ASSIGNMENT, 2020/21 SESSION
ETN3096 DIGITAL SIGNAL PROCESSING

Student ID	Name	Majoring
1181101213	Zhafir Afdhaluddin Roslen	CE
1181101218	Muhammad Nur Irham Firdaus Bin Azmi	CE

To be filled by lecturer:

Filter specification (out of 3 marks)	
FIR filter design (out of 5 marks)	
IIR filter design (out of 5 marks)	
Evaluation of filtered signal (out of 2 marks)	
Total	

Instructions

1. You can work in a group of 2 students, or work individually. Each group only needs to submit a single softcopy pdf report.
2. Please submit the assignment to Dr. Tan Ai Hui through email (htai@mmu.edu.my) by 4 February 2021, 3pm.
3. The assignment may be handwritten or typewritten.
4. Please use this page as the cover page.
5. Penalty for late submission is a deduction of 2 marks out of 15 marks per day.
6. You may use a combination of hand calculations and computations using MATLAB/Octave.
7. Please note that plagiarism is a serious offence. Those involved may get 0 mark for the assignment.

Introduction

Propose a finite impulse response (FIR) filter and an infinite impulse response (IIR) filter to "clean" the noisy signal. You can use any type of filter covered in Chapter 3. Please use the mark distribution as shown in the cover page as a guideline.

Attached noisy_signal.wav and *clean_signal.wav* file shows noticeable difference in sound when played. To analyze it, we need to read the .wav file using the *audioread()* as show in Figure 1.1 and Figure 1.2.

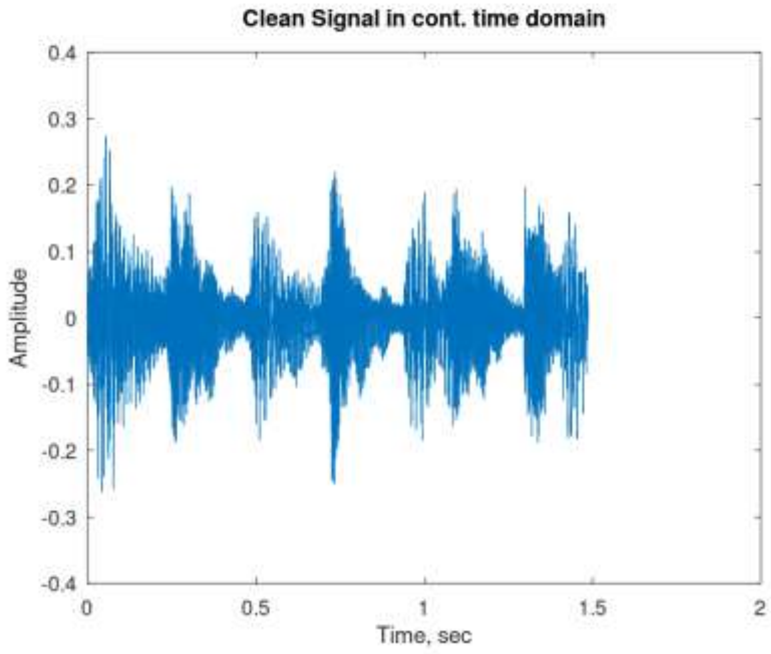

Figure 1.1 shows the clean_signal.wav plotted against cont. Time domain

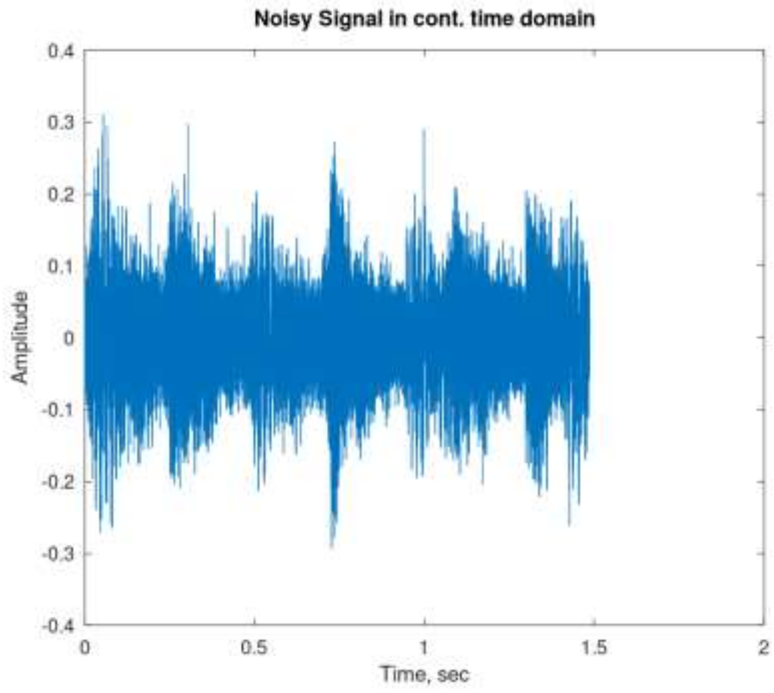

Figure 1.2 shows the noisy_signal.wav plotted against cont. Time domain

We will also be analyzing these signal through *Frequency Spectra Graph* by utilizing the function *fft()* that is available in the octave libraries. Figure below shows the *Frequency Spectra graph* for both clean_signal.wav and *noisy_signal.wav*.

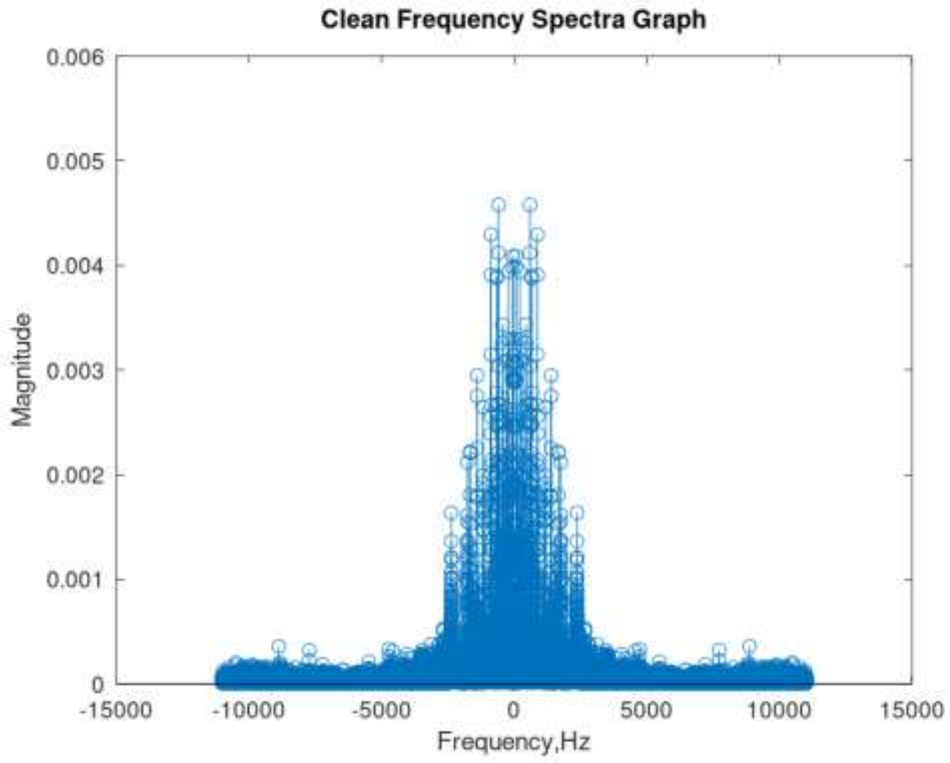

Figure 1.3 shows the Frequency Spectra Graph for clean_signal.wav

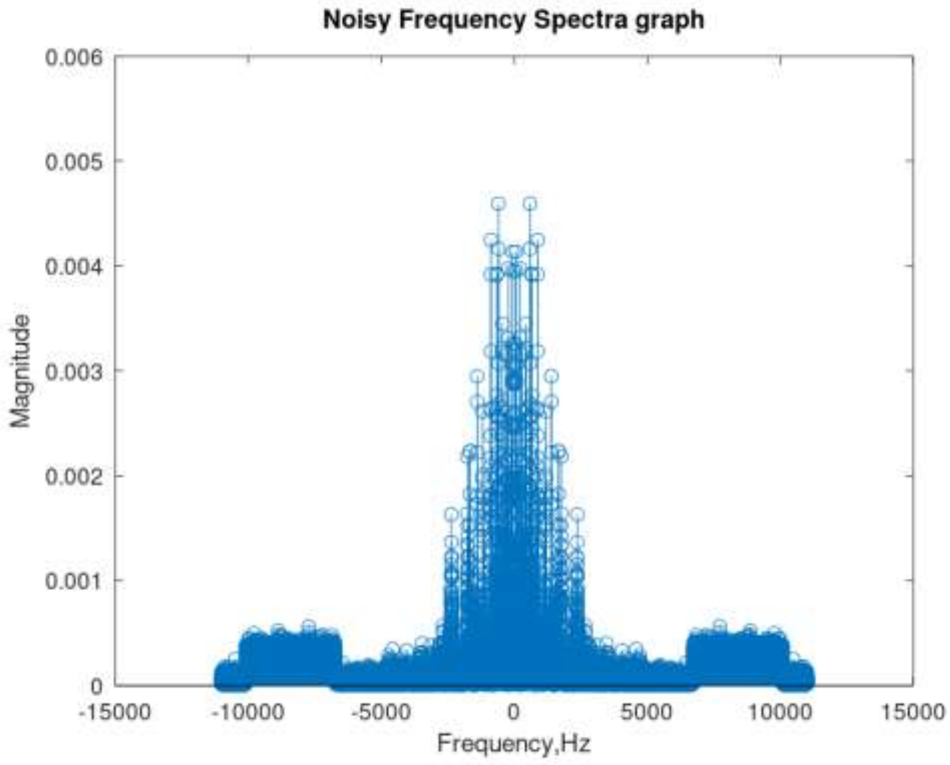

Figure 1.4 shows the Frequency Spectra Graph for noisy_signal.wav

Filter Specification

Analyzing thoroughly Figure 1.3 and Figure 1.4, we could observe the noisy sound playback of *noisy_signal.wav* comes from the higher frequency band. As the *Frequency Spectra Graph* mirrors on the zero frequency, we only consider to observe the positive frequency for simplification. If we observe and compare both plots, there is differences in intensity that resides on the higher frequency band of the *Noisy Frequency Spectra Graph*. This said abnormality would probably corresponds to the noise heard in *noisy_signal.wav*. Therefore, we can assume the said “hissing” noise was embedded between the frequencies of *6729.13Hz* and *10093.7Hz.*

An optimum solution would be a band rejection filter. The filter could attenuate the intensity between this frequencies thus making the hissing noise less noticeable.

Hence, based on the previous data collected, the “hissing” noise in noisy_signal.wav could be refine with bandstop filter of the following specification:-

Type	= FIR Bandstop Filter
Lower cutoff frequency	= 6729.13 Hz
Upper cutoff frequency	= 10093.70 Hz
Sampling Rate	= 22050 Hz
Passband ripple	= 3 dB
Stopband attenuation	=14 dB

However, the filter specification for the lower and upper cutoff frequency is assumed to be plotted in a continuous manner. Thus, it is describe as approximation rounded to the nearest two decimal points. Furthermore, the passband ripple and stopband attenuation could be calculate with the equation below:-

$$\begin{gathered} Passband\ ripple, dB = -20\log_{10}(1-\delta_1); \\ = -20\log_{10}(1-0.3); \\ = 3\ dB \end{gathered}$$

$$\begin{gathered} Stopband\ attenuation, dB = -20\log_{10}(\delta_2); \\ = -20\log_{10}(0.2); \\ = 14\ dB \end{gathered}$$

In which δ_1 and δ_2 is the peak passband ripple and peak stopband ripple respectively.

FIR Filter Design

A rectangular window results in a passband ripple and stopband attenuation of 0.74 dB and 21 dB respectively. Hence, choosing a rectangular window would satisfy the design requirement of the FIR filter. As FIR filter is guaranteed to be stable in most case, we would consider a high tap filter to accommodate a steeper roll-off from the passband to the stopband. Thus, our FIR filter could tolerate a smaller frequency width/transition of 50 Hz. For convenient purpose, we would consider the number of tap is equal to the window length for the chosen window as shown below:-

$$Normalized\ Transition\ Width, \Delta f = \frac{f_{transition}}{f_{sampling}};$$

$$\Delta f = \frac{50}{11025};$$

$$N = \frac{0.9}{\Delta f} = 199\ tap;$$

Therefore, the propose FIR filter specification can be summarize as shown below:-

Type	= FIR Bandstop Filter
Lower cutoff frequency	= 6729.13 Hz
Upper cutoff frequency	= 10093.70 Hz
Sampling Rate	= 22050 Hz
Passband ripple	= 3 dB
Stopband attenuation	=14 dB
Tap	= 199

Using the FIR filter specification above, we can acquire the filter coefficient and frequency response by utilizing the function *fir1()* and *freqz()* respectively as shown in Figure 1.5.

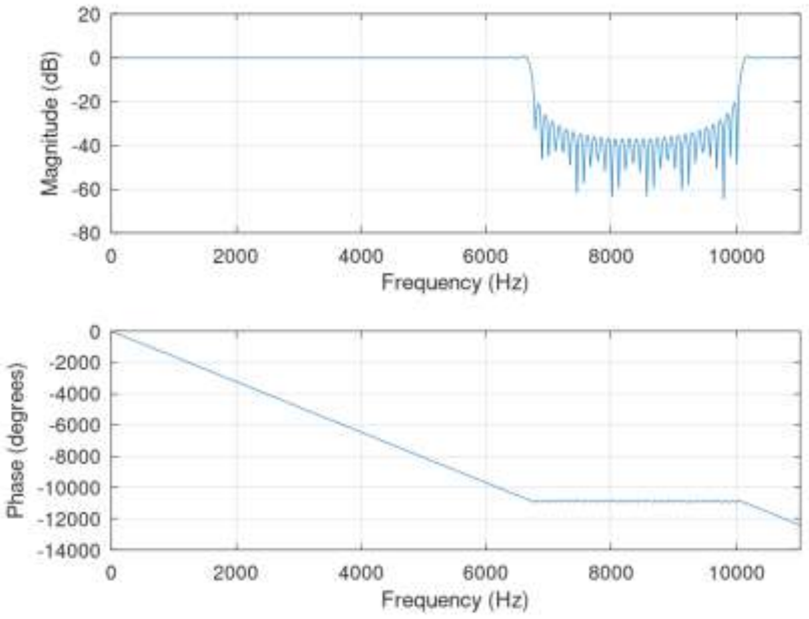

Figure 1.5 shows the frequency response of the FIR filter.

Applying the filter by using the function *filter()*, we can plot the filtered Noisy Signal in Time domain and the Frequency Spectra graph as shown in Figure 1.6 and Figure 1.7 respectively.

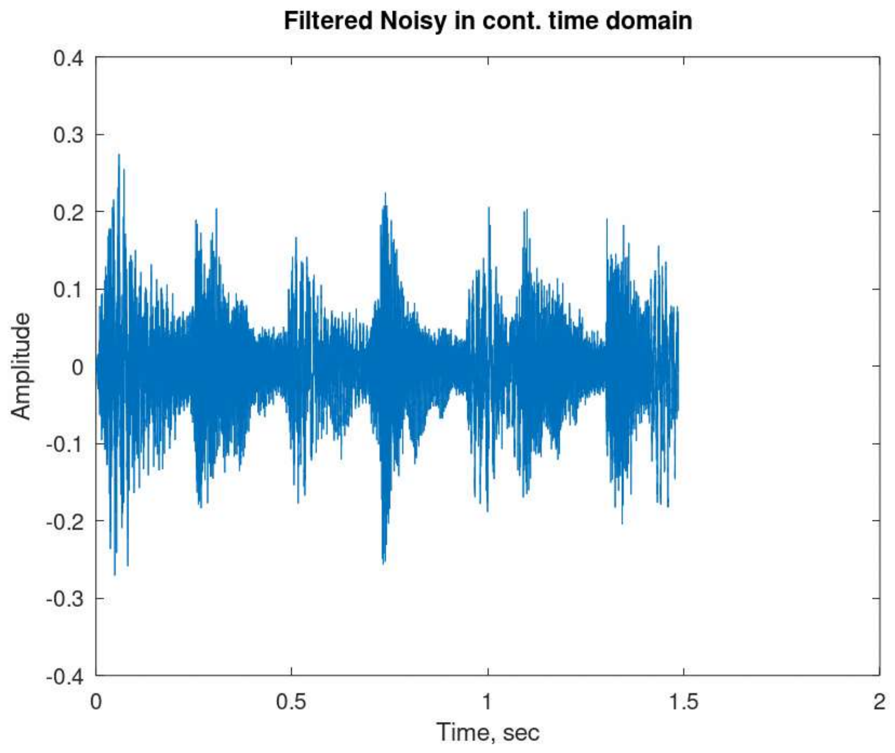

Figure 1.6 shown the filtered Noisy in continuous Time domain FIR filter.

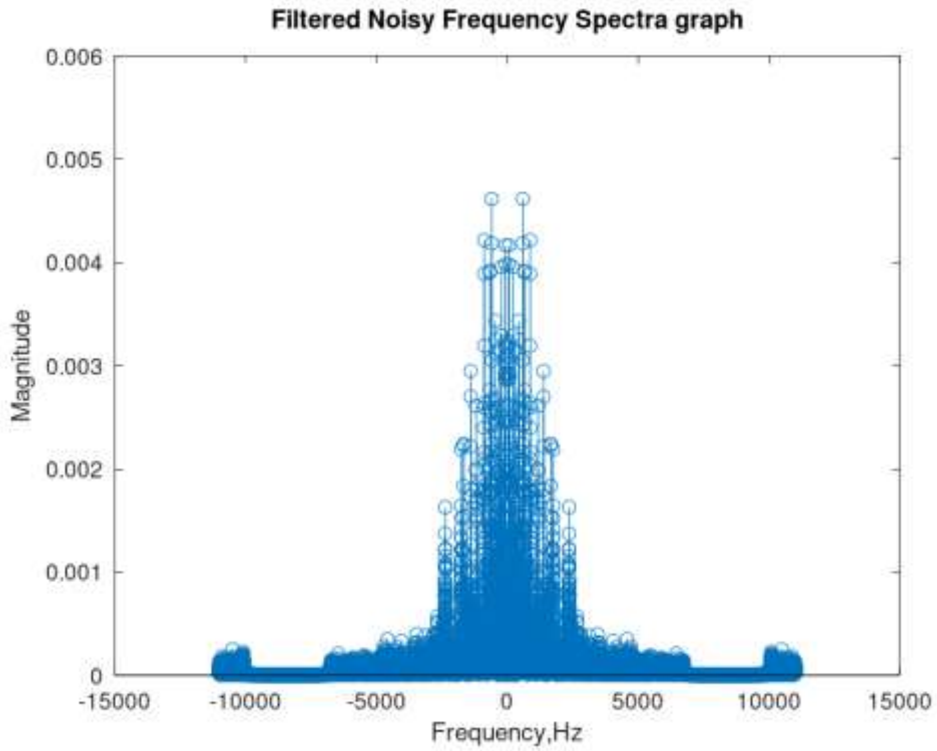

Figure 1.7 shows the filtered Noisy Frequency Spectra graph using FIR filter.

IIR Filter Design

We chose to utilize a Butterworth filter to filter the noise. Although a Chebyshev filter has a steeper roll-off for a smaller frequency width/transition, sound distortion caused by the filter ripples of the passband concerns us. It may cause the sound to be unidentifiable or changed in properties. A Butterworth filter however, has the maximum flat passband ripple for a filter. This is a better solution to only reject the unwanted noise and preserve elsewhere. By utilizing the *buttord(),* we would obtain the recommended Butterworth filter of 13. Therefore, the recommended IIR filter has the following specifications:-

Type	= Butterworth Bandstop Filter
Lower passband frequency	= 6429.13 Hz
Lower stopband frequency	= 6729.13 Hz
Upper stopband frequency	= 10093.70 Hz
Upper passband frequency	= 11025.00 Hz
Sampling Rate	= 22050 Hz
Passband ripple	= 3 dB
Stopband attenuation	= 14 dB
Order	= 13

As higher order Butterworth filter is unstable for digital filters, the following adjustment to tolerate the minimum frequency of 300 Hz and 931.3 Hz is needed for the lower and upper frequency transition respectively. Applying the filter specification by using the *butter()* function, we would obtain the filter frequency response as shown in Figure 1.8 below.

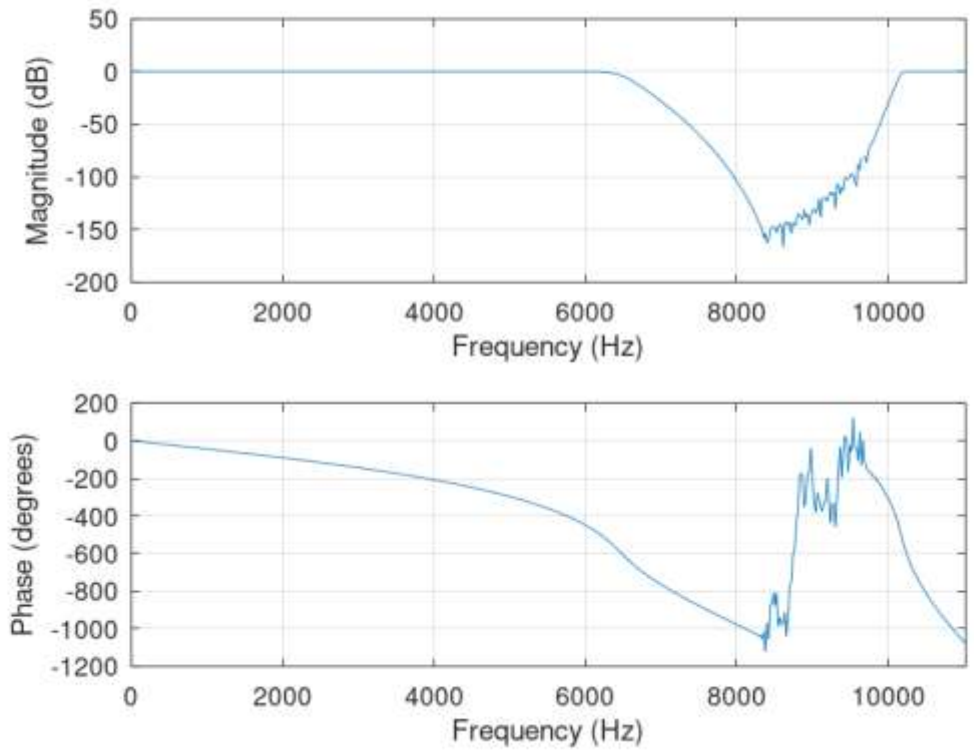

Figure 1.8 shows the Butterworth filter frequency response

Applying the filter by using the function *filter()*, we can plot the *Filtered Noisy Signal in Time domain* and the *Frequency Spectra Graph* as shown in Figure 1.9 and Figure 2.0 respectively.

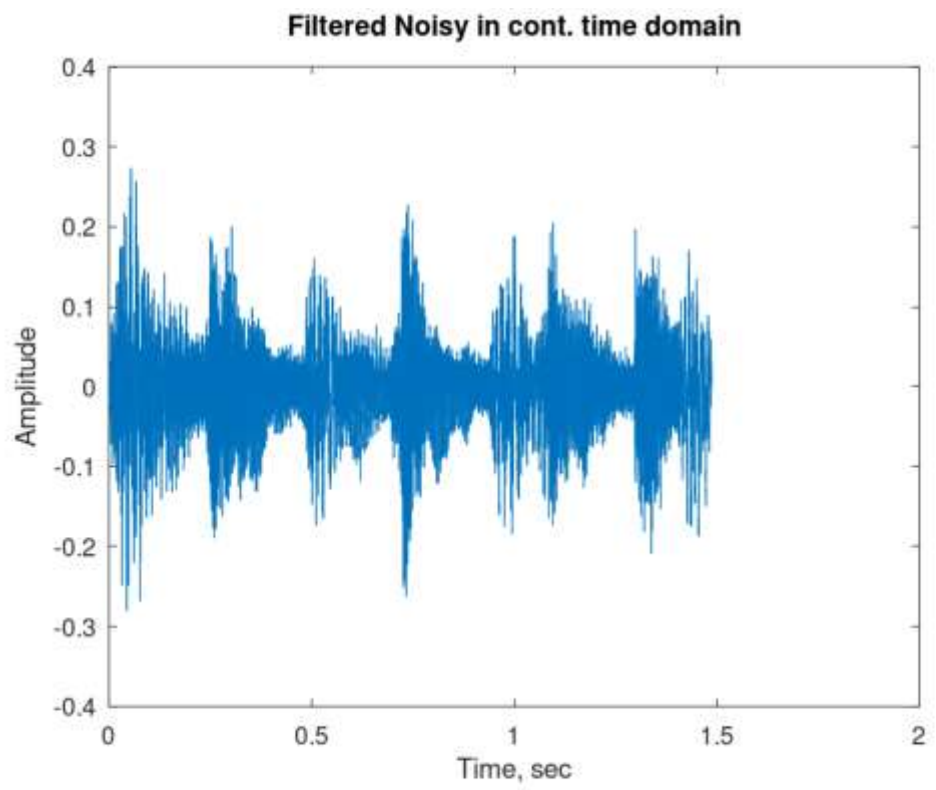

Figure 1.9 shows the filtered Noisy Continuous Time domain using IIR filter

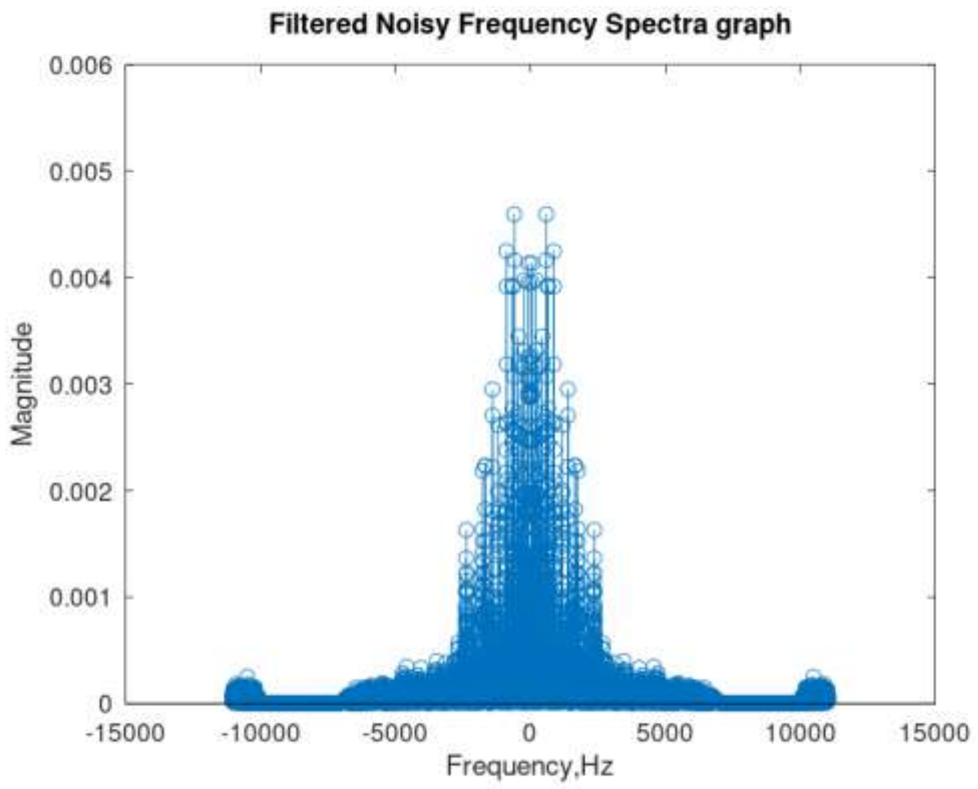

Figure 2.0 shows the filtered Noisy Frequency Spectra Graph using IIR filter

Evaluation of Filtered Signal

The FIR filtered noisy signal has a noticeable difference in both audio and plotted graph. Graphically speaking, our filter has caused a band rejection that satisfy our filter requirement. The *Frequency Spectra Graph* appears to have a large cavity between the frequencies of *6729.8 Hz* and *10093.7 Hz* as shown in Figure 2.1.

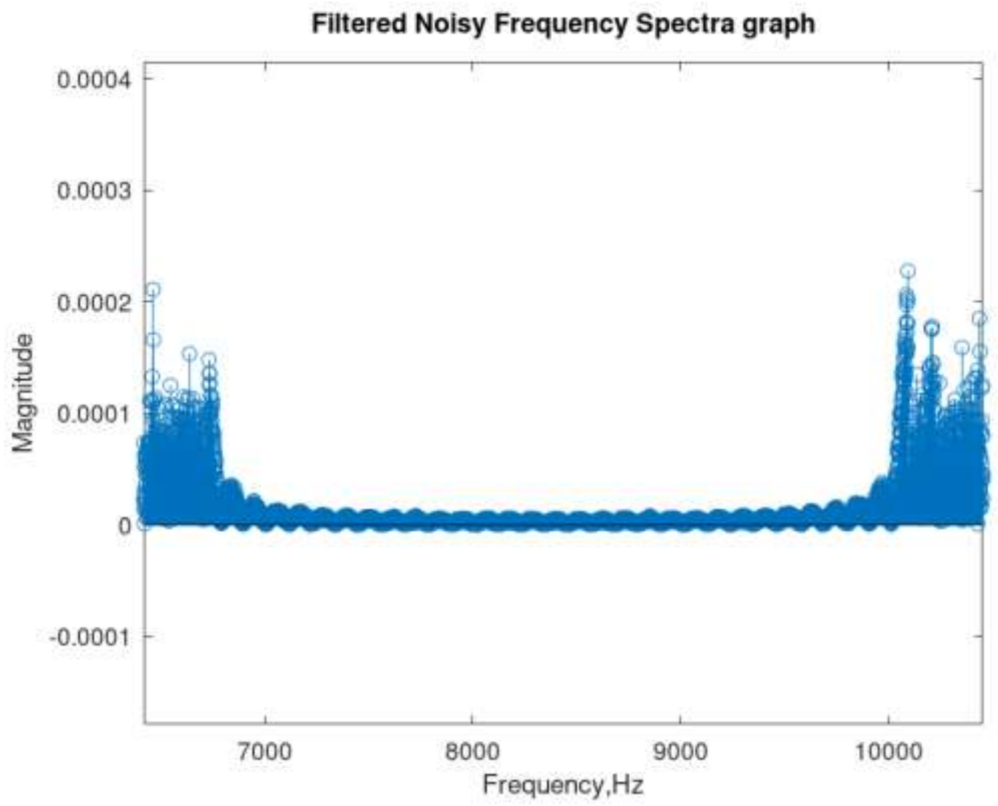

Figure 2.1 shows the cavity caused by FIR filter.

However, comparing the audio of the original noisy signal and the filtered noisy signal, the "hissing" noise sounded muffled with some noise residue. This by means, our design specification for the FIR filter complies with what we are trying to achieve. The noise residue nevertheless, could not be removed completely. This is because the "hissing" noise we heard was embedded in the frequency of *noisy_signal.wav*. Thus, when we apply a bandstop filter, the filter will attenuate the whole frequencies as stated in the filter specification. This would also be a viable explanation to the large cavity we see in Figure 2.1 above. Nevertheless, the *clean_signal.wav* has no noise whatsoever. Furthermore, the Filtered Noisy Signal in Continuous Time domain and the Clean Signal in Continuous Time domain has more similarity compared to Noisy Signal in Continuous Time domain. This would also be evident that we had achieve more similarity towards the clean_signal.wav.

On the other hand, the IIR filtered noisy signal has a noticeable difference in both audio and plotted graph. Graphically speaking, our Butterworth filter also has caused a band rejection that satisfy our filter requirement. Furthermore, the *Frequency Spectra Graph* also appears to have a large cavity between the frequencies of *6729.8 Hz* and *10093.7 Hz* as shown in Figure 2.2.

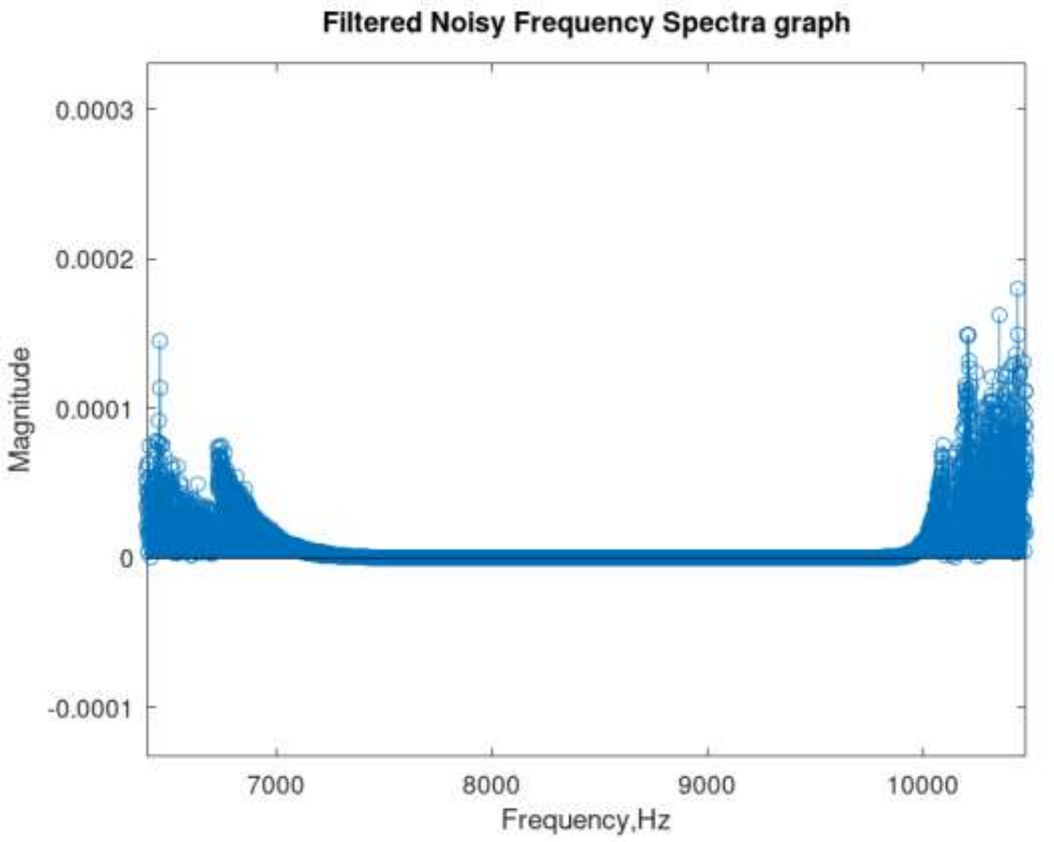

Figure 2.2 shows the cavity caused by the IIR filter.

Despite that, the noise residue could not be removed completely. This is because the noise we heard was embedded in the frequency of *noisy_signal.wav* as explained above. Thus, when we apply a Butterworth bandstop filter, the filter will attenuate the whole frequencies as stated in the filter specification. This also would also be a viable explanation to the large cavity we see in Figure 2.2 above. Moreover, the Filtered Noisy Signal in Continuous Time domain and the Clean Signal in Continuous Time domain has more similarity compared to Noisy Signal in Continuous Time domain. This would also be evident that we had achieve more similarity towards the clean_signal.wav.

Conclusion

A digital audio such as *noisy_signal.wav* and *clean_signal.wav* is considered a digital signal. Therefore, digital filters such as FIR and IIR filter would be applicable to remove unwanted noise. However, in our assignment the noise could not be fully removed as the noise was embedded in the said frequencies. To fully clean the noisy_signal.wav, we can reconstruct the audio signal by each sampling and filtering the noise specifically. This however is not covered in our syllabus.

Appendix

Below is the attached code:-

```
pkg load signal;
close all;
clear;

[ynoisy,fs_noisy] = audioread('noisy_signal.wav');
[y,fs] = audioread('clean_signal.wav');



########################### noisy ###########################################
N_noisy = length(ynoisy);
slength_noisy = N_noisy/fs_noisy; %the length of audio signal in sec

t_noisy = linspace(0, slength_noisy, N_noisy);

figure;
plot(t_noisy,ynoisy); %checking the signal in cont. time domain
xlabel('Time, sec');
ylabel('Amplitude');
title('Noisy Signal in cont. time domain');


%doing DFT to obtain frequency spectra
YNOISY = fft(ynoisy);
Mag_YNOISY = abs(YNOISY);
Mag_YNOISY = fftshift(Mag_YNOISY);

df_noisy = fs_noisy/N_noisy;
w_noisy = (-(N_noisy/2):(N_noisy/2)-1)*df_noisy; % xaxis setting for frequency response graph

figure;
stem(w_noisy, Mag_YNOISY/N_noisy); %normalize to num_samples
xlabel('Frequency,Hz');
ylabel('Magnitude');
title('Noisy Frequency Spectra graph');

############################## clean #########################################
N = length(y);
slength = N/fs;

t= linspace(0,slength,N);

figure;
plot(t,y);
xlabel('Time, sec');
ylabel('Amplitude');
title('Clean Signal in cont. time domain');
```

```
%doing dft to obtain freq spectra
Y = fft(y);
Mag_Y = abs(Y);
Mag_Y = fftshift(Mag_Y);

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

figure;
stem(w, Mag_Y/N_noisy);
xlabel('Frequency,Hz');
ylabel('Magnitude');
title('Clean Frequency Spectra Graph');

############################ Designing FIR filter ##################################
numFreq =512;
tap = 199; %tolerating 50hz of transition width freq for lower and upper
fL = 6729.13; %low cutoff  freq of 6730 Hz
fH = 10093.7; %upper corner freq of 10093.5 Hz
M = tap-1;
Wn_L = fL/(fs_noisy/2); %stop band normalize to fs
Wn_H = fH/(fs_noisy/2); %pass band normalize to fs

Bfir = fir1(M,[Wn_L,Wn_H],'stop',rectwin(tap),'noscale');
figure;
freqz(Bfir,1,numFreq,fs_noisy);

############################  applying filter ######################################
xnoisy = filter(Bfir,1,ynoisy);
figure;
plot(t_noisy,xnoisy);
xlabel('Time, sec');
ylabel('Amplitude');
title('Filtered Noisy in cont. time domain');

XNOISY = fft(xnoisy);
Mag_XNOISY = abs(XNOISY);
Mag_XNOISY = fftshift(Mag_XNOISY);
figure;
stem(w_noisy, Mag_XNOISY/N_noisy);
xlabel('Frequency,Hz');
ylabel('Magnitude');
title('Filtered Noisy Frequency Spectra graph');
```

```
############################ Designing IIR filter ###############################
fstop = [6729.13 10093.7];
fpass = [6429.13 11025]; % fpass = [6429.13 11025];
Rpass = 3;
Rstop = 14;
Wpass = fpass/(fs_noisy/2);
Wstop = fstop/(fs_noisy/2); %normalizing to fs

[order Wn_p Wn_s] = buttord(Wpass, Wstop, Rpass, Rstop);
[B_butt A_butt] = butter(order,Wn_s,'stop');

figure;
freqz(B_butt,A_butt,numFreq,fs_noisy);

###########################  applying filter #####################################
xnoisy = filter(B_butt,A_butt,ynoisy);
figure;
plot(t_noisy,xnoisy);
xlabel('Time, sec');
ylabel('Amplitude');
title('Filtered Noisy in cont. time domain');

XNOISY = fft(xnoisy);
Mag_XNOISY = abs(XNOISY);
Mag_XNOISY = fftshift(Mag_XNOISY);
figure;
stem(w_noisy, Mag_XNOISY/N_noisy);
xlabel('Frequency,Hz');
ylabel('Magnitude');
title('Filtered Noisy Frequency Spectra graph');
```