Concordia University
Engineering and Computer Science

LABORATORY REPORT

ELEC 342 Discrete Time Signals and Systems

FALL 2023

Course:	ELEC 342	**Lab Section:**	UN-X
Experiment No.:	4	**Date Performed:**	2023 – 11 –07 (YYYY – MM – DD)

Experiment Title: Introduction to Simulink and Filter Design using MATLAB.

Name:	BAYAN ALSALEM	**ID No.:**	40105034

I certify that this submission is my original work and meets the Faculty's Expectations of Originality

Signature: [illegible] **Date:** 2023 – 11 – 19 (YYYY – MM – DD)

Table of Contents

Objectives 3

Theory 3

Tasks / Results / Discussion 3

Pre Lab 3

- Question 1 3
- Question 2 4
- Question 3 7

Lab Problems 10

- Question 1 10
- Question 2 23
- Question 3 27

Objectives

For this experiment, the main objective is to gain further knowledge on analog and digital filter designs methods within the MATLAB environment. Furthermore, there are two important tools that will be used to analyze filter design methods and to simulate and design signal systems which are Filter Design & Analysis Tool (FDAT) and Simulink respectively.

Theory

Filters are systems that are used to remove certain parts of a signal at desired frequencies. There are four filters that are commonly used and investigated; the low pass filter, high pass filter, ideal band pass filter and band stop filter. Filters allow the modification of signal to remove specific frequency components from the output signal without disturbing the other components of a signal. Filters come in various types to remove different frequency ranges in a signal and have different curves that affect the degree of filtering. Tools such as FDATool and Simulink can be used to construct digital filters to be applied to signals while MATLAB plots and scopes can demonstrate the effect on the output signal.

Tasks / Results / Discussion

Pre Lab

Question 1

Use the audioread command to read the audio file "lab_5_Audio_1.wav"
[y,fs]=audioread('lab_5_Audio_1.wav');

After reading, listen to

a) sound(y,fs)

This command played the original sound at the designated sampling frequency fs.

b) sound(0.25*y, fs) and sound (4*y, fs)

These commands altered the magnitude of the signals, decreasing and increasing the volume of the sound, respectively.

Explain in what way these are different from what you heard in the part (a).

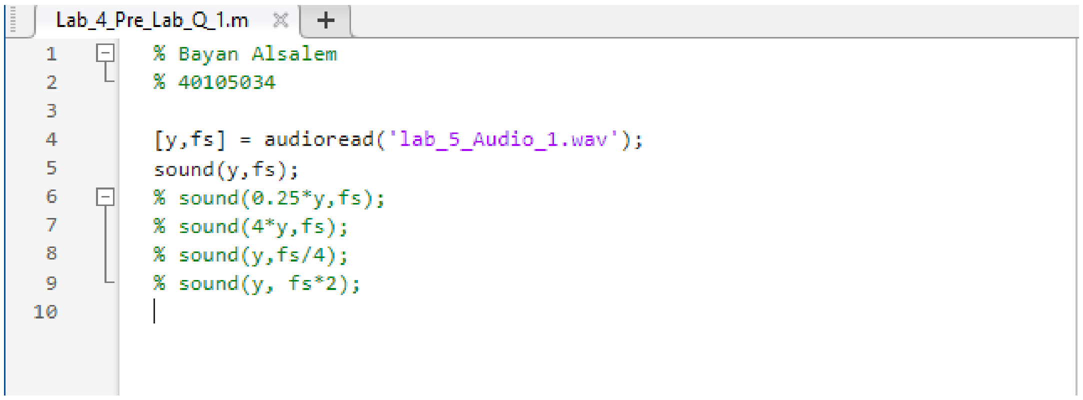
- **Sound(0.25*y, fs):** Amplitude scaling of the original audio signal by 1/4 with a sampling frequency fs. The new amplitude is decreased because of the 1/4. Hence the sound is softer.
- **Sound (4*y, fs):** amplitude scaling of the original audio signal multiplied by 4 with a sampling frequency fs, the sound is louder.

c) sound(y, fs/2) and sound(y, fs*2)

These commands increased and decreased the playback frequency, making the sound play slower and faster respectively.

Explain how these are different from what you heard in the part (a).

- **Sound(y, fs/2):** the sampling frequency is divided by 2, hence it decreases the number of samples which leads to a longer interval between the samples. Hence the signal is slower.
- **Sound(y, fs*2):** we have a higher sample rate, which leads to a better-quality audio reproduction since we are taking more samples per second, shorter interval between the samples, hence the signal is faster.

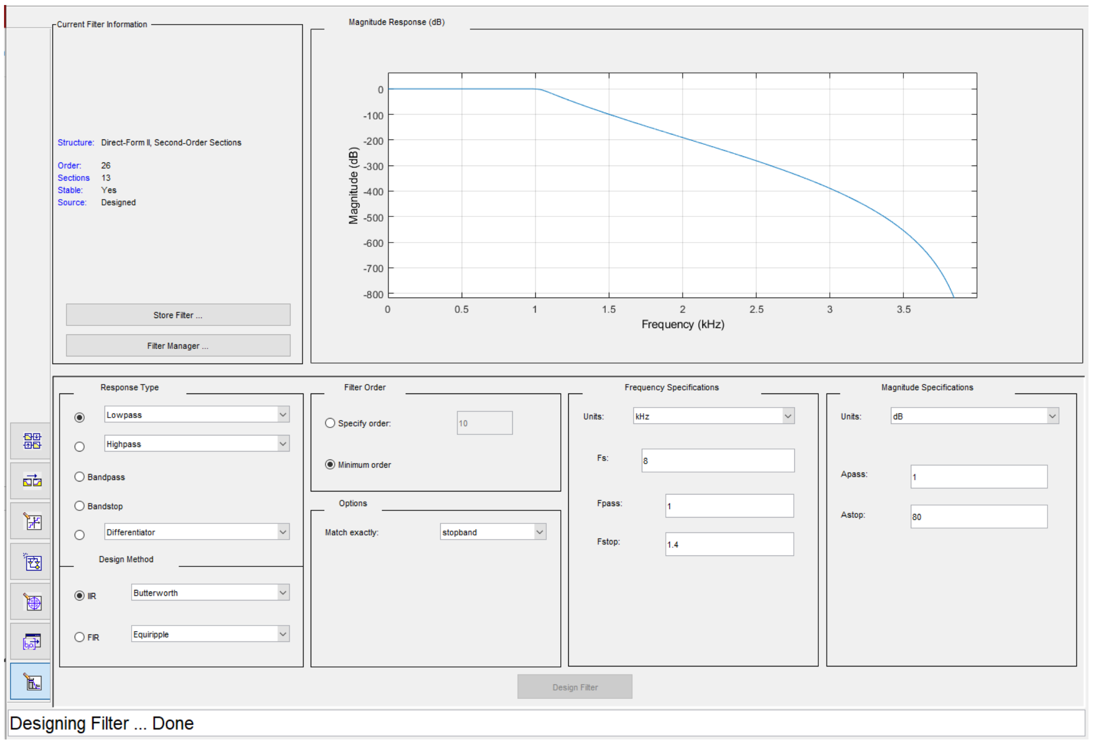


```
% Bayan Alsalem
% 40105034

[y,fs] = audioread('lab_5_Audio_1.wav');
sound(y,fs);
% sound(0.25*y,fs);
% sound(4*y,fs);
% sound(y,fs/4);
% sound(y, fs*2);
```

Question 2

Designing the Filters Using FDATOOL, perform the following tasks, assuming a sampling rate of 8 kHz:

I. Design a minimum order, stable, lowpass *Butterworth* filter with a passband frequency of 1 kHz and a stopband frequency of 1.4 kHz. Make the attenuation 1 dB at the passband frequency and 80 dB at the stopband frequency.



Current Filter Information

Structure: Direct-Form II, Second-Order Sections

Order:	26
Sections	13
Stable:	Yes
Source:	Designed

Response Type: [✓ Lowpass] Highpass Bandpass Bandstop Differentiator

Design Method: [✓ IIR] Butterworth; FIR Equiripple

Filter Order: Specify order: 10; [✓ Minimum order]

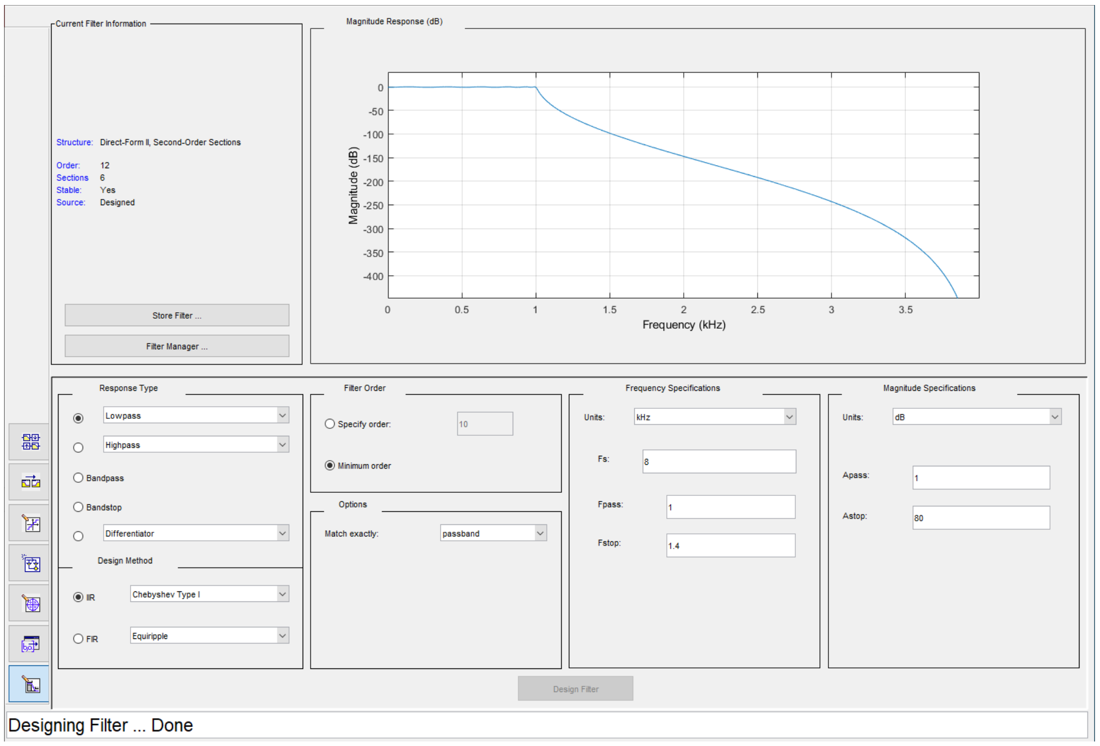
Options: Match exactly: stopband

Frequency Specifications: Units: kHz; Fs: 8; Fpass: 1; Fstop: 1.4

Magnitude Specifications: Units: dB; Apass: 1; Astop: 80

Designing Filter ... Done

II. Design a minimum order, stable, lowpass *Chebyshev* Type I filter with the same specifications as the Butterworth filter.



Current Filter Information

Structure: Direct-Form II, Second-Order Sections

Order:	12
Sections	6
Stable:	Yes
Source:	Designed

Response Type: [✓ Lowpass] Highpass Bandpass Bandstop Differentiator

Design Method: [✓ IIR] Chebyshev Type I; FIR Equiripple

Filter Order: Specify order: 10; [✓ Minimum order]

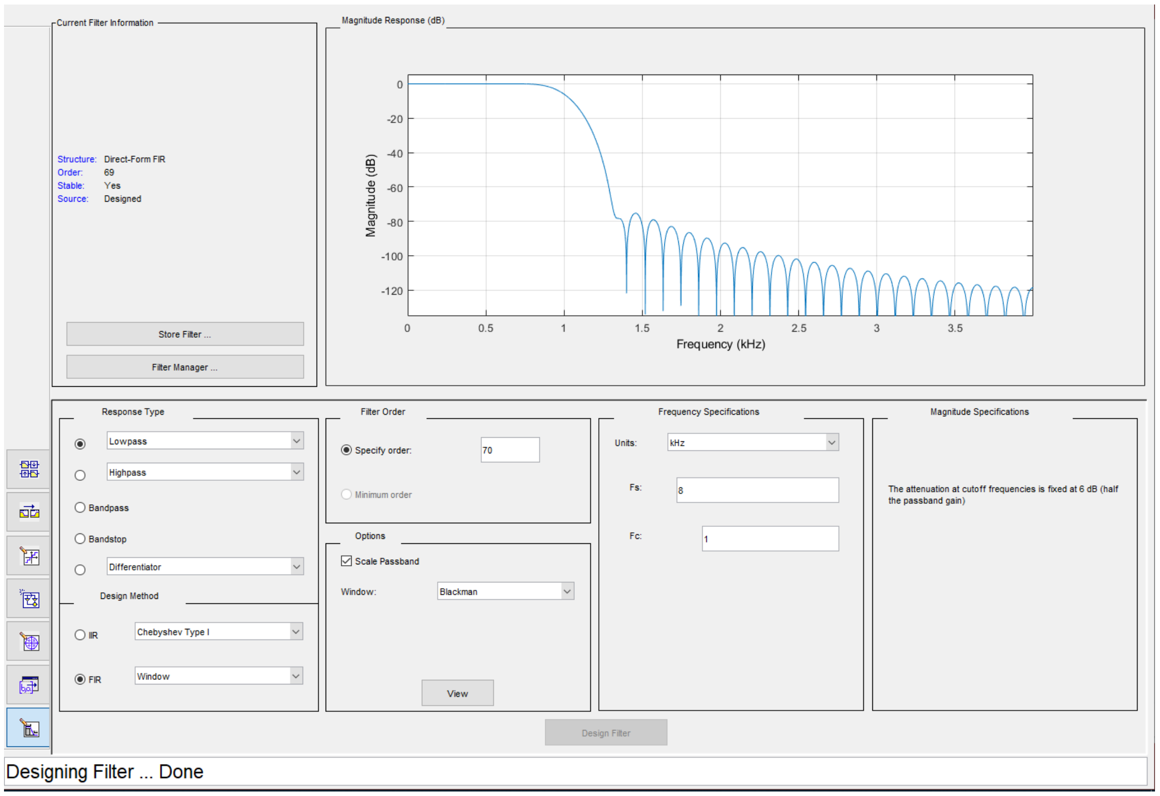
Options: Match exactly: passband

Frequency Specifications: Units: kHz; Fs: 8; Fpass: 1; Fstop: 1.4

Magnitude Specifications: Units: dB; Apass: 1; Astop: 80

Designing Filter ... Done

III. Design a lowpass *FIR* filter using the Blackman Window with a cutoff frequency of 1 kHz. Specify the order of the filter such that the first minimum in the stopband (preceding the first lobe) is as close to 1.4 kHz as possible without exceeding it.



Current Filter Information

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

Response Type
[✓ Lowpass]
Highpass
Bandpass
Bandstop
Differentiator

Design Method
IIR Chebyshev Type I
[✓ FIR] Window

Filter Order
[✓ Specify order:] 70
Minimum order

Options
[✓ Scale Passband]
Window: Blackman

Frequency Specifications
Units: kHz
Fs: 8
Fc: 1

Magnitude Specifications
The attenuation at cutoff frequencies is fixed at 6 dB (half the passband gain)

Designing Filter ... Done

Now answer the following

a) What is the order of the lowpass Butterworth filter you designed?

The order number is 26.

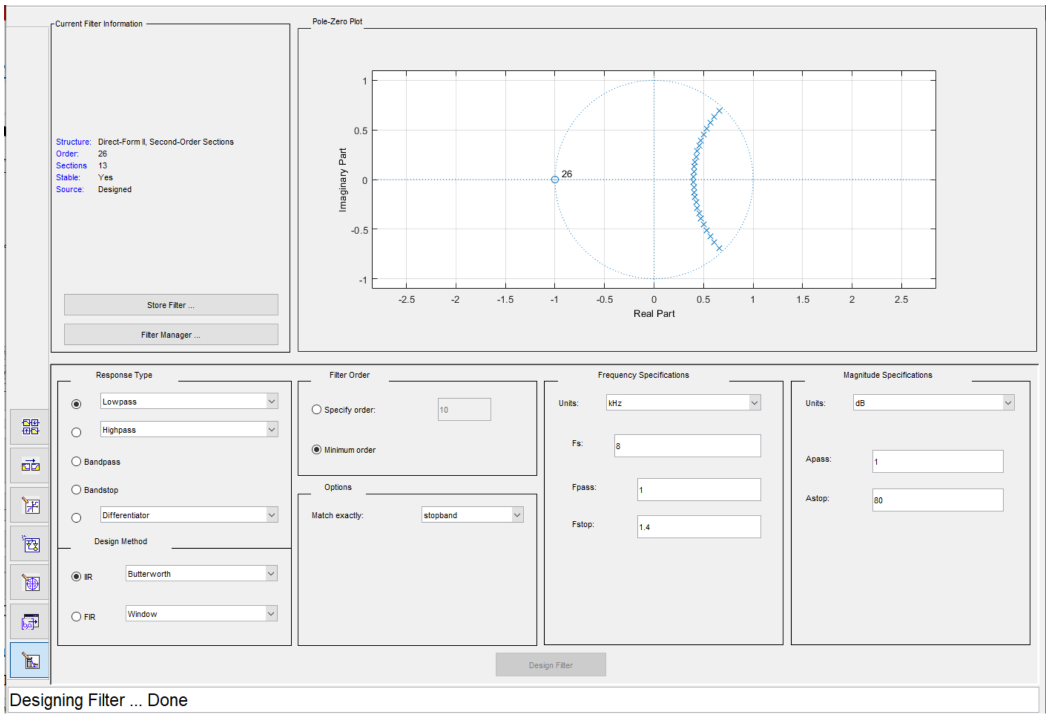
b) What is the order of the lowpass Chebyshev Type I filter you designed?

The order number is 12.

c) Compare the implementation cast of each of the 3 filters. Do you see how inefficient the windowing technique is? How much more expensive in terms of memory is the windowing technique from the best IIR filter?

The Blackman window filter has a significantly higher order resulting in a much larger quantity of multiplication operations and data points, becoming very computationally costly on system resources.

d) For the filter you designed in Part I, round the filter coefficients (b,a) to the nearest integer value.



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

Response Type: [✓ Lowpass] Highpass Bandpass Bandstop Differentiator

Design Method: [✓ IIR] Butterworth; FIR Window

Filter Order: Specify order: 10 [✓ Minimum order]

Options: Match exactly: stopband

Frequency Specifications: Units: kHz; Fs: 8; Fpass: 1; Fstop: 1.4

Magnitude Specifications: Units: dB; Apass: 1; Astop: 80

Designing Filter ... Done

Question 3

Creating a Noisy Signal using Simulink

You will create a sinusoid single corrupted by high frequency noise. Both the sine wave and the noise will be discrete; thus, your whole Simulink model will be discrete (this is purposely done to simplify things). Here's how to create such a signal:

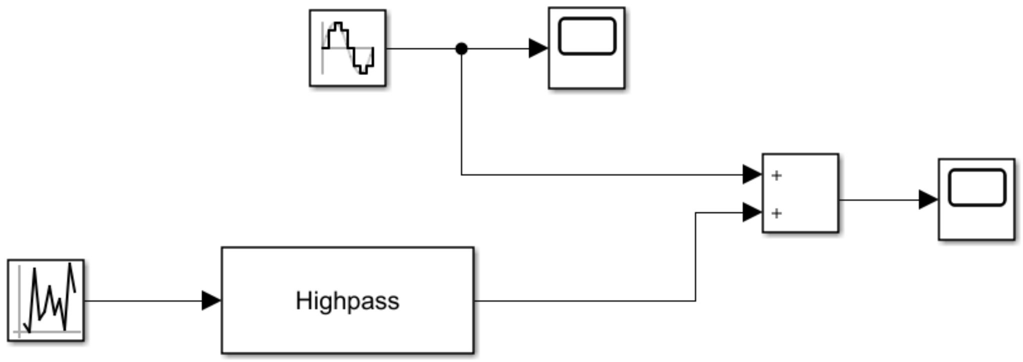
I. Add a Sine Wave block to your model. Set the frequency to 100 Hz and the Sample time to 1/8000.

II. Add a Uniform Random Number block to your model. Set Sample time to 1/8000 and Maximum and Minimum to 5 and -5, respectively. As its name suggests, this block outputs random numbers, which for our purposes is a good model for noise.

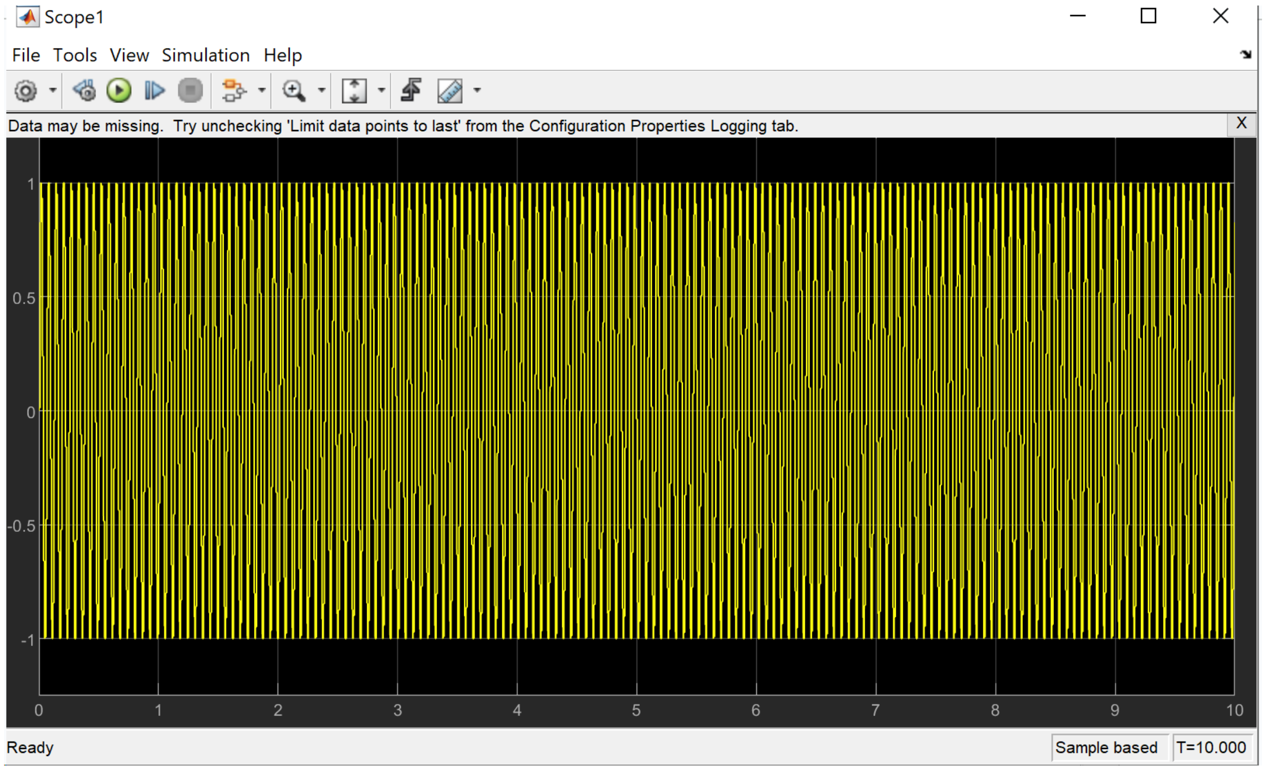
III. We want to restrict our noise to only have high frequency components. To do this, we are going to send it through a highpass filter. Add a Highpass Filter block and open the Block Parameters. Under Filter specifications, set Impulse response to IIR and Order mode to Minimum. Under Frequency specifications, set Frequency units to Hz, Input Fs to 8000, Fstop to 3600, and Fpass to 3800. Under Magnitude specifications, set Magnitude units to dB, Astop to 60, and Apass to 1. Finally, under Algorithm, set Design method to Elliptic and Structure to Direct-form II SOS. Right now, all that you have to understand about this filter is that it only passes frequencies above 3800 Hz (Fpass); anything below 3600 Hz (Fstop) is attenuated by at least 60 dB (the region between these two frequencies is a transition region).

IV. Send the random number signal from Step II through the highpass filter and add it to the sine wave you made in Step I.

V. To view the noisy sine wave signal, use Scope blocks. Also, use another scope to view the original sine wave. Use the Scope parameters to set ‘limit data points to last’ parameter and to save the data to your workspace.

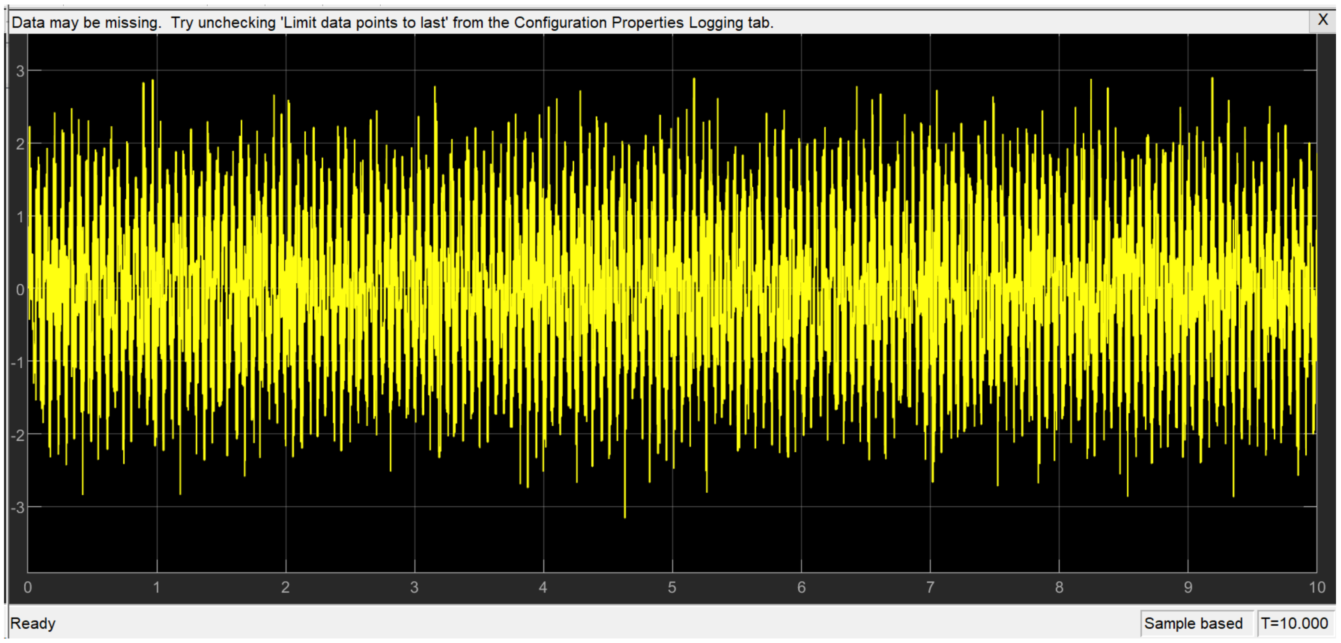


The sine wave:



Data may be missing. Try unchecking 'Limit data points to last' from the Configuration Properties Logging tab.

The filtered signal


Data may be missing. Try unchecking 'Limit data points to last' from the Configuration Properties Logging tab.

Lab Problems

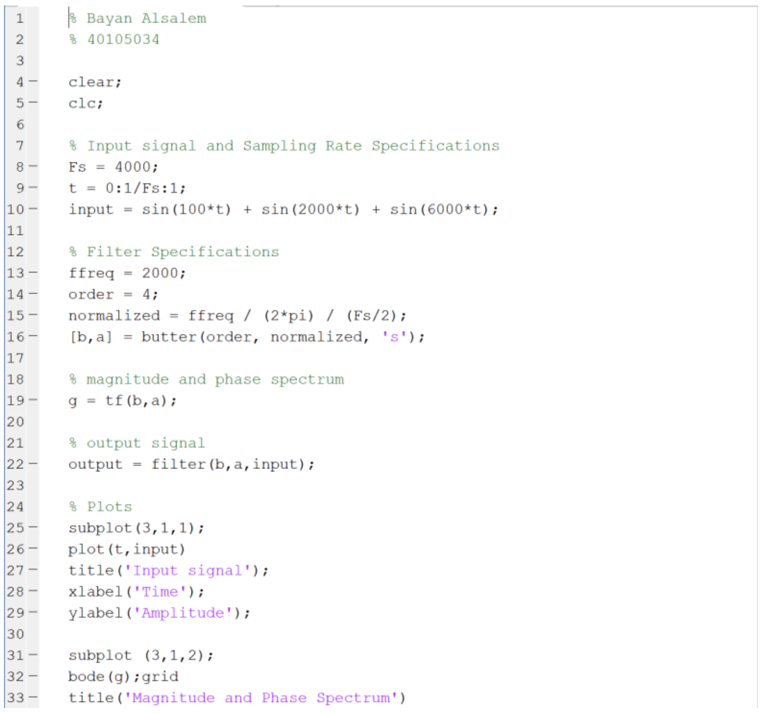
Question 1

Consider the input signal:

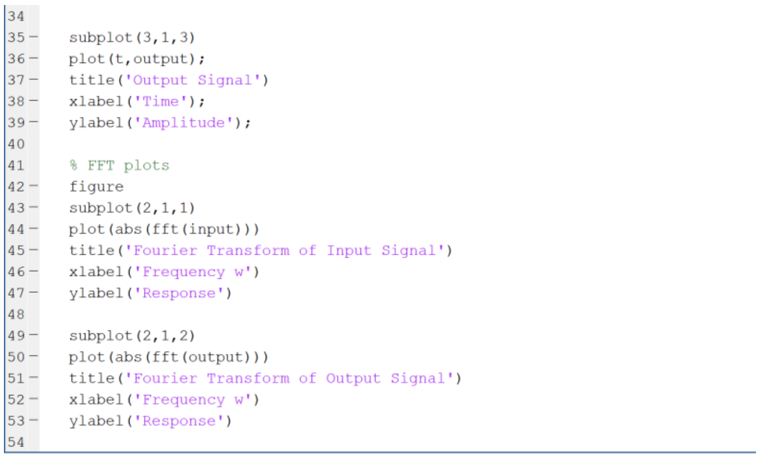
$$x\,(t)=sin\,(100\,t)\,+sin(2000t)\,+sin(6000t)$$

We need to pass the term $sin(2000t)$ and to attenuate the other terms.

a) Design a digital fourth and eighth order IIR Butterworth filters. Plot the input signal along with the magnitude response of the filter and its output signal for each filter.

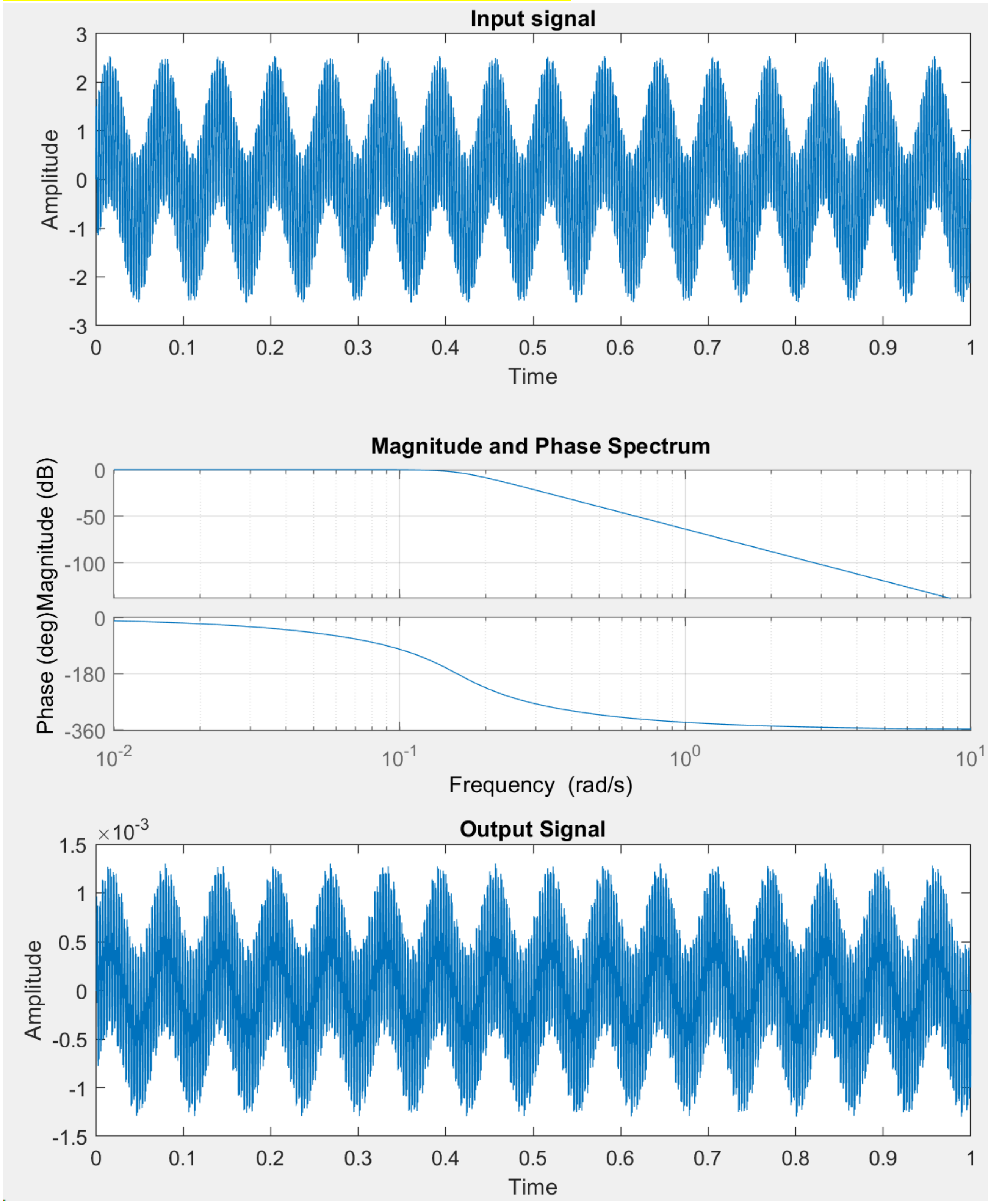


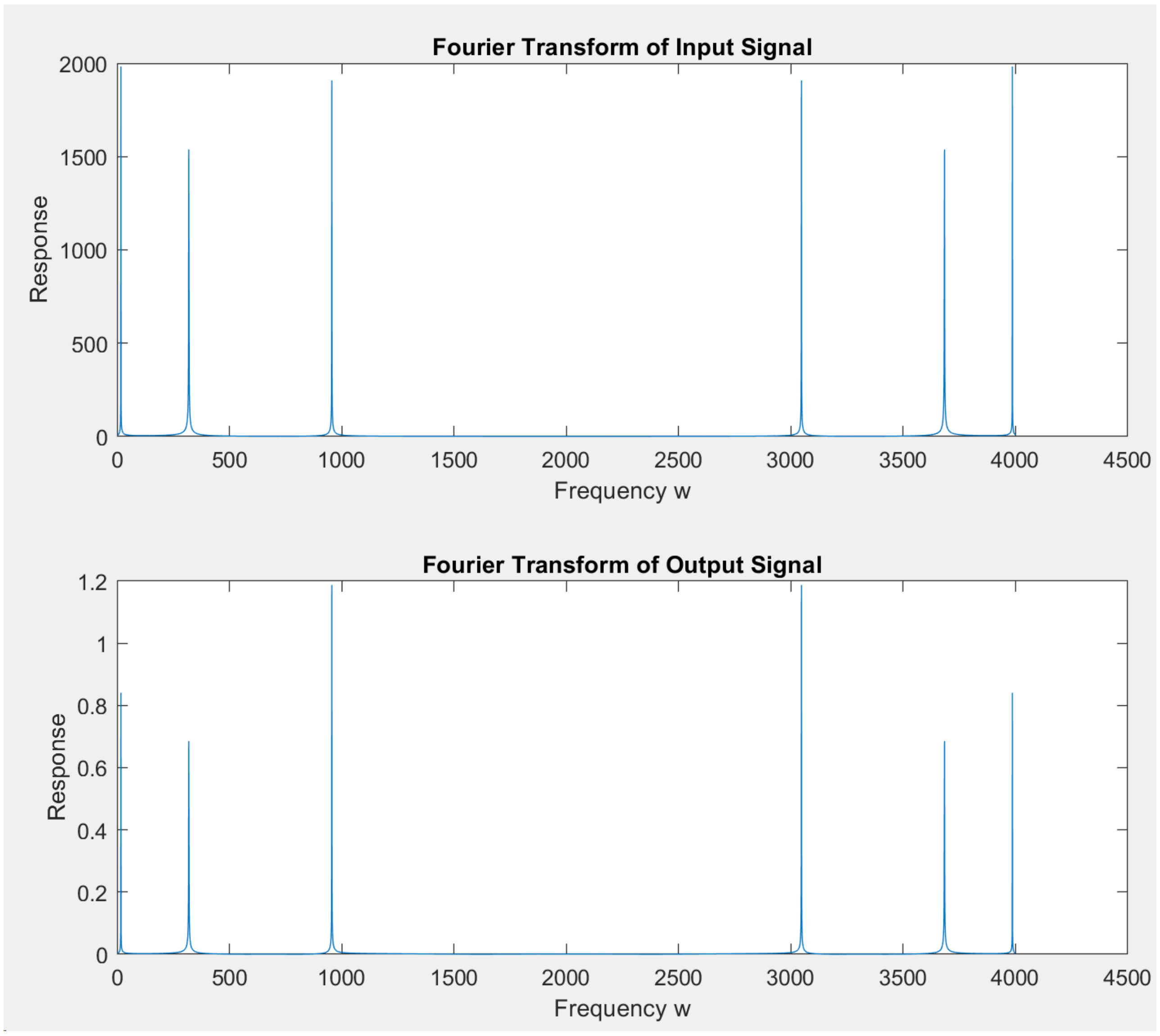
```
% Bayan Alsalem
% 40105034

clear;
clc;

% Input signal and Sampling Rate Specifications
Fs = 4000;
t = 0:1/Fs:1;
input = sin(100*t) + sin(2000*t) + sin(6000*t);

% Filter Specifications
ffreq = 2000;
order = 4;
normalized = ffreq / (2*pi) / (Fs/2);
[b,a] = butter(order, normalized, 's');

% magnitude and phase spectrum
g = tf(b,a);

% output signal
output = filter(b,a,input);

% Plots
subplot(3,1,1);
plot(t,input)
title('Input signal');
xlabel('Time');
ylabel('Amplitude');

subplot (3,1,2);
bode(g);grid
title('Magnitude and Phase Spectrum')
```



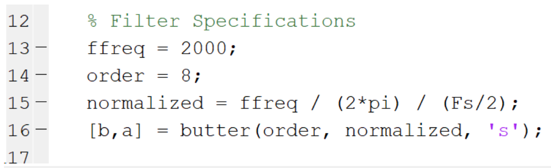
```

subplot(3,1,3)
plot(t,output);
title('Output Signal')
xlabel('Time');
ylabel('Amplitude');

% FFT plots
figure
subplot(2,1,1)
plot(abs(fft(input)))
title('Fourier Transform of Input Signal')
xlabel('Frequency w')
ylabel('Response')

subplot(2,1,2)
plot(abs(fft(output)))
title('Fourier Transform of Output Signal')
xlabel('Frequency w')
ylabel('Response')

```

Plots of the fourth order IIR Butterworth filter

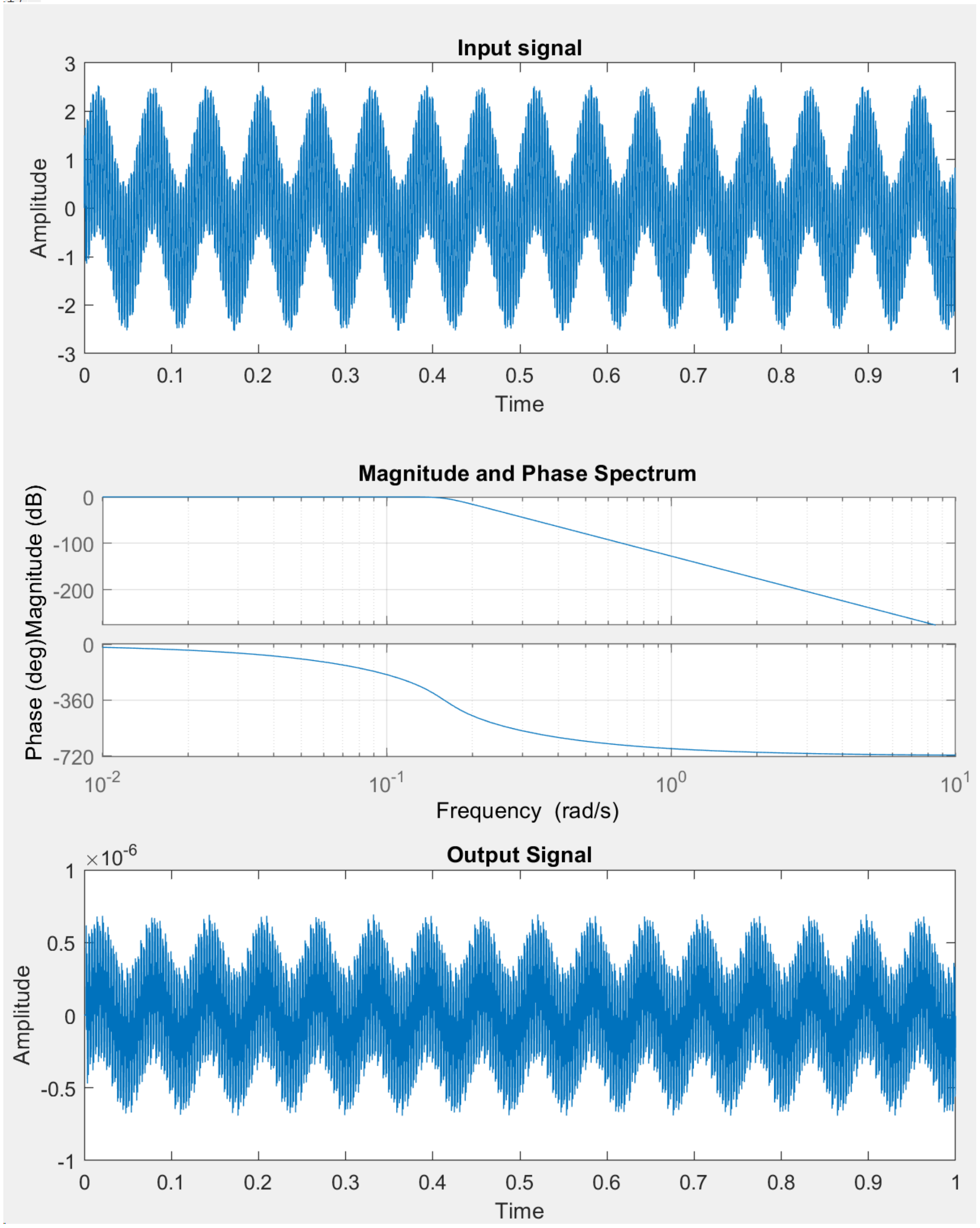


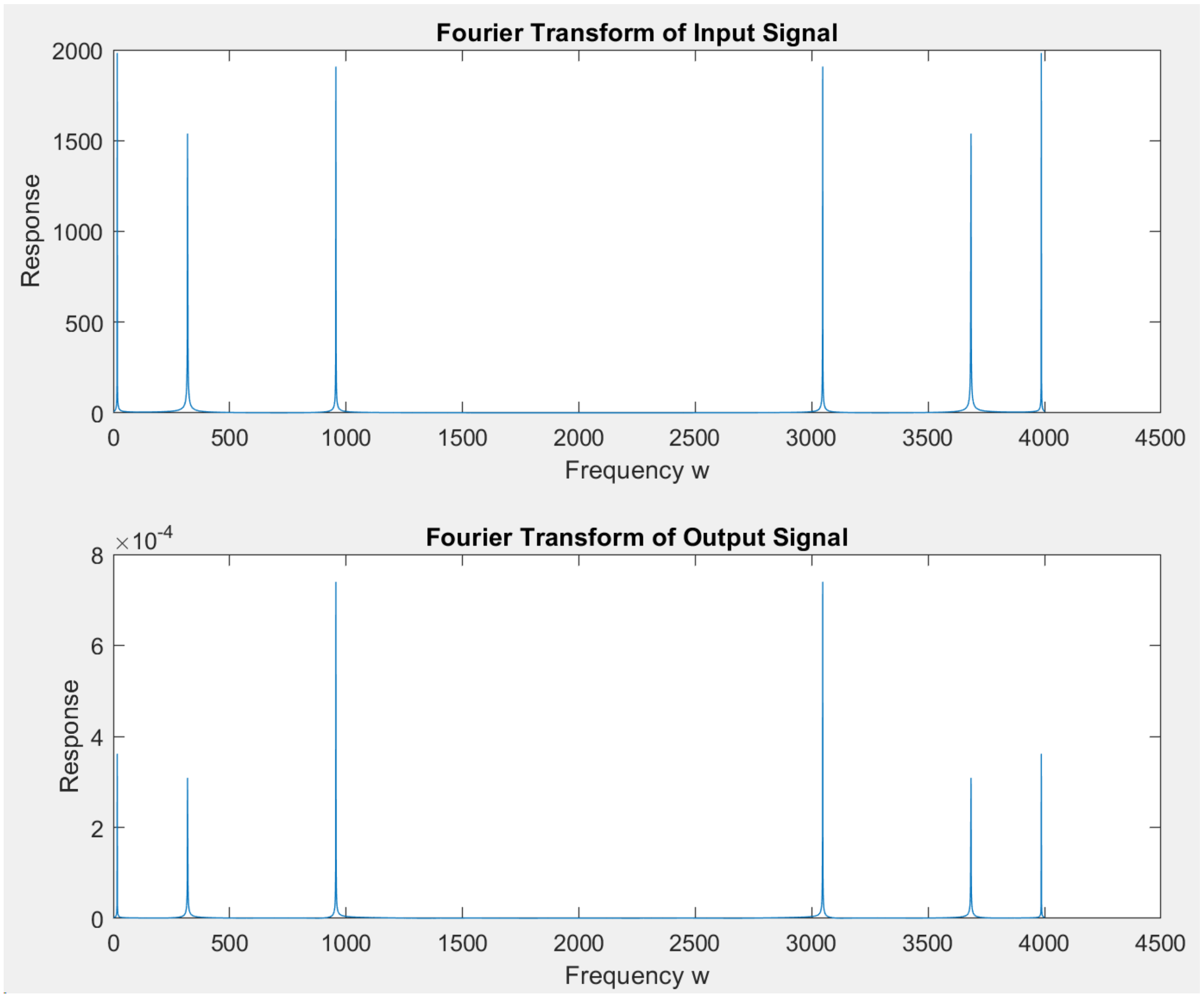

Fourier Transform of Input Signal
Response
Frequency w
Fourier Transform of Output Signal
Response
Frequency w

Plots of the eight order IIR Butterworth filter



```
% Filter Specifications
ffreq = 2000;
order = 8;
normalized = ffreq / (2*pi) / (Fs/2);
[b,a] = butter(order, normalized, 's');
```

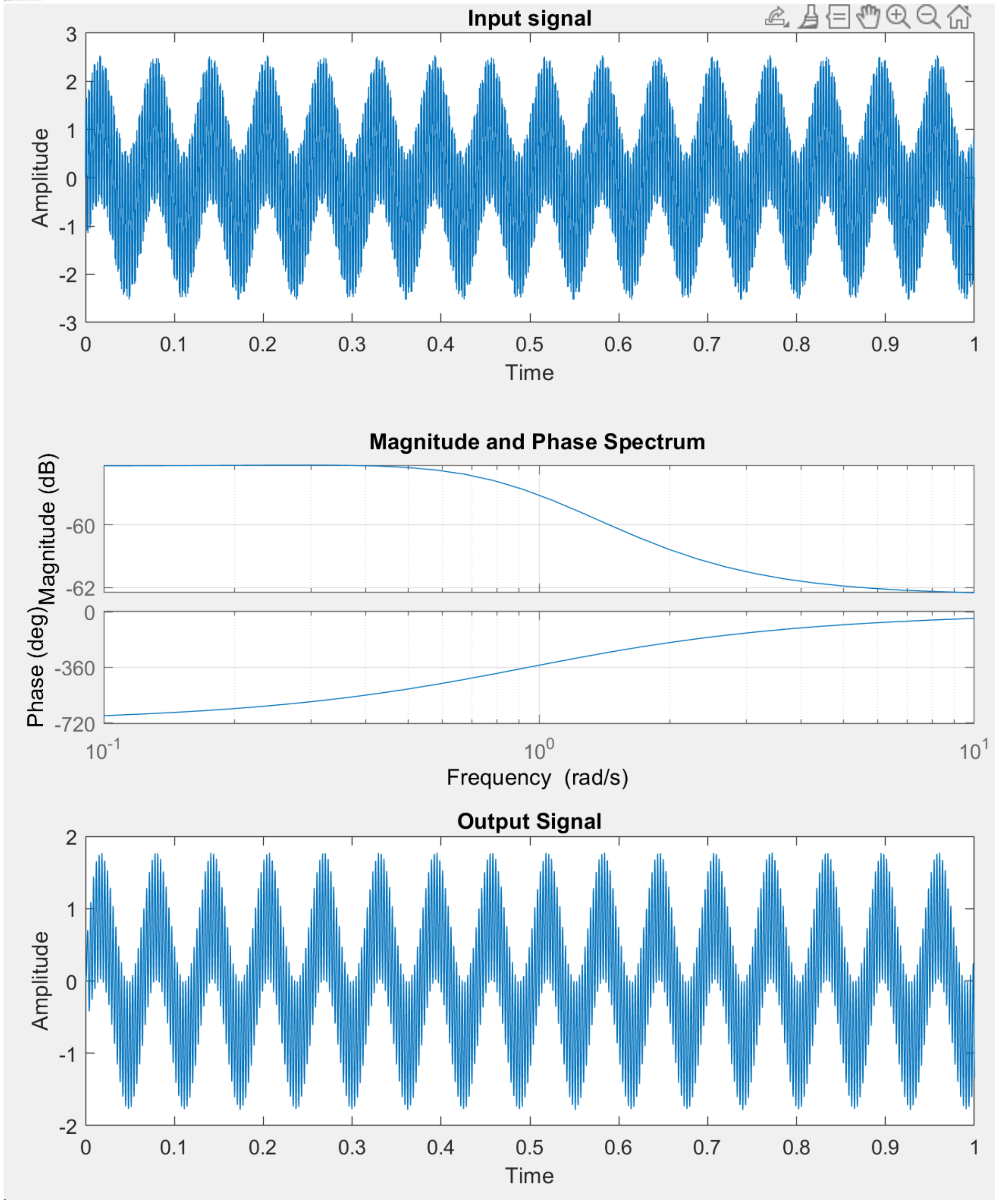


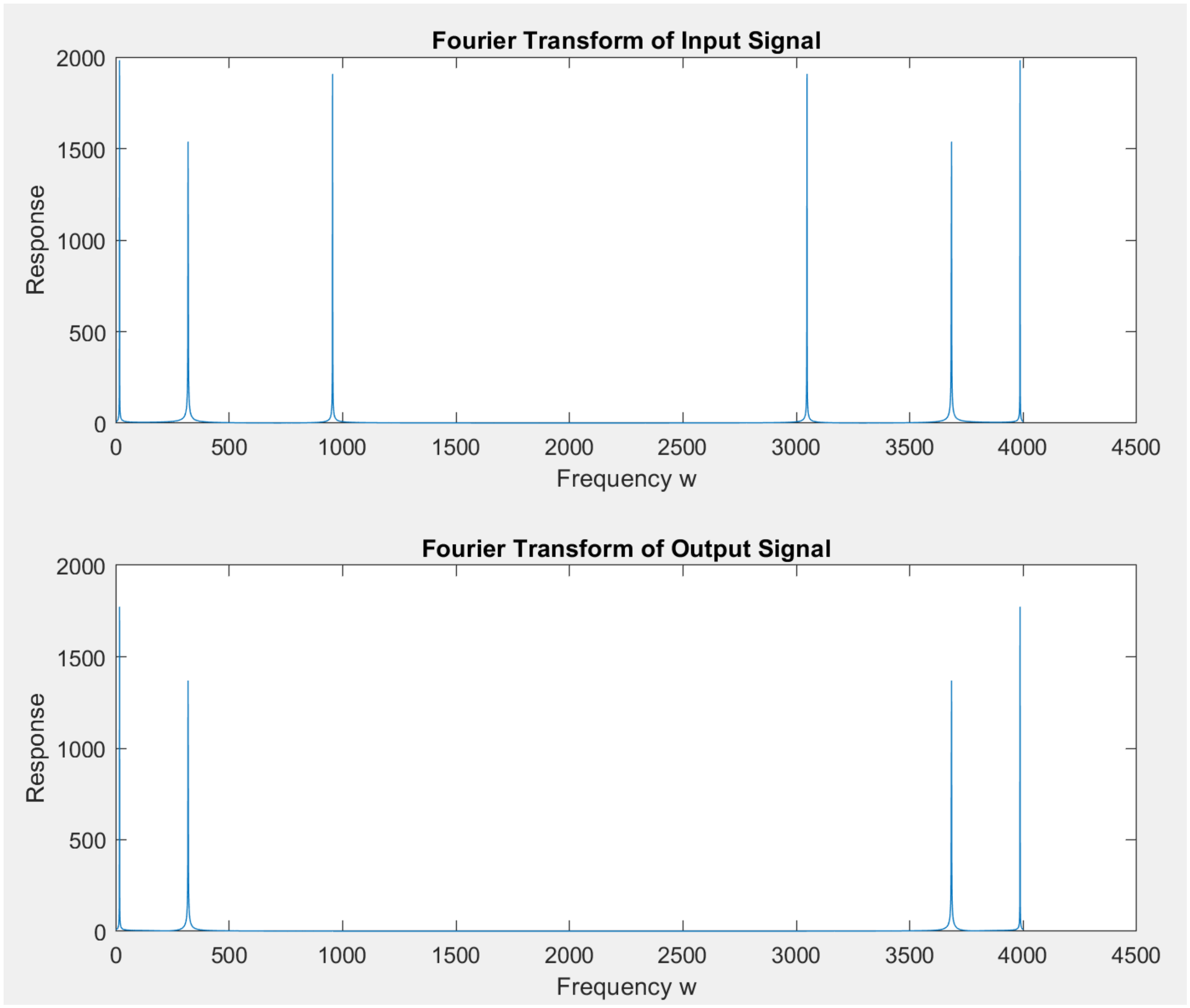

Fourier Transform of Input Signal
Response
Frequency w
Fourier Transform of Output Signal
$\times 10^{-4}$
Response
Frequency w

b) Design a digital fourth and eighth order IIR Chebyshev type I filters. Plot the input signal along with the magnitude response of the filter and its output signal for each filter. (consider $Rp = 1dB$).

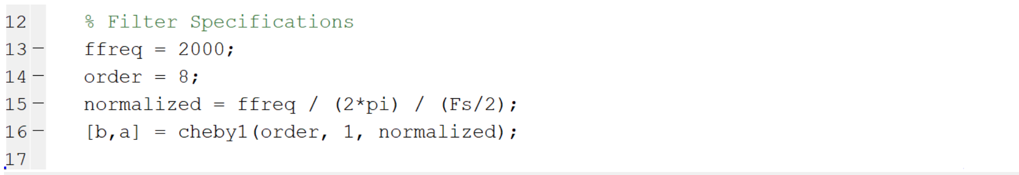
Plots of the fourth order IIR Chebyshev type 1 filter

```
% Filter Specifications
ffreq = 2000;
order = 4;
normalized = ffreq / (2*pi) / (Fs/2);
[b,a] = cheby1(order, 1, normalized);
```

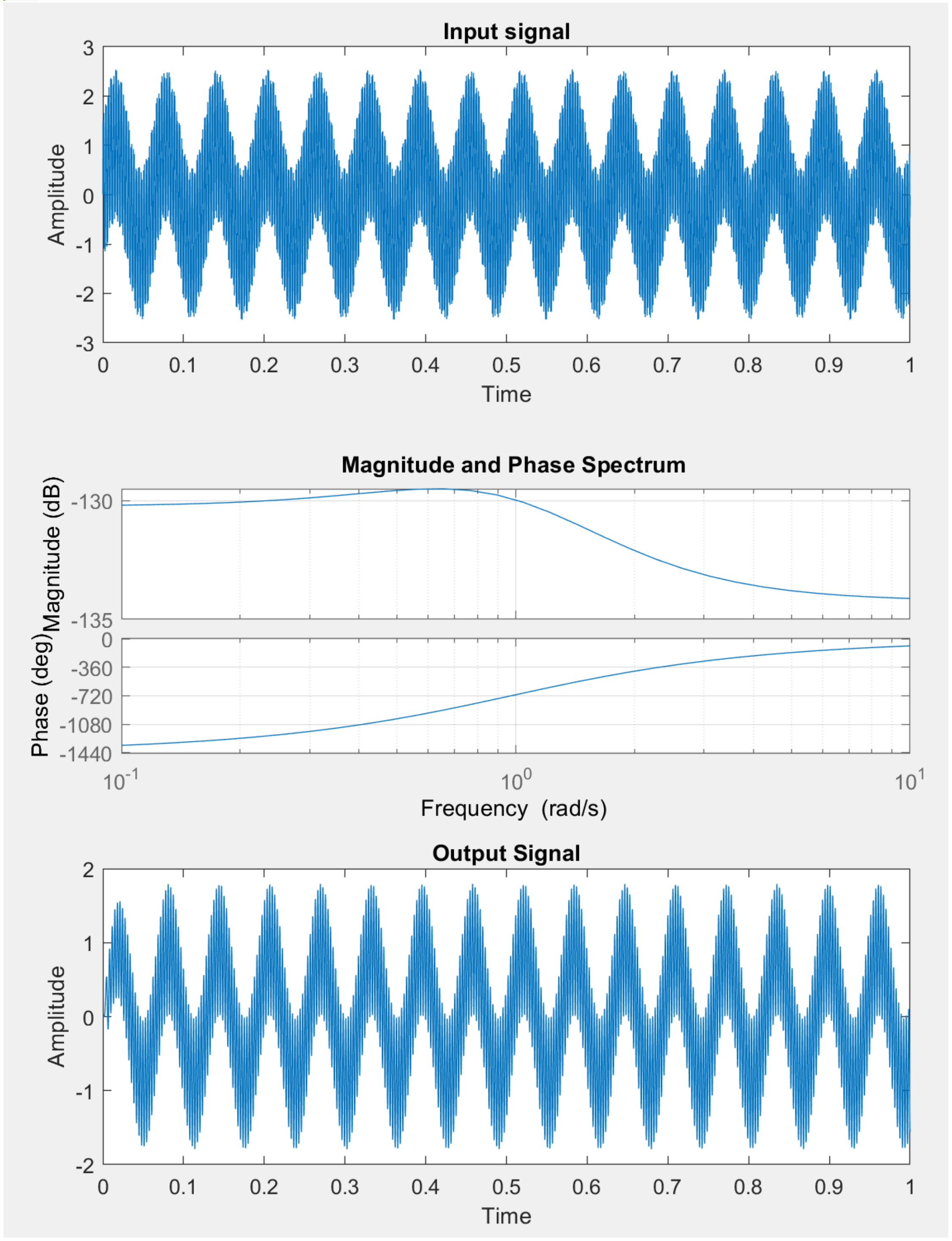


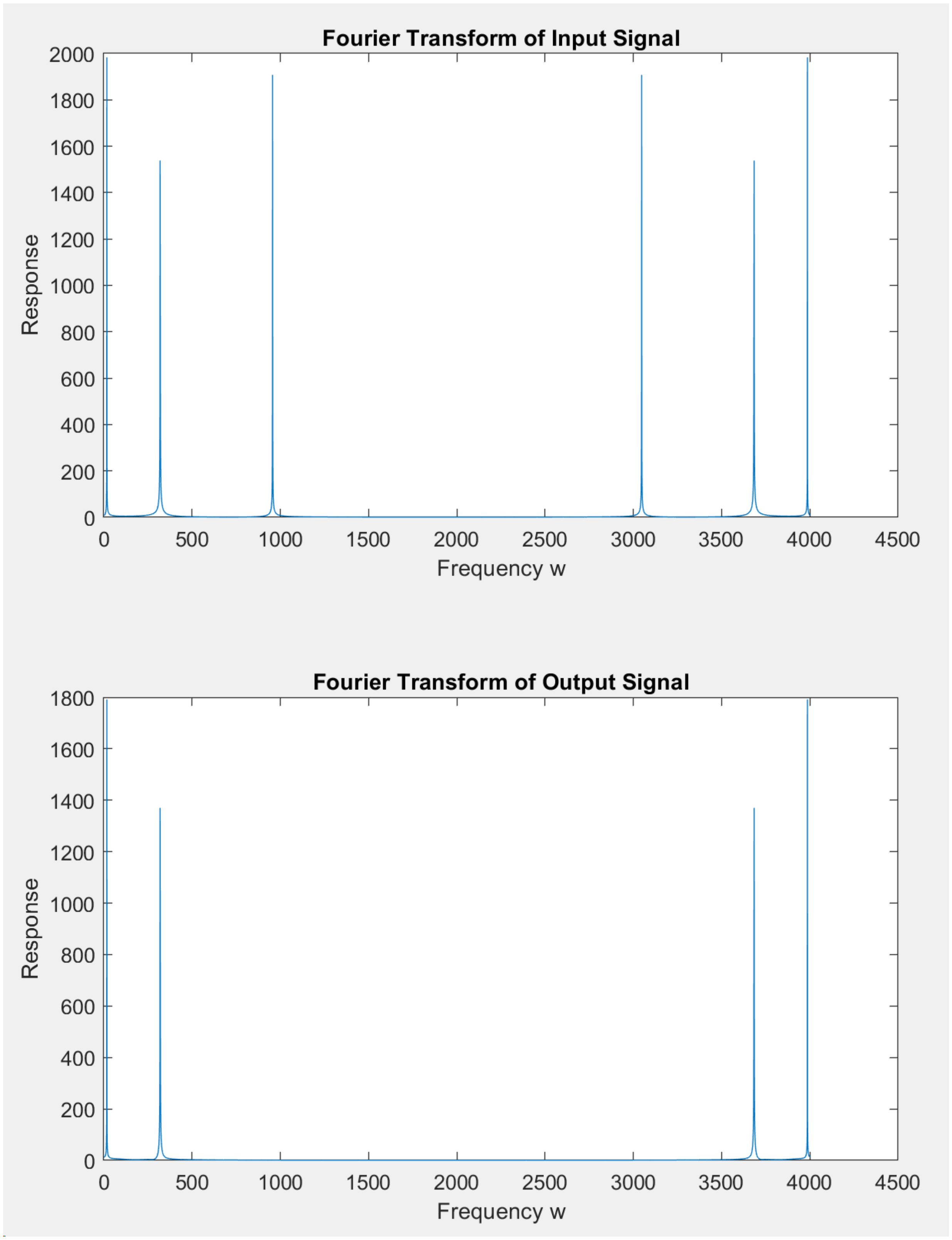

Fourier Transform of Input Signal
Response
Frequency w
Fourier Transform of Output Signal
Response
Frequency w

Plots of the eight order IIR Chebyshev type 1 filter



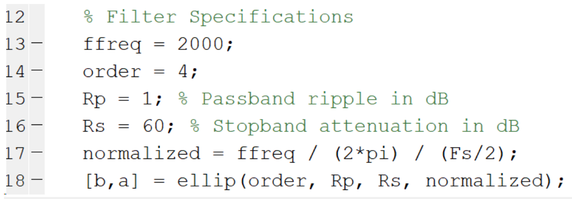
```
% Filter Specifications
ffreq = 2000;
order = 8;
normalized = ffreq / (2*pi) / (Fs/2);
[b,a] = cheby1(order, 1, normalized);
```



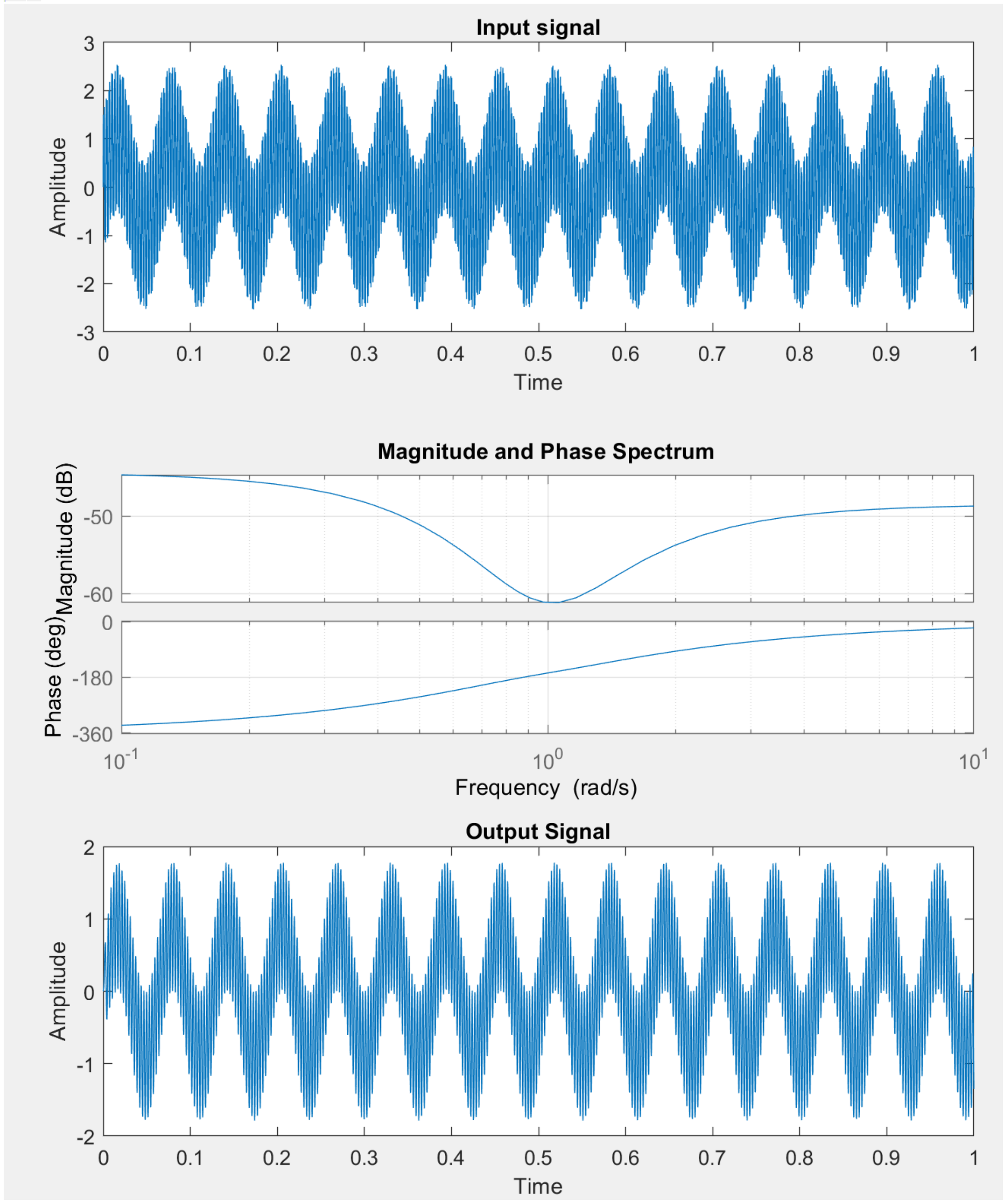

Fourier Transform of Input Signal
Response
Frequency w
0
500
1000
1500
2000
2500
3000
3500
4000
4500
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Fourier Transform of Output Signal
Response
Frequency w
0
200
400
600
800
1000
1200
1400
1600
1800

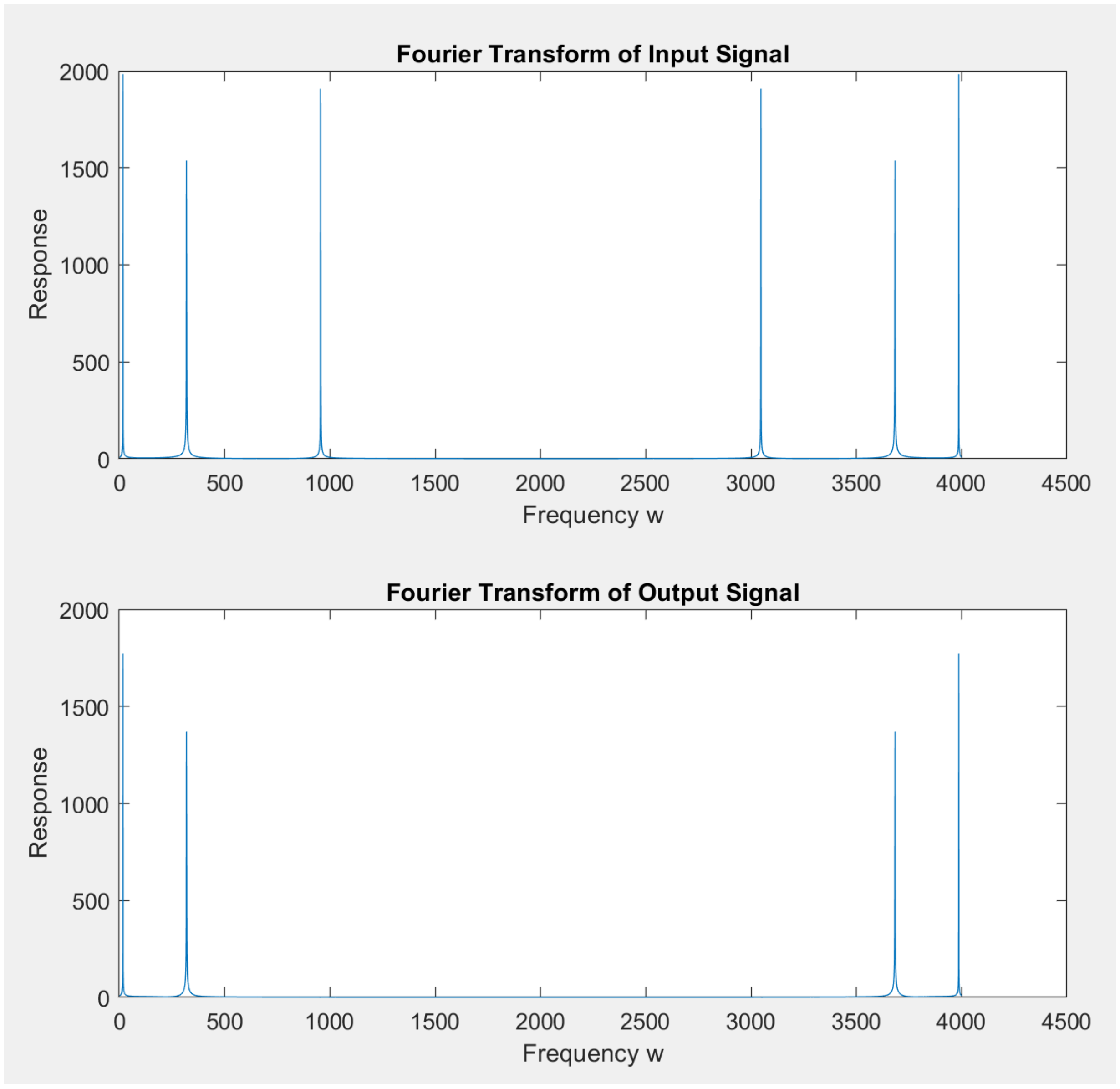
c) Design a digital fourth and eighth order IIR Elliptic filters. Plot the input signal along with the magnitude response of the filter and its output signal for each filter. (consider Rp= 1dB, Rs=60dB) =1, =60)

Plots of the fourth order IIR Elliptic filter

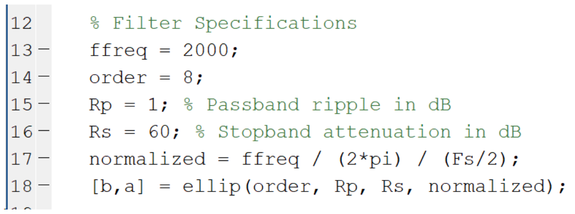


```
% Filter Specifications
ffreq = 2000;
order = 4;
Rp = 1; % Passband ripple in dB
Rs = 60; % Stopband attenuation in dB
normalized = ffreq / (2*pi) / (Fs/2);
[b,a] = ellip(order, Rp, Rs, normalized);
```

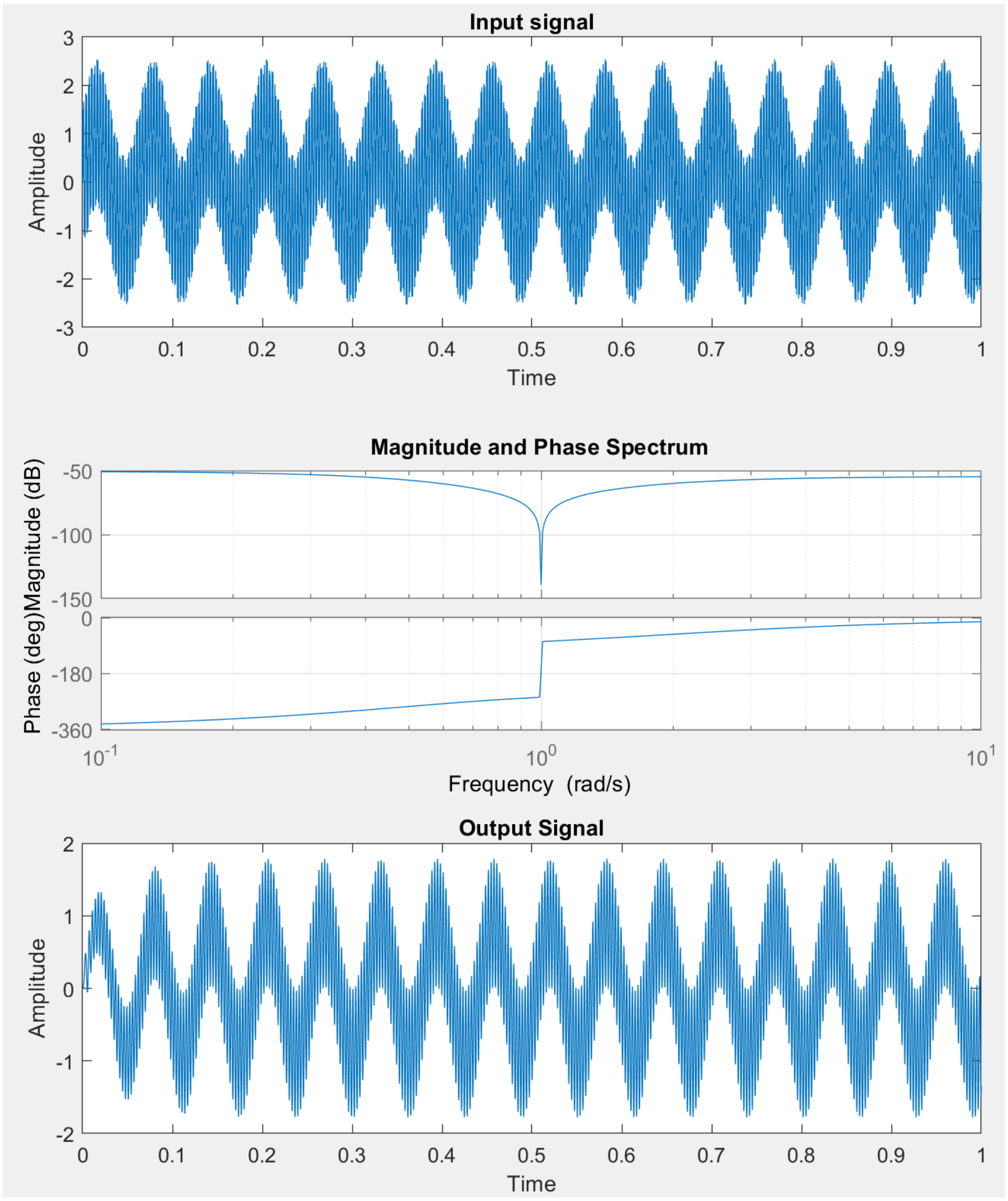


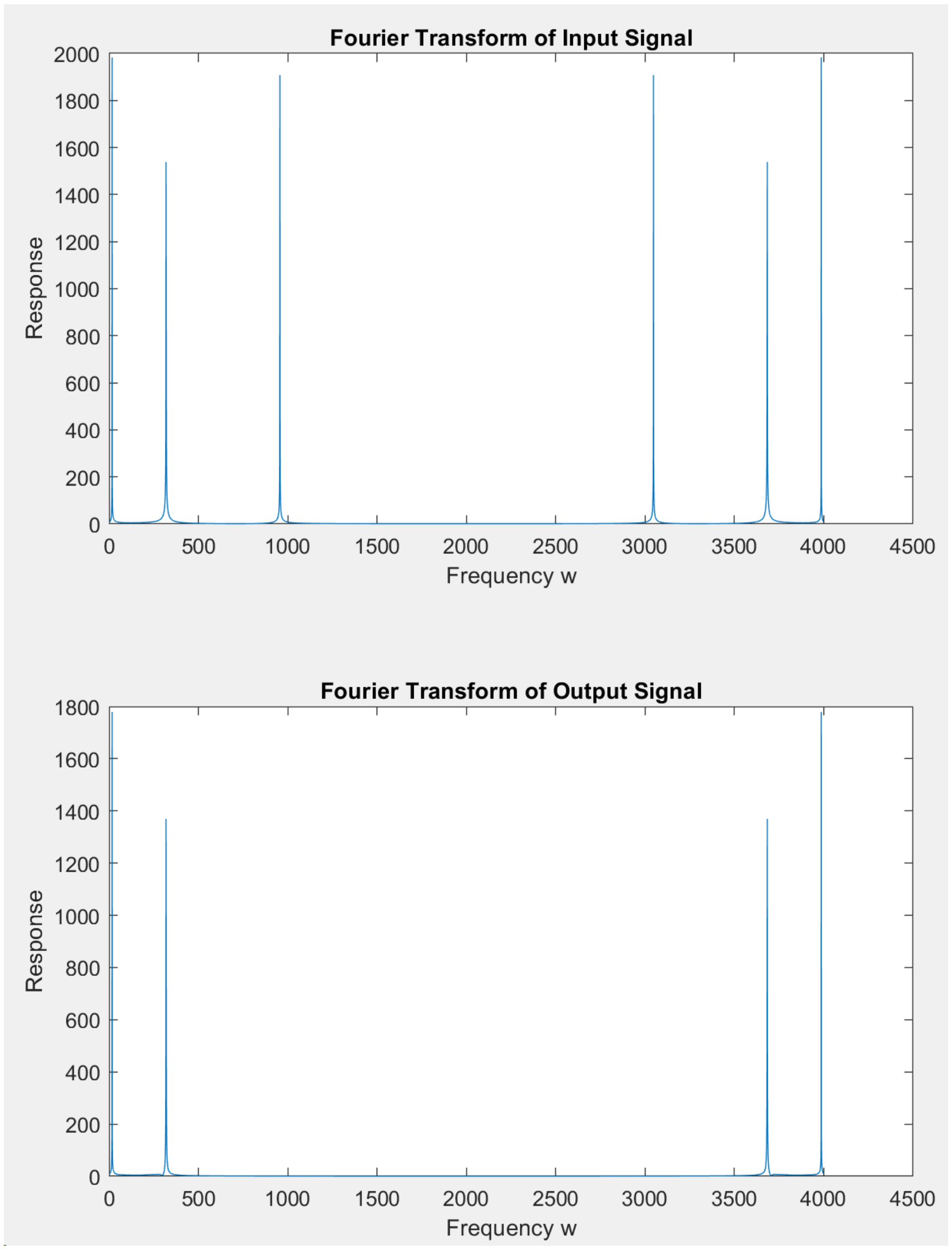

Fourier Transform of Input Signal
Response
Frequency w
Fourier Transform of Output Signal
Response
Frequency w

Plots of the eight order IIR Elliptic filter



```
% Filter Specifications
ffreq = 2000;
order = 8;
Rp = 1; % Passband ripple in dB
Rs = 60; % Stopband attenuation in dB
normalized = ffreq / (2*pi) / (Fs/2);
[b,a] = ellip(order, Rp, Rs, normalized);
```





d) For parts a), b), and c) plot the discrete Fourier transform of the input signal and the output signal for eighth order filters, you can use FFT command.

The plots were provided in the previous questions.

Question 2

Part A

Create a signal with three seconds duration. The signal contains three tones of equal amplitude, as following.

a) Contain a 200 Hz tone for $0 < t < 3$.

b) Contain a 330 Hz tone for $1 < t < 3$.

c) Contain a 480 Hz tone for $2 < t < 3$.

Note: The signal should have a sampling frequency equal to 8192 Hz.

View the signal before proceeding (in time domain and in frequency domain) and listen to the signal using soundsc command.

Part B

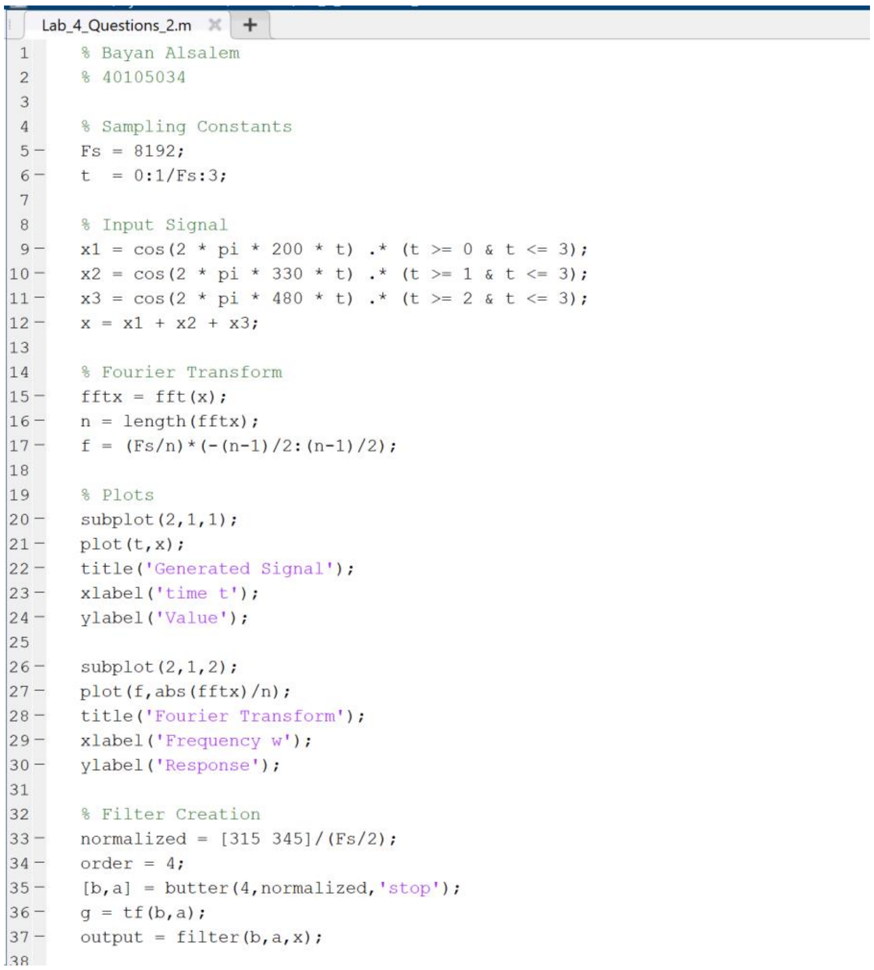
Design a filter to remove the 330 Hz component from the main signal.

a) Plot the designed response (magnitude and phase) of the filter.

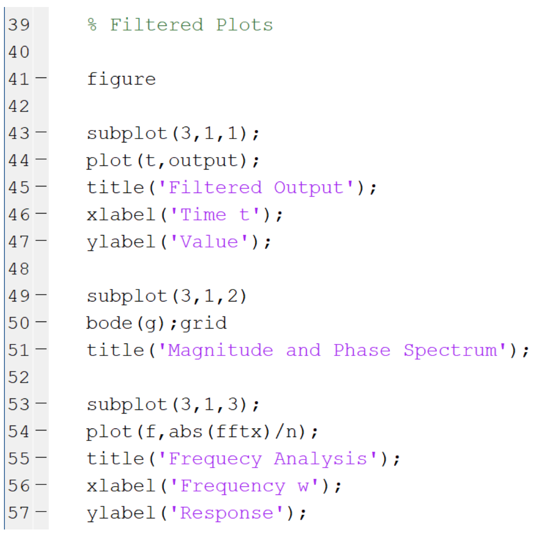
b) Filter the main signal with the designed filter.

c) Compare signals and spectra (in time domain and in frequency domain) before and after filtering and listen to the filtered signal using soundsc command.

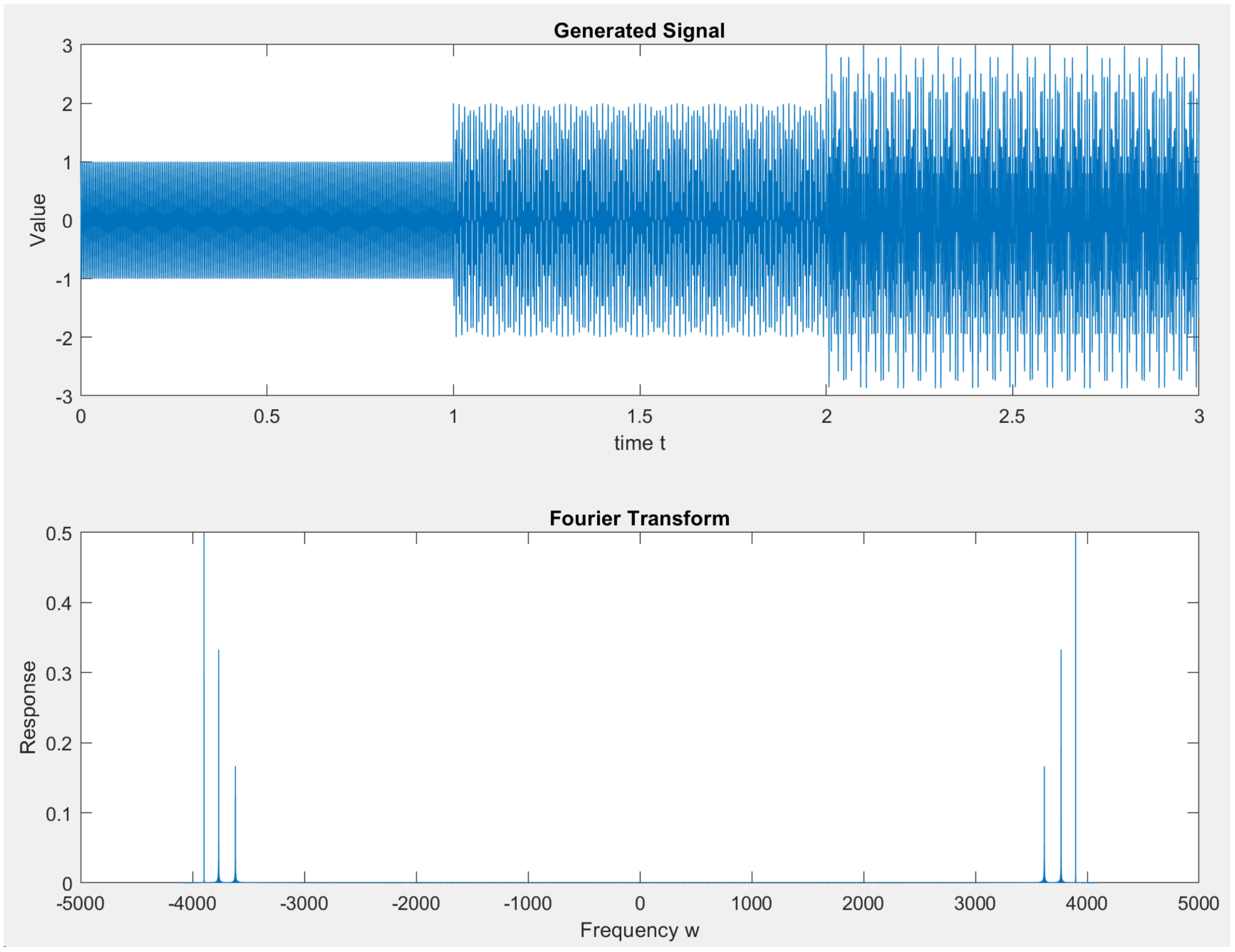
The MATLAB code and figures of part A and part B are shown below.

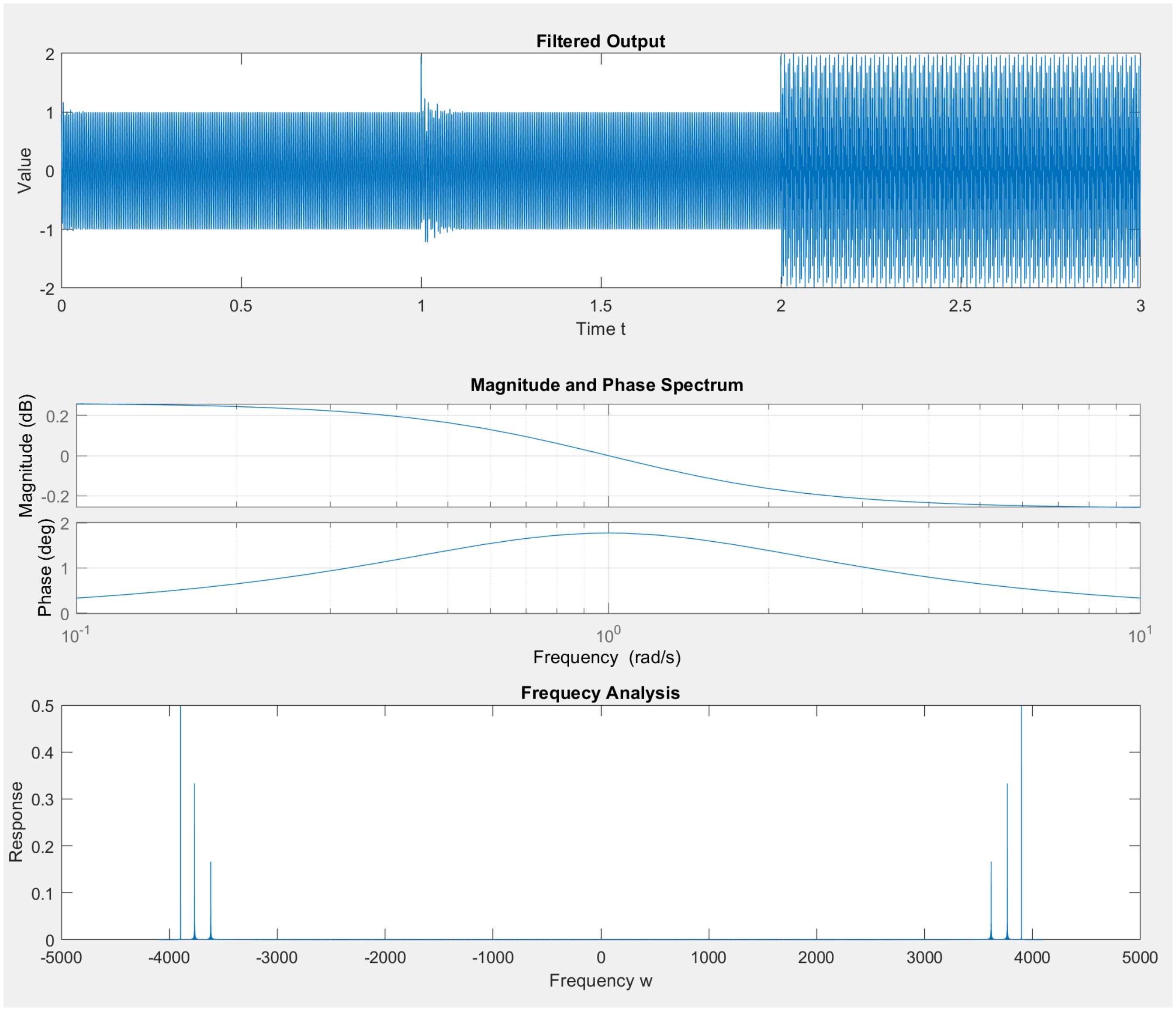


```matlab
% Bayan Alsalem
% 40105034

% Sampling Constants
Fs = 8192;
t  = 0:1/Fs:3;

% Input Signal
x1 = cos(2 * pi * 200 * t) .* (t >= 0 & t <= 3);
x2 = cos(2 * pi * 330 * t) .* (t >= 1 & t <= 3);
x3 = cos(2 * pi * 480 * t) .* (t >= 2 & t <= 3);
x = x1 + x2 + x3;

% Fourier Transform
fftx = fft(x);
n = length(fftx);
f = (Fs/n)*(-(n-1)/2:(n-1)/2);

% Plots
subplot(2,1,1);
plot(t,x);
title('Generated Signal');
xlabel('time t');
ylabel('Value');

subplot(2,1,2);
plot(f,abs(fftx)/n);
title('Fourier Transform');
xlabel('Frequency w');
ylabel('Response');

% Filter Creation
normalized = [315 345]/(Fs/2);
order = 4;
[b,a] = butter(4,normalized,'stop');
g = tf(b,a);
output = filter(b,a,x);

```



```matlab
% Filtered Plots

figure

subplot(3,1,1);
plot(t,output);
title('Filtered Output');
xlabel('Time t');
ylabel('Value');

subplot(3,1,2)
bode(g);grid
title('Magnitude and Phase Spectrum');

subplot(3,1,3);
plot(f,abs(fftx)/n);
title('Frequecy Analysis');
xlabel('Frequency w');
ylabel('Response');
```


Generated Signal
Value
time t
Fourier Transform
Response
Frequency w


Filtered Output
Value
Time t
Magnitude and Phase Spectrum
Magnitude (dB)
Phase (deg)
Frequency (rad/s)
Frequecy Analysis
Response
Frequency w

Question 3

Building Simulink model to read an audio file and manipulate the sound by adding upsampling or downsampling block.

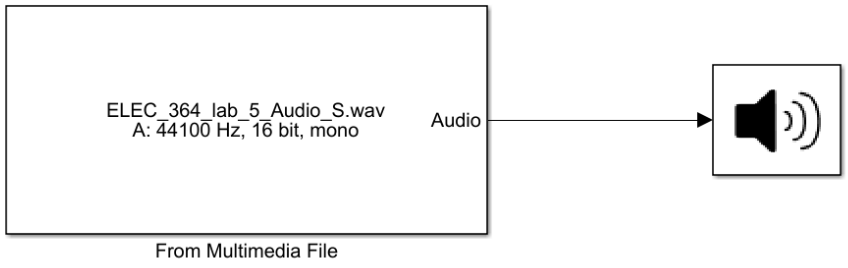
a) Download the audio file "ELEC_364_lab_5_Audio_S.wav" from Moodle, find lab data.

b) Open new Simulink model window, and import the audio file

"ELEC_364_lab_5_Audio_S.wav". Set Audio output data type to double.

c) Add To Audio Device block and connect the output of the audio block to the input of the To Audio Device block. Click run button and listen.

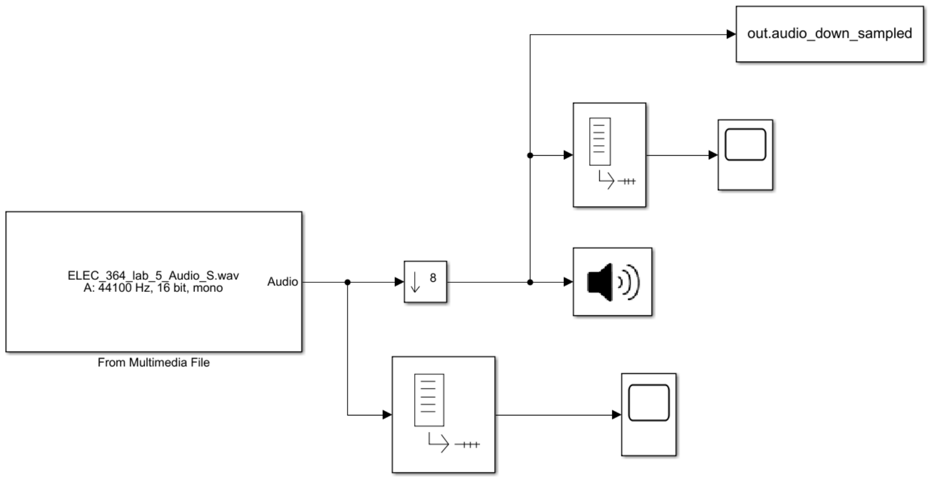
The audio file was downloaded, and the Simulink model was built.



Part A

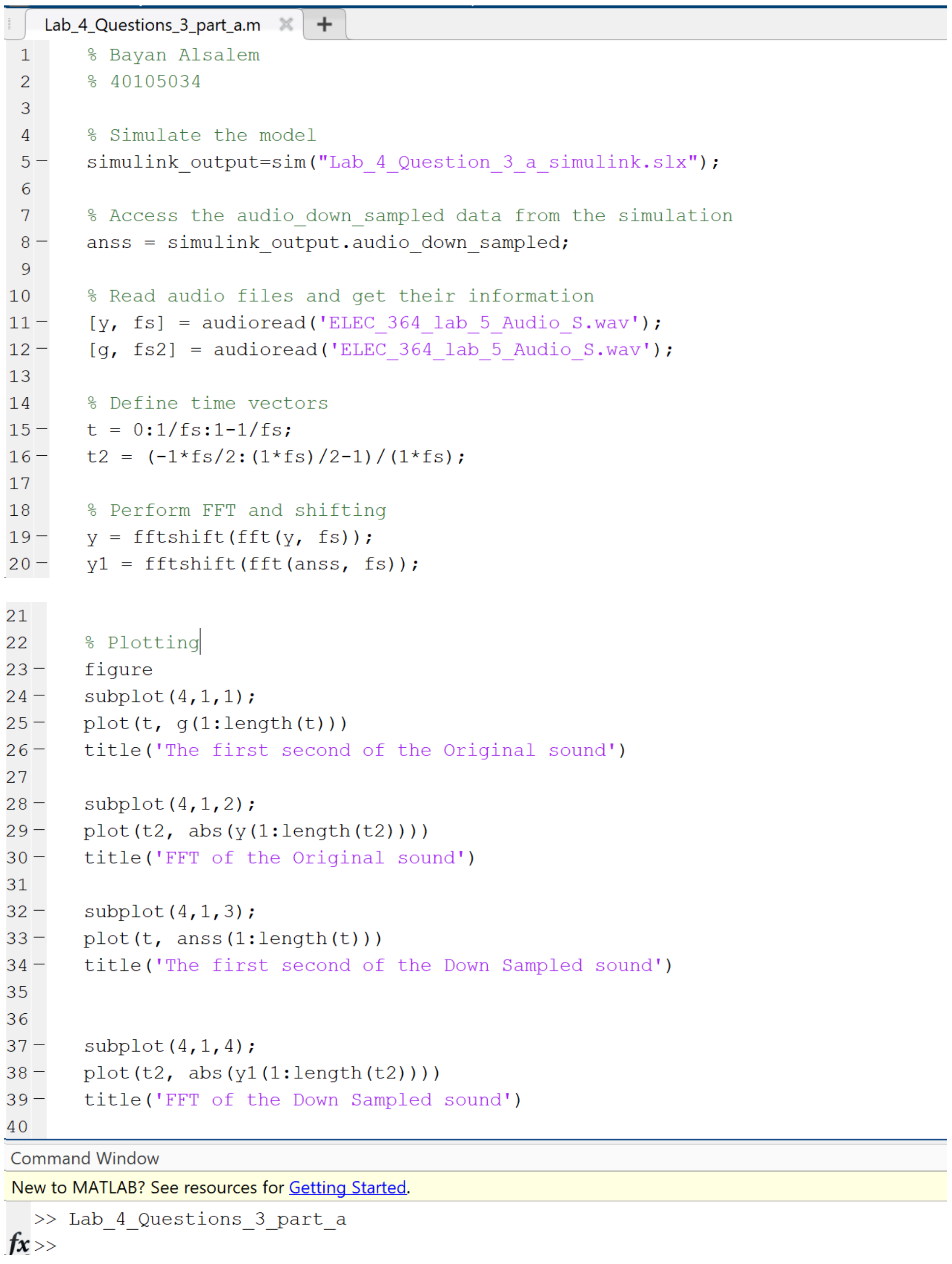
d) After listening, add Downsample block, and set downsample factor k to 8.

e) Build the Simulink model shown in Figure 6. Set the Scope parameter "limit data point to last" to 327680.



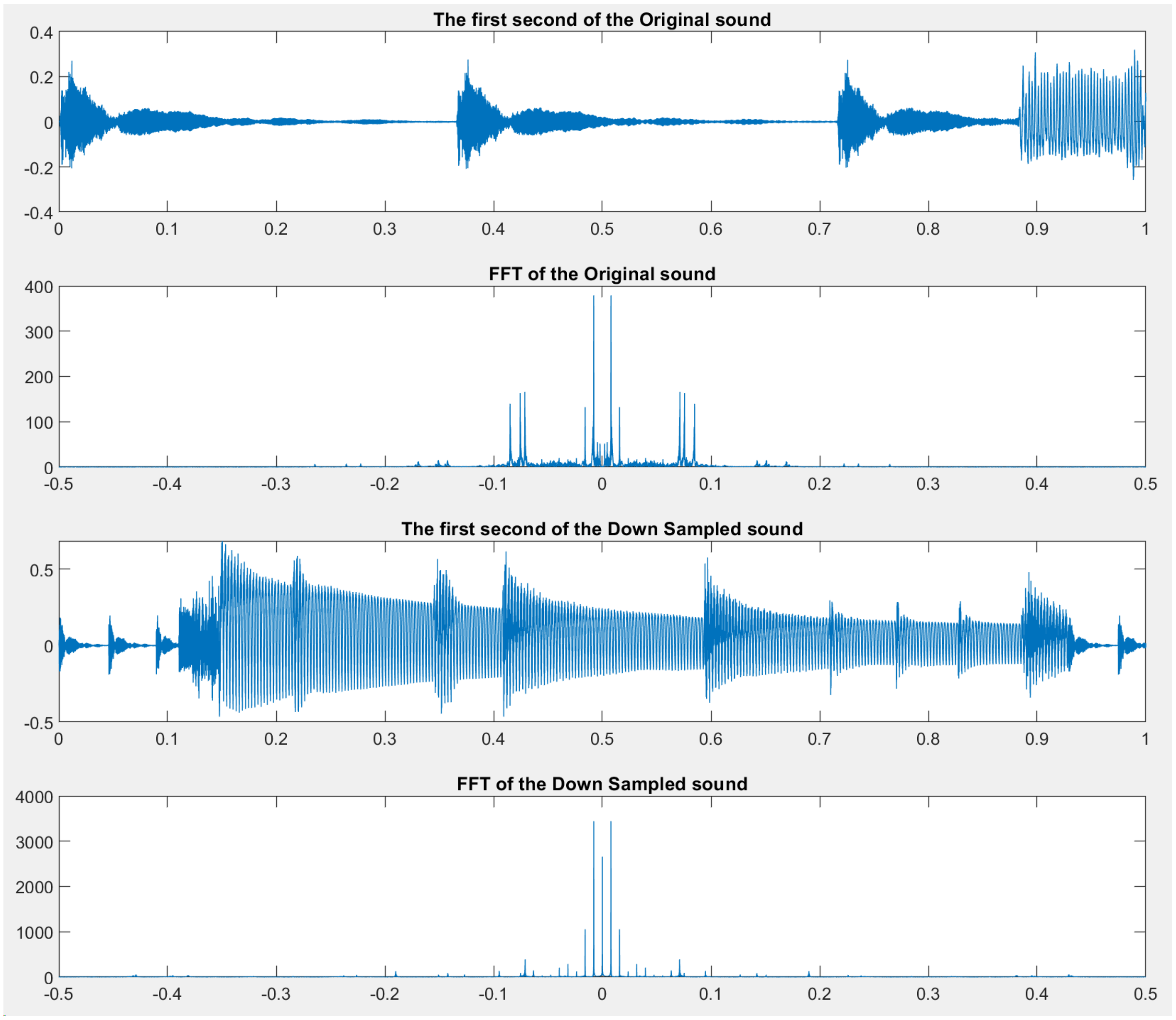
f) Click run button and listen to the sound. Did you notice any difference; write that in your lab report. Use single figure to plot the original sound signal in time and frequency domain (use FFT) and audio_down_sampled signal in time and frequency domain (use FFT).

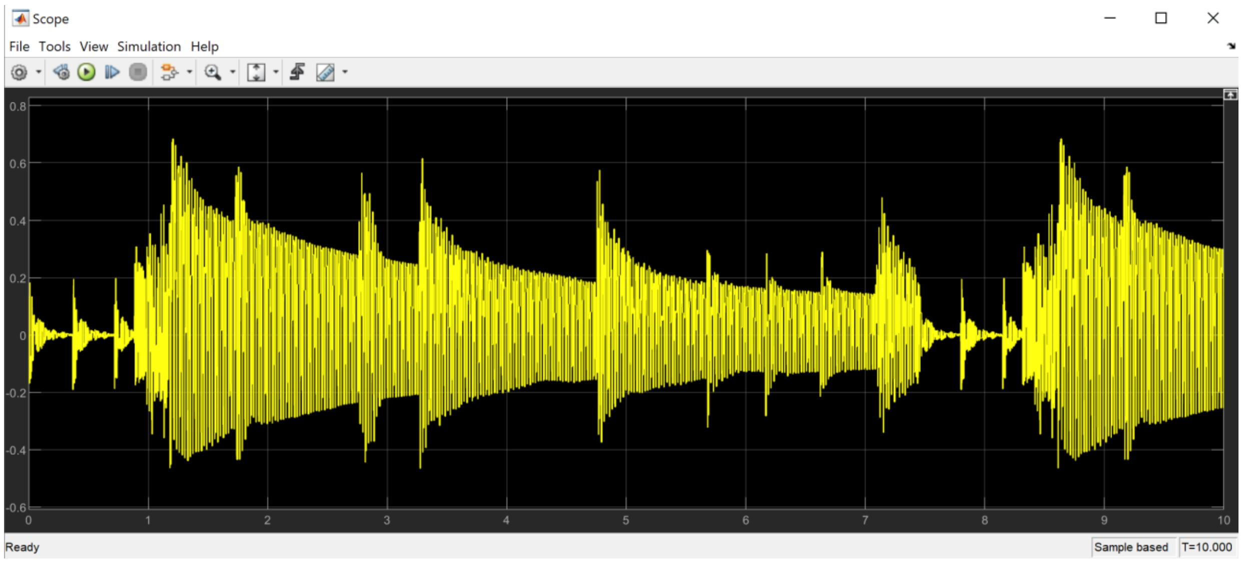
I felt that the quality of the audio has reduced a little bit. The sound was much lower in tone, with all the high pitch tones effectively being removed.

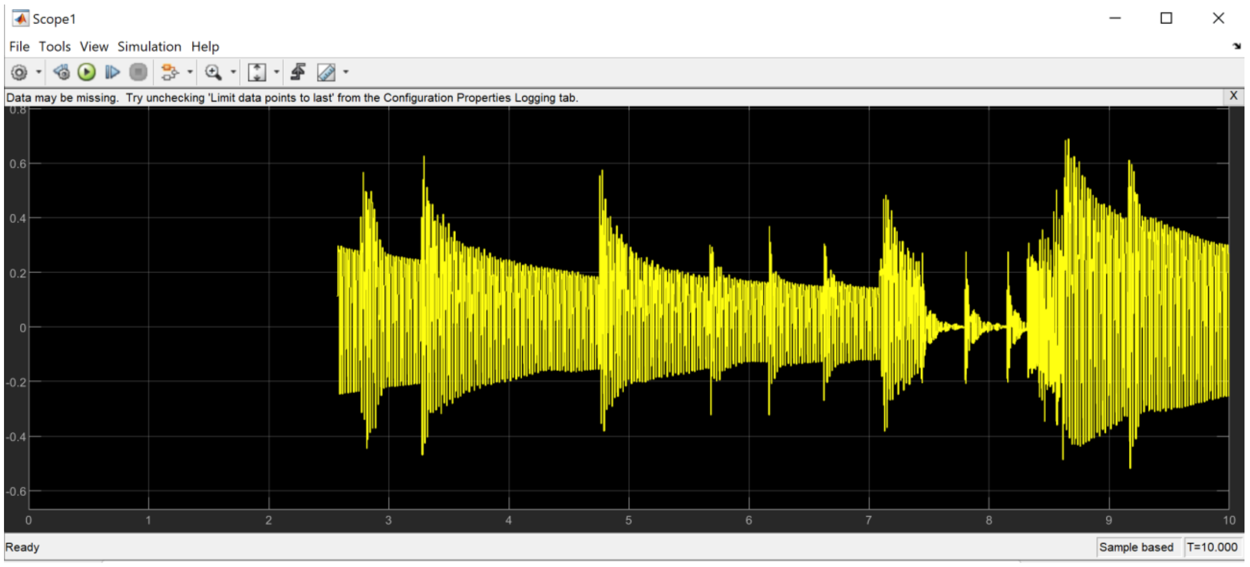


```
% Bayan Alsalem
% 40105034

% Simulate the model
simulink_output=sim("Lab_4_Question_3_a_simulink.slx");

% Access the audio_down_sampled data from the simulation
anss = simulink_output.audio_down_sampled;

% Read audio files and get their information
[y, fs] = audioread('ELEC_364_lab_5_Audio_S.wav');
[g, fs2] = audioread('ELEC_364_lab_5_Audio_S.wav');

% Define time vectors
t = 0:1/fs:1-1/fs;
t2 = (-1*fs/2:(1*fs)/2-1)/(1*fs);

% Perform FFT and shifting
y = fftshift(fft(y, fs));
y1 = fftshift(fft(anss, fs));


% Plotting
figure
subplot(4,1,1);
plot(t, g(1:length(t)))
title('The first second of the Original sound')

subplot(4,1,2);
plot(t2, abs(y(1:length(t2))))
title('FFT of the Original sound')

subplot(4,1,3);
plot(t, anss(1:length(t)))
title('The first second of the Down Sampled sound')


subplot(4,1,4);
plot(t2, abs(y1(1:length(t2))))
title('FFT of the Down Sampled sound')

```

```
>> Lab_4_Questions_3_part_a
fx >>
```


The first second of the Original sound
0.4
0.2
0
-0.2
-0.4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
FFT of the Original sound
400
300
200
100
0
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
The first second of the Down Sampled sound
0.5
0
-0.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
FFT of the Down Sampled sound
4000
3000
2000
1000
0
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5

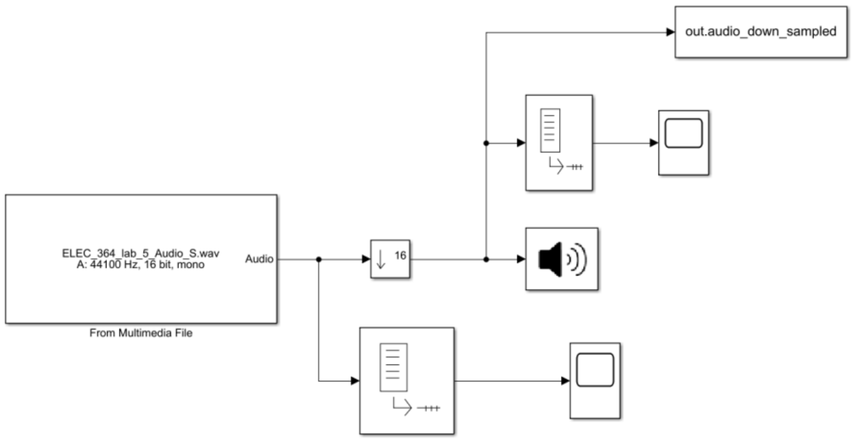


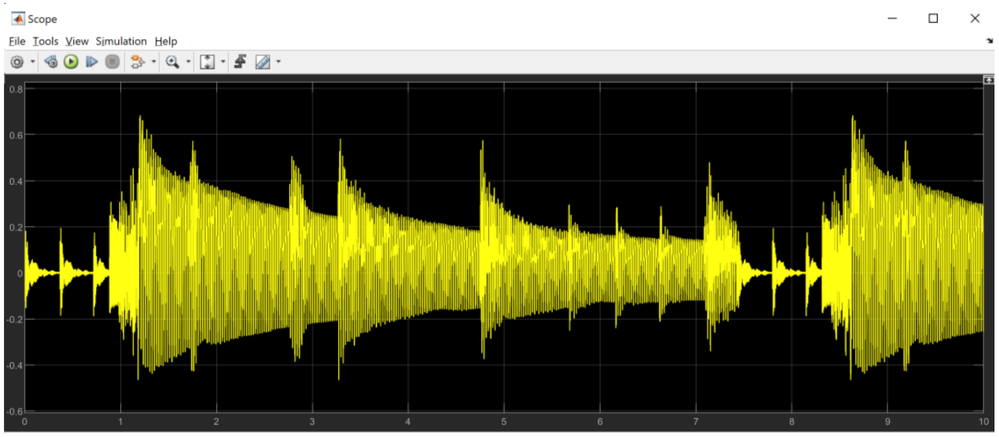


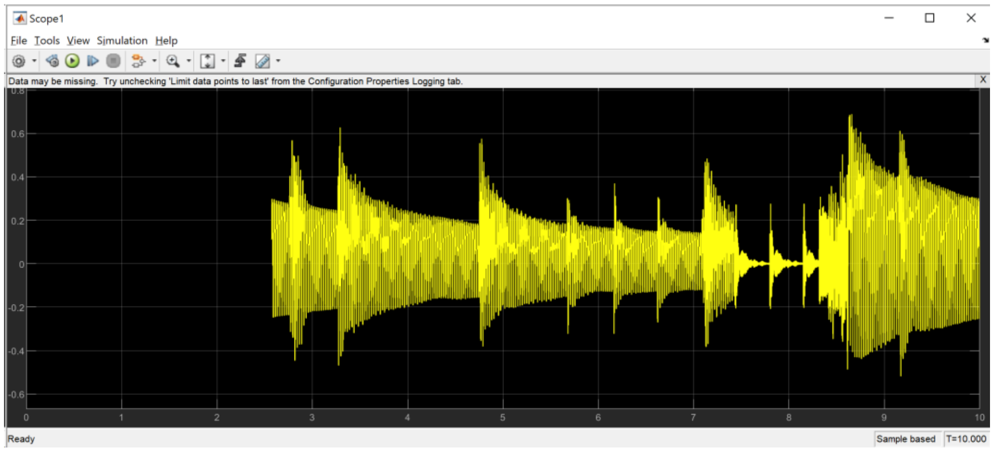
Data may be missing. Try unchecking 'Limit data points to last' from the Configuration Properties Logging tab.

g) Now, reset the downsample factor k to 16 and repeat the step f).

The down sampling by factor 16 decreased the distortion and further lowered the tone. There was reducing in the high pitch sound components.



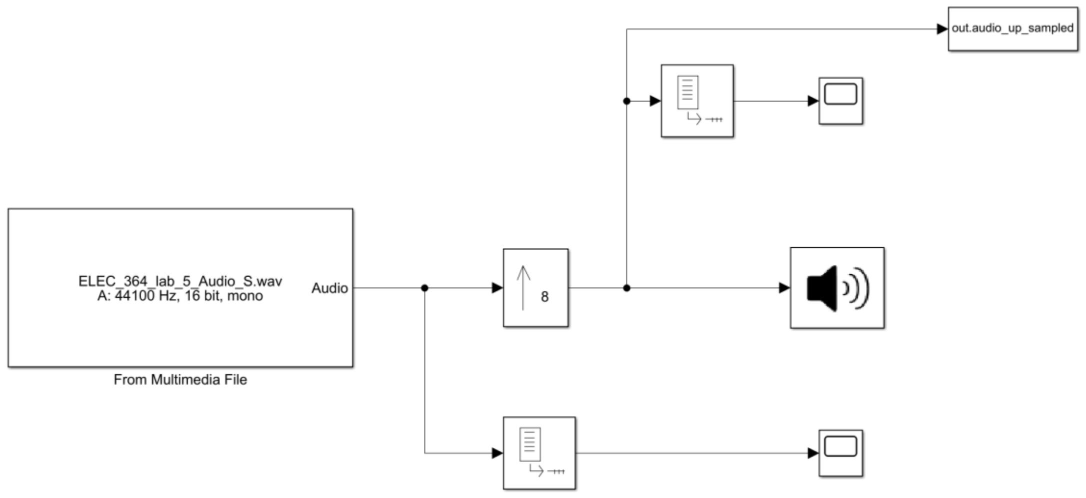




Data may be missing. Try unchecking 'Limit data points to last' from the Configuration Properties Logging tab.

Part B

h) Build the Simulink model shown in Figure 8 and add Upsample block. Set the upsample factor L to 8 and set the Scope parameter "limit data point to last" to 327680.

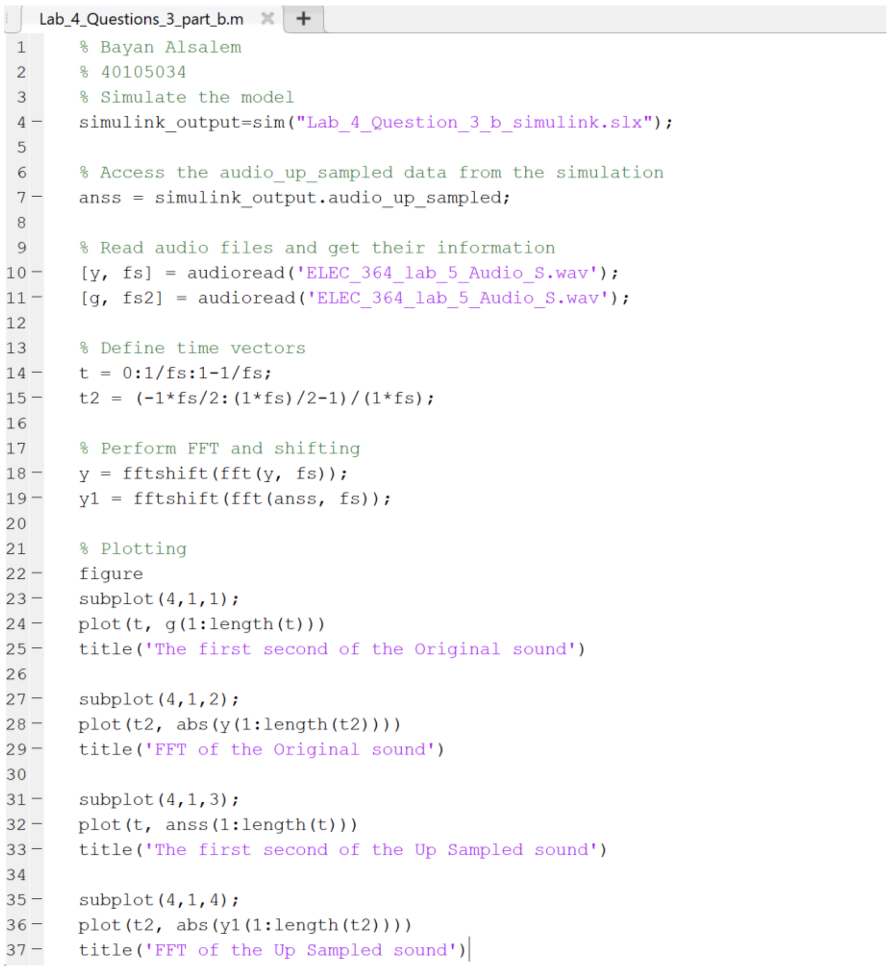


i) Double click of the To Audio Device block and uncheck the Inherit sample rate from input option.

j) Click run button and listen to the sound. Did you notice any difference; write that in your lab report. Use single figure to plot the original sound signal in time and frequency domain.

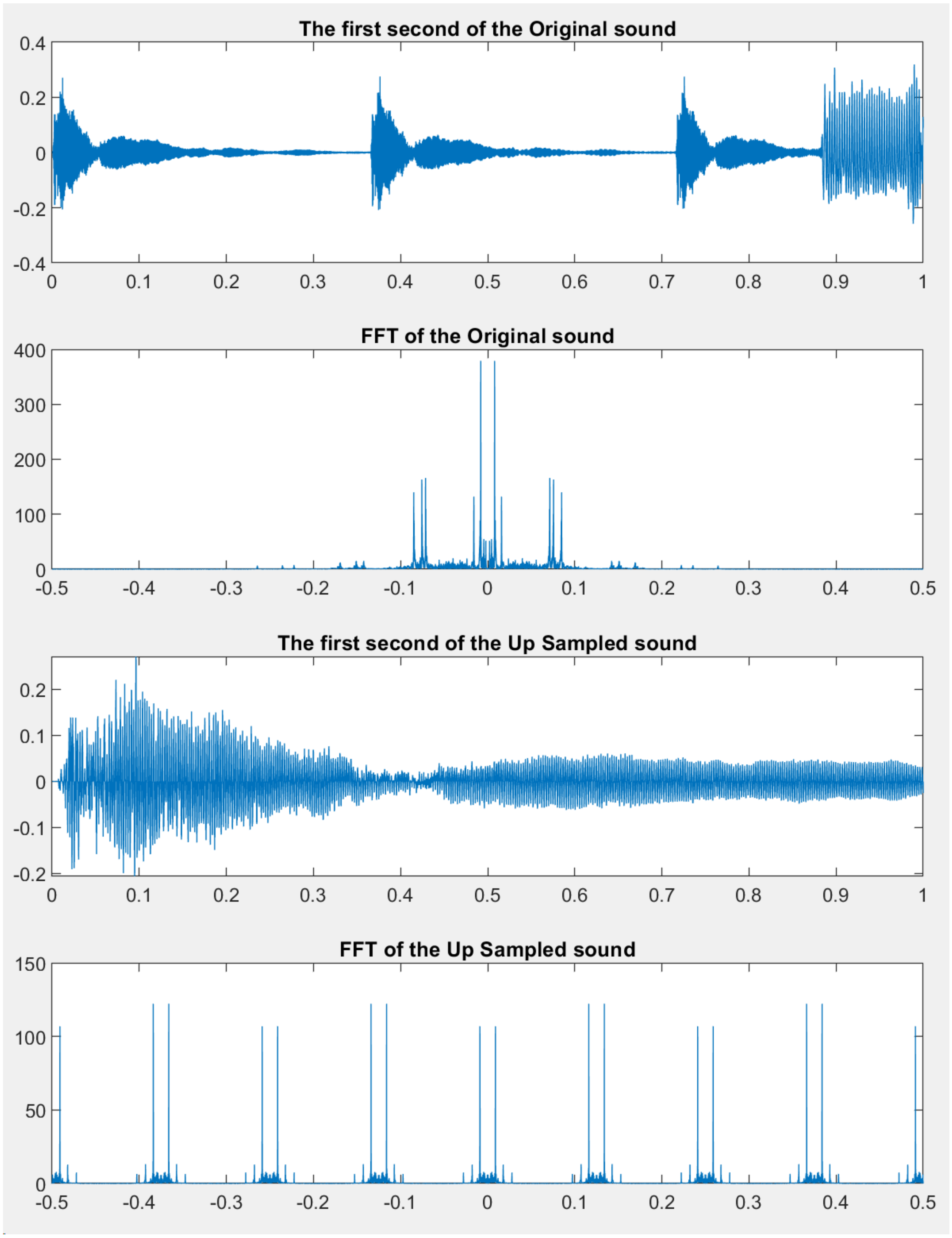
There is a huge difference between the original signal and the sampled one by factor 8. Up Sampled audio caused the playback time of the sample to be extended. The increased samples created gaps, and as the audio block output at a set frequency, the sound was left with gaps as the sampled frequency was now significantly higher.

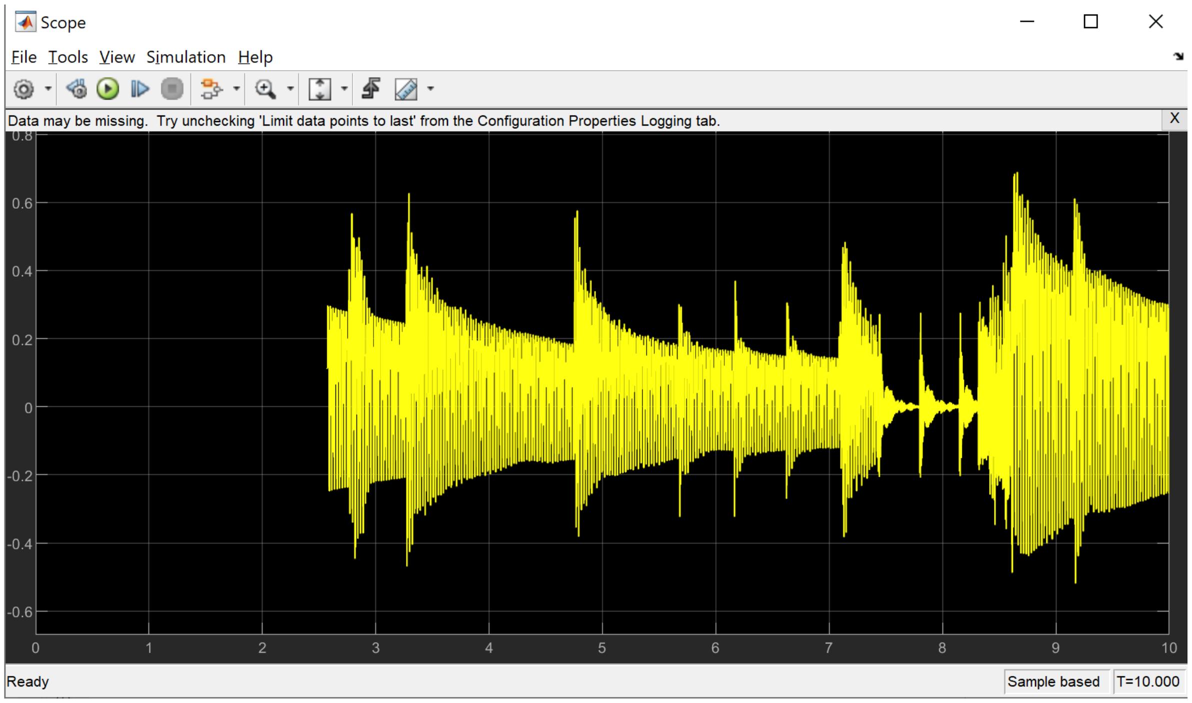
The scope output demonstrated this with a much larger gap before sound playback started.



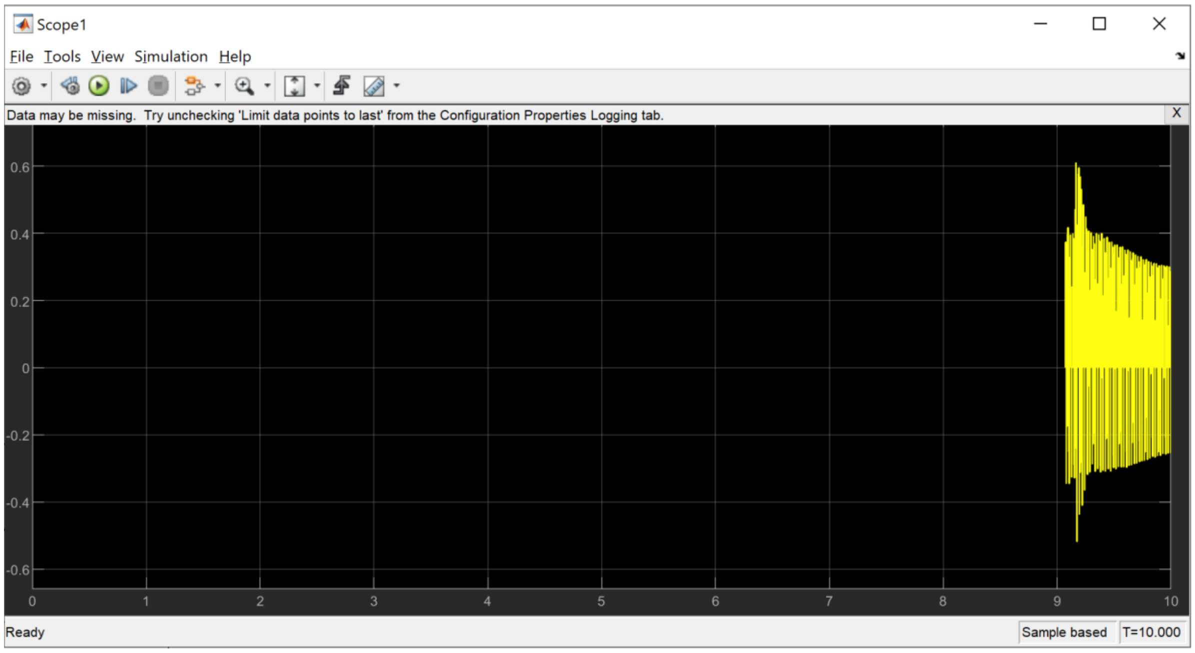
Lab_4_Questions_3_part_b.m

```
% Bayan Alsalem
% 40105034
% Simulate the model
simulink_output=sim("Lab_4_Question_3_b_simulink.slx");

% Access the audio_up_sampled data from the simulation
anss = simulink_output.audio_up_sampled;

% Read audio files and get their information
[y, fs] = audioread('ELEC_364_lab_5_Audio_S.wav');
[g, fs2] = audioread('ELEC_364_lab_5_Audio_S.wav');

% Define time vectors
t = 0:1/fs:1-1/fs;
t2 = (-1*fs/2:(1*fs)/2-1)/(1*fs);

% Perform FFT and shifting
y = fftshift(fft(y, fs));
y1 = fftshift(fft(anss, fs));

% Plotting
figure
subplot(4,1,1);
plot(t, g(1:length(t)))
title('The first second of the Original sound')

subplot(4,1,2);
plot(t2, abs(y(1:length(t2))))
title('FFT of the Original sound')

subplot(4,1,3);
plot(t, anss(1:length(t)))
title('The first second of the Up Sampled sound')

subplot(4,1,4);
plot(t2, abs(y1(1:length(t2))))
title('FFT of the Up Sampled sound')
```


The first second of the Original sound
0.4
0.2
0
-0.2
-0.4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
FFT of the Original sound
400
300
200
100
0
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
The first second of the Up Sampled sound
0.2
0.1
0
-0.1
-0.2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
FFT of the Up Sampled sound
150
100
50
0
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5



Data may be missing. Try unchecking 'Limit data points to last' from the Configuration Properties Logging tab.

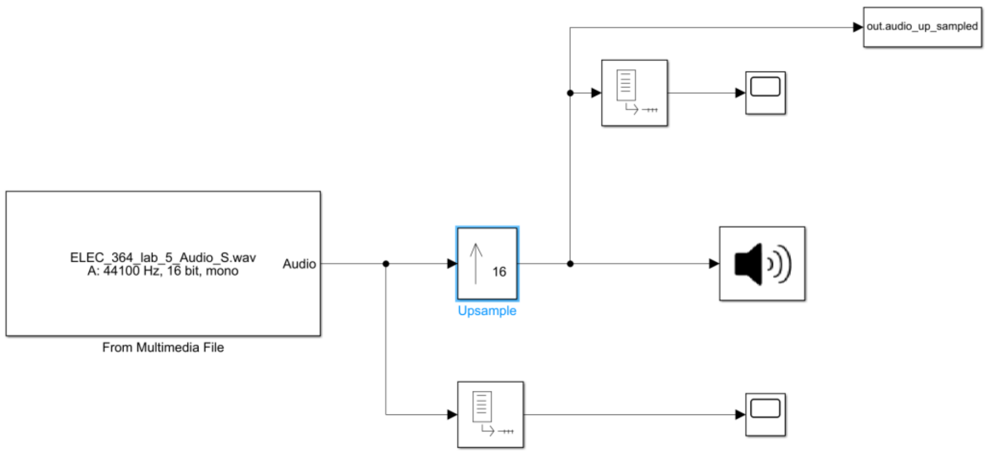


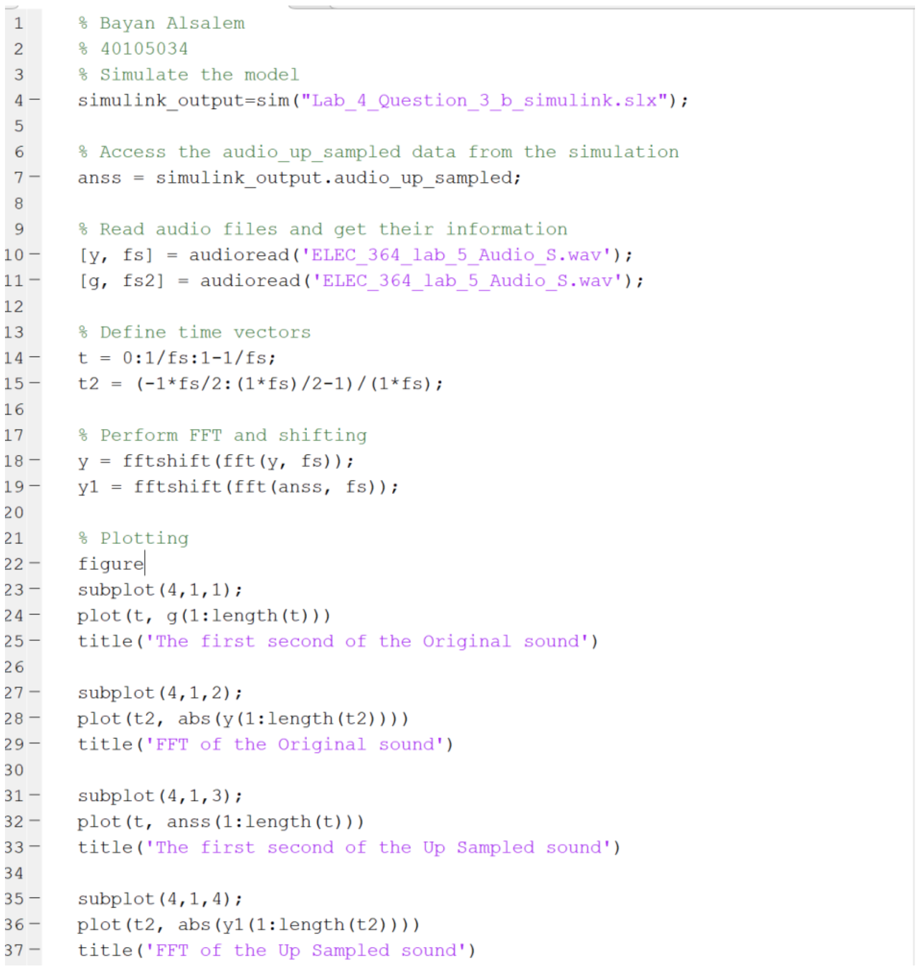
Data may be missing. Try unchecking 'Limit data points to last' from the Configuration Properties Logging tab.

k) Set the upsample factor L to 16 and repeat the step j).

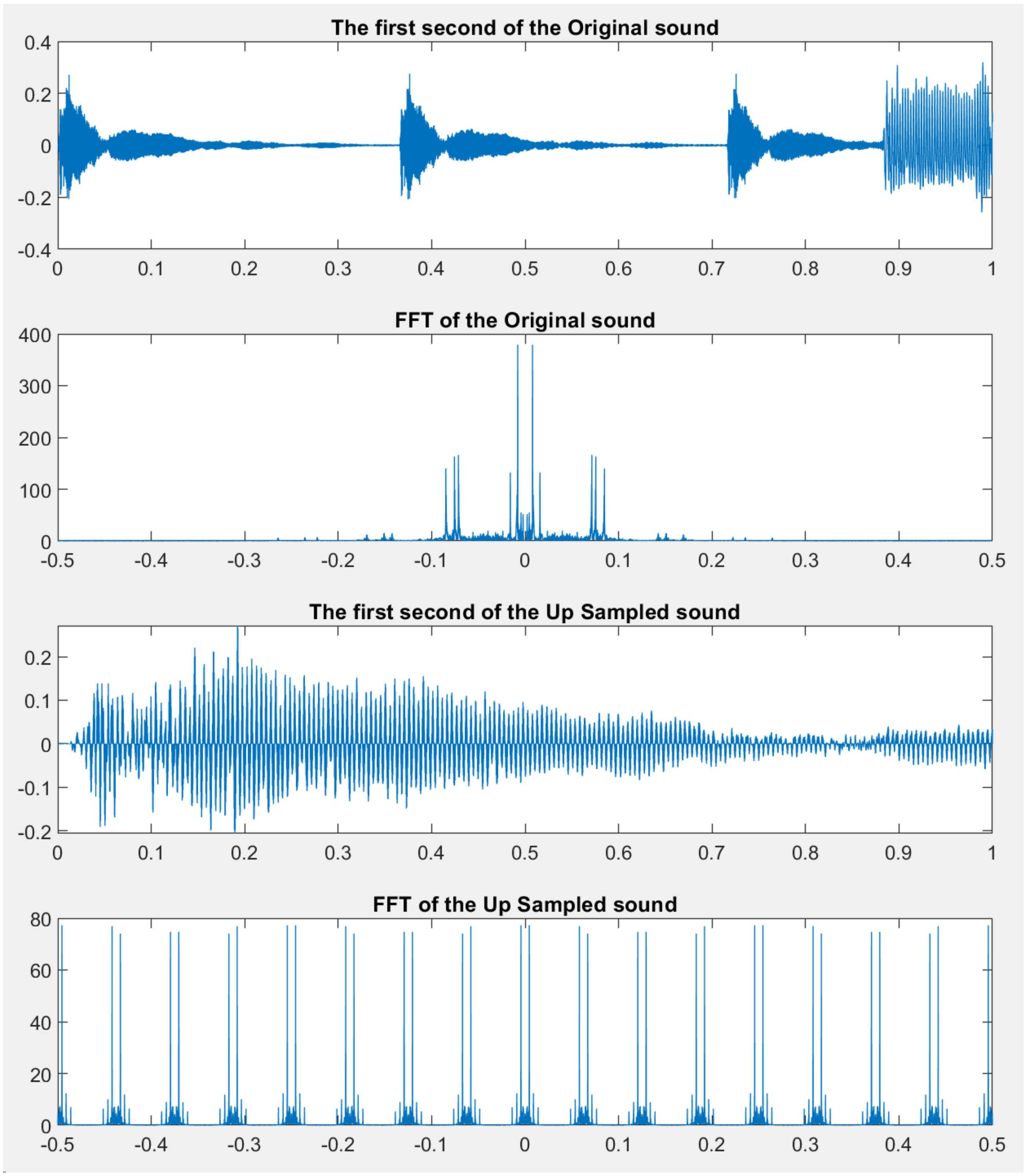
Increasing the sampling rate further increased the gaps in audio playback and showed ta larger gap in the scope readout.

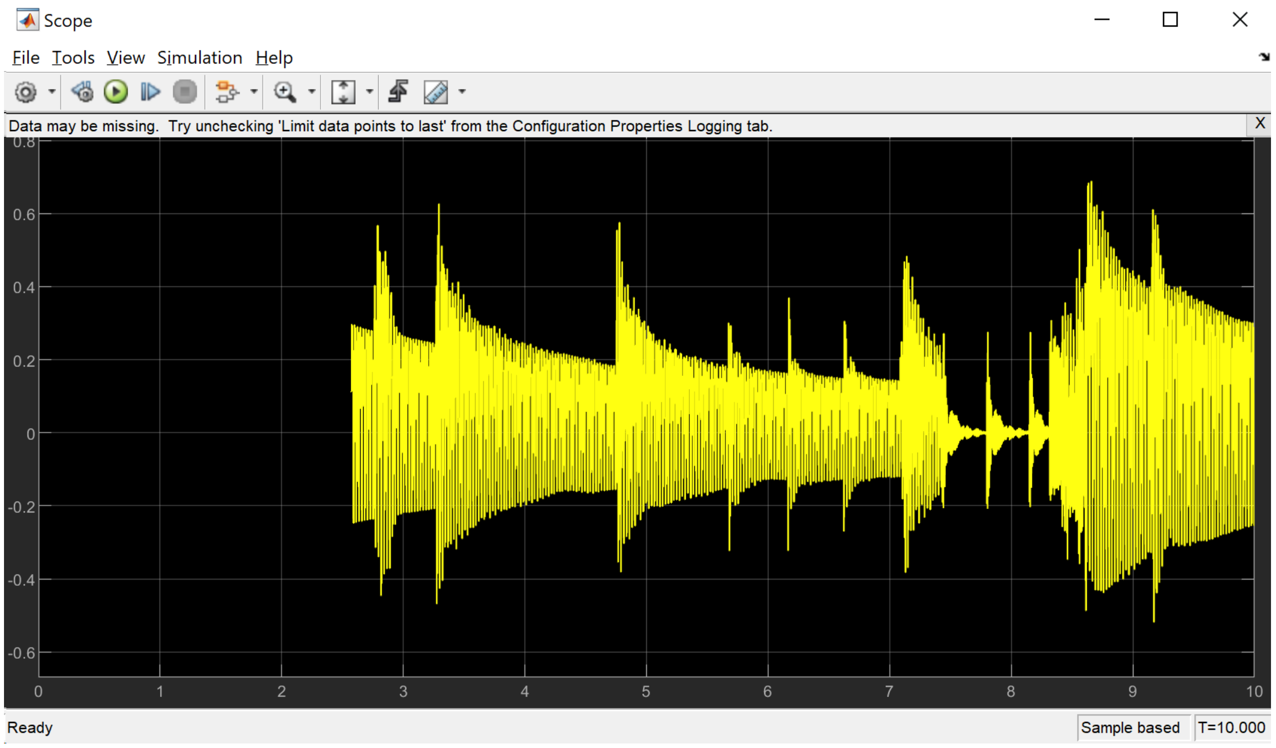
The sampled audio was also greatly increased in duration in real time.



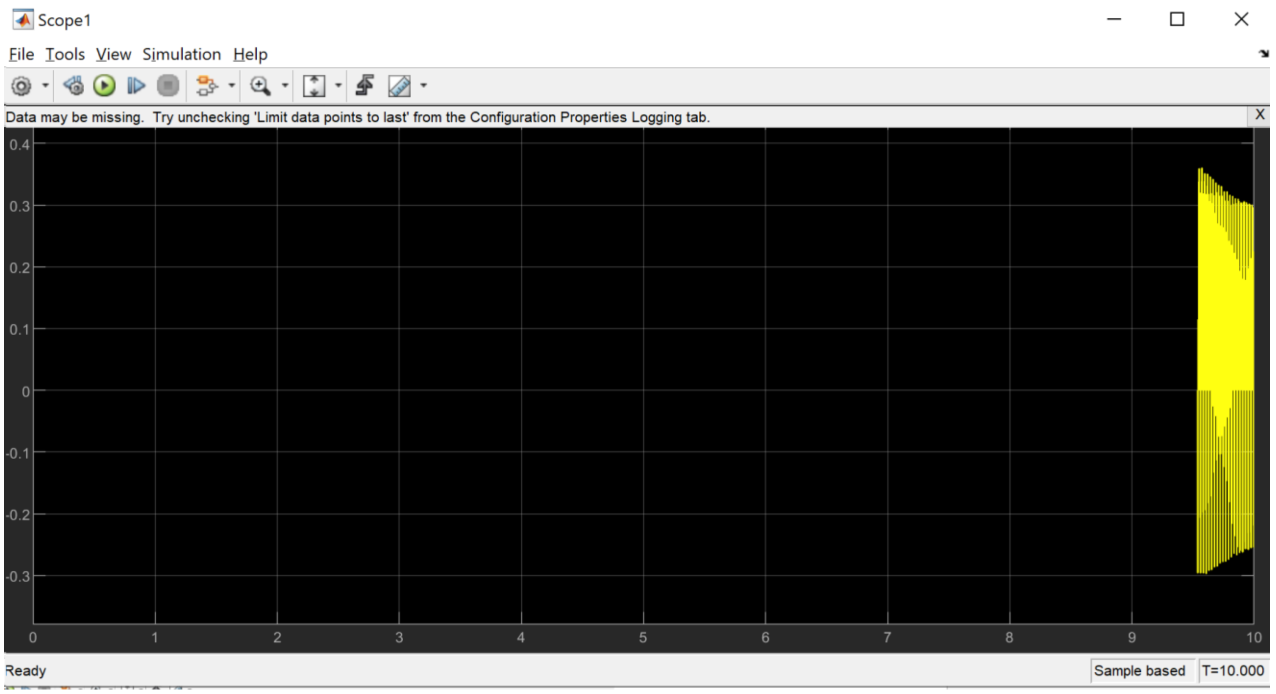


```
% Bayan Alsalem
% 40105034
% Simulate the model
simulink_output=sim("Lab_4_Question_3_b_simulink.slx");

% Access the audio_up_sampled data from the simulation
anss = simulink_output.audio_up_sampled;

% Read audio files and get their information
[y, fs] = audioread('ELEC_364_lab_5_Audio_S.wav');
[g, fs2] = audioread('ELEC_364_lab_5_Audio_S.wav');

% Define time vectors
t = 0:1/fs:1-1/fs;
t2 = (-1*fs/2:(1*fs)/2-1)/(1*fs);

% Perform FFT and shifting
y = fftshift(fft(y, fs));
y1 = fftshift(fft(anss, fs));

% Plotting
figure
subplot(4,1,1);
plot(t, g(1:length(t)))
title('The first second of the Original sound')

subplot(4,1,2);
plot(t2, abs(y(1:length(t2))))
title('FFT of the Original sound')

subplot(4,1,3);
plot(t, anss(1:length(t)))
title('The first second of the Up Sampled sound')

subplot(4,1,4);
plot(t2, abs(y1(1:length(t2))))
title('FFT of the Up Sampled sound')
```


The first second of the Original sound
0.4
0.2
0
-0.2
-0.4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
FFT of the Original sound
400
300
200
100
0
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
The first second of the Up Sampled sound
0.2
0.1
0
-0.1
-0.2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
FFT of the Up Sampled sound
80
60
40
20
0
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5



Data may be missing. Try unchecking 'Limit data points to last' from the Configuration Properties Logging tab.



Data may be missing. Try unchecking 'Limit data points to last' from the Configuration Properties Logging tab.