Mamadou Kaba

27070179

ELEC 342 CN-X

01/08/2024

08/08/2024

Nadia Abdolkhani

Objective

This lab aims to provide a comprehensive understanding of analog and digital filter design techniques using MATLAB. It includes the use of the Filter Design & Analysis Tool (FDAT), representing, playing, constructing, and plotting audio signals in MATLAB, and an introduction to Simulink for system modeling. The lab covers the design and implementation of lowpass, highpass, bandpass, bandstop, and notch filters, demonstrating their practical applications and analysis.

Theory

This lab involves the design and analysis of filters using MATLAB and Simulink, which are essential tools in signal processing. Filters can be classified as analog or digital, operating on continuous-time or discrete-time signals, respectively. The lab covers lowpass, highpass, bandpass, bandstop, and notch filters.

Lowpass filters allow frequencies below a certain cutoff to pass while attenuating higher frequencies, useful for removing high-frequency noise. Highpass filters do the opposite, eliminating low-frequency noise. Bandpass filters isolate frequencies within a specific range, while bandstop and notch filters remove frequencies within a specified range, with notch filters targeting very narrow bands for precision.

Key design parameters include passband frequency (ωp), stopband frequency (ωs), passband ripple (Rp), and stopband attenuation (Rs). Butterworth filters are known for their flat magnitude response, Chebyshev filters for their steep roll-off, and elliptic filters for the steepest roll-off with ripples in both passband and stopband.

MATLAB's Filter Design & Analysis Tool (FDATool) offers an interactive graphical interface for designing and analyzing filters. Simulink provides a visual environment for modeling and simulating systems, including filter design and application. Filters are implemented based on difference equations or transfer functions, with MATLAB functions like filtfilt used to apply filters without phase distortion.

Overall, this lab provides a thorough understanding of filter design and application, demonstrating how to use MATLAB and Simulink for effective signal processing.

Results/Discussion

Part 1

Question 1

a) and b)

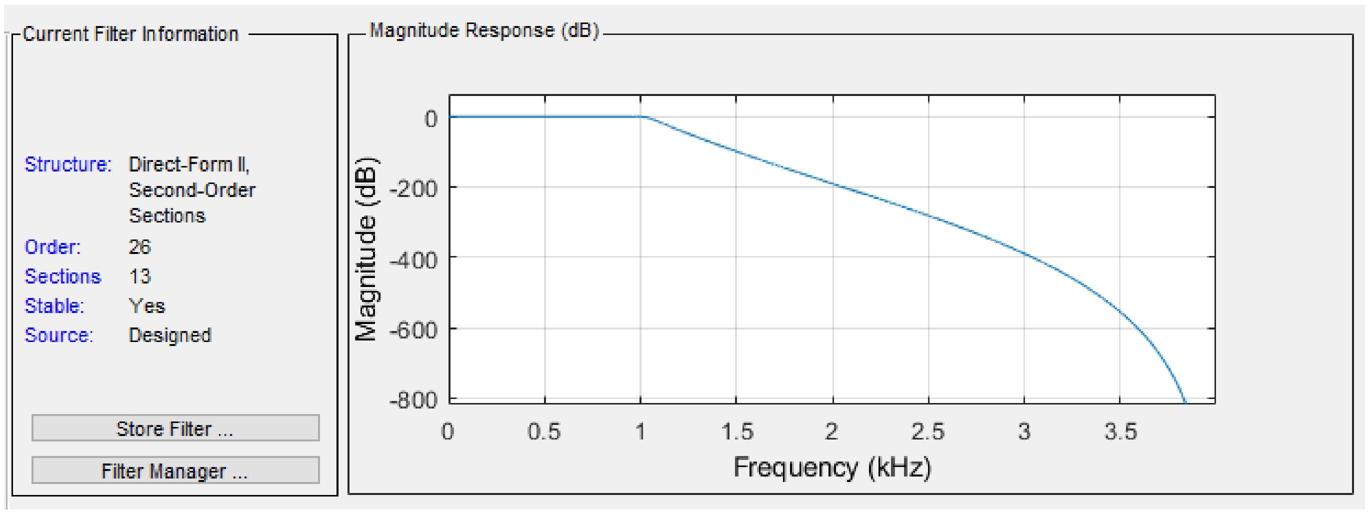
Scaling the amplitude of the audio signal by 0.25 reduces the volume to a quarter of the original. The sound is quieter compared to the original audio. Whereas scaling the amplitude of the audio signal by 4 increases the volume fourfold, making the sound louder compared to the original audio. We also experience some clipping and distortion.

c)

Playing the audio at half the original sampling rate (fs / 2) slows down the playback speed, lowering the pitch and making the sound longer in duration. It sounds deeper and slower compared to the original audio. And playing the audio at double the original sampling rate (fs * 2) speeds up the playback speed, raising the pitch and shortening the duration, making it sound higher-pitched and faster compared to the original audio.

Question 2

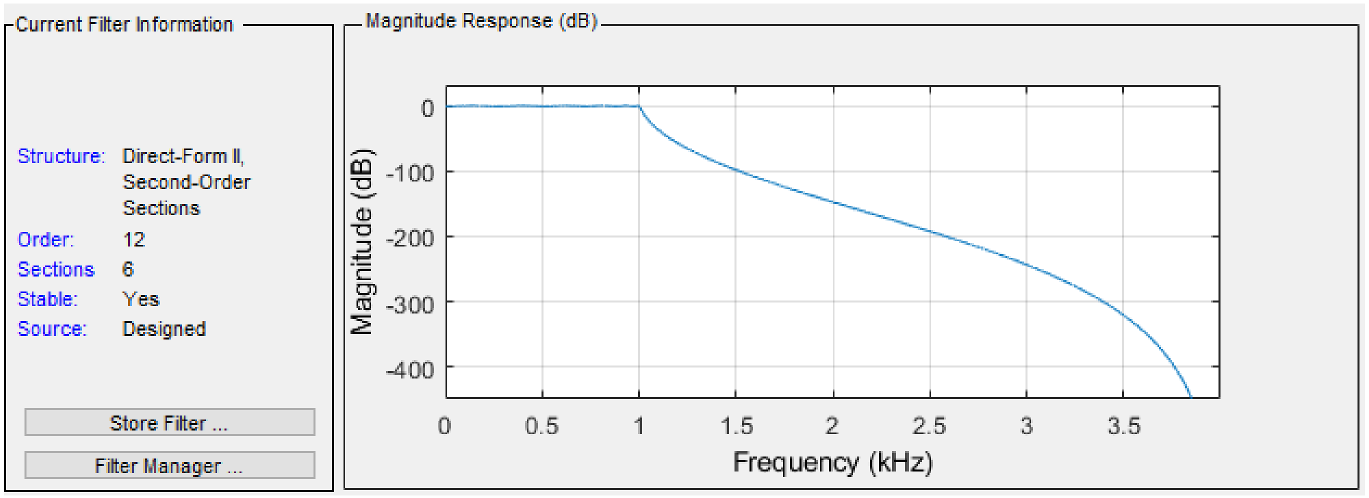
a) The order is 26.



Current Filter Information

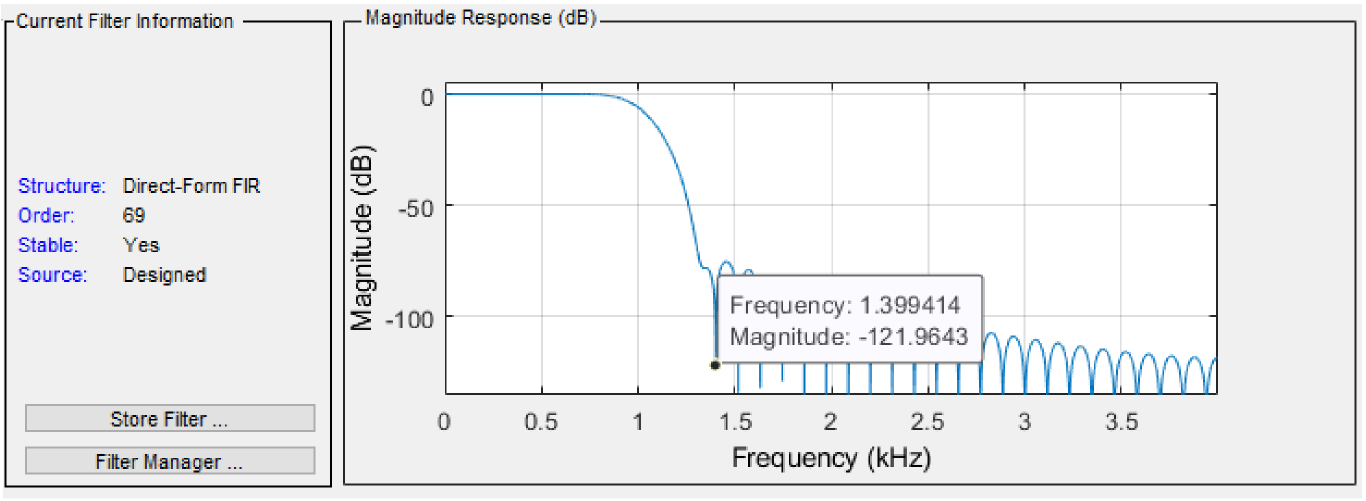
Structure: Direct-Form II, Second-Order Sections
Order: 26
Sections 13
Stable: Yes
Source: Designed

b) The order is 12.


Current Filter Information

Structure: Direct-Form II, Second-Order Sections
Order: 12
Sections 6
Stable: Yes
Source: Designed

c)

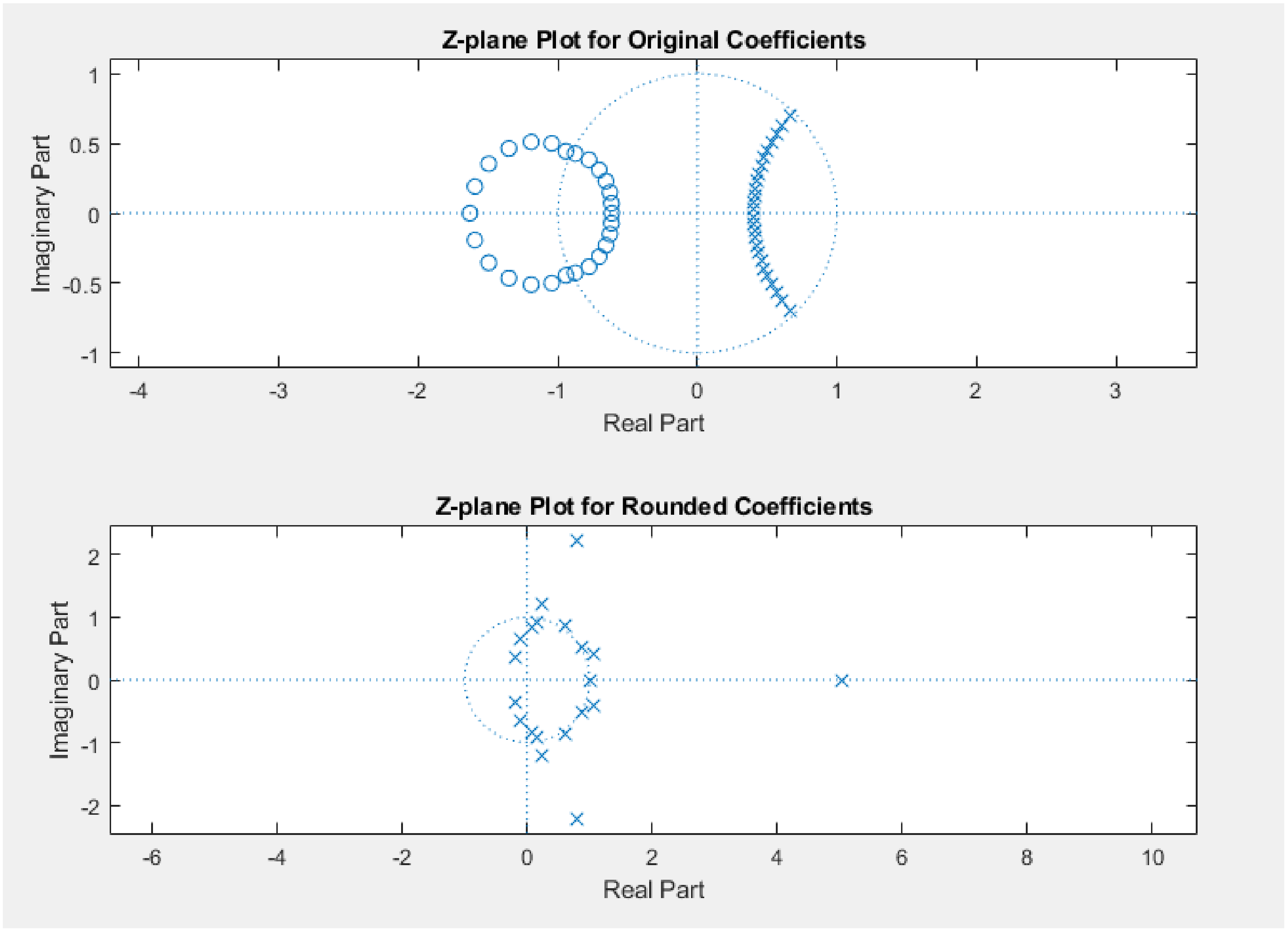

Current Filter Information

Structure: Direct-Form FIR
Order: 69
Stable: Yes
Source: Designed

Frequency: 1.399414
Magnitude: -121.9643

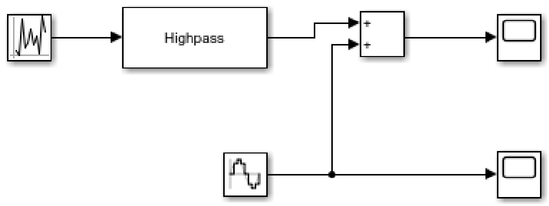
The implementation costs of the three filters show that the FIR filter using the windowing technique is highly inefficient compared to the IIR filters. The FIR filter has an order of 69, making it significantly more computationally intensive and memory-demanding. In contrast, the Butterworth filter has an order of 12, and the Chebyshev Type I filter has an order of 26. This indicates that the FIR filter is approximately 5.75 times more expensive in terms of memory than the Butterworth filter, highlighting the inefficiency of the windowing technique in practical applications.

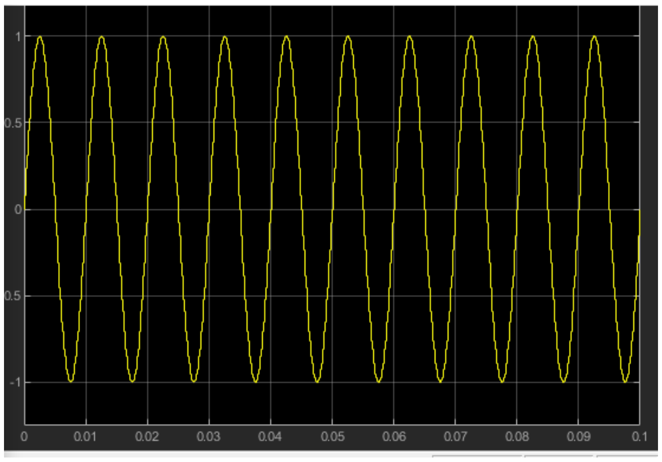
d)



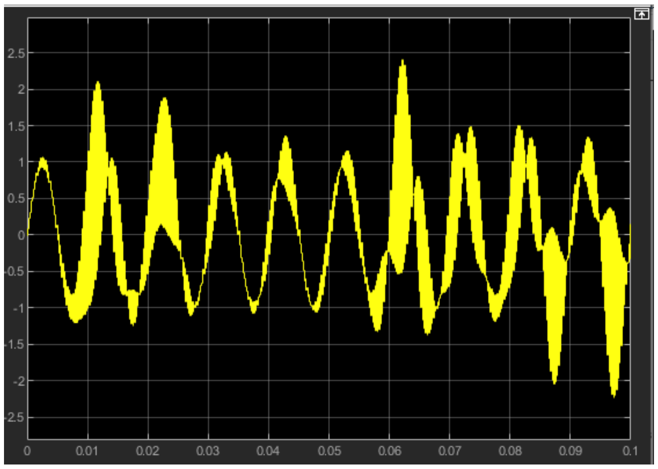
The pole-zero plot for the original filter coefficients (top plot) shows a well-defined distribution of poles and zeros, indicating a stable and well-behaved filter. However, after rounding the coefficients to the nearest integer (bottom plot), the pole-zero plot becomes significantly distorted, with poles and zeros scattered more widely and even extending outside the unit circle.

Question 3





The first plot displays the original sine wave, generated with a frequency of 100 Hz and a sample time of 1/8000 seconds. The waveform is smooth and consistent, representing a clean sine wave without any noise or distortion.

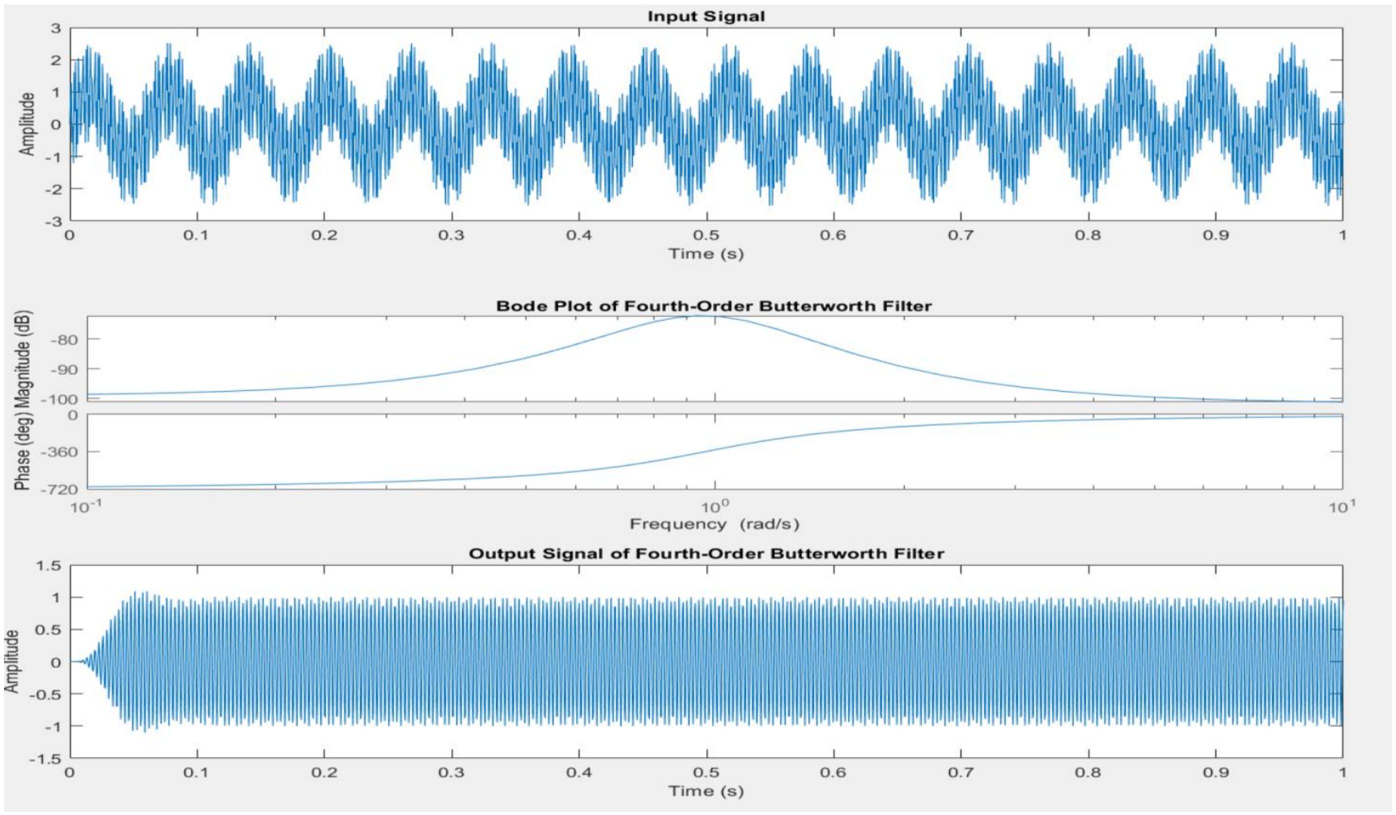


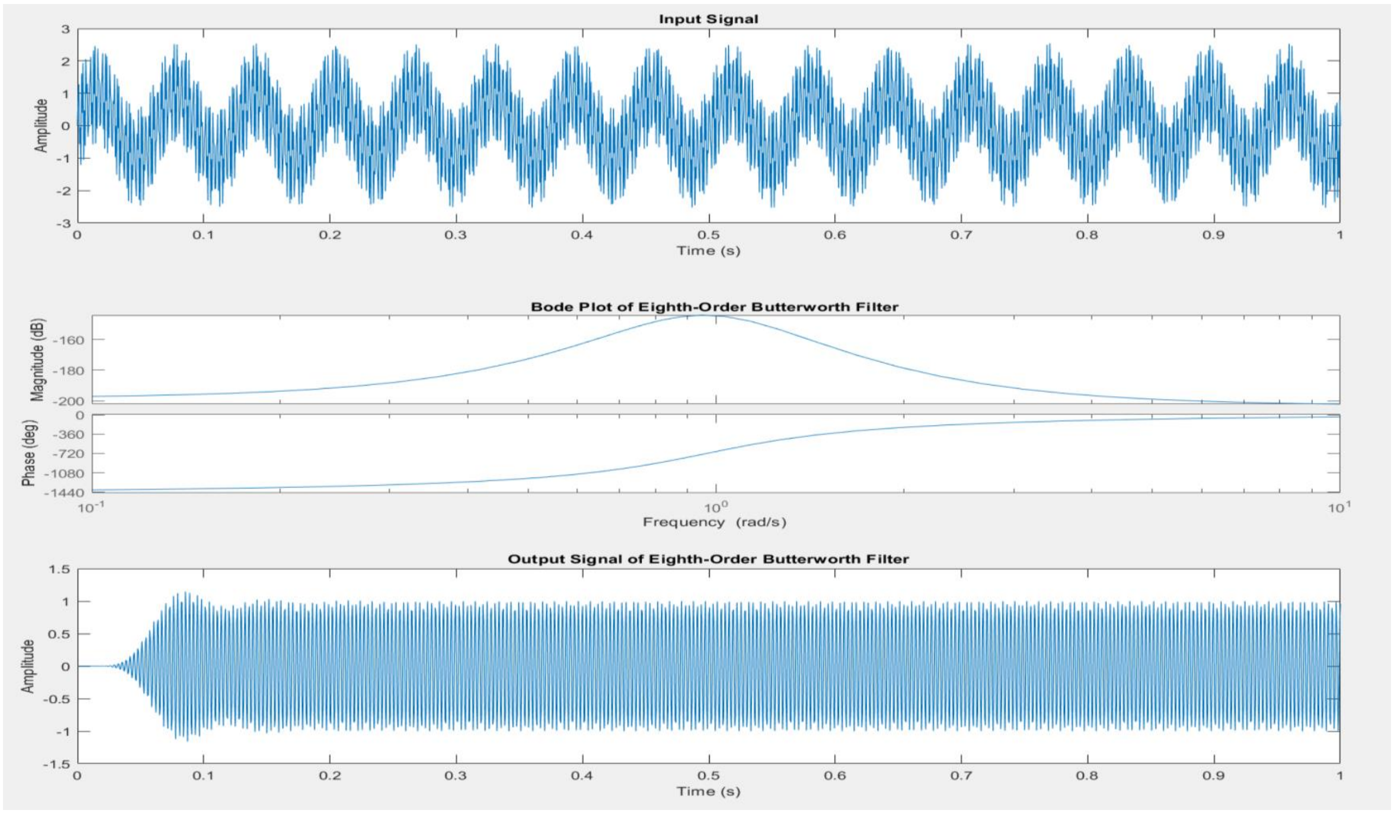
The second plot shows the noisy sine wave, where high-frequency noise has been added. The noise was generated using a Uniform Random Number block, set to produce random values between -5 and 5, and then passed through a highpass filter to ensure only frequencies above 3800 Hz were included. The resulting waveform in the second plot exhibits rapid oscillations superimposed on the original sine wave, indicating the presence of high-frequency noise.

Part 2

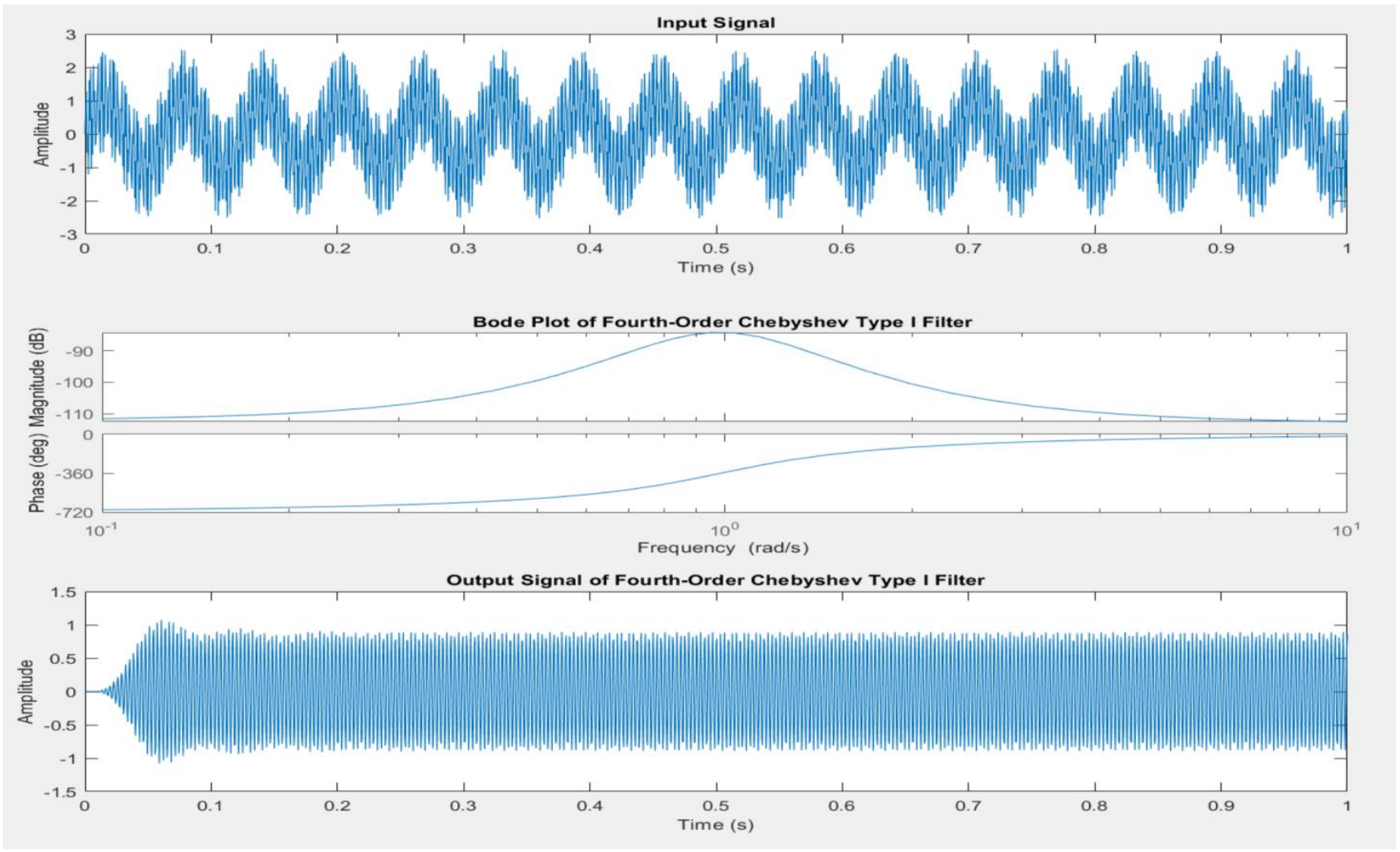
Question 1

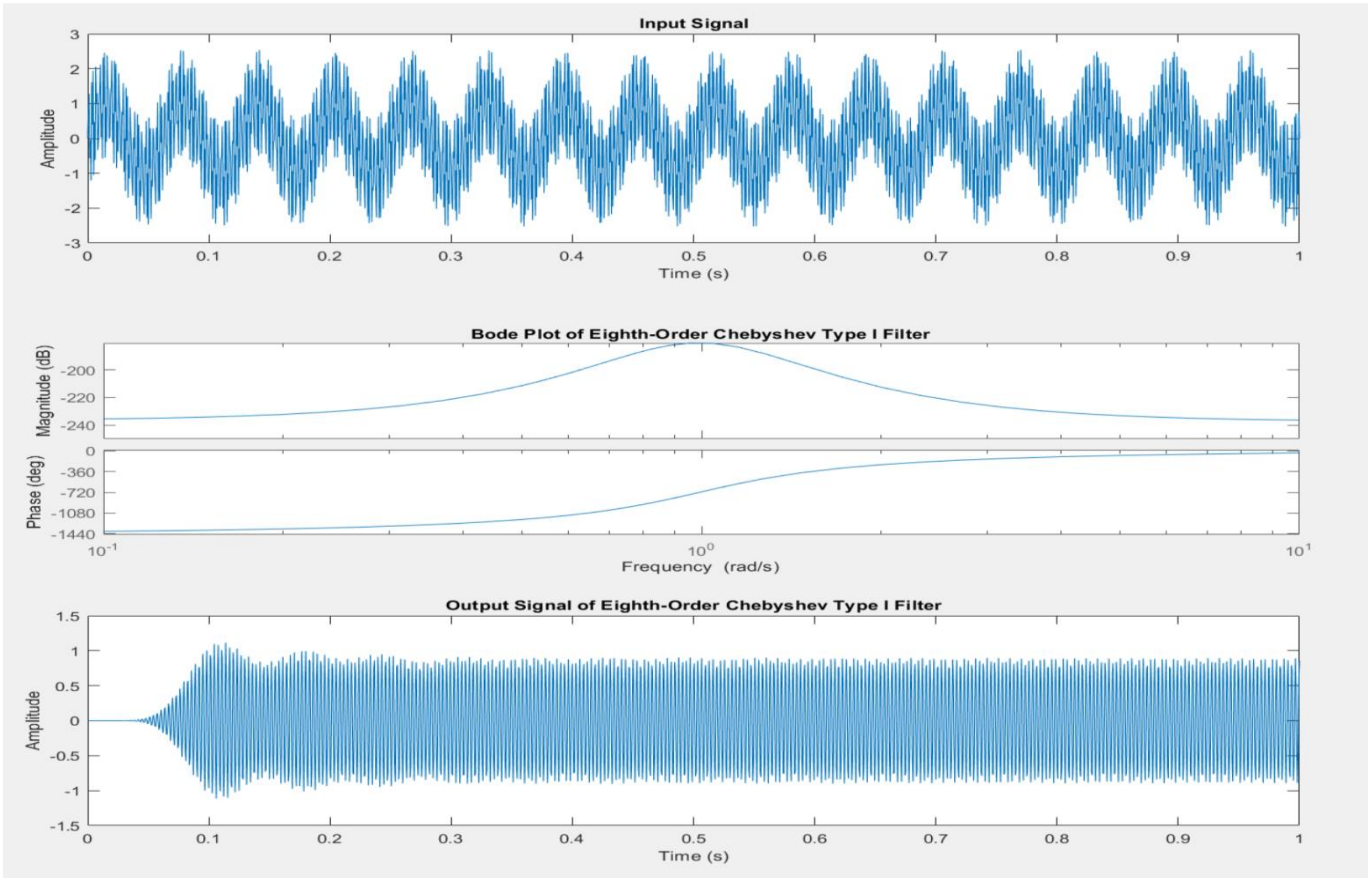
a)



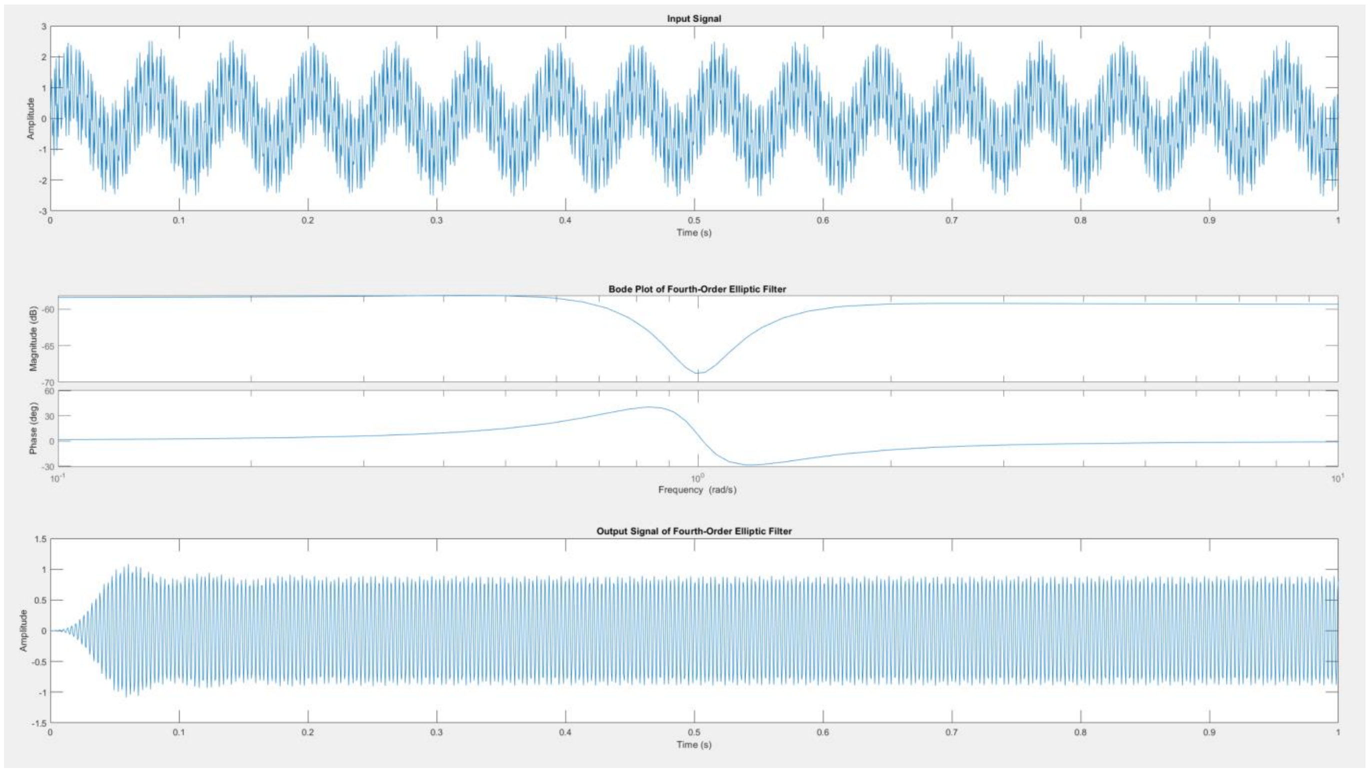


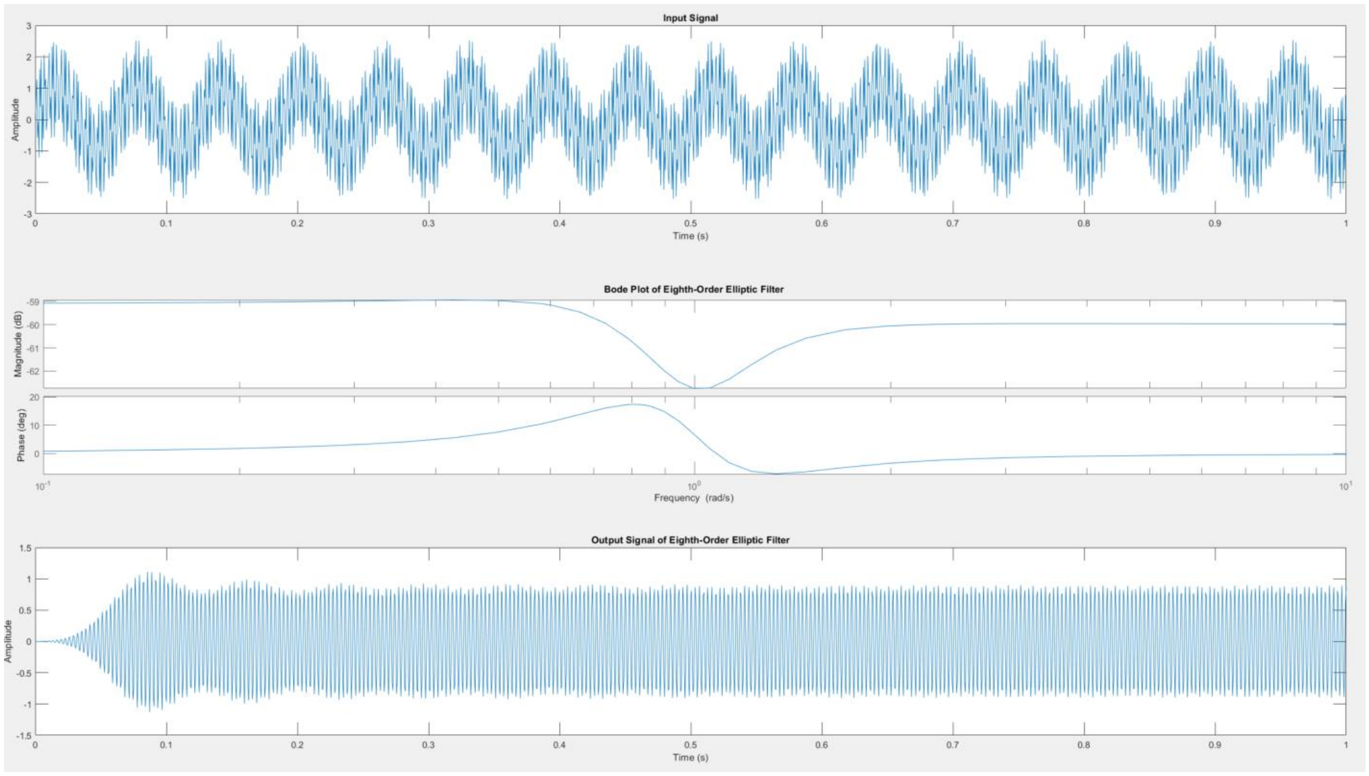
b)


Input Signal
Amplitude
Time (s)
Bode Plot of Fourth-Order Chebyshev Type I Filter
Magnitude (dB)
Phase (deg)
Frequency (rad/s)
Output Signal of Fourth-Order Chebyshev Type I Filter
Amplitude
Time (s)

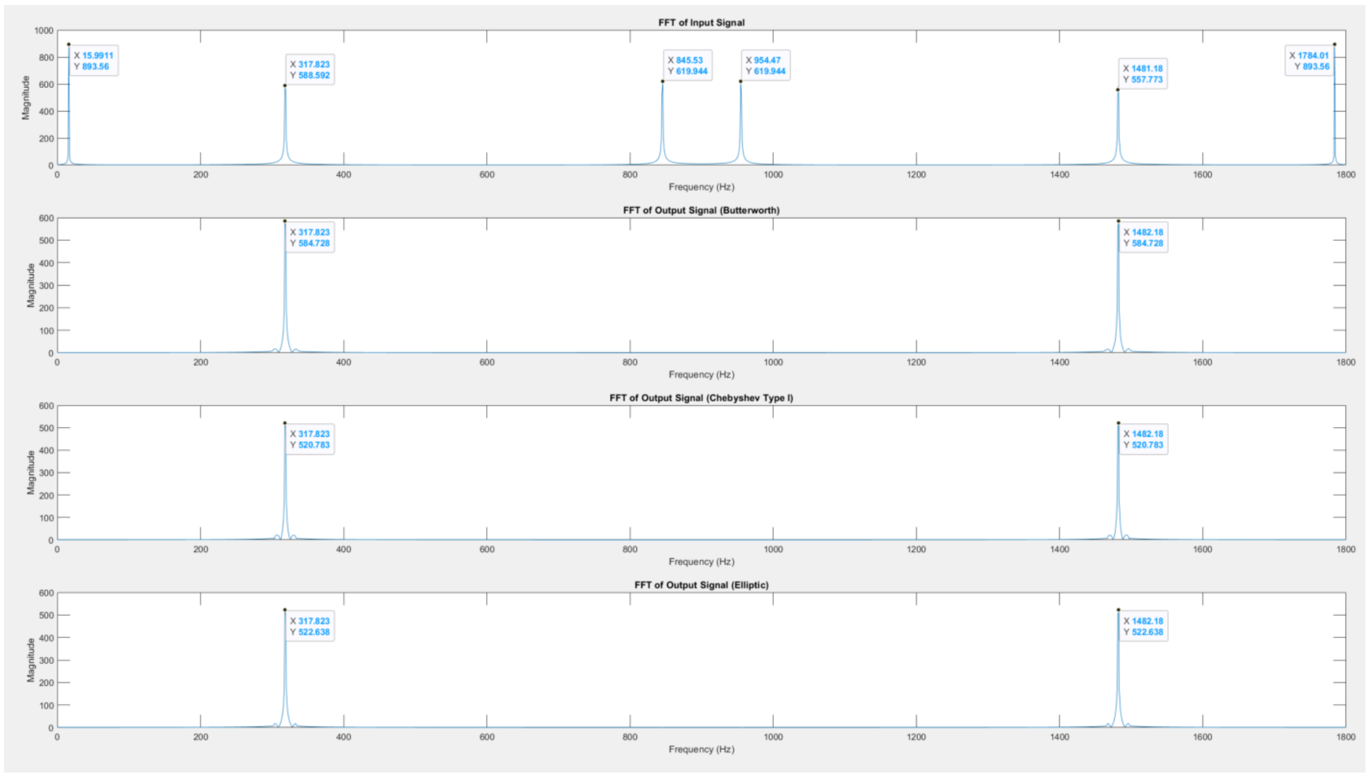

Input Signal
Amplitude
Time (s)
Bode Plot of Eighth-Order Chebyshev Type I Filter
Magnitude (dB)
Phase (deg)
Frequency (rad/s)
Output Signal of Eighth-Order Chebyshev Type I Filter
Amplitude
Time (s)

c)


Input Signal
Amplitude
Time (s)
Bode Plot of Fourth-Order Elliptic Filter
Magnitude (dB)
Phase (deg)
Frequency (rad/s)
Output Signal of Fourth-Order Elliptic Filter
Amplitude
Time (s)


Input Signal
Amplitude
Time (s)
Bode Plot of Eighth-Order Elliptic Filter
Magnitude (dB)
Phase (deg)
Frequency (rad/s)
Output Signal of Eighth-Order Elliptic Filter
Amplitude
Time (s)

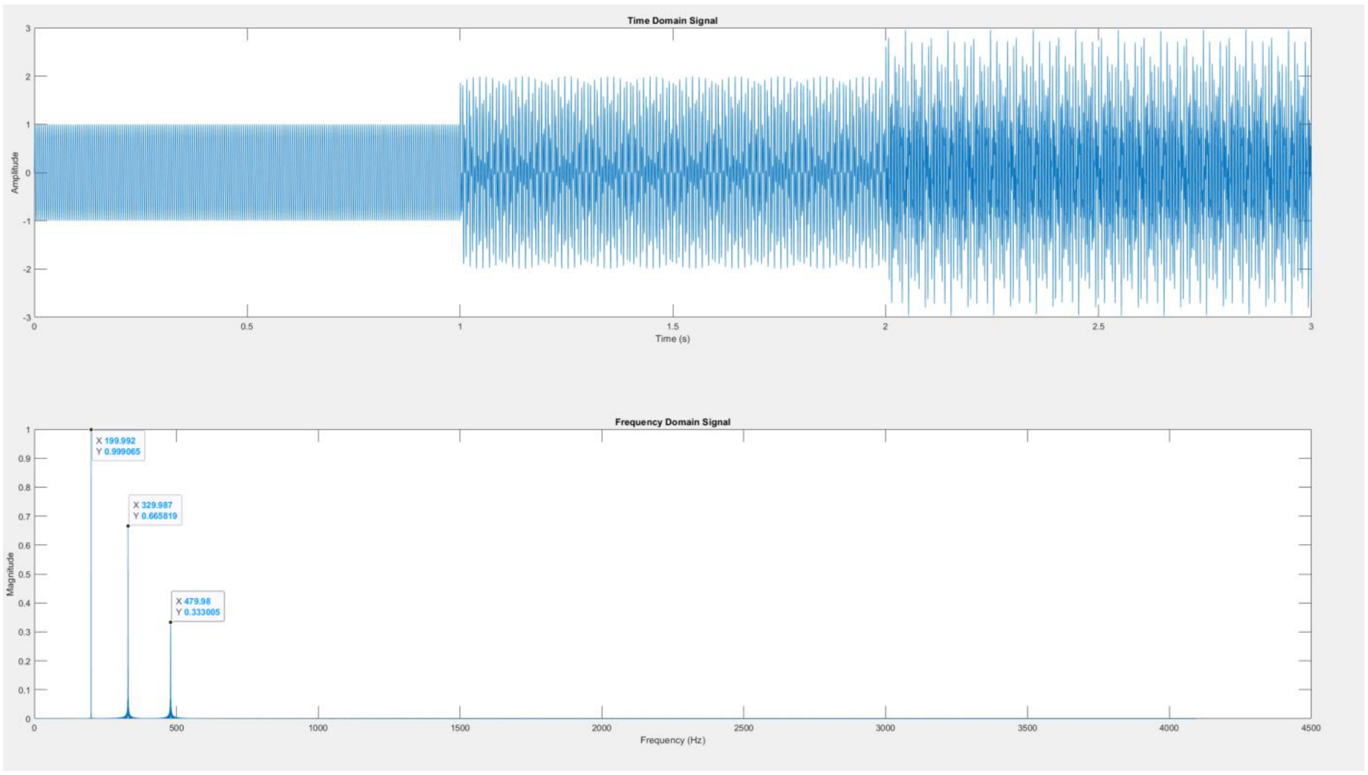
d)



All three filters (Butterworth, Chebyshev Type I, and Elliptic) successfully isolate the desired frequency components (around 317.823 Hz and 1482.18 Hz) from the input signal. However, the Butterworth filter demonstrates slightly superior performance in preserving the magnitudes of these components, followed closely by the Chebyshev Type I and Elliptic filters. This analysis highlights the effectiveness of each filter type in achieving frequency isolation, with the Butterworth filter showing a marginal advantage in maintaining signal strength.

Question 2

Part A

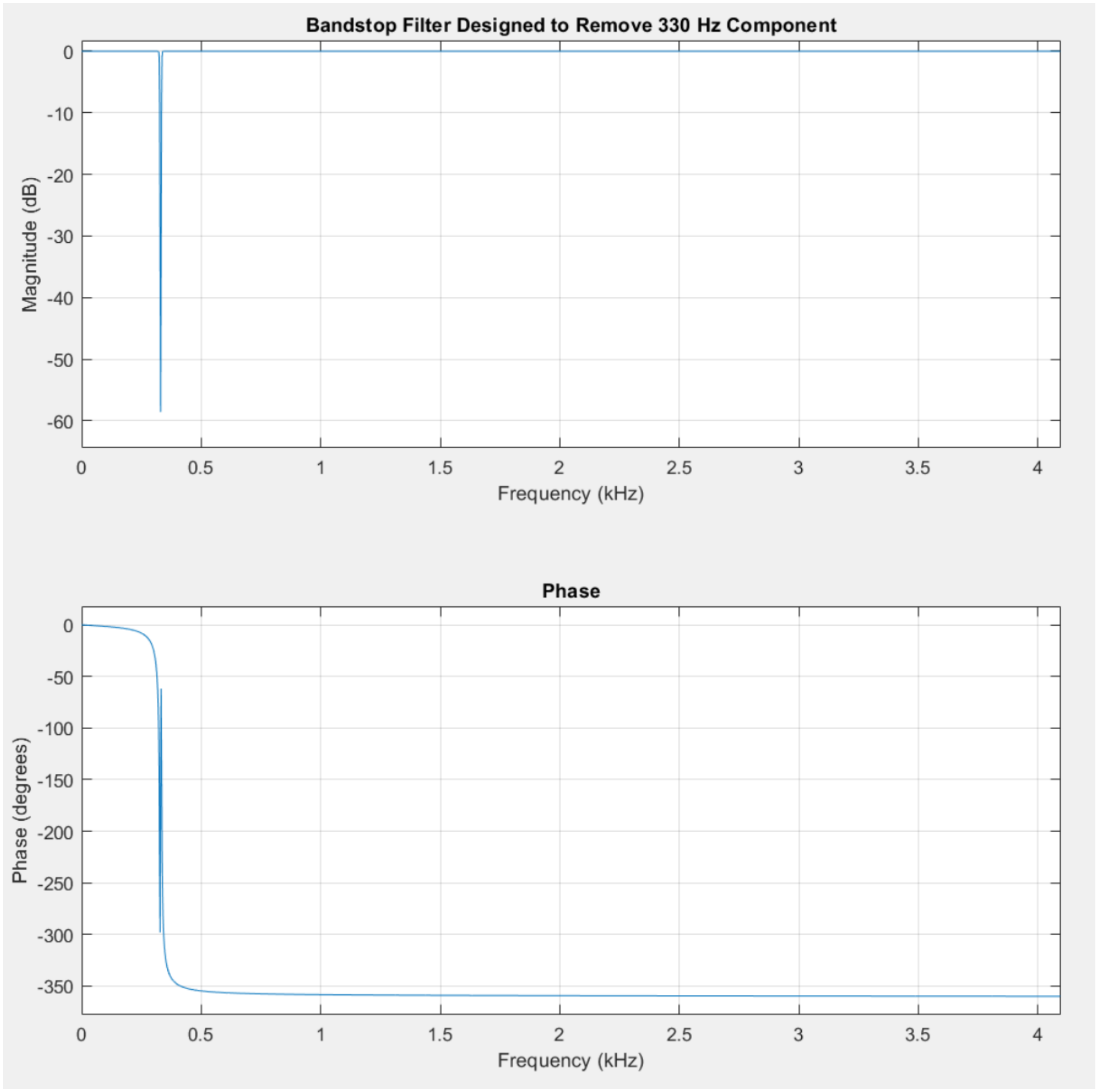


The time-domain plot of the signal shows the presence of three distinct tones over the three-second duration, starting with a 200 Hz tone, followed by the addition of a 330 Hz tone at 1 second, and finally a 480 Hz tone at 2 seconds. The frequency-domain plot accurately identifies these tones with clear peaks at 200 Hz, 330 Hz, and 480 Hz, confirming the correct composition of the signal.

The original signal, when played, contains three distinct tones that change over time, corresponding to the frequencies 200 Hz, 330 Hz, and 480 Hz.

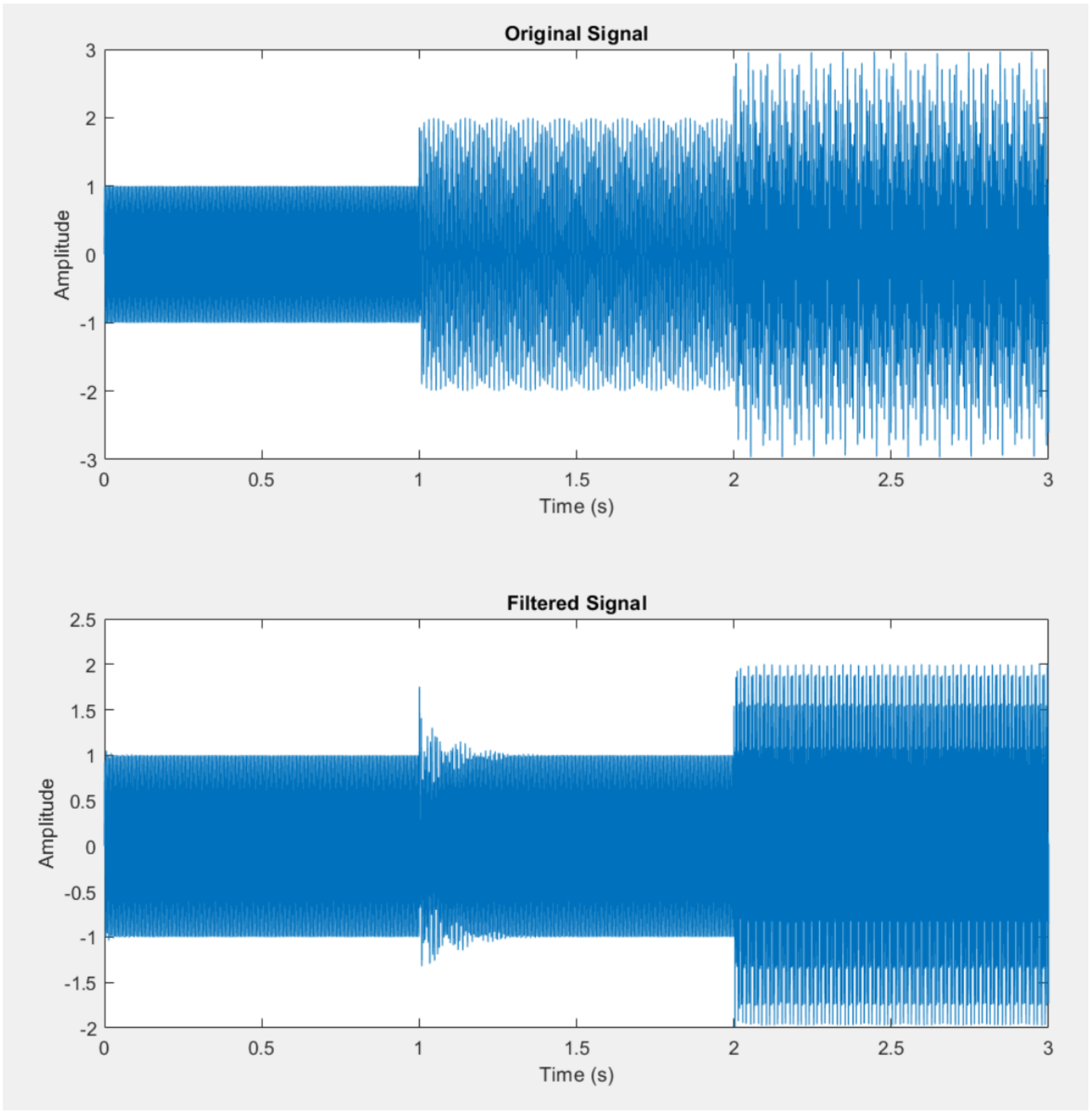
Part B

a)



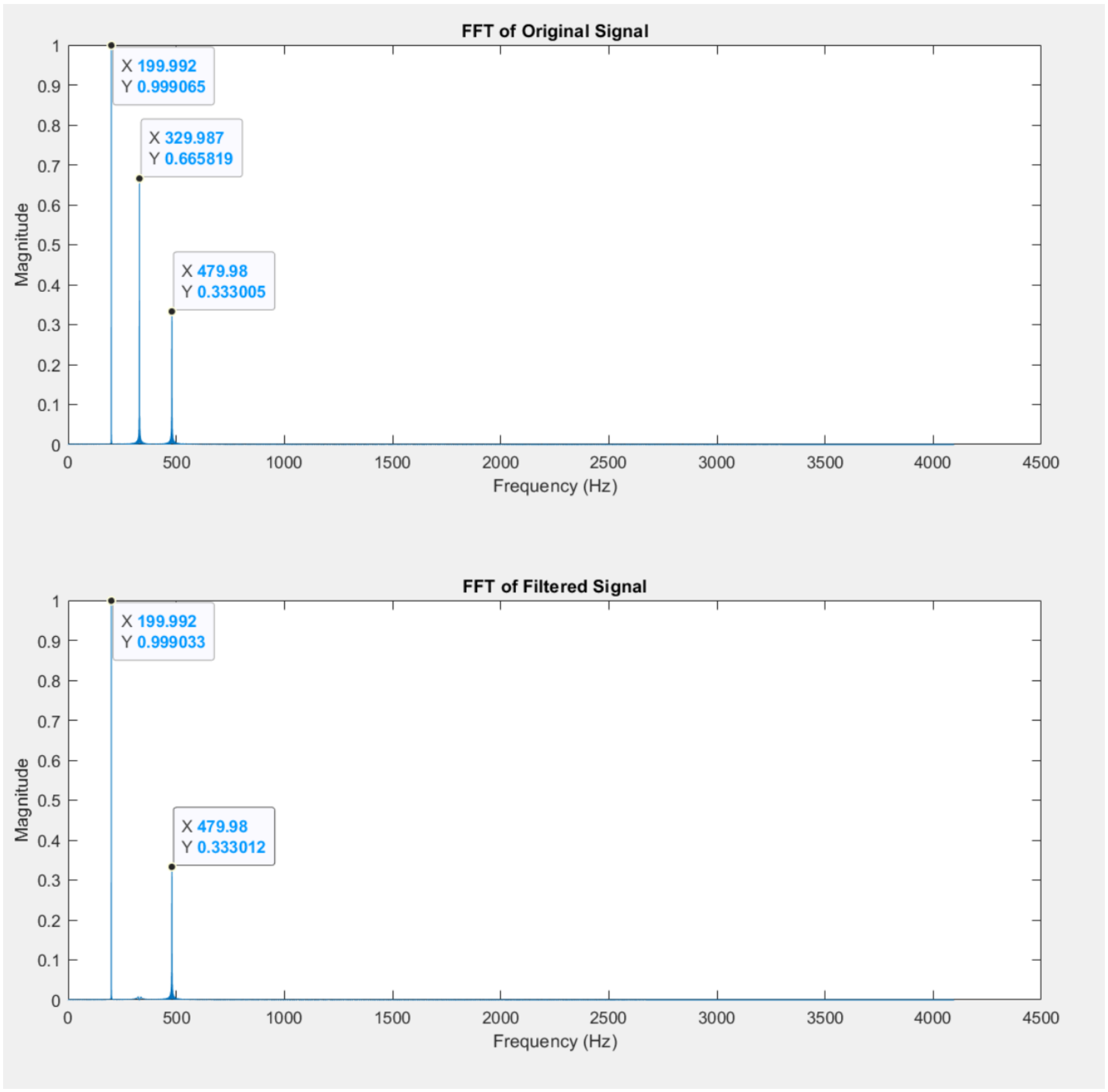
The plots above show the magnitude and phase responses of a fourth-order Butterworth bandstop filter designed to remove the 330 Hz component from the main signal. The magnitude response clearly indicates a notch around 330 Hz, effectively attenuating frequencies in this range. The phase response shows a sharp transition at the cutoff frequencies, which is characteristic of Butterworth filters. This filter is expected to suppress the 330 Hz component while allowing other frequencies to pass with minimal attenuation, thereby preserving the overall integrity of the signal.

b)



The time-domain plots show the original signal and the filtered signal after applying the bandstop filter designed to remove the 330 Hz component. While the bandstop filter has successfully attenuated the 330 Hz component for the most part, there is noticeable residual signal between 1 to 1.25 seconds. This indicates that the filter may not have perfectly removed the 330 Hz component during this interval, possibly due to the filter's transition band or other signal processing artifacts.

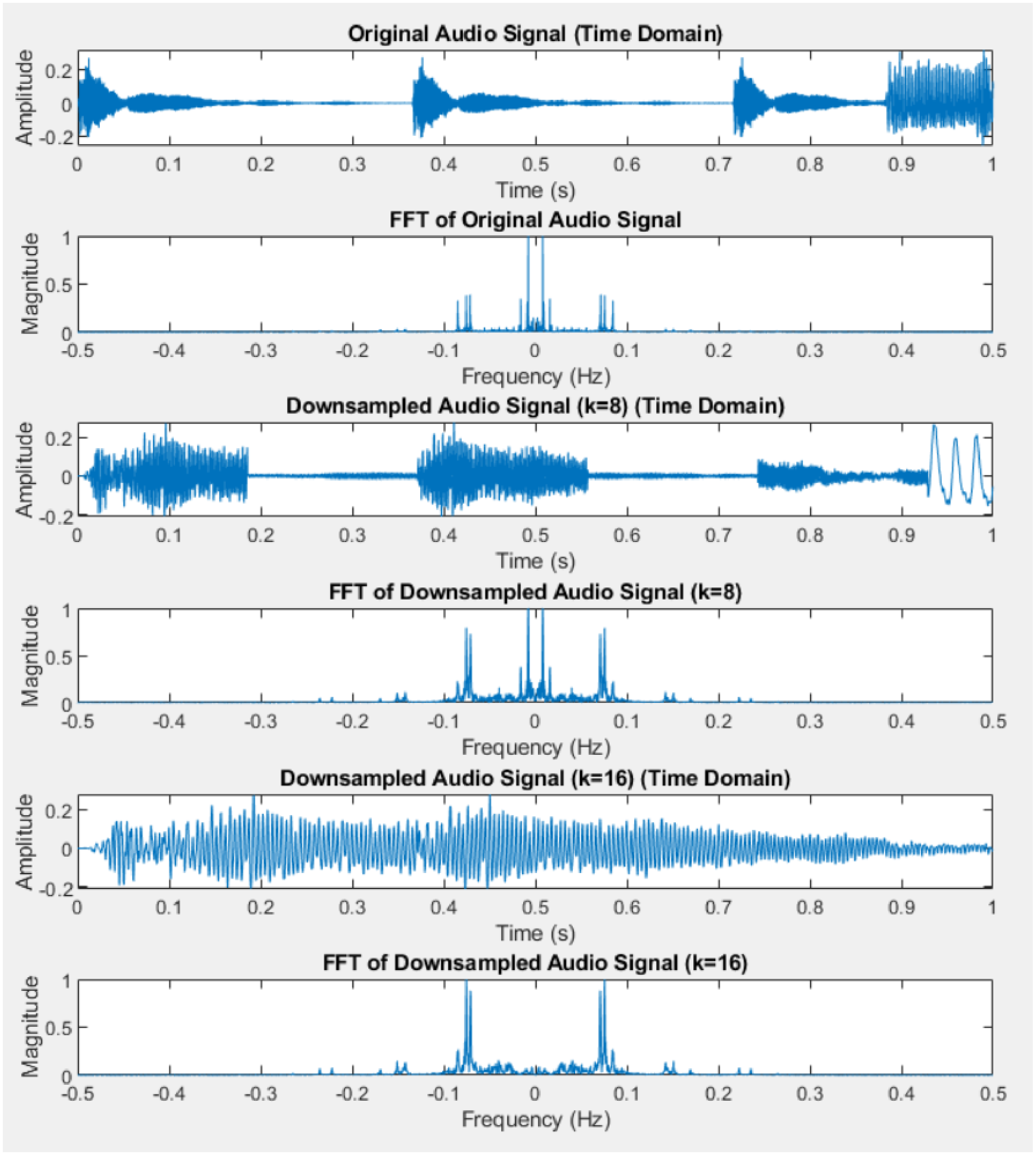
c)



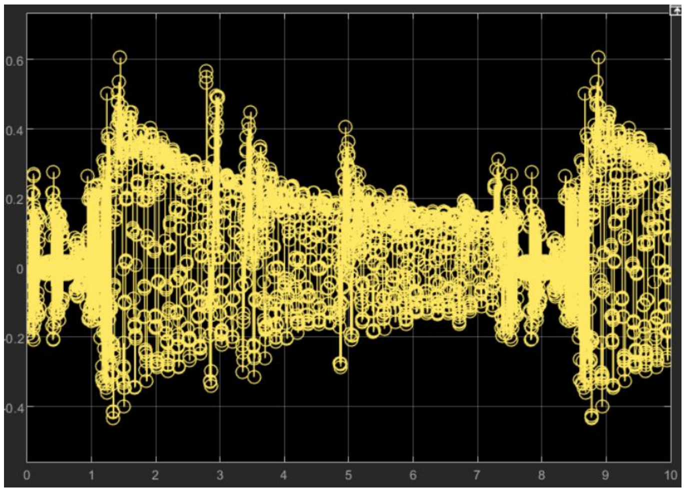
The FFT plots show the frequency domain representation of the original signal and the filtered signal. In the original signal, three prominent peaks are visible at approximately 200 Hz, 330 Hz, and 480 Hz, corresponding to the three tones present in the signal. After applying the bandstop filter designed to remove the 330 Hz component, the FFT of the filtered signal shows that the peak at 330 Hz has been significantly attenuated. However, the peaks at 200 Hz and 480 Hz remain largely unaffected, which indicates that the bandstop filter effectively targeted the 330 Hz component while preserving the other frequencies. Upon listening to the filtered signal using the soundsc command, it becomes evident that the 330 Hz component has been attenuated. After filtering, the sound no longer contains the 330 Hz tone, which is especially noticeable in the segment from 1 to 2 seconds where the 330 Hz component was present in the original signal.

Question 3

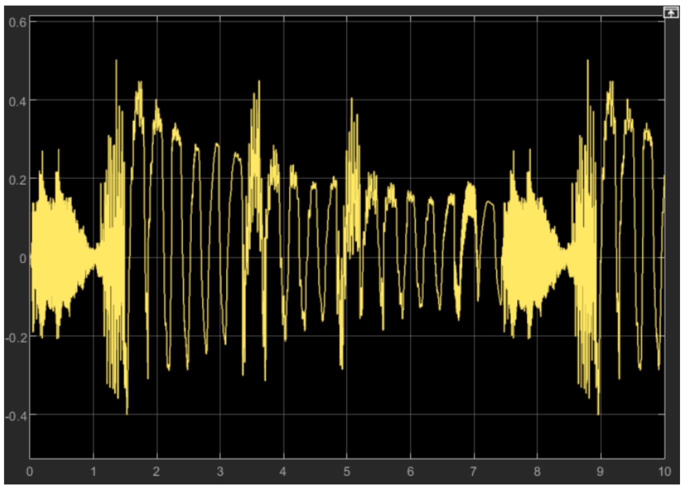
Part A



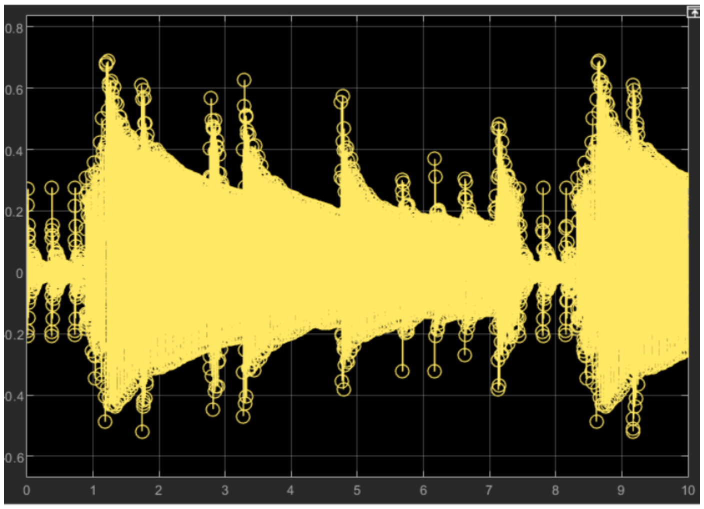
When downsampling the audio signal with k=8, the sound still has some of its original quality but becomes a bit muffled, losing some high-frequency details. With k=16, the sound quality drops even more, becoming very muffled and dull, with noticeable distortions. Compared to the original audio, both downsampled versions lack the clarity and natural sound, with k=16 sounding worse.



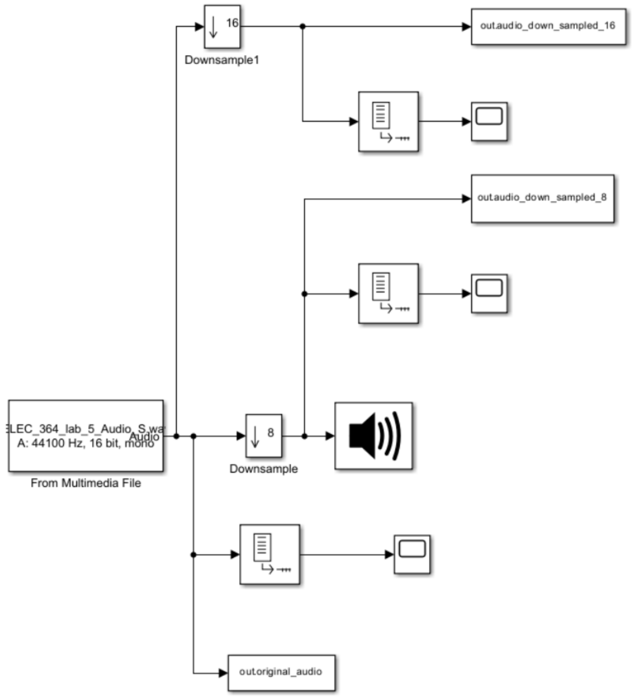
K = 8



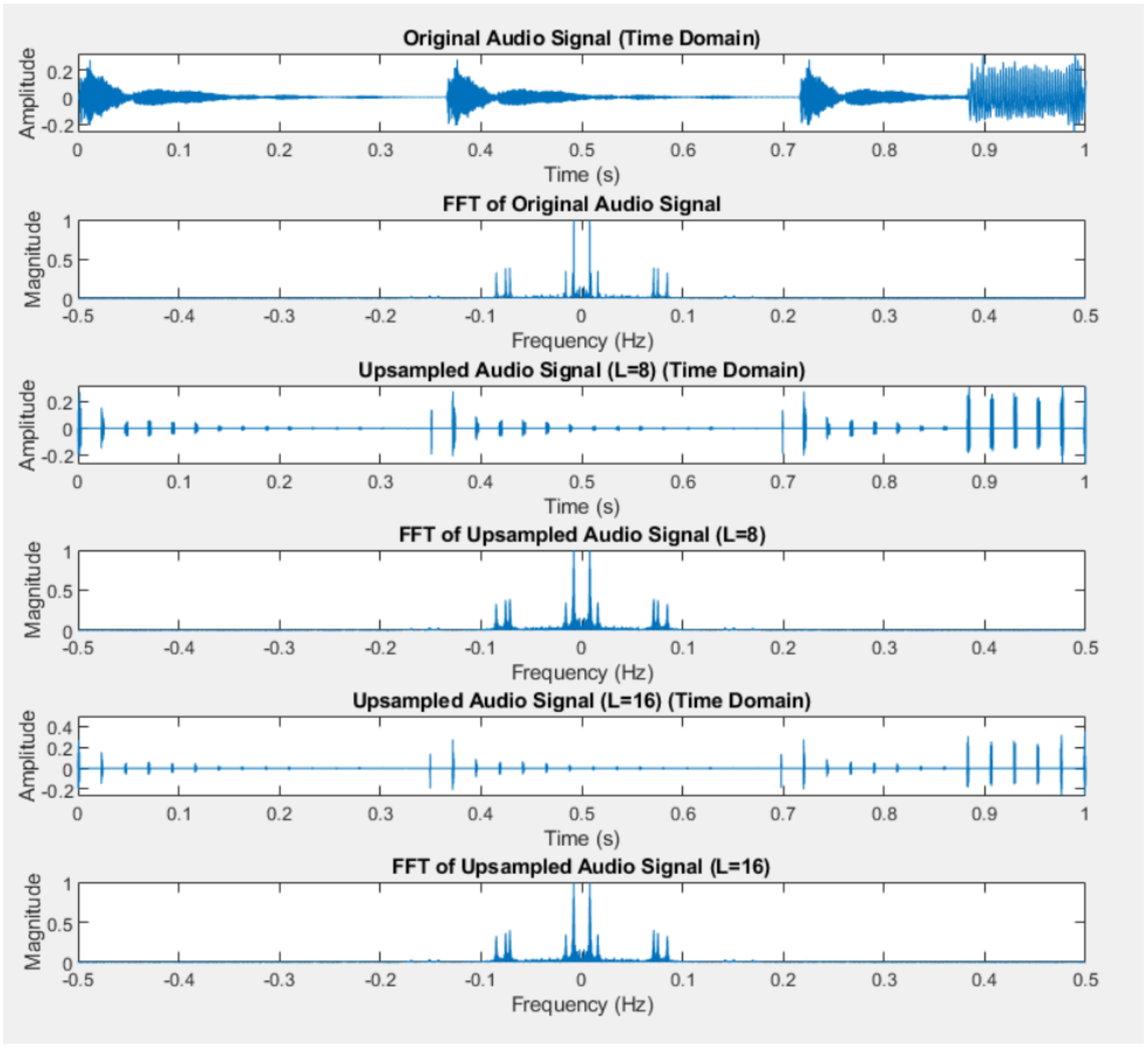
k = 16



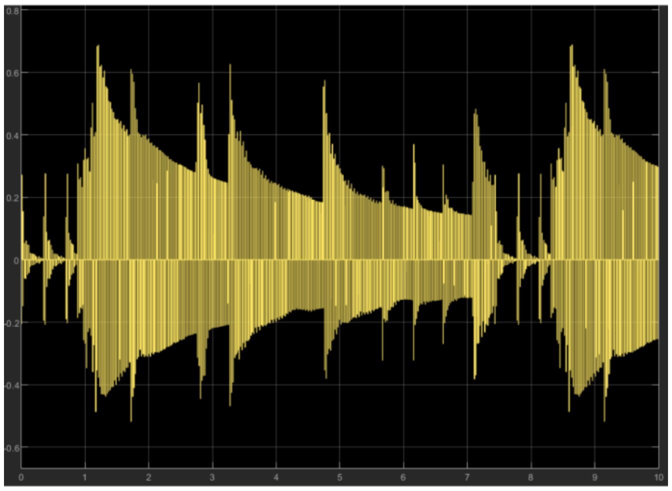
Original



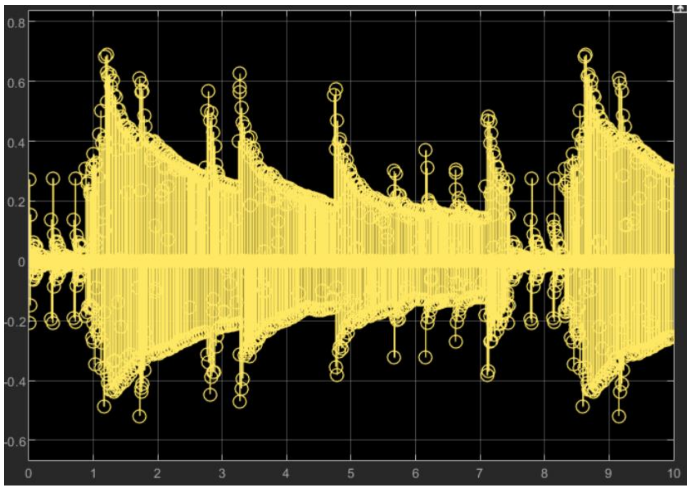
Part B



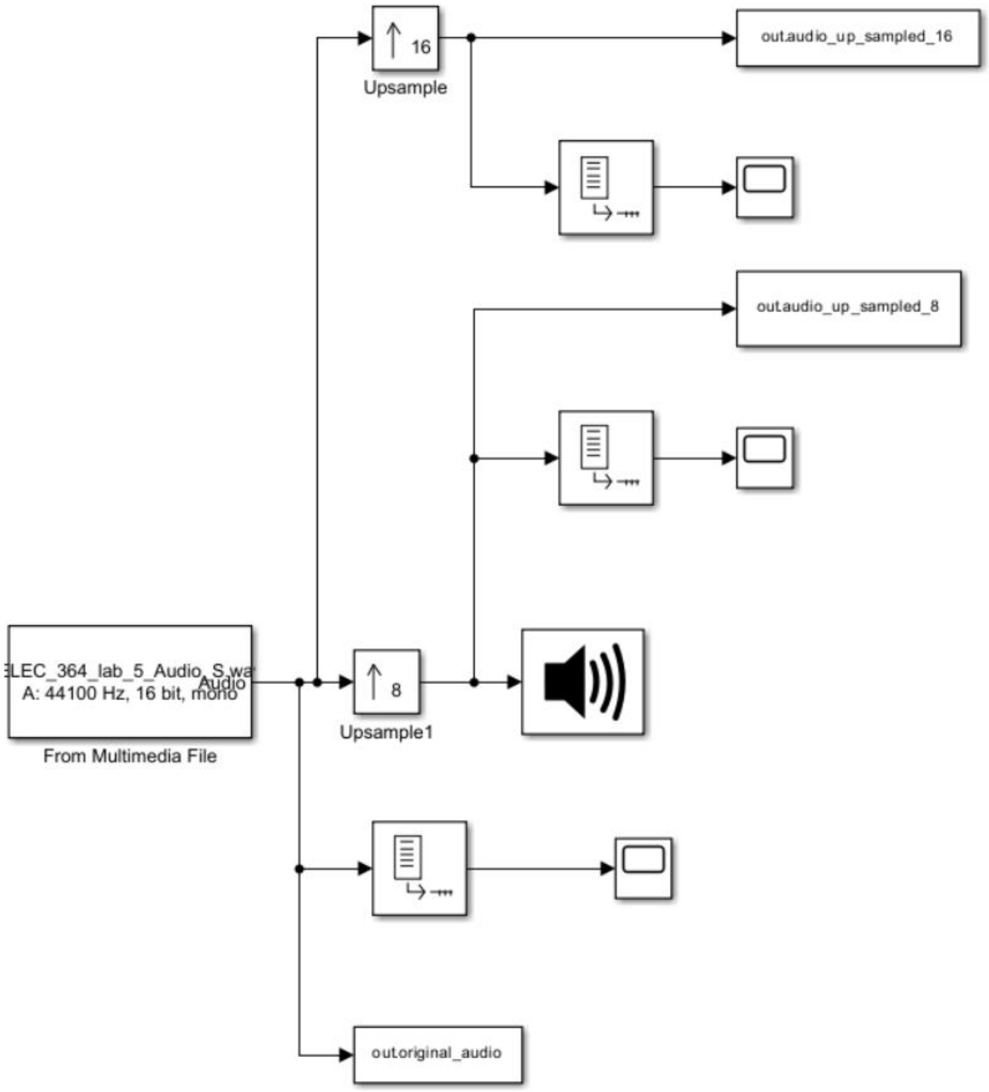
When upsampling the audio signal by a factor of 8 (L = 8), the sound becomes smoother making it a bit similar to the original audio but introduce minor distortions. These distortions make the sound feel slightly unnatural compared to the original. With an upsampling factor of 16 (L = 16), the distortions are more noticeable, making the sound quality worse.



L = 16



L = 8


16
Upsample
out.audio_up_sampled_16
out.audio_up_sampled_8
ELEC_364_lab_5_Audio_S.wa
A: 44100 Hz, 16 bit, mono
Audio
From Multimedia File
8
Upsample1
out.original_audio

Conclusion

This lab successfully provided a comprehensive understanding of analog and digital filter design techniques using MATLAB. Through the use of the Filter Design & Analysis Tool (FDAT), we effectively designed and analyzed various filters, including lowpass, highpass, bandpass, bandstop, and notch filters. We also explored the practical applications of these filters by representing, playing, constructing, and plotting audio signals in MATLAB. Additionally, the introduction to Simulink for system modeling enhanced our ability to visualize and simulate filter effects on audio signals. Overall, this lab demonstrated the practical implementation and analysis of different filter types, reinforcing the importance of these tools and techniques in audio signal processing.

Appendix

Lab4_part1_question1

```
% Mamadou Kaba 27070179
% Part 1 Question 1

clc; clear; close all;

[y,fs]=audioread('lab_5_Audio_1.wav');

% a)

%sound(y,fs)

% b)

%sound(0.25*y,fs)
%sound(4*y,fs)

% c)

%sound(y, fs/2)
sound(y, fs*2)
```

Lab4_part1_question2d

```
% Mamadou Kaba 27070179
% Part 1 Question 2 d)

clc;clear;
Hd = fda_butterworth;
sosMatrix = Hd.sosMatrix;
ScaleValues = Hd.ScaleValues;
[b,a] = sos2tf(sosMatrix, ScaleValues);
b2=round(b);
a2=round(a);

subplot(2,1,1);
zplane(b,a);
title('Z-plane Plot for Original Coefficients');

subplot(2,1,2);
zplane(b2,a2);
title('Z-plane Plot for Rounded Coefficients');
```

Lab4_part2_question1b

```
% Mamadou Kaba 27070179
% Part 2 Question 1 a)
clc; clear; close all;

Fs = 1800;
t = 0:1/Fs:1;
```

```
x = sin(100*t) + sin(2000*t) + sin(6000*t);


Wn = [1900 2100]/(Fs*pi);

% Fourth-order Butterworth filter
[b4, a4] = butter(4, Wn, 'bandpass');
x_filt4 = filter(b4, a4, x);

% Eighth-order Butterworth filter
[b8, a8] = butter(8, Wn, 'bandpass');
x_filt8 = filter(b8, a8, x);

% Plot the input signal
figure;
subplot(3,1,1);
plot(t, x);
title('Input Signal');
xlabel('Time (s)');
ylabel('Amplitude');

% Bode plot for fourth-order Butterworth filter
subplot(3,1,2);
g4 = tf(b4, a4);
bode(g4);
title('Bode Plot of Fourth-Order Butterworth Filter');

% Output signal of the fourth-order Butterworth filter
subplot(3,1,3);
plot(t, x_filt4);
title('Output Signal of Fourth-Order Butterworth Filter');
xlabel('Time (s)');
ylabel('Amplitude');

% Plot the input signal again for eighth-order Butterworth filter
figure;
subplot(3,1,1);
plot(t, x);
title('Input Signal');
xlabel('Time (s)');
ylabel('Amplitude');

% Bode plot for eighth-order Butterworth filter
subplot(3,1,2);
g8 = tf(b8, a8);
bode(g8);
title('Bode Plot of Eighth-Order Butterworth Filter');

% Output signal of the eighth-order Butterworth filter
subplot(3,1,3);
plot(t, x_filt8);
title('Output Signal of Eighth-Order Butterworth Filter');
xlabel('Time (s)');
ylabel('Amplitude');
```

Lab4_part2_question1b

```matlab
% Mamadou Kaba 27070179
% Part 2 Question 1 b)
clc; clear; close all;

Fs = 1800;
t = 0:1/Fs:1;


x = sin(100*t) + sin(2000*t) + sin(6000*t);


Wn = [1900 2100]/(Fs*pi);

% Fourth-order Chebyshev Type I filter with 1 dB passband ripple
[b4, a4] = cheby1(4, 1, Wn, 'bandpass');
x_filt4 = filter(b4, a4, x);

% Eighth-order Chebyshev Type I filter with 1 dB passband ripple
[b8, a8] = cheby1(8, 1, Wn, 'bandpass');
x_filt8 = filter(b8, a8, x);

% Plot the input signal
figure;
subplot(3,1,1);
plot(t, x);
title('Input Signal');
xlabel('Time (s)');
ylabel('Amplitude');

% Bode plot for fourth-order Chebyshev Type I filter
subplot(3,1,2);
g4 = tf(b4, a4);
bode(g4);
title('Bode Plot of Fourth-Order Chebyshev Type I Filter');

% Output signal of the fourth-order Chebyshev Type I filter
subplot(3,1,3);
plot(t, x_filt4);
title('Output Signal of Fourth-Order Chebyshev Type I Filter');
xlabel('Time (s)');
ylabel('Amplitude');

% Plot the input signal again for eighth-order Chebyshev Type I filter
figure;
subplot(3,1,1);
plot(t, x);
title('Input Signal');
xlabel('Time (s)');
ylabel('Amplitude');

% Bode plot for eighth-order Chebyshev Type I filter
subplot(3,1,2);
g8 = tf(b8, a8);
bode(g8);
title('Bode Plot of Eighth-Order Chebyshev Type I Filter');
```

```
% Output signal of the eighth-order Chebyshev Type I filter
subplot(3,1,3);
plot(t, x_filt8);
title('Output Signal of Eighth-Order Chebyshev Type I Filter');
xlabel('Time (s)');
ylabel('Amplitude');
```

Lab4_part2_question1c

```
% Mamadou Kaba 27070179
% Part 2 Question 1 c)
clc; clear; close all;

Fs = 1800;
t = 0:1/Fs:1;


x = sin(100*t) + sin(2000*t) + sin(6000*t);

% Normalized cutoff frequencies for bandpass filter
Wn = [1900 2100]/(Fs*pi);

% Fourth-order Elliptic filter with 1 dB passband ripple and 60 dB stopband attenuation
[b4, a4] = ellip(4, 1, 60, Wn, 'bandpass');
x_filt4 = filter(b4, a4, x);

% Eighth-order Elliptic filter with 1 dB passband ripple and 60 dB stopband attenuation
[b8, a8] = ellip(8, 1, 60, Wn, 'bandpass');
x_filt8 = filter(b8, a8, x);

% Plot the input signal
figure;
subplot(3,1,1);
plot(t, x);
title('Input Signal');
xlabel('Time (s)');
ylabel('Amplitude');

% Bode plot for fourth-order Elliptic filter
subplot(3,1,2);
g4 = tf(b4, a4);
bode(g4);
title('Bode Plot of Fourth-Order Elliptic Filter');

% Output signal of the fourth-order Elliptic filter
subplot(3,1,3);
plot(t, x_filt4);
title('Output Signal of Fourth-Order Elliptic Filter');
xlabel('Time (s)');
ylabel('Amplitude');

% Plot the input signal again for eighth-order Elliptic filter
figure;
subplot(3,1,1);
plot(t, x);
title('Input Signal');
xlabel('Time (s)');
ylabel('Amplitude');
```

```
% Bode plot for eighth-order Elliptic filter
subplot(3,1,2);
g8 = tf(b8, a8);
bode(g8);
title('Bode Plot of Eighth-Order Elliptic Filter');

% Output signal of the eighth-order Elliptic filter
subplot(3,1,3);
plot(t, x_filt8);
title('Output Signal of Eighth-Order Elliptic Filter');
xlabel('Time (s)');
ylabel('Amplitude');
```

Lab4_part2_question1d

```
% Mamadou Kaba 27070179
% Part 2 Question 1 d)
clc; clear; close all;

Fs = 1800;
t = 0:1/Fs:1;

% Input signal
x = sin(100*t) + sin(2000*t) + sin(6000*t);

% Normalized cutoff frequencies for bandpass filter (adjusted for lower frequencies)
Wn = [1900 2100]/(Fs*pi);

% Eighth-order Butterworth filter
[b_butter, a_butter] = butter(8, Wn, 'bandpass');
x_filt_butter = filter(b_butter, a_butter, x);

% Eighth-order Chebyshev Type I filter with 1 dB passband ripple
[b_cheby1, a_cheby1] = cheby1(8, 1, Wn, 'bandpass');
x_filt_cheby1 = filter(b_cheby1, a_cheby1, x);

% Eighth-order Elliptic filter with 1 dB passband ripple and 60 dB stopband attenuation
[b_ellip, a_ellip] = ellip(8, 1, 60, Wn, 'bandpass');
x_filt_ellip = filter(b_ellip, a_ellip, x);

% Compute the FFT of the input and filtered signals
X = fft(x);
X_filt_butter = fft(x_filt_butter);
X_filt_cheby1 = fft(x_filt_cheby1);
X_filt_ellip = fft(x_filt_ellip);

% Frequency vector
f = (0:length(X)-1)*Fs/length(X);

% Plot the FFT of the input signal and filtered signals
figure;
subplot(4,1,1);
plot(f, abs(X));
title('FFT of Input Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
```

```
subplot(4,1,2);
plot(f, abs(X_filt_butter));
title('FFT of Output Signal (Butterworth)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

subplot(4,1,3);
plot(f, abs(X_filt_cheby1));
title('FFT of Output Signal (Chebyshev Type I)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

subplot(4,1,4);
plot(f, abs(X_filt_ellip));
title('FFT of Output Signal (Elliptic)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
```

Lab4_part2_question2

```
% Mamadou Kaba 27070179
% Part 2 Question 2
clc; clear; close all;

% Sampling frequency and time vector
Fs = 8192;
t = 0:1/Fs:3;

% Create the signal
signal = sin(2*pi*200*t).*(t >= 0 & t < 3) + sin(2*pi*330*t).*(t >= 1 & t < 3) +
sin(2*pi*480*t).*(t >= 2 & t < 3);

% Plot the time domain signal
figure;
subplot(2,1,1);
plot(t, signal);
title('Time Domain Signal');
xlabel('Time (s)');
ylabel('Amplitude');

% Compute the FFT of the signal
L = length(signal);
Y = fft(signal);
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
f = Fs*(0:(L/2))/L;

% Plot the frequency domain signal
subplot(2,1,2);
plot(f, P1);
title('Frequency Domain Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Design a bandstop filter to remove the 330 Hz component
```

```matlab
Wn = [325 335]/(Fs/2);
[b, a] = butter(4, Wn, 'stop');

% Plot the filter response
figure;
freqz(b, a, 2048, Fs);
title('Bandstop Filter Designed to Remove 330 Hz Component');

% Filter the signal
filtered_signal = filter(b, a, signal);

% Plot the original and filtered signals in time domain
figure;
subplot(2,1,1);
plot(t, signal);
title('Original Signal');
xlabel('Time (s)');
ylabel('Amplitude');

subplot(2,1,2);
plot(t, filtered_signal);
title('Filtered Signal');
xlabel('Time (s)');
ylabel('Amplitude');

% Compute the FFT of the filtered signal
Y_filtered = fft(filtered_signal);
P2_filtered = abs(Y_filtered/L);
P1_filtered = P2_filtered(1:L/2+1);
P1_filtered(2:end-1) = 2*P1_filtered(2:end-1);

% Plot the frequency domain of the original and filtered signals
figure;
subplot(2,1,1);
plot(f, P1);
title('FFT of Original Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

subplot(2,1,2);
plot(f, P1_filtered);
title('FFT of Filtered Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');


soundsc(signal, Fs);
pause(4);
soundsc(filtered_signal, Fs);
```

Lab4_part2_question3A

```
% Mamadou Kaba 27070179
% Part 2 Question 3 A

% Load the original and downsampled signals from the workspace
original_audio = out.original_audio;
audio_down_sampled_8 = out.audio_down_sampled_8;
audio_down_sampled_16 = out.audio_down_sampled_16;

% Convert timeseries to double and reshape to 1D array
original_audio = double(original_audio.Data(:));
audio_down_sampled_8 = double(audio_down_sampled_8.Data(:));
audio_down_sampled_16 = double(audio_down_sampled_16.Data(:));

% Define sampling frequency
Fs = 44100;

% Time vector for plotting (first second)
t = (0:Fs-1) / Fs;

% Plot the original signal in time domain
figure;
subplot(6,1,1);
plot(t, original_audio(1:Fs));
title('Original Audio Signal (Time Domain)');
xlabel('Time (s)');
ylabel('Amplitude');

% Compute and plot FFT of the original signal
L = length(original_audio);
f = (-L/2:L/2-1)/L;
Y_orig = fftshift(fft(original_audio));

subplot(6,1,2);
plot(f, abs(Y_orig)/max(abs(Y_orig)));
title('FFT of Original Audio Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Adjust time vector for the downsampled signal (k=8)
t_down_8 = (0:Fs/8-1) / (Fs/8);

% Plot the downsampled signal (k=8) in time domain
subplot(6,1,3);
plot(t_down_8, audio_down_sampled_8(1:Fs/8));
title('Downsampled Audio Signal (k=8) (Time Domain)');
xlabel('Time (s)');
ylabel('Amplitude');

% Compute and plot FFT of the downsampled signal (k=8)
L_down_8 = length(audio_down_sampled_8);
f_down_8 = (-L_down_8/2:L_down_8/2-1)/L_down_8;
Y_down_8 = fftshift(fft(audio_down_sampled_8));

subplot(6,1,4);
plot(f_down_8, abs(Y_down_8)/max(abs(Y_down_8)));
title('FFT of Downsampled Audio Signal (k=8)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
```

```
% Adjust time vector for the downsampled signal (k=16)
t_down_16 = (0:Fs/16-1) / (Fs/16);

% Plot the downsampled signal (k=16) in time domain
subplot(6,1,5);
plot(t_down_16, audio_down_sampled_16(1:Fs/16));
title('Downsampled Audio Signal (k=16) (Time Domain)');
xlabel('Time (s)');
ylabel('Amplitude');

% Compute and plot FFT of the downsampled signal (k=16)
L_down_16 = length(audio_down_sampled_16);
f_down_16 = (-L_down_16/2:L_down_16/2-1)/L_down_16;
Y_down_16 = fftshift(fft(audio_down_sampled_16));

subplot(6,1,6);
plot(f_down_16, abs(Y_down_16)/max(abs(Y_down_16)));
title('FFT of Downsampled Audio Signal (k=16)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
```

Lab4_part2_question3B

```
% Mamadou Kaba 27070179
% Part 2 Question 3 B

% Load the original and upsampled signals from the workspace
original_audio = out.original_audio;
audio_up_sampled_8 = out.audio_up_sampled_8;
audio_up_sampled_16 = out.audio_up_sampled_16;

% Convert timeseries to double and reshape to 1D array
original_audio = double(original_audio.Data(:));
audio_up_sampled_8 = double(audio_up_sampled_8.Data(:));
audio_up_sampled_16 = double(audio_up_sampled_16.Data(:));

% Define sampling frequency
Fs = 44100;

% Time vector for plotting (first second)
t = (0:Fs-1) / Fs;

% Plot the original signal in time domain
figure;
subplot(6,1,1);
plot(t, original_audio(1:Fs));
title('Original Audio Signal (Time Domain)');
xlabel('Time (s)');
ylabel('Amplitude');

% Compute and plot FFT of the original signal
L = length(original_audio);
f = (-L/2:L/2-1)/L;
Y_orig = fftshift(fft(original_audio));

subplot(6,1,2);
```

```matlab
plot(f, abs(Y_orig)/max(abs(Y_orig)));
title('FFT of Original Audio Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Adjust time vector for the upsampled signal (L=8)
t_up_8 = (0:Fs*8-1) / (Fs*8);

% Plot the upsampled signal (L=8) in time domain
subplot(6,1,3);
plot(t_up_8, audio_up_sampled_8(1:Fs*8));
title('Upsampled Audio Signal (L=8) (Time Domain)');
xlabel('Time (s)');
ylabel('Amplitude');

% Compute and plot FFT of the upsampled signal (L=8)
L_up_8 = length(audio_up_sampled_8);
f_up_8 = (-L_up_8/2:L_up_8/2-1)/L_up_8;
Y_up_8 = fftshift(fft(audio_up_sampled_8));

subplot(6,1,4);
plot(f_up_8, abs(Y_up_8)/max(abs(Y_up_8)));
title('FFT of Upsampled Audio Signal (L=8)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Adjust time vector for the upsampled signal (L=16)
t_up_16 = (0:Fs*16-1) / (Fs*16);

% Plot the upsampled signal (L=16) in time domain
subplot(6,1,5);
plot(t_up_16, audio_up_sampled_16(1:Fs*16));
title('Upsampled Audio Signal (L=16) (Time Domain)');
xlabel('Time (s)');
ylabel('Amplitude');

% Compute and plot FFT of the upsampled signal (L=16)
L_up_16 = length(audio_up_sampled_16);
f_up_16 = (-L_up_16/2:L_up_16/2-1)/L_up_16;
Y_up_16 = fftshift(fft(audio_up_sampled_16));

subplot(6,1,6);
plot(f_up_16, abs(Y_up_16)/max(abs(Y_up_16)));
title('FFT of Upsampled Audio Signal (L=16)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
```