised by
    sample frenquency
b=fir1(N_coefficients,frequencies,'bandpass',hamming(
    N_coefficients+1)); %FIR definition
ecg_filtered=filter(b,1,ecg); %apply filter to signal
```

```

%plot settings
subplot(2,1,2);
plot(t,ecg_filtered);
title('Filtered ECG');
xlabel('Time (s)');
ylabel('Amplitude');


sgtitle('ECG data processing');
```

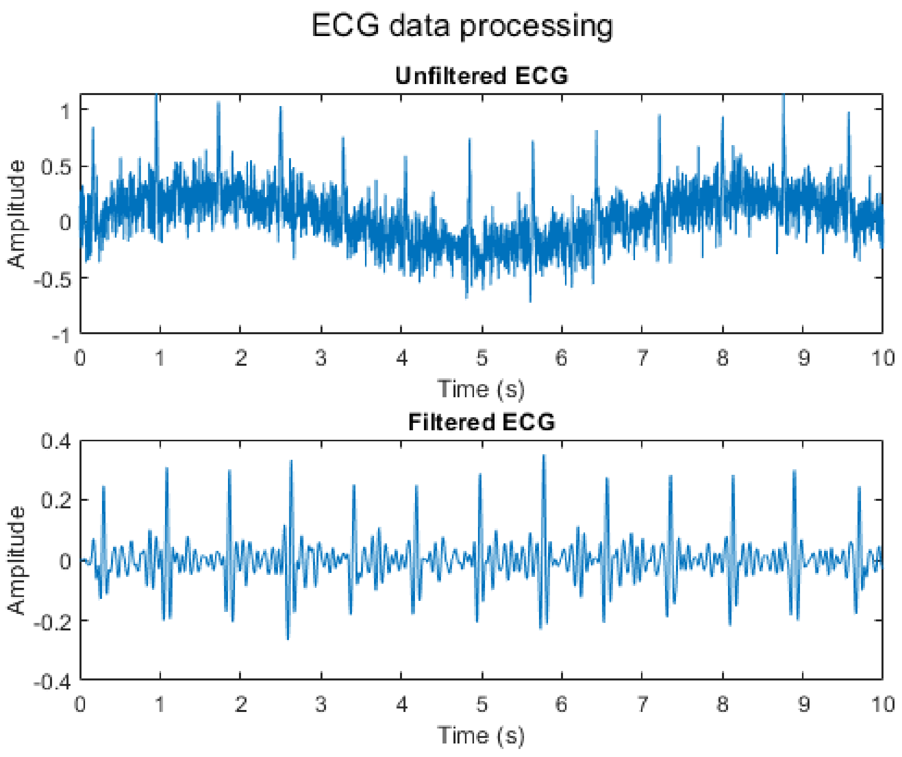

Figure 6: Comparison of filtered and unifiltered ECG signals

Applying the FIR filter with 66 coefficients (order 65) and bandpass between 7-16Hz, it was possible to greatly refine the ECG data and obtain the R-peaks. The default options of filter function apply a hamming window with size ORDER+1, but it was decided to show this parameter explicitly.

1.7 Exercise 4.19:

The file ECG_60HZ_data.mat contains an ECG signal in variable x that was sampled at fs = 250 Hz and has been corrupted by 60-Hz noise. The 60-Hz noise is at a high frequency compared to the ECG signal, so it appears as a thick line superimposed on the signal. Construct a 127-coefficient FIR rectangular bandstop filter with a center frequency of 60 Hz and a bandwidth of 10 Hz (i.e., ±5 Hz) and apply it to the noisy signal. Implement the filter using either filter or conv with the 'same' option, but note the time shift if you use the former. Plot the signal before and after filtering and also plot the filter's magnitude spectrum to ensure the filter spectrum is what you want. [Hint: You can easily modify a portion of the code in Example 4.7 to implement the bandstop filter.]

The MATLAB's code:

```
%% dados
clear all; close all; clc;

load ECG_60Hz_data.mat; % load the file
fs = 250; % Hz
Ts = 1/fs;
N = length(x);
bandwidth = 5;
center = 60;
order = 27;

t = (0:N-1)*Ts; % time axis

f = (0:N-1)*fs/N; % frequency axis
fnyq = (1:N/2); % Nyquist frequency

%% filtering process

fc1 = (center - bandwidth)/(fs/2); % normalized fc1 -¿
    fc1/(fs/2)
fc2 = (center + bandwidth)/(fs/2); % normalized fc2 -¿
    fc2/(fs/2)
fc1_orig = fc1/2; % only fc1/fs
fc2_orig = fc2/2; % only fc2/fs
```

```matlab

% rectangular stopband filter
k = -floor(order/2):-1;
b1_fc1 = sin(2*pi*fc1_orig*k)./(pi*k); % first fc - fc1
b1_fc2 = sin(2*pi*fc2_orig*k)./(pi*k); % second fc -
    fc2
b1 = (b1_fc1 - b1_fc2); % construct negative b[k]
b1 = [b1 1 - 2*(fc2_orig - fc1_orig), fliplr(b1)]; %
    flip for positive b[k]
%b1 = b1 .* hamming(order)'; % hamming windowed coeffs
Hb1_m = abs(fft(b1, N)); % Filter Magnitude Spectrum


% fir rectangular bandstop
figure(1);
subplot(2,1,1);
plot(f(fnyq), Hb1_m(fnyq), 'b', 'linewidth', 1.3);
title ('FIR filter - Rectangular Bandstop');
xlabel ('Time (s)');
ylabel ('Amplitude');
xlim([0 120]);

% fir1() function
subplot(2,1,2)
b2 = fir1(order, [fc1 fc2], 'stop'); % stopband filter
Hb2 = fft(b2, N);
Hb2_m = abs(Hb2);
plot(f(fnyq), Hb2_m(fnyq), 'b', 'linewidth', 1.3);
title ('FIR filter - fir1() function');
xlabel ('Time (s)');
ylabel ('Amplitude');
xlim([0 120]);
saveas(gcf, 'fig1_ex4_19.png')

% signal before filtering
figure(2);
subplot(3,1,1)
plot(t, x, 'b', 'linewidth', 1.3);
title ('ECG signal before filtering');
xlabel ('Time (s)');
ylabel ('Amplitude');
```

```


% signal after filtering
subplot(3,1,2);
x_filt_b1 = filter(b1, 1, x); % rect window
plot(t, x_filt_b1, 'b', 'linewidth', 1.3);
title ('ECG signal filtered - Rectangular Bandstop');
xlabel ('Time (s)');
ylabel ('Amplitude');

subplot(3,1,3);
x_filt_b2 = filtfilt(b2, 1, x); % fir1() function
plot(t, x_filt_b2, 'b', 'linewidth', 1.3);
title ('ECG signal filtered - fir() function');
xlabel ('Time (s)');
ylabel ('Amplitude');

saveas(gcf, 'fig2_ex4_19.png')
```

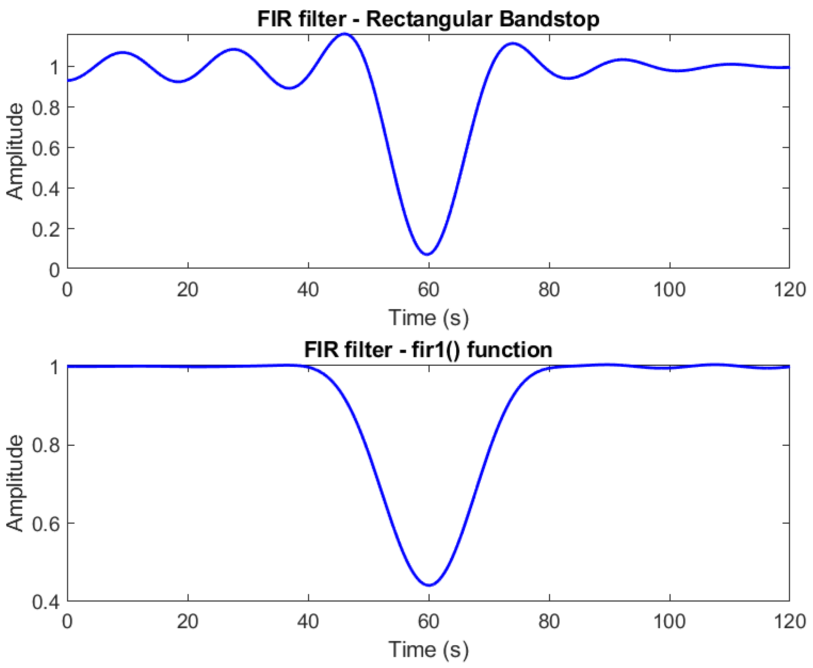

Figure 7: 127-coefficient FIR rectangular bandstop filter.

Initially, the Figure 7 represents the 127-coefficient FIR rectangular bandstop filter with a center frequency of 60 Hz and a bandwidth of 10 Hz. The first subplot was implemented using the analytics equations, and the second subplot created this filter using the function fir1(). As the graph shows, both methods were efficient and generated the filter required in this question.

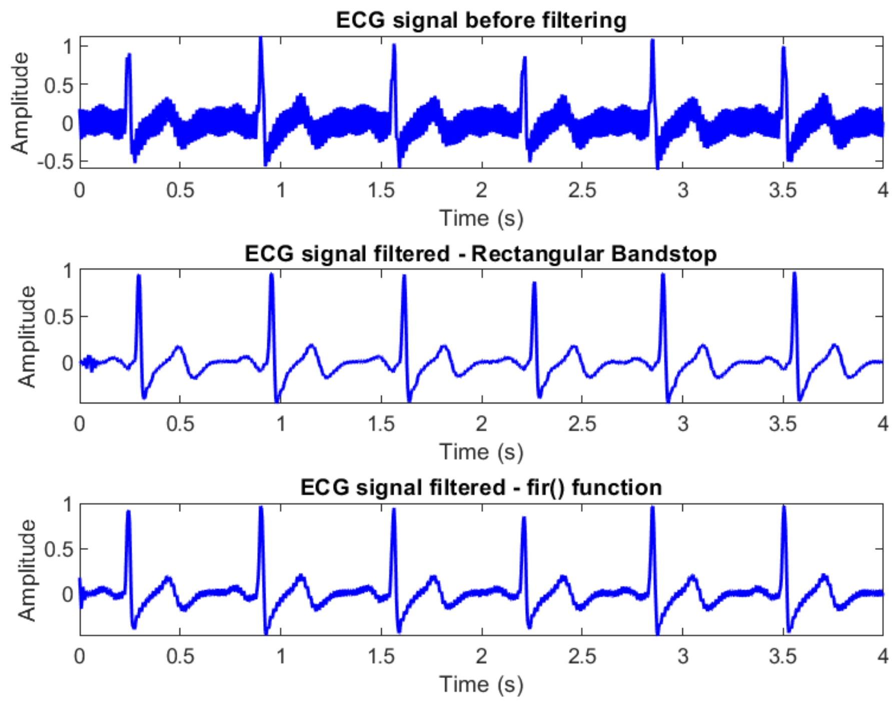

Figure 8

In sequence, the ECG signal with the 60 Hz noise was filtered with the two versions of the required filter implemented and showed above. The original noisy signal was plotted with the filtered signals to evidence the filtering process.

It is notorious that the filtered process was successfully applied and the 60 Hz was eliminated to the ECG signal. The filter developed in both form acted correctly, but the hands-on filtered make the ECG signal cleaner and with less noise.

1.8 Exercise 4.20:

This problem compares causal and noncausal FIR filter implementation. Generate the filter coefficients of a 65th-order rectangular window filter with a cutoff frequency of 40 Hz. Apply a Blackman–Harris window to the filter. Then apply the filter to the noisy sawtooth wave, x, in file sawth.mat. This waveform was sampled at fs = 1000 Hz. Implement the filter in two ways. Use the causal filter routine and noncausal conv with the 'same' option. (Without this option, conv is like filter except that it produces extra points.) Plot the two waveforms along with the original, superimposed for comparison. Note the obvious differences. Also note that while the filter removes much of the noise, it also reduces the sharpness of the transitions.

The MATLAB's code:

```matlab
close all
clear all
load sawth.mat

fs = 1000; % Sampling frequency
N=256;
f1 = (1:N)*fs/N; % Frequency vector for plotting
t = (0:N-1)*fs/N; %Time vector for plotting
fc = 40/fs; % Cutoff frequency (normalized to fs)
L = 65; % Requested filter length

ffiltro=[0,2*fc,2*fc,1];
G=[0,0,1,1];
b=fir2(L,ffiltro,G);
bblack=b.*blackmanharris(1);

y=filter(bblack,1,x);
y1=conv(x,bblack,'same');

figure;
plot(t,x); hold on; plot(t,y,'k'); hold on; plot(t,y1,'
    r');
legend('Default Signal','Causal','Noncasual');
```

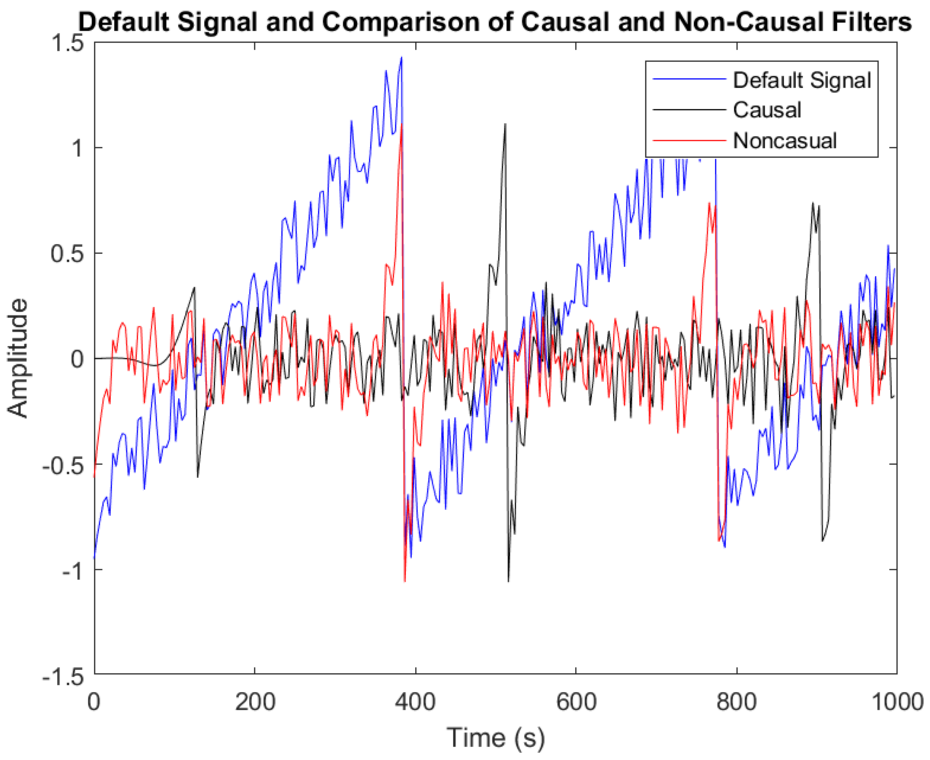

Figure 9: Plot of the comparison of causal and non-causal filters and original signal.

The exercise requested us to compare causal and noncausal FIR filters. We applied the filters on a sawtooth wave. As we can see, in the figure, with the superimposed signals, the difference between the signals with the causal and the noncausal filters is noticeable. We can see a difference on where the transitions are.

1.9 Exercise 4.21:

Given the advantage of a non-causal filter with regard to the time shift shown in Problem 4.20, why not use non-causal filters routinely? This problem shows the downsides of non-causal FIR filtering. Generate the filter coefficients of a 33rd-order rectangular window filter with a cutoff frequency of 100 Hz, assuming fs = 1 kHz. Use a Blackman–Harris window on the truncated filter coefficients. Generate an impulse function consisting of a 1 followed by 255 zeros. Now apply the filter to the impulse function in two ways: causally using the MATLAB filter routine, and noncausally using the conv routine with the 'same' option. (The latter generates a non-causal filter since it performs symmetrical convolution.) Plot the two time responses separately, limiting the x axis to 0–0.05 s to better visualize the responses. Then take the Fourier transform of each out put and plot the magnitude and phase. (For a change, you can plot the phase in radians.). Use the MATLAB unwrap routine on the phase data before plotting. Note the strange spectrum produced by the non-causal filter (i.e., conv with the ‘same’ option). This is because the non-causal filter has truncated the initial portion of the impulse response. To confirm this, rerun the program using an impulse that is delayed by 10 sample intervals (i.e., impulse = [zeros(1,10) 1 zeros(1,245)];). Note that the magnitude spectra of the two filters are now the same, although the phase curves are different due to the different delays produced by the two filters. The phase spectrum of the non=causal filter shows reduced phase shift with frequency as would be expected. This problem demonstrates that non-causal filters can create artifact with the initial portion of an input signal because of the way it compensates for the time shift of causal filters.

The MATLAB’s code:

```
clearvars;
N=256;
fs=1e3;
t=(0:N-1)/fs; %time vector

order=33; %order of filter
cutoff_freq=100/(fs/2); %normalised cut off frequency
b=fir1(order,cutoff_freq,blackmanharris(order+1)); %FIR
    definition

x=[1 zeros(1,255)]; %impulse function

causal=filter(b,1,x);

non_causal=conv(x,b,'same');

%Plot settings
figure(1);
subplot(2,1,1);
plot(t,causal);
title('Causal filter');
xlabel('Time (s)');
```

```matlab
ylabel('Amplitude');
xlim([0 0.05]);

subplot(2,1,2);
plot(t,non_causal);
title('Non causal filter');
xlabel('Time (s)');
ylabel('Amplitude');
xlim([0 0.05]);

%% Frequency domain

f=(-N/2:N/2-1)*fs/N;

causal_f=fft(causal);
causal_f=fftshift(causal_f);

non_causal_f=fft(non_causal);
non_causal_f=fftshift(non_causal_f);

%Plot settings

figure(2);
subplot(2,2,1);
plot(f,abs(causal_f));
title('Causal filter');
xlabel('Frequency (Hz)');
ylabel('Amplitude');

subplot(2,2,2);
plot(f,unwrap(angle(causal_f)));
title('Causal filter');
xlabel('Frequency (Hz)');
ylabel('Angle');

subplot(2,2,3);
plot(f,abs(non_causal_f));
title('Non causal filter');
xlabel('Frequency (Hz)');
ylabel('Amplitude');

```

```
subplot(2,2,4);
plot(f,unwrap(angle(non_causal_f)));
title('Non causal filter');
xlabel('Frequency (Hz)');
ylabel('Angle');

%% Shifted
x=[zeros(1,10) 1 zeros(1,245)]; %impulse function

causal=filter(b,1,x);

non_causal=conv(x,b,'same');

causal_f=fft(causal);
causal_f=fftshift(causal_f);

non_causal_f=fft(non_causal);
non_causal_f=fftshift(non_causal_f);

%Plot settings
figure(3);
subplot(2,2,1);
plot(f,abs(causal_f));
title('Causal filter');
xlabel('Frequency (Hz)');
ylabel('Amplitude');

subplot(2,2,2);
plot(f,unwrap(angle(causal_f))); %phase spectrum in
    radians
title('Causal filter');
xlabel('Frequency (Hz)');
ylabel('Angle');

subplot(2,2,3);
plot(f,abs(non_causal_f));
title('Non causal filter');
xlabel('Frequency (Hz)');
ylabel('Amplitude');

subplot(2,2,4);
```

```
plot(f,unwrap(angle(non_causal_f)));  %phase spectrum
    in radians
title('Non causal filter');
xlabel('Frequency (Hz)');
ylabel('Angle');

sgtitle('Analysis of shifted input');
```

In Figure 10 it is possible to notice the expected delay of causal filters. The non causal filter already corrected this and is centered in the y-axis.

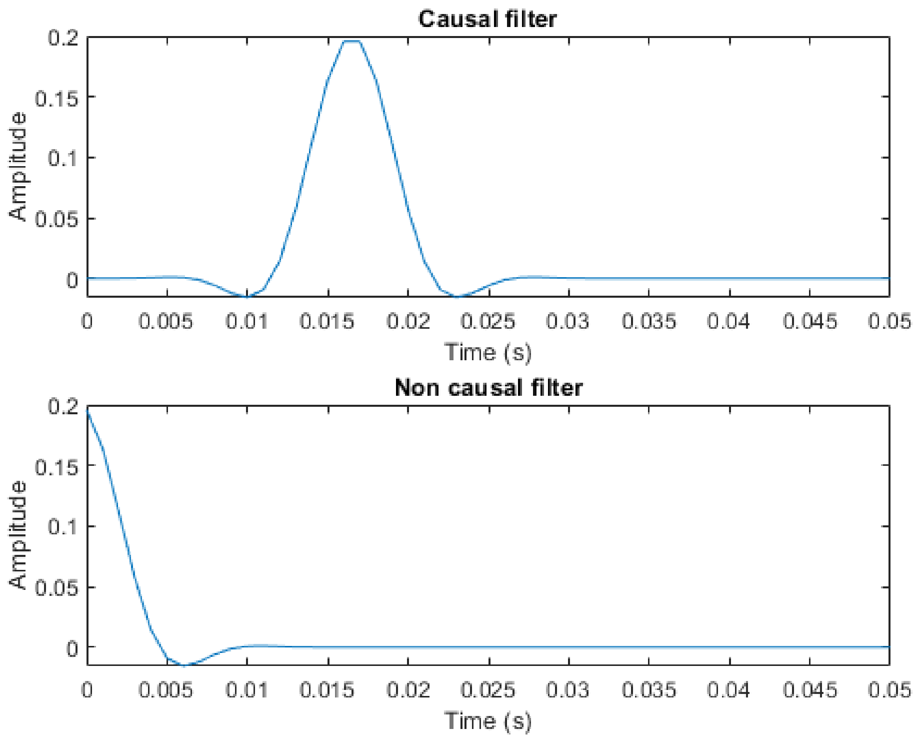

Figure 10: Result of the causal and non-causal filters on input

In Figure 11 the magnitude spectrum of the non causal filter is deformed. According to the exercise, this occurs because of the truncation of the initial portion of the impulse response.

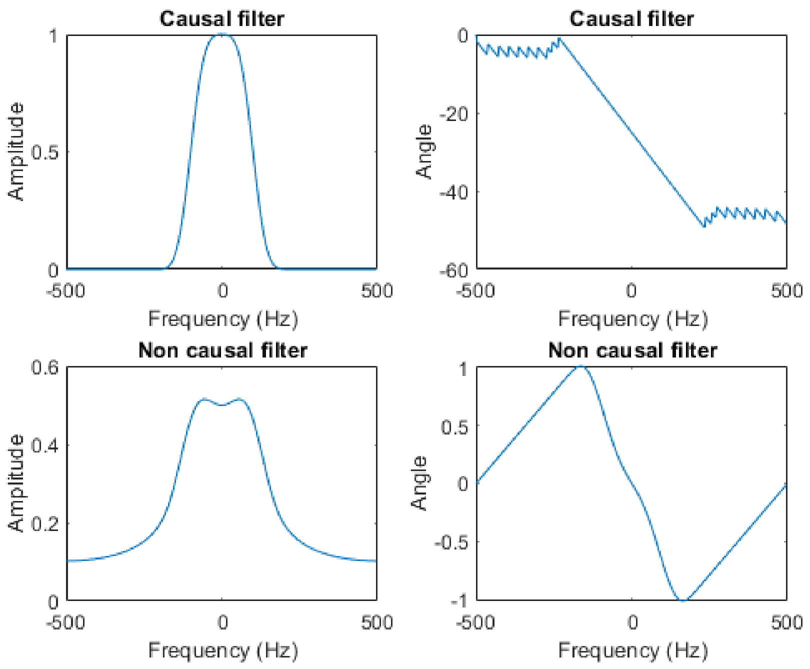

Figure 11: Graph 1 - Exercise 4.21

In Figure 12, both magnitude spectra are very similar. When the impulse was shifted, the truncation of did not occur anymore.

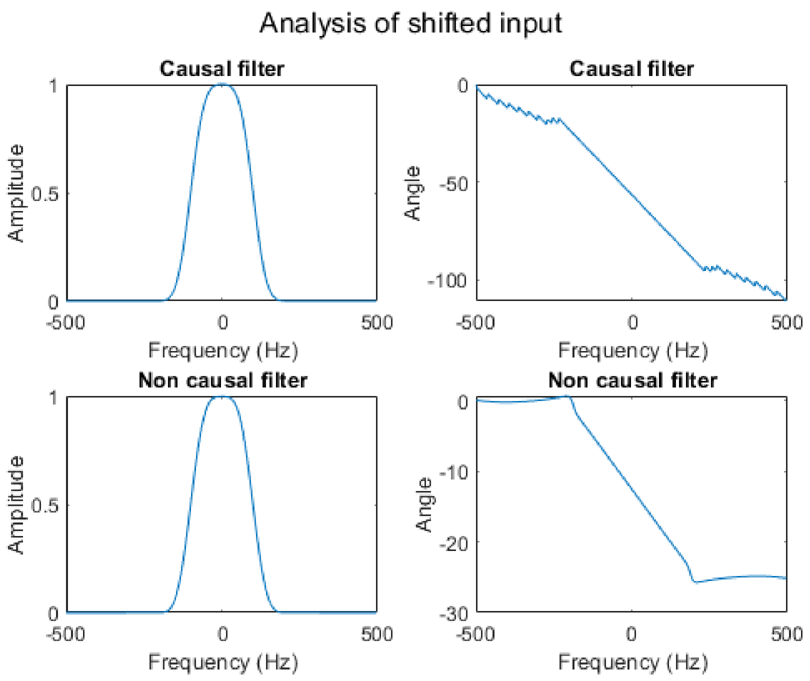

Figure 12: Graph 2 - Exercise 4.21

1.10 Exercise 4.24:

This problem demonstrates a comparison of a causal and a noncausal IIR filter implementation. Load file Resp_noise1.mat containing a noisy respiration signal in variable resp_noise1. Assume a sample frequency of 125 Hz. Construct a 14th-order Butterworth filter with a cutoff frequency of 0.15 fs/2. Filter that signal using both filter and filtfilt and plot the original and both filtered signals. Plot the signals offset on the same graph to allow for easy comparison. Also plot the noise-free signal found as resp in file Resp_noise1.mat below the other signals. Note how the original signal compares with the two filtered signals, in terms of the restoration of features in the original signal and the time shift.

The MATLAB's code:

```
%% data
clear all; close all; clc;
load Resp_noise1.mat; % load the file

```

```
fs = 125; % Hz
Ts = 1/fs;
resp = resp; % noise free-signal
resp_noise1 = resp_noise1; % noisy signal
N = length(resp);
order = 14;
fc_orig = 0.15*(fs/2);
fc = fc_orig / (fs/2);

t = (0:N-1)*Ts; % time axis
f = (0:N-1)*fs/N; % frequency axis
fnyq = 1:N/2; % Nyquist frequency

%% butterworth filter
[b, a] = butter(order, fc);

% causal IIR filter
x_filter = filter(b, a, resp_noise1); % filter() func

% noncausal IIR filter
x_filtfilt = filtfilt(b, a, resp_noise1); % filter with
    no delay

%% plots

% magnitude spectrum of butterworth filter
H_filt_mag = abs(fft(b, N) ./ fft(a, N));
figure(1);
plot(f(fnyq), H_filt_mag(fnyq), 'b', 'linewidth', 1.3);
title ('Butterworth Filter');
xlabel ('Frequency (Hz)');
ylabel ('Magnitude');
xlim([0 62]);
saveas(gcf, 'fig1_ex4_24.png')


% original signal - noisy signal
figure(2);
subplot(3,1,1);
plot(t, resp_noise1, 'b', 'linewidth', 1.3);
title ('Signal - Noisy Original');
```

```
xlabel ('Time (s)');
ylabel ('Amplitude');
xlim([0 5.5]);

% original signal - noise-free signal
figure(2);
subplot(3,1,2);
plot(t, resp, 'b', 'linewidth', 1.3);
title ('Signal - Noise-free');
xlabel ('Time (s)');
ylabel ('Amplitude');
xlim([0 5.5]);

% signal filtered with filter() function -- causal IIR filter
subplot(3,1,3);
plot(t, x_filter, 'b', 'linewidth', 1.3, 'DisplayName', 'filter()');
title ('Signal Filtered - filter() - causal');
xlabel ('Time (s)');
ylabel ('Amplitude');
hold on; % plot in the same graph

% signal filtered with filtfilt() function -- noncausal IIR filter
plot(t, x_filtfilt, 'r', 'linewidth', 1.3, 'DisplayName', 'filtfilt()');
title ('Signal Filtered - filtfilt() - noncausal');
xlabel ('Time (s)');
ylabel ('Amplitude');
xlim([0 5.5]);
legend();

saveas(gcf, 'fig2_ex4_24.png')
```

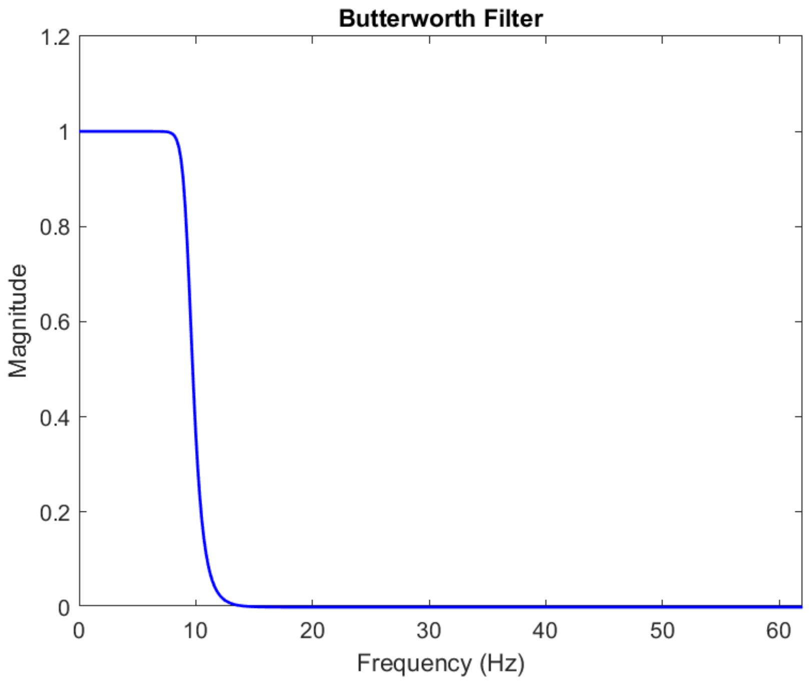

Figure 13: 14th-order Butterworth Filter - cutoff frequency of $0.15\frac{fs}{2}$.

The first step adopted was to construct the 14-th order Butterworth Filter with cutoff frequency of $0.15\frac{fs}{2}$. It was used the butter() function to define the a[k] and the b[k] coefficientes of this filter, and its magnitude spectrum of were implemented in the Figure 13.

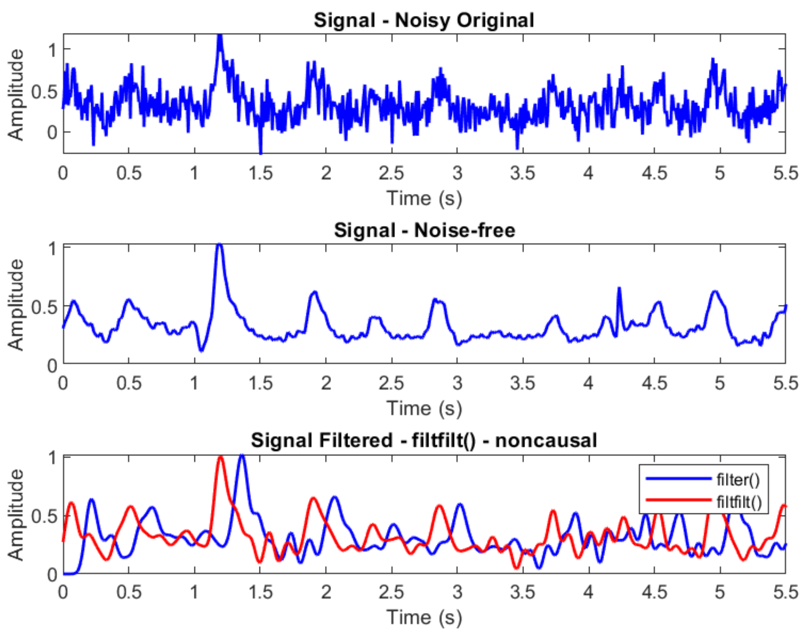

Figure 14: Comparison between the noisy signal of respiration and the filtered signals.

The second and last step was to use the noisy signal of respiration to compare the effect of a causal (filter()) and a non-causal (filtfilt()) filters.

As expected in the theory of filter design, the causal filter (filter()) generates the signal with no noise and introduces some delay. Differently, the non-causal filter (filtfilt()) filters the signal with no delay. The Figure 14 shows this delay effect and evidence the phase difference of the filters. The noisy signal and the free noise signal were also plotted to make this view clearer and to improve the analysis of the respective differences.

1.11 Exercise 4.25:

This problem is similar to Problem 4.21 in that it illustrates problems with noncausal filtering, except that an IIR filter is used and the routine filtfilt is used to implement the noncausal filter. Generate the filter coefficients of an eighth-order Butterworth filter with a cutoff frequency of 100 Hz assuming fs = 1 kHz. Generate an impulse function consisting of a 1 followed by 255 zeros. Now apply the filter to the impulse function using both the MATLAB filter routine and the filtfilt routine. The latter generates a noncausal filter. Plot the two time responses separately, limiting the x axis to 0–0.05 s to better visualize the responses. Then take the Fourier transform of each output and plot the magnitude and phase. Use the MATLAB unwrap routine on the phase data before plotting. Note the differences in the magnitude spectra. The noncausal filter (i.e., filtfilt) has ripple in the passband. Again, this is because the noncausal filter has truncated the initial portion of the impulse response. To confirm this, rerun the program using an impulse that is delayed by 20 sample intervals (i.e., impulse = [zeros(1,20) 1 zeros(1,235)];). Note that the magnitude spectra of the two filters are now the same. The phase spectrum of the noncausal filter shows reduced phase shift with frequency, as would be expected. However, even after changing the delay, the noncausal implementation has a small amount of ripple in the passband of the magnitude spectrum.

The MATLAB's code:

```
close all
clear all

fs = 1000; % Sampling frequency
fc = 100/fs; % Cutoff frequency (normalized to fs)

[b,a] = butter(8,2*fc);         %Getting the coefficients
imp=[1 zeros(1,255)];           %adding the impulse and the
    zeros
fil=filter(b,a,imp);            %Applying filter()
filfil=filtfilt(b,a,imp);       %Applying filtfilt()

t=(0:255)/fs;   %time vector for plotting

%Plotting
subplot(2,1,1)
plot(t,fil)
title('Response using filter function')
xlim([0 0.05])
xlabel('Time(s)')
ylabel('Amplitude')

subplot(2,1,2)
plot(t,filfil)
title('Response using filtfilt function')
```

```
xlim([0 0.05])
xlabel('Time(s)')
ylabel('Amplitude')

%Fourier
M_fil=(2/256)*abs(fft(fil));
 M_filfil=(2/256)*abs(fft(filfil));
 Phase_fil=unwrap(phase(fft(fil)))*(180/pi);
 Phase_filfil=unwrap(phase(fft(filfil)))*(180/pi);

f = (1:256)*fs/256; % Frequency vector for plotting
%Plotting Fourier
figure;
subplot(2,2,1)
plot(f(1:end/2),M_fil(1:end/2), 'k')
title('Magnitude')
xlabel('Frequency')
ylabel('Amplitude')

subplot(2,2,2)
plot(f,Phase_fil, 'r')
title('Phase')
xlabel('Frequency')
ylabel('Degrees')

subplot(2,2,3)
plot(f(1:end/2),M_filfil(1:end/2), 'k')
title('Magnitude')
xlabel('Frequency')
ylabel('Amplitude')


subplot(2,2,4)
plot(f,Phase_filfil, 'r')
title('Phase')
xlabel('Frequency')
ylabel('Degrees')
```

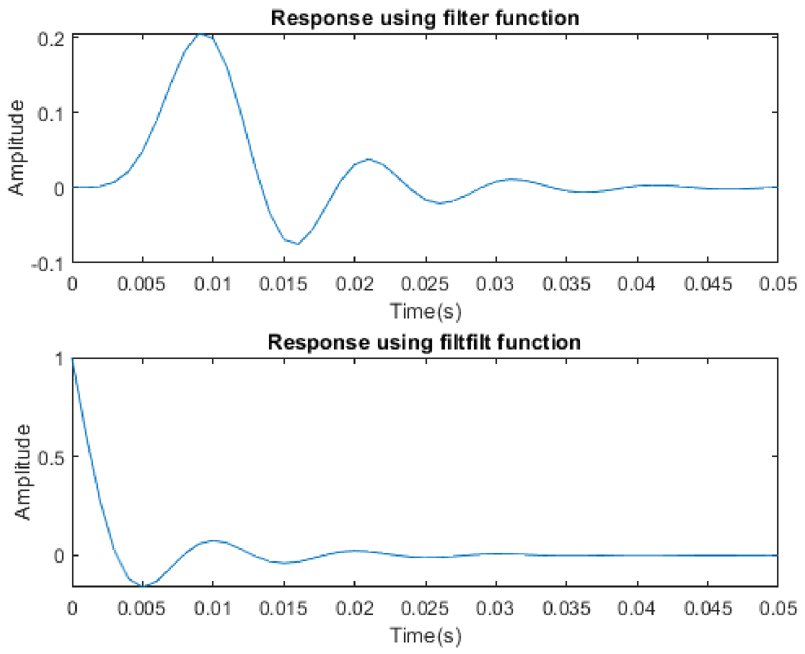

Figure 15: Graph 1 - Exercise 4.25

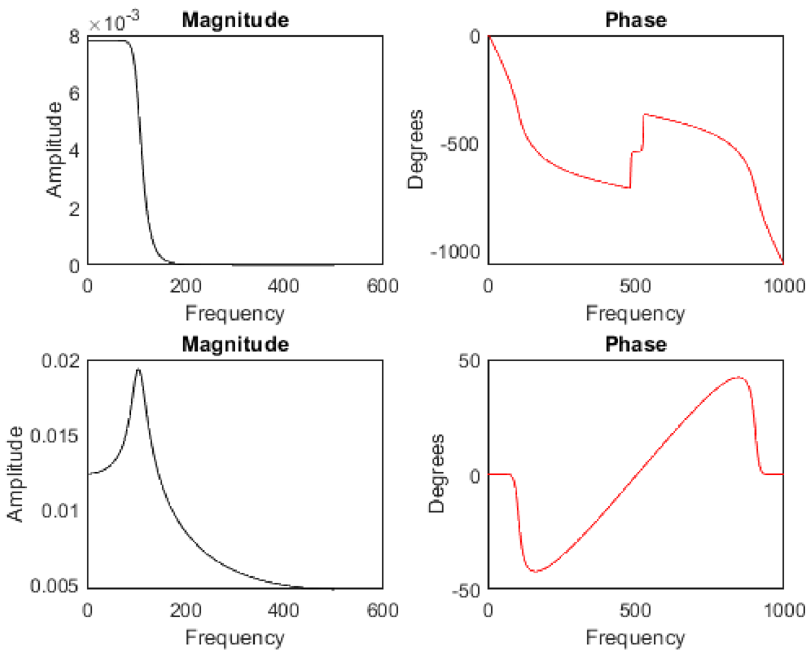

Figure 16: Graph 2 - Exercise 4.25

Now, rerunning the code but with the variable "imp" as [zeros(1,20) 1 zeros(1,235)], the magnitude and phase that is given form the Fourier transform is:

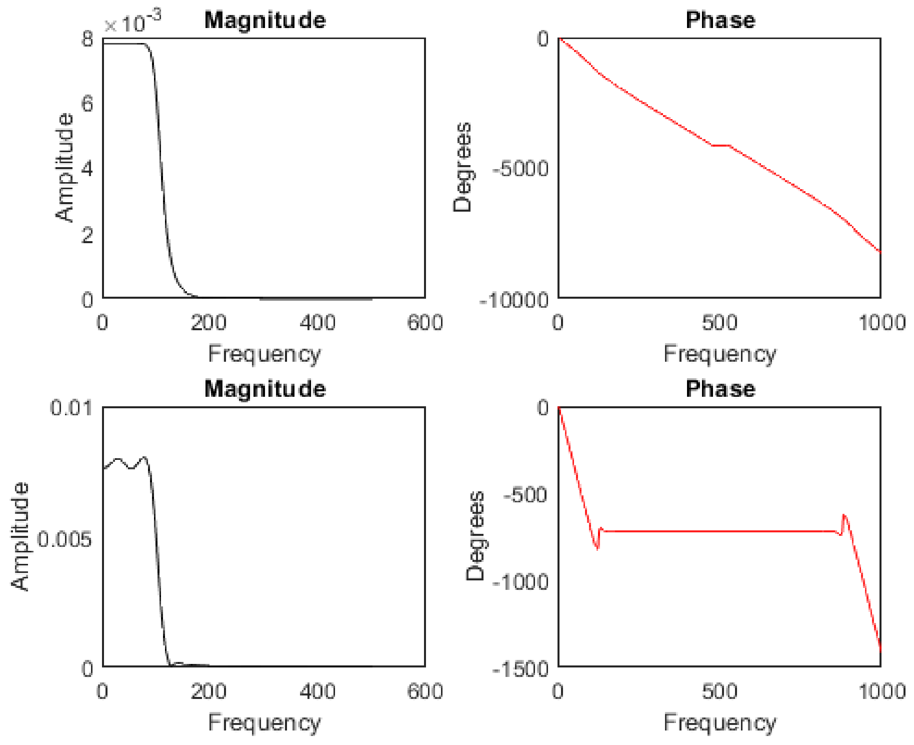

Figure 17: Graph 3 - Exercise 4.25

The exercise requested us to apply a Butterworth filter with a 100Hz cutoff frequency on an impulse, applying it with the filter() and filtfilt() functions. As we can see, the response seems to differ on time, as expected, since one filter is causal and the other is noncausal. However, the magnitude and phase spectres are things that differ considerably. But as the problem states, when we delay the impulse, the magnitude spectrum of both filters become very similar.

1.12 Exercise 4.27:

This problem compares FIR–IIR filters. Construct a 12th-order Butterworth highpass filter with a cutoff frequency of 80 Hz assuming fs = 300 Hz. Construct an FIR high-pass filter having the same cutoff frequency using Equation 4.26. Apply a Blackman–Harris window to the FIR filter. Plot the spectra of both filters and adjust the order of the FIR filter to approximately match the slope of the IIR filter. Use Equation 4.13 to find the spectra of both filters. Count and display the number of b coefficients in the FIR filter and the number of a and b coefficients in the IIR filter.

The MATLAB's code:

```
clearvars;
fs=300; %sample frequency
N=256; %number of points choosen
f=(1:N)*fs/N; %frequency axis

order1=12; %order of filter IIR
cutoff_frequency=80/(fs/2);%normalised cutoff frequency

% Butterworth filter
[b1,a1]=butter(order1,cutoff_frequency,'high'); %
    Butterworth definition

% FIR filter
order2=63; %order of filter FIR
cutoff_frequency2=80/fs;
k=-floor(order2/2):-1;
b_negative = -sin((2*pi*cutoff_frequency2)*k)./(pi*k);
    %negative coefficients
b_zero=1-2*cutoff_frequency2; %index zero coefficient
b2=[b_negative, b_zero, fliplr(b_negative)]; %complete
    coefficient vector
b2=b2.*blackmanharris(length(b2))'; %apply blackman-
    harris

% Transfer function butter

%Transfer function IIR
H1=abs(fft(b1,N)./fft(a1,N));

%Transfer function FIR
H2=abs(fft(b2,N)); % fft b coefficients


%Plot settings
```

```
plot(f(1:N/2),H1(1:N/2),'DisplayName','IIR');
title('IIR Filter');

xlabel('Frequency (Hz)');
ylabel('Amplitude');

hold on;
plot(f(1:N/2),H2(1:N/2),'r','DisplayName','FIR');
title('FIR Filter');

xlabel('Frequency (Hz)');
xticks([0 10 20 30 40 50 60 70 80 90 100 110 120 130
    140 150]); %specific frequencies ticks to show
    cutoff frequency (80)
ylabel('Amplitude');
txt = {['IIR: ','Number of A coefficients-¿ ',num2str(
    length(a1)),' and number of B coefficients-¿ ',
    num2str(length(b1))
     ],['FIR: ','Number of B coefficients-¿ ',num2str(
        length(b2))]}; %string that contain the number
        of coefficients
subtitle(txt);
legend;
```

After adjusting the coefficients in order to match the slope of the IIR filter, we obtained a FIR filter with 63 coefficients (order 62 as the number coefficients is one unit more than the order). It is also possible to notice that the FIR filter is more restrictive to lower frequencies (Figure 18), as it rises after the IIR Filter.

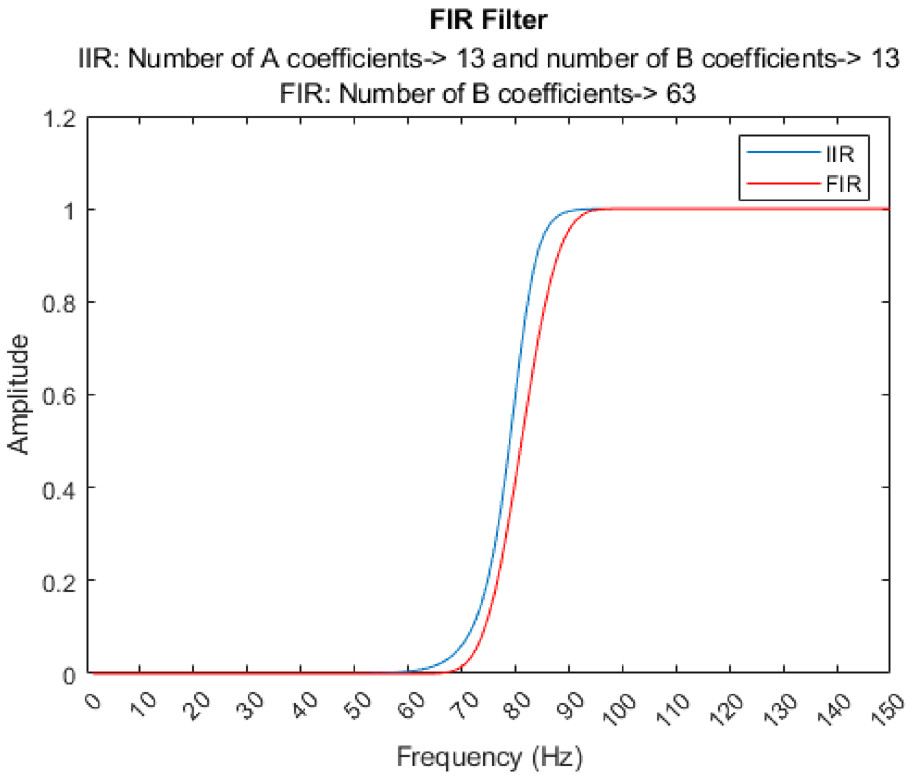

Figure 18: Output of the code of exercise 4.27