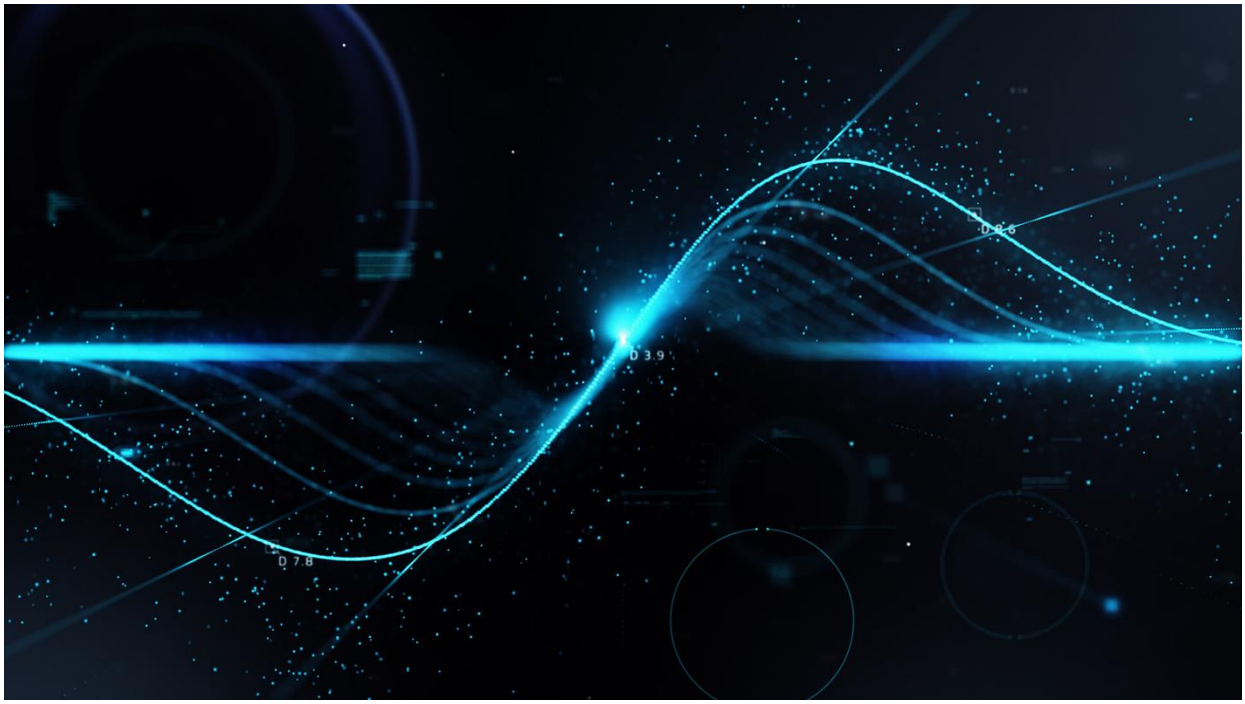
Digital Signal Processing

Digital Filter Design Project

Prepared By: Farida Ahmed, Habiba Hassan, and Haidy Magdy.



Abstract

Going digital is no longer an option, it's the default. ~ Natarajan Chandrasekaran.

Communication is the exchange of information. Information comes in the form of signals that are carried on electromagnetic waves to travel through different mediums, these signals are either Analog (Natural Signals) or Digital (Device Signals). The main difference between these two types is that analog signals are continuous in time and in amplitude, i.e. For every instant of time there is a value for the signal. On the other hand, digital signals represent data in a sequence of discrete values i.e. It can only take one value from a finite set of possible values at a given time. Digital signal processing involves various techniques for improving the accuracy, and reliability of digital communications. This paper discusses the effect of noise on an audio signal, and how to banish it using different types of digital filters on MATLAB.

Contents

List Of Figures

Introduction

Filters are circuits that pass or amplify certain frequencies all the while attenuating other ones. In other words, a filter can extract important frequencies from signals that contain undesirable ones. There are four primary types of filters; these are Low-Pass Filter, High-Pass Filter, Band-Pass Filter, and Band-Stop (Notch) Filter. Filters can be categorized into passive and active according to their components. Passive components may be resistors, capacitors or inductors, while active ones include Op-amps. Some applications that strictly acquire filter usage are: Radio communications, DC power supplies, and analog to digital converters. Response curves are used to describe how a filter behaves. They are plots that show the attenuation in decibels (dB) vs frequency. There are two fundamental types of digital filters: FIR (Finite Impulse Response) and IIR (Infinite Impulse Response). Both of these can come in the four primary forms mentioned above. Further more information about these two types are to be discussed later. In this project, different types of noise are applied to an audio signal, and it is required to use FIR & IIR filters to recover the original signal again.

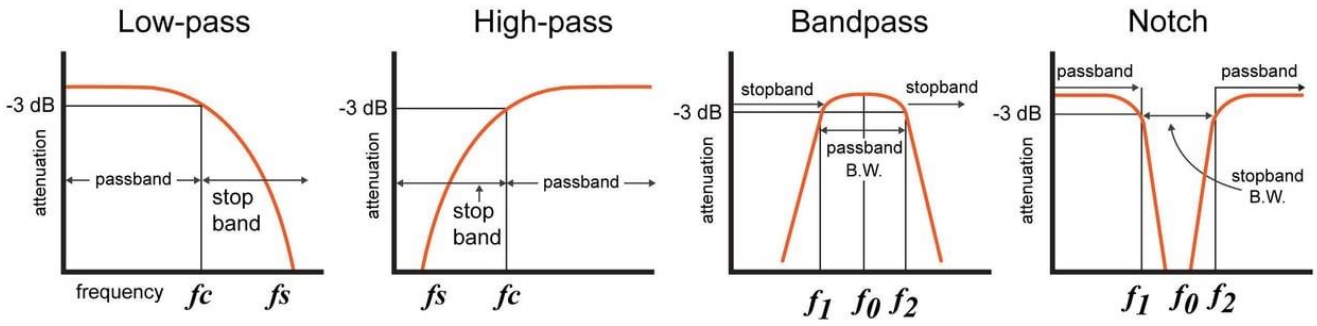


Figure 1: Primary Filter Types

FIR Filter Using Basic Principles

- Over view

This task required the design of an FIR (Non-Recursive) filter to remove the noise at 60 Hz frequency. This means that a band stop FIR filter is needed, with its notch exactly at the affected place. This can be noticed upon switching to the frequency domain as it is more effective in showing the exact noise place.

➢ Objectives

To observe the effect of 60 Hz noise on an audio signal

To design an FIR filter and see how it behaves at a small order i.e. N=3.

➢ Filter Design Steps

1) Transform the signal into the frequency domain using Fast Fourier Transform. It is important to note that upon switching from time domain to frequency domain, the signal values become complex, so we get the magnitude of the FFT transformed signal only.

2) It is not necessary to count the –ve frequency scale as it is just an image to the positive one.

3) Note that the step size is equal to the number of samples divided by the sampling frequency, this is how we get the frequency scale from 0 to fs, also note that we subtract one step in our MATLAB code because sample 0 is already counted.

4) The impulse response h(n) of an FIR filter is = The feed forward coefficient only, meaning there is no recurrence (Feedback), or in other words, a previous output does not get to be an input to the next one.

5) The filter is then applied to achieve the required impulse response.

6) Finally, the original, the noisy, and the filtered signal are plotted.

➢ Filter Design Analysis

H(n)=σ(n)-2cosGↃ0 σ(n-1)+ σ(n-2) *Impulse Response*

Y(n)=X(n)-2cos(2πf)X(n-1)+X(n-2)

Remember that:

$H(e^{j\omega}) = \sum_{n=0}^{\infty} h(n)e^{-nj\omega}$ *Fourier Transform*

$Z = e^{j\omega}$, *so H(z) =* $H(e^{j\omega})$

$H(z) = \frac{Y(z)}{x(z)} = 1-2\cos(2\pi f)z^{-1}+z^{-2}$ *Transfer Function*

$H(z) = 1-2\cos(2\pi f)z^{-1}+z^{-2}$

$H(z) = 1-2z^{-1}+z^{-2}$

➢ Plots Comparison

To start with, The original signal plot in the frequency domain is shown down below:

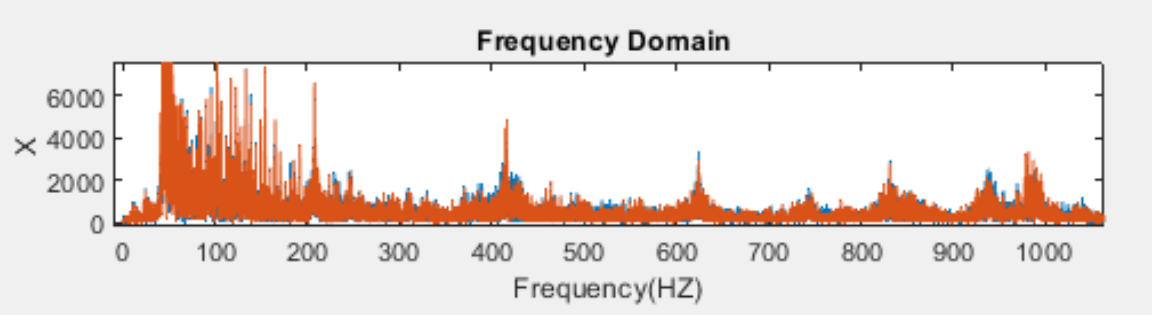


Figure 2 Original Signal Frequency Domain

The contaminated signal comes next with the noise at 60 Hz:

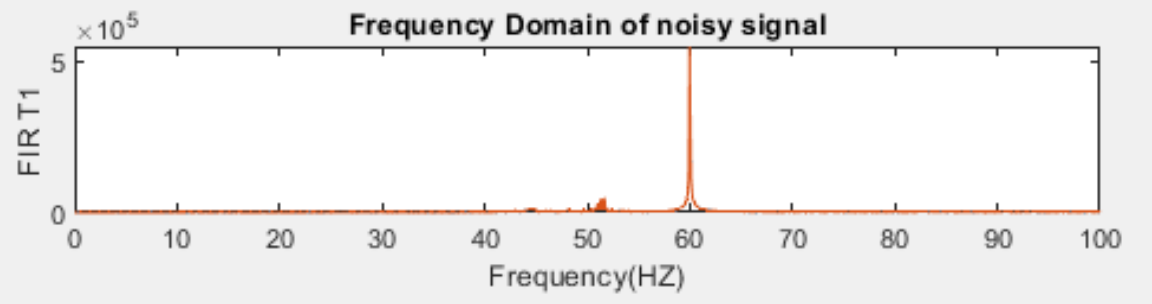


Figure 3 Noisy Signal Frequency Domian

As shown above, the needed filter is obviously a band stop (Notch) filter: Shown below is the filter response curve. The filter will minimize the song's original amplitude to approximately -90 dB.

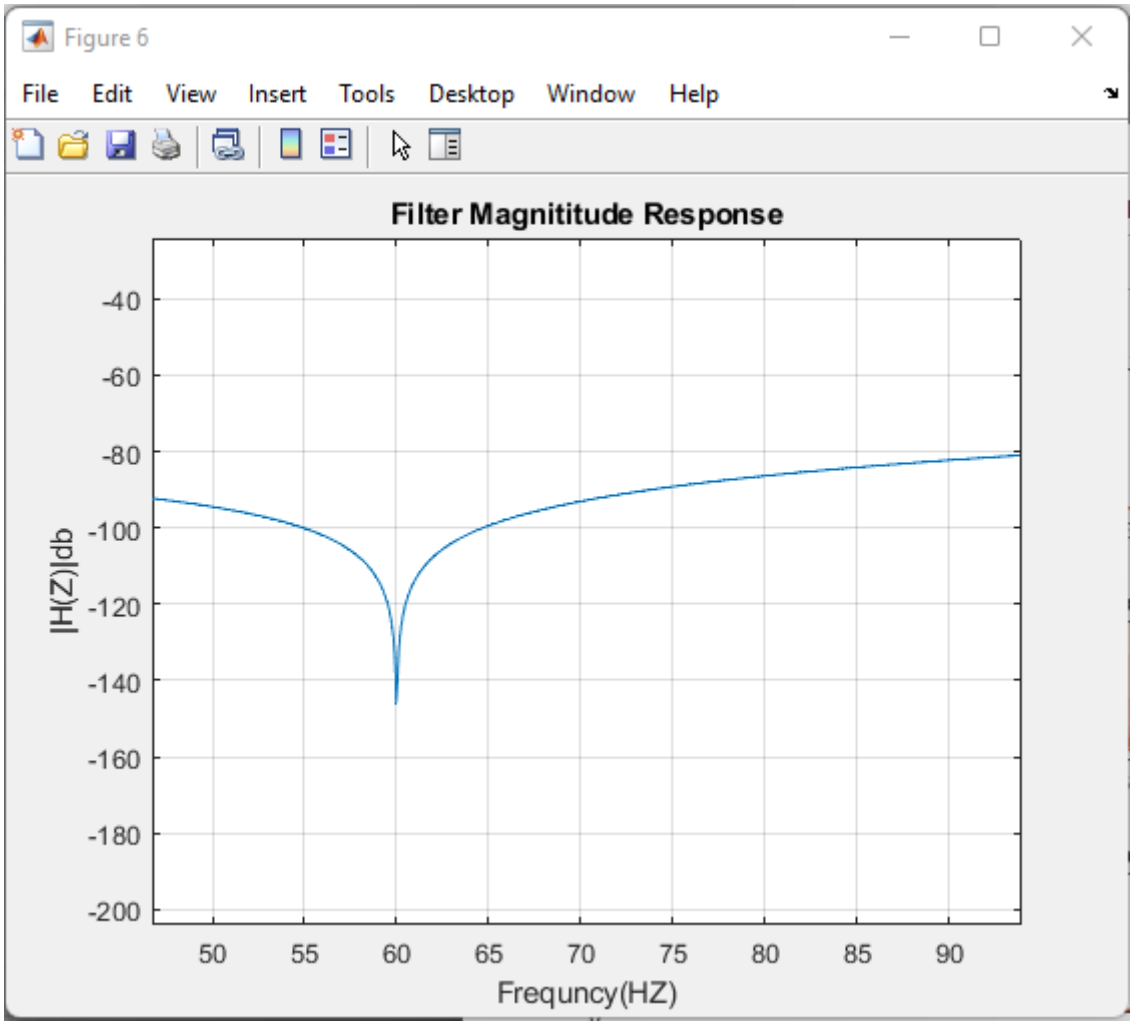


Figure 4 FIR Frequency Response

Upon applying the filter to the contaminated signal, these are the obtained results from a close view:

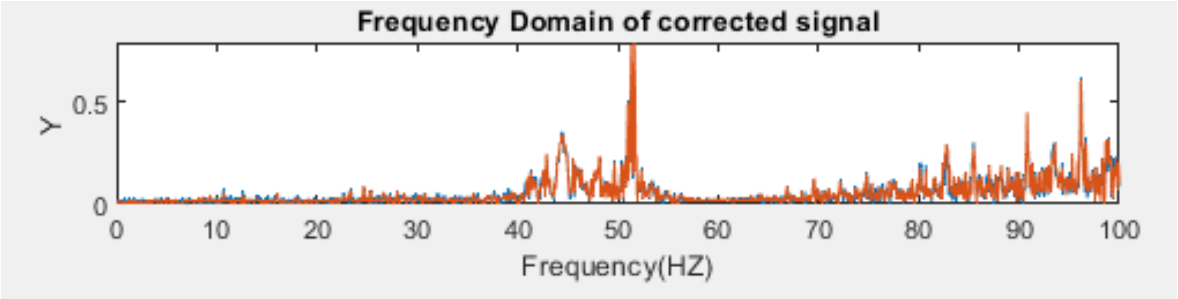


Figure 5 FIR Filterd Signal

A final comparison to wrap up is shown next:

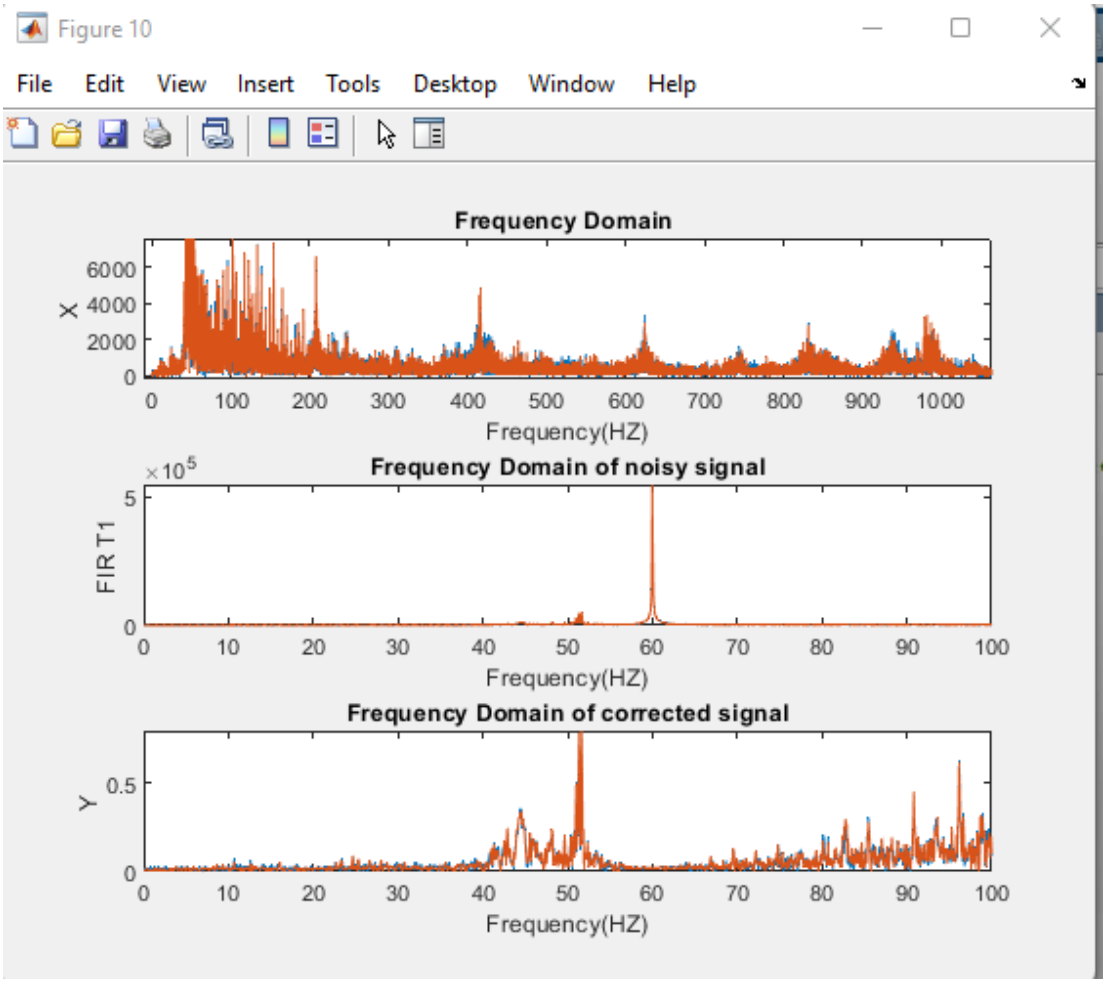


Figure 6 FIR Comparison Between Signals

To finalize, the expected results of applying this filter are not very promising because of the low order we've used. A better approach is yet to follow.

FIR Filter Using Window Technique

- Overview

This task required the design of an FIR (Non-Recursive) filter using a window to remove noise.

- Objective

To test the effect of increasing the filter order and using the window in eliminating the noise from audio signal.

- Filter Design Steps

Firstly, we needed to know which kind of noise we had before we could design the filter. And to know that we used the frequency domain plot of the noisy signal.

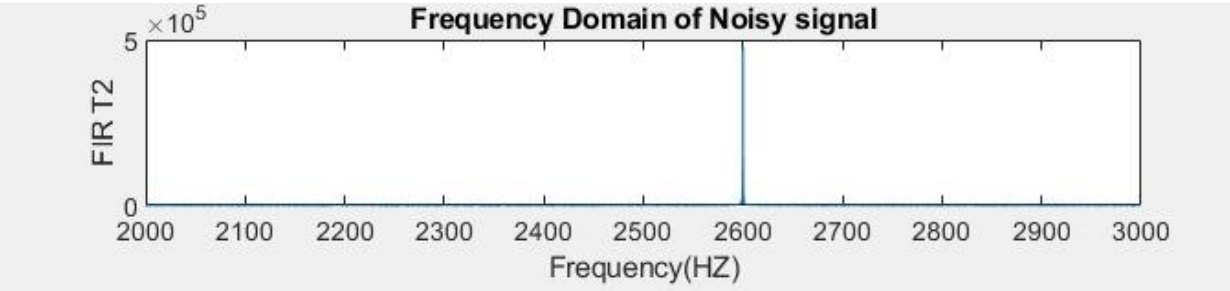


Figure 7 FIR Task 2 Noisy Signal

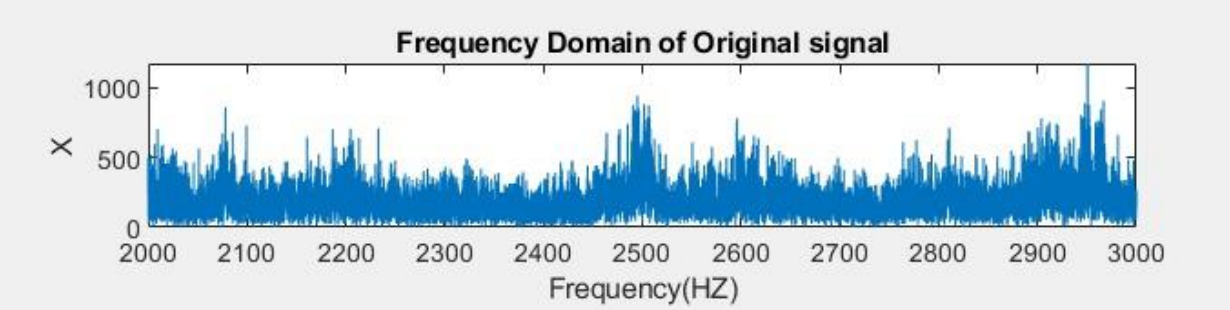


Figure 8 FIR Task 2 orginal signal

And by comparing it with the original song, we found that the noise is at 2600 Hz and it's type high frequency noise. Consequently, a band stop filter will be used.

It was stated in the project description that the required stop band attenuation should not be less than 50 dB. By knowing this, we chose to use the Hamming window technique.

For Hamming window, this is the formula that gets the filter order:

$$\frac{3.3}{N} = \frac{Transition\ Band}{fs}$$

We'll apply this formula in two cases:

- **Case 1 Transition band = 500 Hz**

$$\frac{3.3}{N} = \frac{500}{48000}$$

$\boldsymbol{N = 317}$By multiplying the filter impulse response to the hamming window, this is the magnitude response that we get at an order of 317.

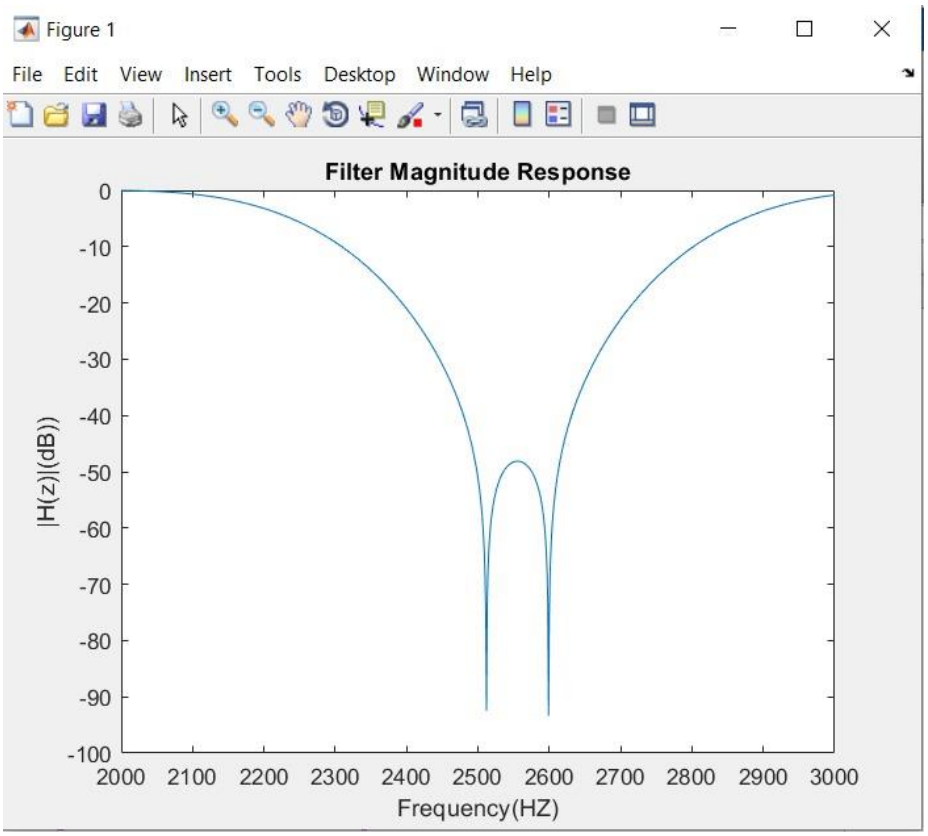


Figure 9 FIR Task 2 Frequency Response

As shown in the figure below that the noise was completely reduced and we won't listen to the noise in the sound file we extracted.

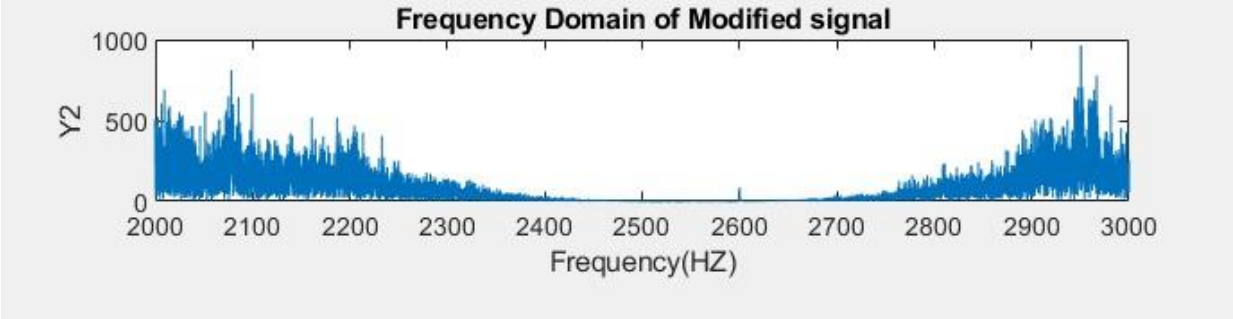


Figure 10 FIR Task 2Filtred Signal

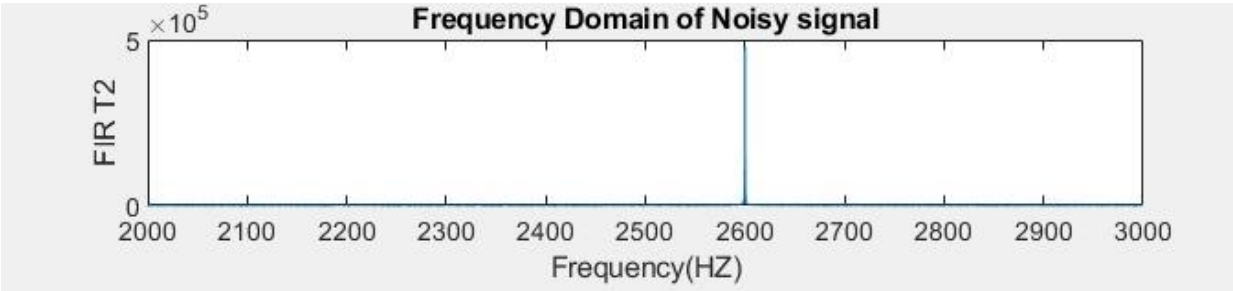


Figure 11 FIR Task 2 Noisy Signal

- **Case 2 Transition band = 750 Hz**

$$\frac{3.3}{N} = \frac{750}{48000}$$

$$\boldsymbol{N = 212}$$

By multiplying the filter impulse response to the hamming window, this is the magnitude response that we get at an order of 212.

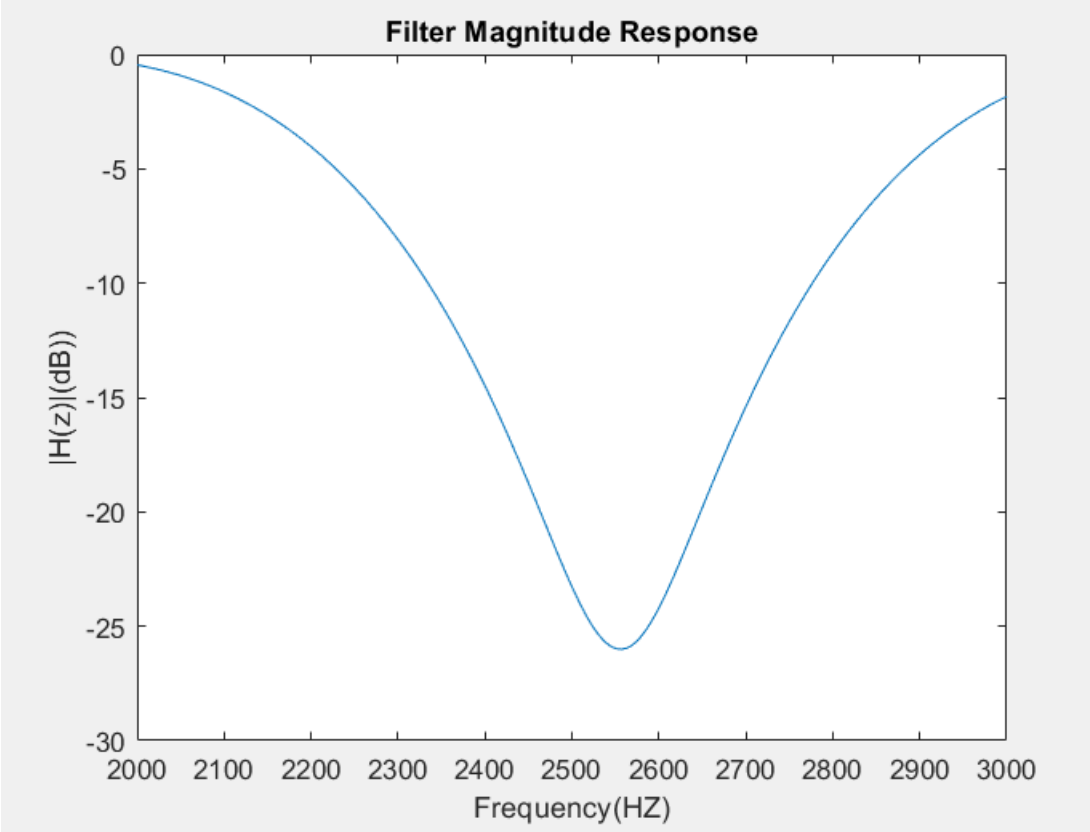


Figure 12 FIR Task 2 Frequency Response 700Hz

As shown in the figure below that the noise cropped and still there with smaller amplitude but we can still hear it after applying the filter.

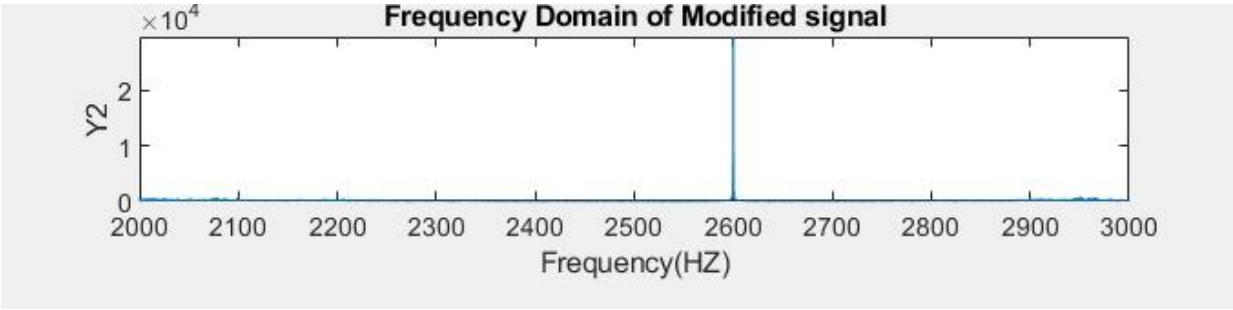


Figure 13 FIR Task 2 Output for 700 Hz Trans. Band

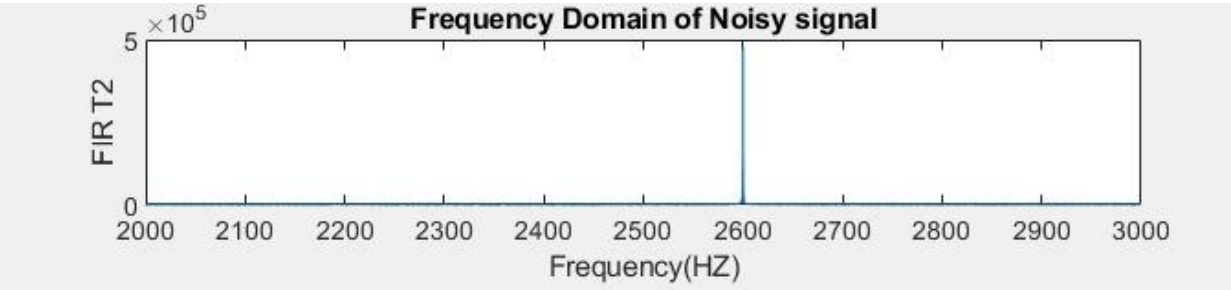


Figure 14 FIR Task 2 Noisy Signal

By observing the two cases, we found that the transition band 500 Hz has the higher order and will be more efficient.

- *Band Stop Filter Analysis*

Before the analysis please remember that:

$$\sin(x) = \frac{e^{jx} - e^{-jx}}{2j}$$

$$\mathrm{Cos}(x) = \frac{e^{jx} + e^{-jx}}{2}$$

$$\mathrm{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$

$$h_d(n) = \frac{1}{w_s} \int_{-\frac{w_s}{2}}^{\frac{w_s}{2}} H_d e^{j\omega} e^{j\omega n T_s} \, d\omega$$

$$= \frac{1}{w_s} [\int_{-\frac{w_s}{2}}^{-w_{c2}} e^{j\omega n T_s} \, d\omega$$

$$+ \int_{-w_{c1}}^{w_{c1}} e^{j\omega n T_s} \, d\omega$$

$$+ \int_{w_{c2}}^{\frac{w_s}{2}} e^{j\omega n T_s} \, d\omega]$$

By Applying the integration:

$$= \frac{1}{jnT_s}[e^{j\omega nT_s}]_{-\frac{w_s}{2}}^{-w_{c2}}$$

$$+ \frac{1}{jnT_s}[e^{j\omega nT_s}]_{-w_{c1}}^{w_{c1}}$$

$$+ \frac{1}{jnT_s}[e^{j\omega nT_s}]_{w_{c2}}^{\frac{w_s}{2}}$$

$$= \frac{f_s}{j2\pi nf_s}([e^{-jw_{c2}nT_s} - e^{-j\frac{w_s}{2}nT_s}]$$

$$+ [e^{jw_{c1}nT_s} - e^{-jw_{c1}nT_s}]$$

$$+ [e^{j\frac{w_s}{2}nT_s} - e^{jw_{c2}nT_s}])$$

$$= \frac{1}{j2\pi n}(e^{-j2\pi n(f_{c2}/f_s)}$$

$$- e^{-j\pi n}$$

$$+ e^{j2\pi n(f_{c1}/f_s)}$$

$$- e^{-j2\pi n(f_{c1}/f_s)}$$

$$+ e^{j\pi n} - e^{j2\pi n(f_{c2}/f_s)})$$

Finally,

$$h_d(n) = \frac{1}{\pi n}(\sin(\pi n) + \sin\left(2\pi n(\frac{fc1}{fs})\right) - \sin\left(2\pi n(\frac{fc2}{fs})\right)$$

Note that :

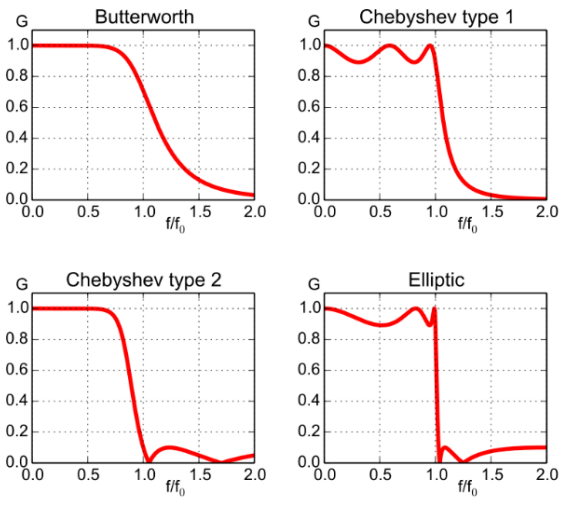
Where f_{c1} is the beginning of the cutoff band , and f_{c2} is the end of the cutoff band.

IIR Filter Design

➢ Overview

In this task, the required filter to design is the recursive IIR that is known for its effectiveness in noise elimination even with little order due to the feedback it's distinguished with. Having a feedback means that the past output can be used as an input to generate a new output. In this task, the noise is located at a low frequency, precisely at the origin, consequently, a High-Pass Filter is required to banish it. There are many types of IIR filters, some of these are:

- **Butterworth Filter**: This one has smooth pass and stop bands i.e. No ripples, however its transition band is relatively wide compared to some of the others. It is considered the simplest in its design with no complicated formulas to get its order.
- **Chebychev Type 1 Filter:** This one has ripples in its passband, yet its stop band is smooth. Its transition band is narrower than that of the Butterworth filter. But, why are ripples considered a problem ? These fluctuations can be heard in the filtered signal, that's to say, they affect the digital filter performance.
- **Chebychev Type 2 Filter:** It has no ripples in its pass band, however, it has ripples in its stop band instead. Its transition band is a bit similar to that of cheb 1.
- **Elliptic Filter:** This has the narrowest transition bandwidth (closest to ideal), yet it has ripples in both the pass band & the stop band. Its order formula is complicated as it's based on series and on data that's set in tabular forms.
- **Bessel Filter:** It has the widest transition band of all mentioned. Its stop and pass bands are smooth like the butterworth filter.



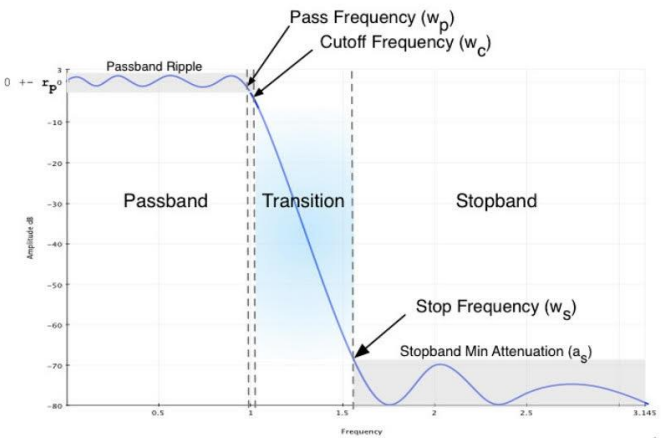


Figure 15 Different Filter Bands

➢ Objective

To observe the behaviour of the IIR filter towards noisy audio signals.

To apply the Bilinear Transformation method analysis in order to design the Butterworth filter manually.

To test & compare the effect of different IIR filter types.

➢ Filter Design Steps

The procedure is previously and in details mentioned in task one, however, the difference here is that Recursive (IIR) filters' transfer functions H(z) have feed forward and feedback coefficients i.e. denominator is not = 0.

Remember: H(z) = b/a. where b is an array holding the feed forward coefficients, and a is the array holding the feed back ones.

But before going further with H(z), what does the original signal look like ? where is the noise located ? and last but not least, how does it sound like ?

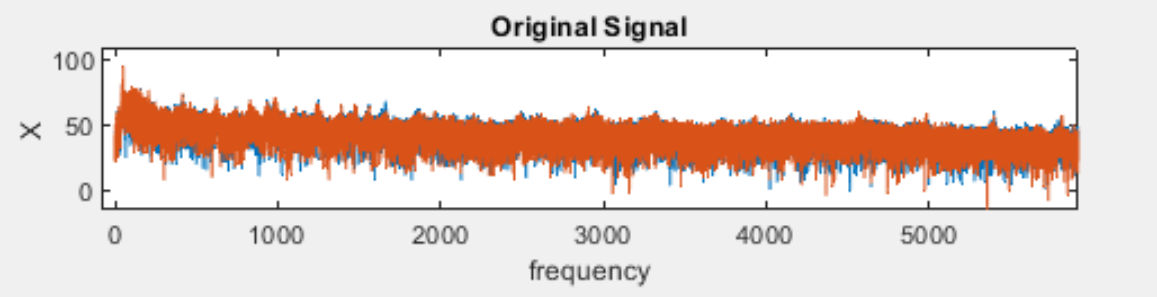


Figure 16 Original Signal

As the title implies, this is how the original signal looks like.

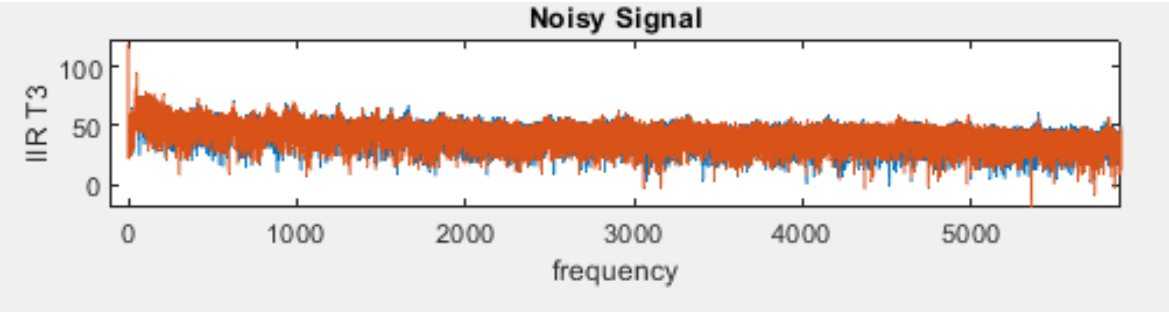


Figure 17 Noisy Signal At Origin

Though hard to notice, being around the origin in the low frequency, the noise sounds like the signal is stuttering. This is definitely a High-Pass filter type of noise.

To make sure of our noise location, we called for the spectrum Analyzer to come to the rescue:

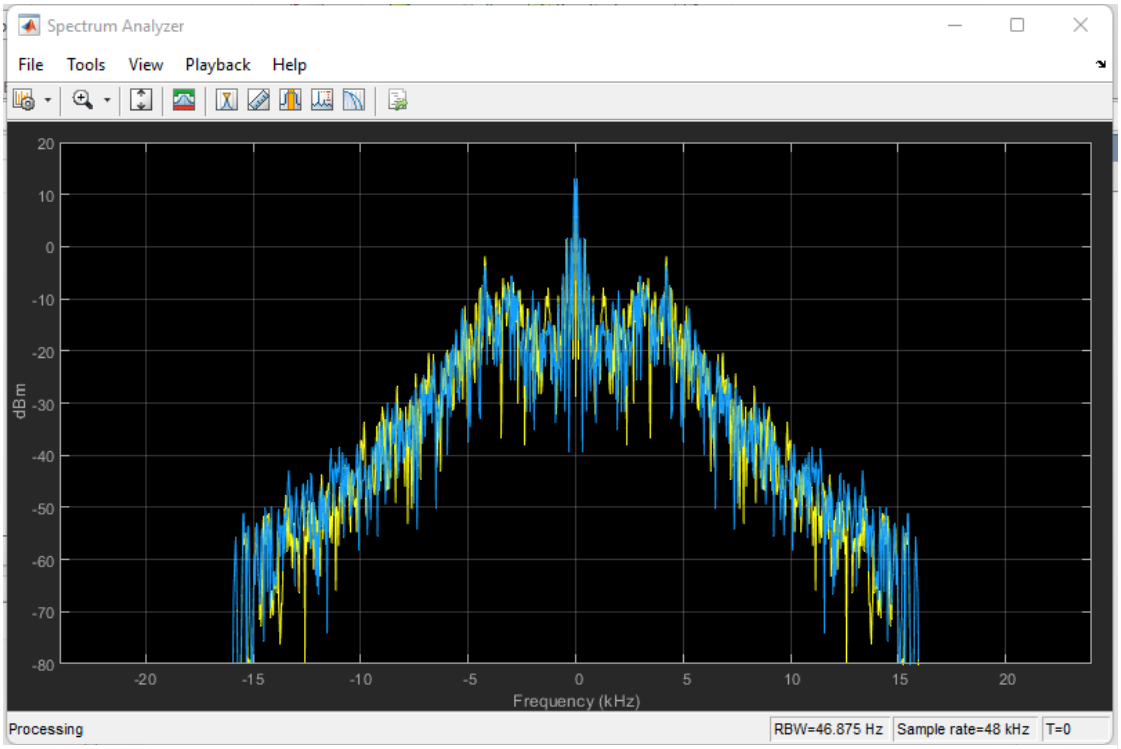


Figure 18Original Signal Spectrum Analyzer

Our original signal shown on the spectrum analyzer (Amplitude in dBm VS frequency in KHz)

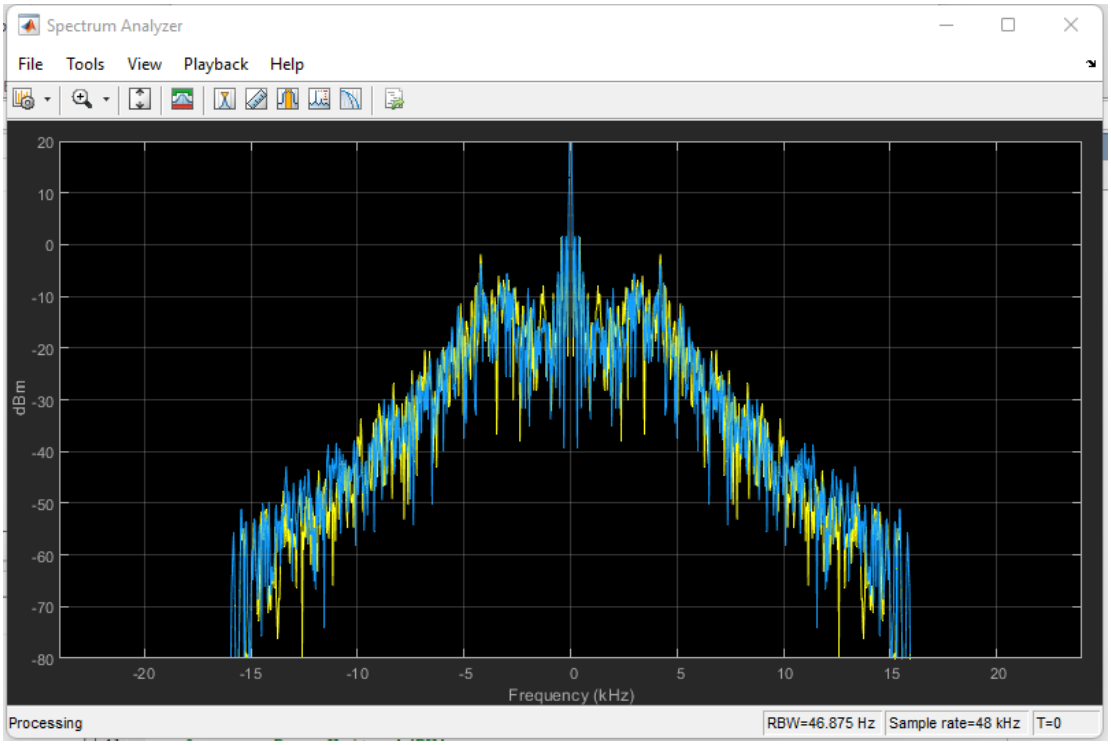


Figure 19Noisy Signal Spectrum Analyzer

Notice the glitch at the 0 Hz frequency ? Exactly this is where our noise is located.

Not enough of a proof for the noise location ? we've also plotted our signal using the spectrogram and the periodogram as shown below:

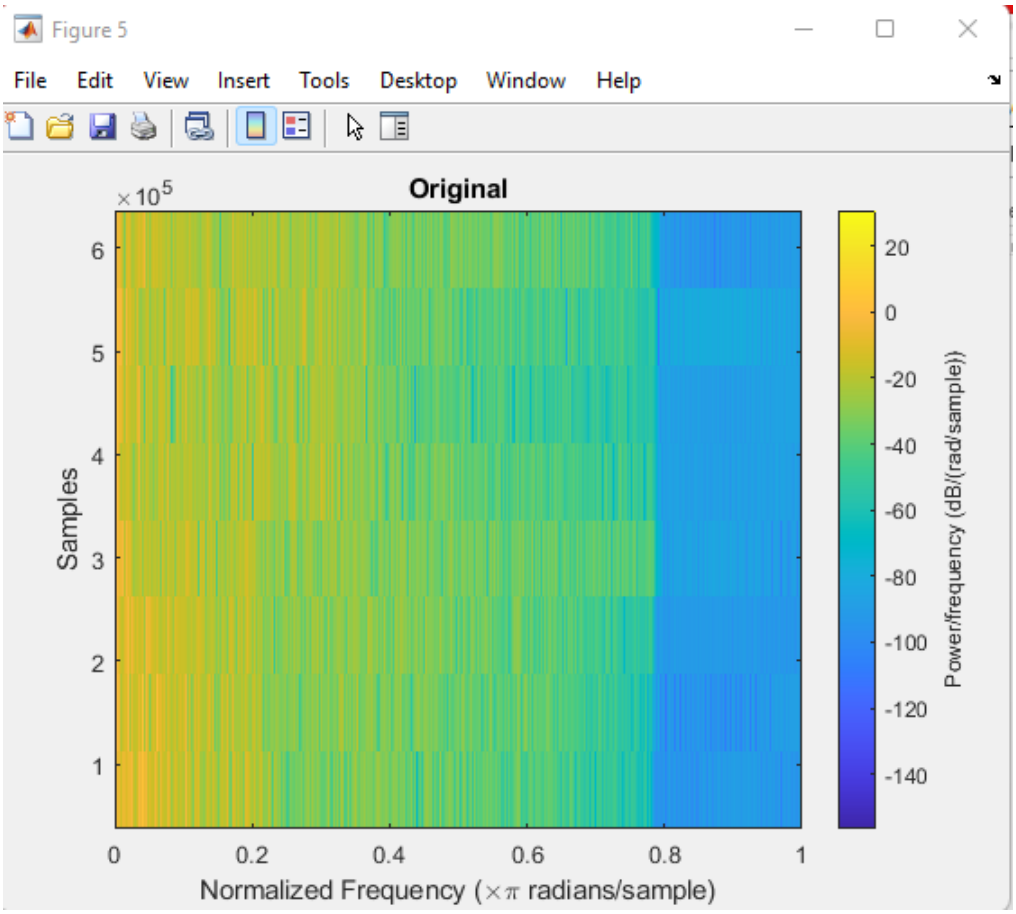


Figure 20 Spectrogram Original Signal

Spectrograms show the Fourier transform of the input vector. We had to put our signal in the form of a vector in order for it to work. Time increases from left to right, and frequency increases from up to down all the while decreasing the signal power, so we can expect that the low frequency part (Yellow) will be affected by the noise addition as shown below:

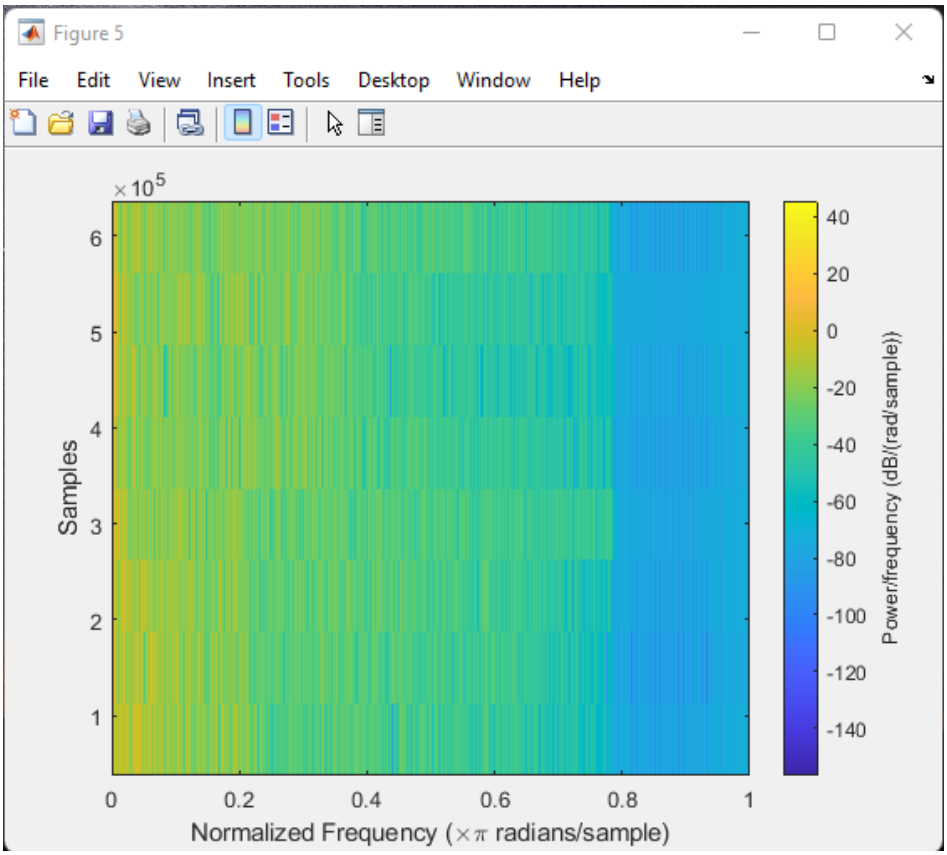


Figure 21Spectrogram Noisy Signal

Finally, the periodogram which is used to show the power spectral density of the signal. It computes the PSD of every column of the input matrix and stores it in the corresponding column of the output matrix.

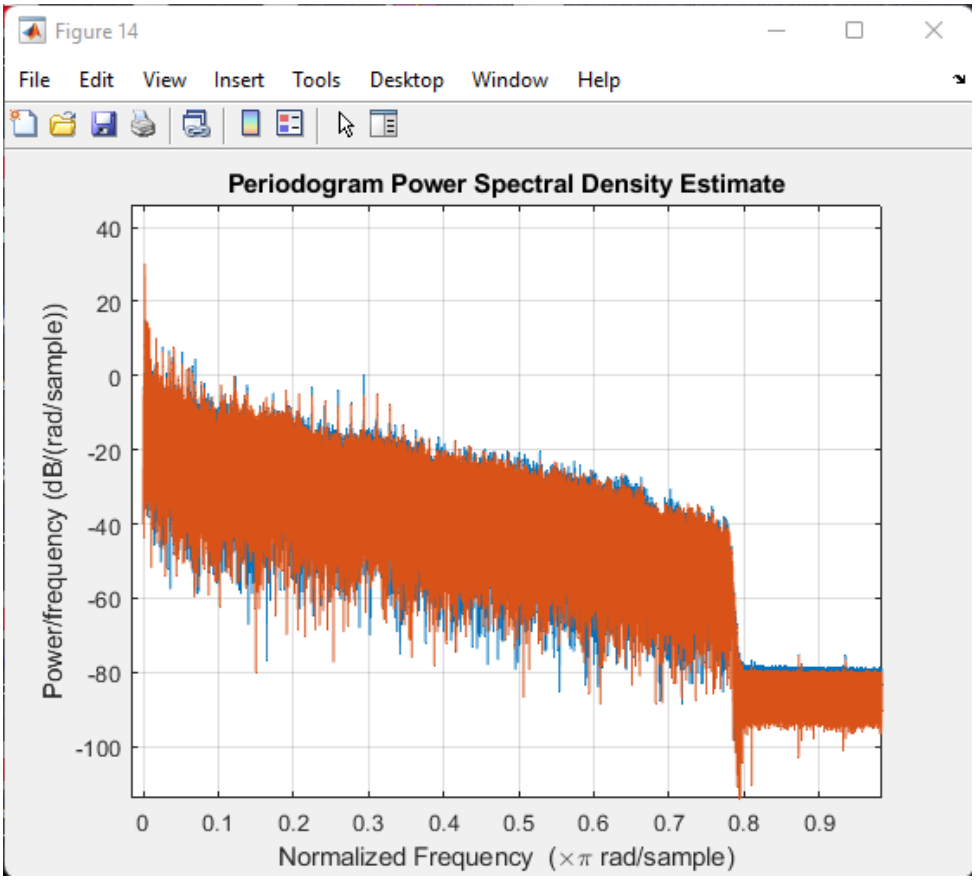


Figure 22 Periodogram Original Signal

Notice the power at low frequency is about 30 dB/(rad/sec) for the original signal and is about 55 dB/(rad/sec) for the Contaminated one.

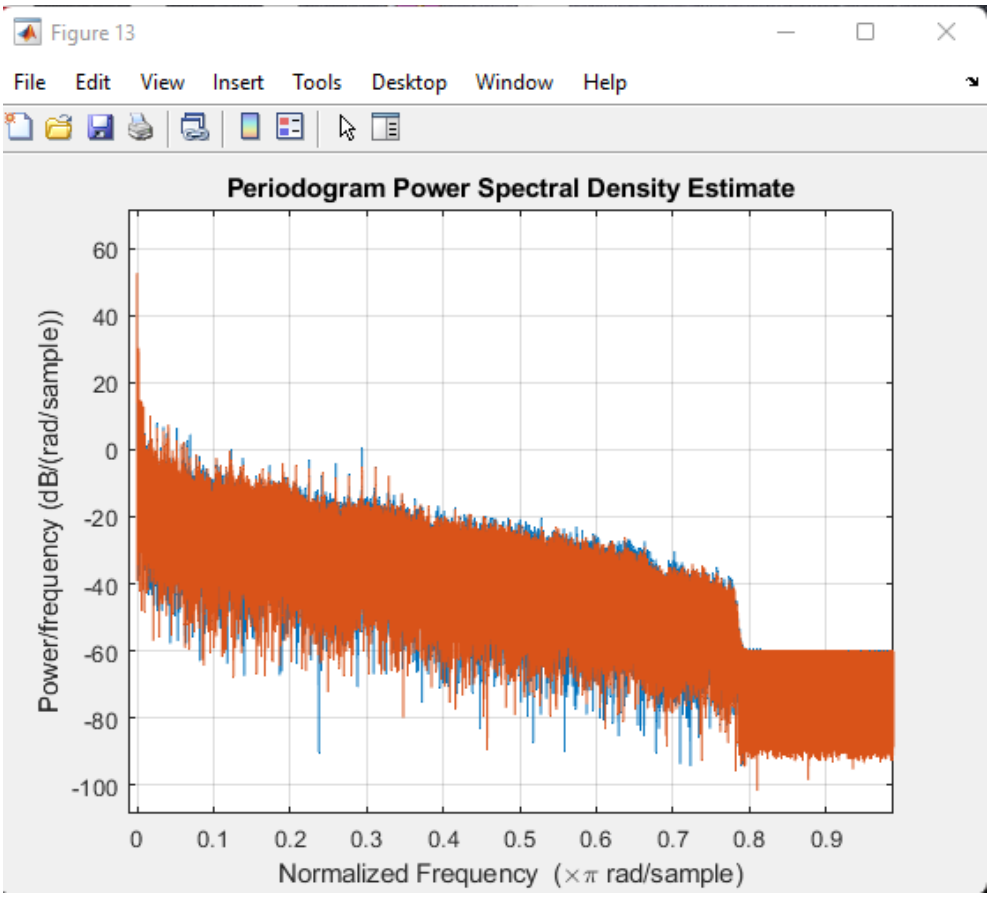


Figure 23Preiodogram Noisy Signal

Enough with the plotting, now this is how we dealt with the noise:

- First Approach: Using The Elliptic Filter Function

By calculating our pass band edge and stop band edge angular frequencies, and normalizing them where:

- Our stop band: From 0 to 50 Hz
- Our transition band: 50+500 or 50 +750 Hz
- Our Pass band: From 550 or from 800

and by choosing to follow the MATLAB suggestion of choosing the pass band ripples to be = 0.5, the ellipord function got us our order as well as the elliptic natural frequency needed for the ellip function to do its job of

designing the filter this is the designed elliptic filter for the 500 Hz Transition band:

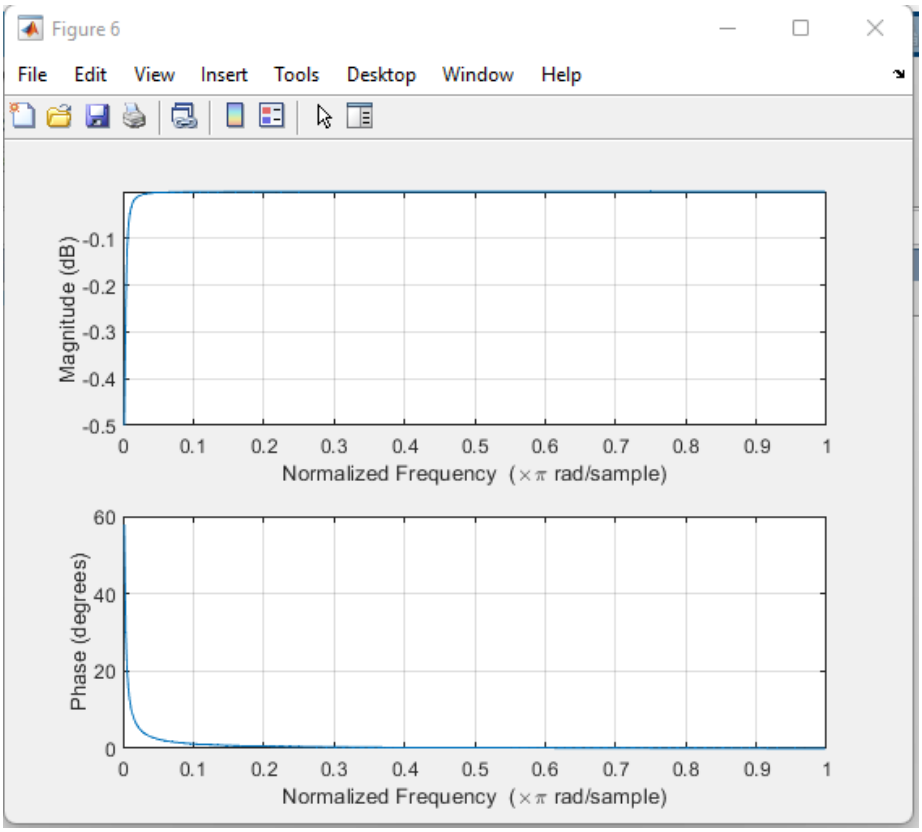


Figure 24 Elliptic Filter Frequency Response

- Second Approach: Using The ChebyChev 1 Filter Function

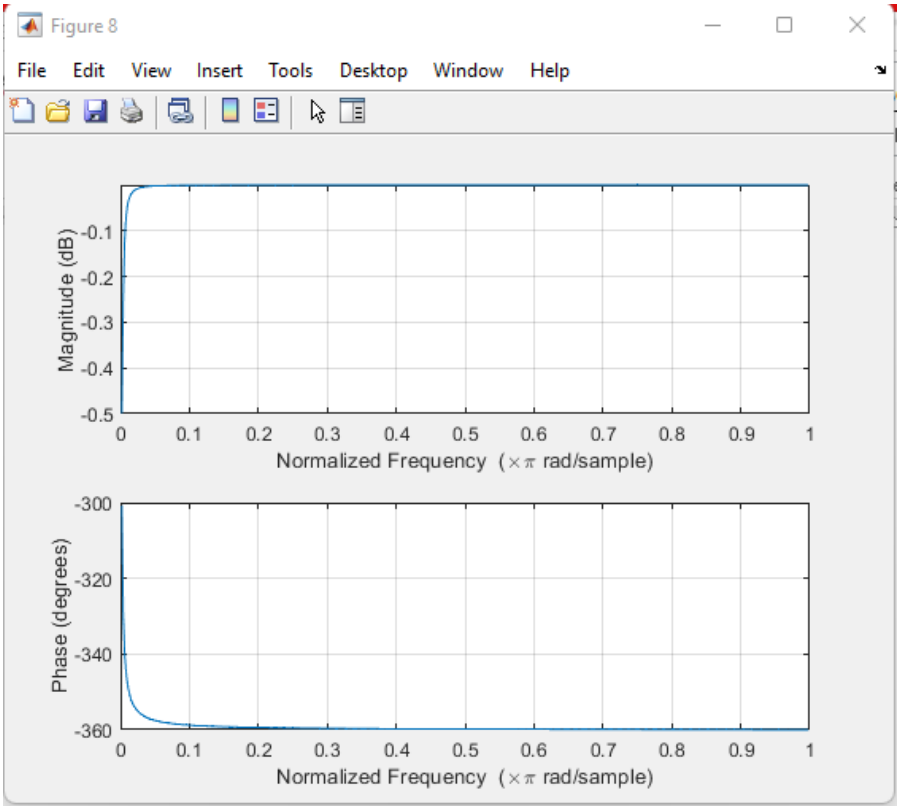


Figure 25ChebyChev 1 Frequency Response

- Third Approach: Using The ChebyChev 2 Filter Function

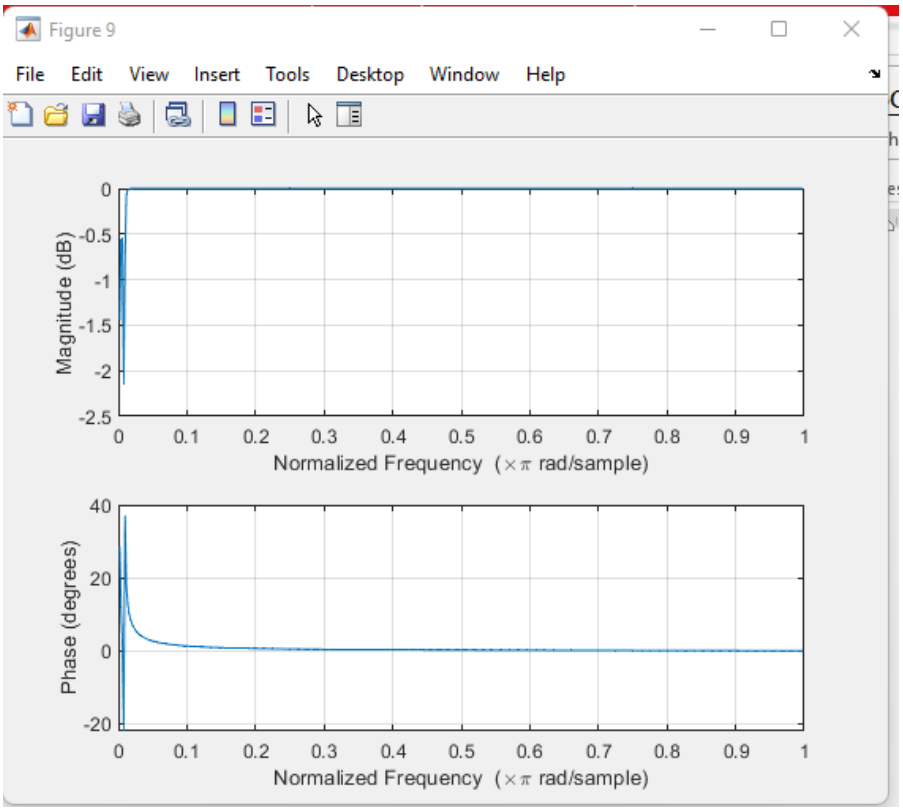


Figure 26 ChebyChev 2 Frequency Response

Can you notice how the stop band looks like ?

- Fourth Approach: Using The Butterworth Filter Function

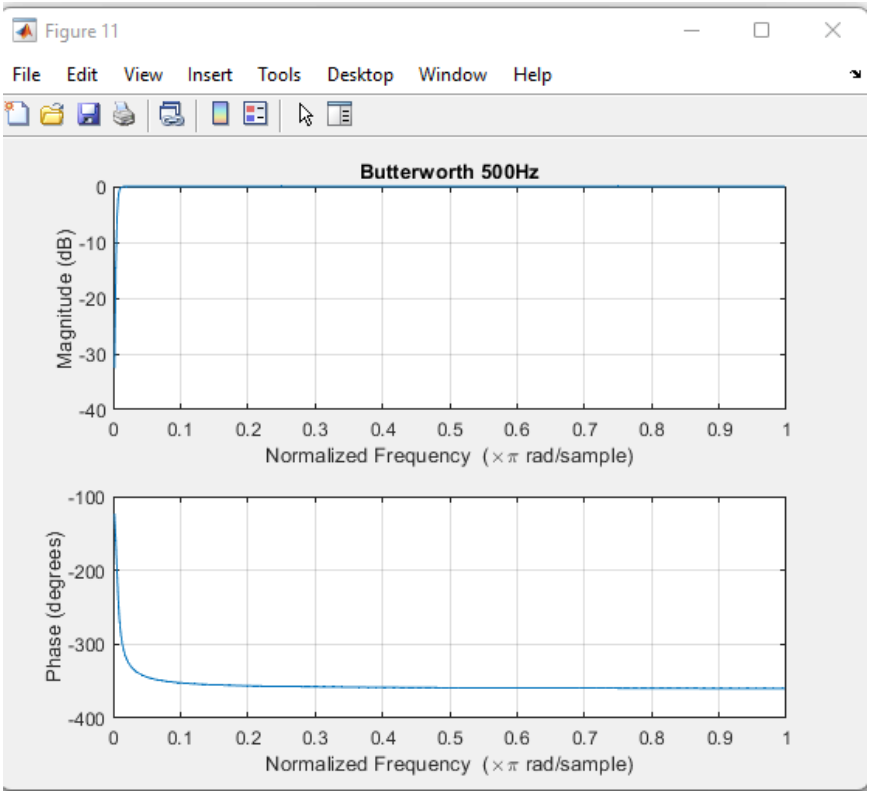


Figure 27 Butterworth Trans. Band = 500 Hz

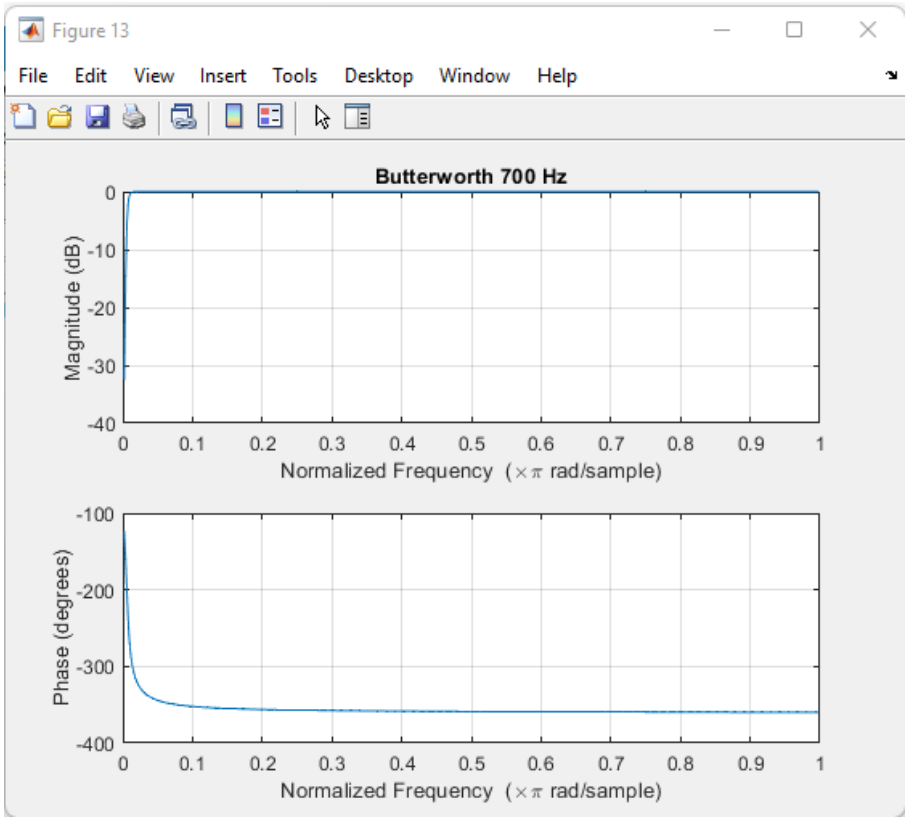


Figure 28 Butterworth Trans. Band = 700 Hz

- Fifth Approach: Using The Analog Bessel Filter Analysis

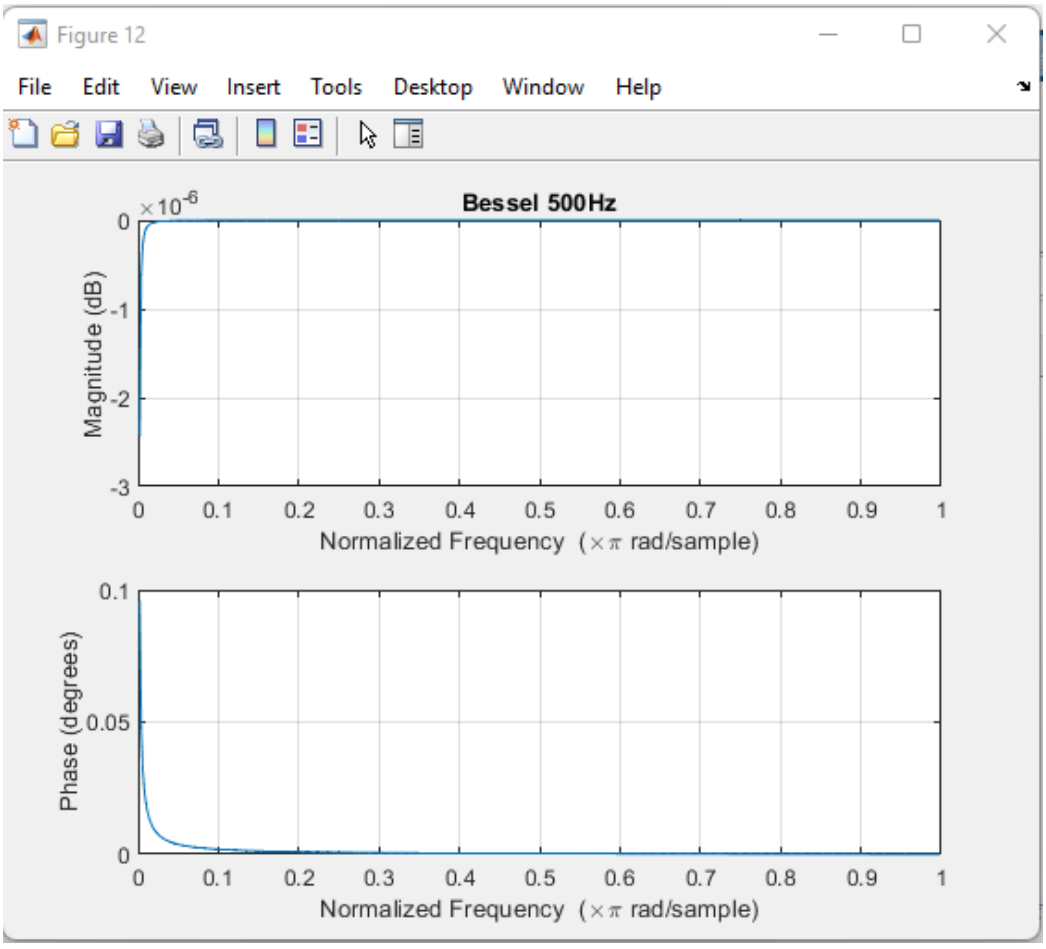


Figure 29 Bessel Frequency Response 500Hz

- Sixth Approach: Using The Butterworth Filter Analysis

Using the bilinear transformation method:

This method involves analog to digital transformation from La Place domain to Z-domain. The core of this transformation in the Cutoff frequency.

$$\omega a = \frac{2}{Ts} * \tan(\frac{\omega d\, Ts}{2})$$

Due to the missing corresponding values of ωd to ωa at certain values (Because tan function curve tends to ∞ at some points (Non Linear Relation)) which is called the Warping effect the best approach is to get the digital omega first then the corresponding analog (Pre-warping).

Steps to design a Butterworth HPF:

We got fpd, fcd, At (min. attenuation), and consequently we got ωpd & ωcd from our signal plot. With these we started our analysis:

Note: Due to the close values of $\omega cutoff$ & $\omega stopband$ we chose to put them = the same value.

Full analysis is provided down below in the appendix code 3 section.

Conclusion & Comparison

Although FIR filters are powerful for the Linear Phase they provide, they can be a bit unreliable at low orders, so in order to achieve a good quality a higher order is required i.e More hardware. However, windows can be very helpful in eliminating noisy signals when multiplied to the impulse response of the filter especially at lower transition bands. But in the end IIR filters are the masters of all, they are

complex, yet they provide high quality filtration at low orders. This Project has been very helpful in showing how digital filters behave as a step in the vast field of Digital Signal Processing.

References

https://www.mathworks.com/help/dsp/ug/designing-low-pass-fir-filters.html

Configure Dynamic Filter Visualizer - MATLAB & Simulink (mathworks.com)

https://www.section.io/engineering-education/how-to-design-infinite-impulse-response-filters-using-matlab/

IIR digital filter matlab code | MATLAB source code (rfwireless-world.com)

https://users.encs.concordia.ca/~realtime/elec342/manuals/lab5.pdf

https://www.ece.rutgers.edu/~orfanidi/ece521/notes.pdf

MATLAB Program for IIR(Infinite Impulse Response) High Pass Filter using the Window function | IT1254 - DSP and Communications Systems Lab - Source Code Solutions

Butterworth filter Matlab | Examples of Butterworth filter Matlab (educba.com)

Python scipy.signal IIR Filter Design - Christopher Felton (dsprelated.com)

Appendix

Task One MATLAB Code:

```
clc
clear
%loading the song pre coding
[x,fs]=audioread ('17.wav');
%c=2 channels n=samples 2 colomns
load('FIR_T1.mat') %matrix file
%  sound(x ,fs);
%  sound(FIR_T1,fs)
figure
spectrogram(x(:,1));
 title('Original Signal');
Nsamps = length(x);
f=0:fs/Nsamps:fs-fs/Nsamps; %to see +ve half cycle (step)
```

```
%Divide every sample by fs to get time scale
w0=(2*pi*60)/fs; %f0=60 , to get normalized dived by fs
h=[1 -2*cos(w0) 1 ]; %feed forward coffecient and there is no feed back it
will be =1
a=1;%feed back backward coffecient
y=filter(h,a,FIR_T1);
figure
spectrogram(FIR_T1(:,1))
 title('Noisy Signal');
%  sound(y,fs); %sampling song after filter
H=freqz(h,a,f,fs); %freqz convert func. of time domain to freq. domain h,a
impulse response then all the frequncy
%H=frequncy response (freq domain )
figure
 spectrogram(y(:,1))
 title('Filtered Signal');
 figure
plot(f,20*log10(abs(H))); %graph to see H in freq Domain
% Plotting The frequency domain of the three signals to compare between
them %
title('Filter Magnititude Response');
xlabel('Frequncy(HZ)');
ylabel('|H(Z)|db');
grid on
figure
subplot(3,1,1)
xlim([0,100])
plot(f,(abs(fft(x))));
title('Frequency Domain ');
xlabel('Frequency(HZ)');
ylabel('X');
subplot(3,1,2)
plot(f,abs(fft(FIR_T1)))
xlim([0,100])
title('Frequency Domain of noisy signal');
xlabel('Frequency(HZ)');
ylabel('FIR T1');
subplot(3,1,3)
plot(f,abs(fft(y)))
title('Frequency Domain of corrected signal');
xlabel('Frequency(HZ)');
ylabel('Y');
xlim([0,100])
%To save the filtered signal to a specific location
audiowrite('C:\Users\Hassan Ghandoor\Desktop\17\FilteredsigT1.wav',y,fs);

%%%%%%Conolution Trials %%%%%%%%
% B=[0.5 0.5]; % coefficient of the x(n), x(n-1)
% A= [1 -1 0.5];% coefficient of the y(n), y(n-1), y(n-2)
% fs=48000;
% n=1*fs;
% f1=100;
% f2=3000;
% x=0.2*rand(n,1);
% c=conv(x,fs);
% x=2*sin(2*pi*f1/fs*n)+4*sin(2*pi*f2/fs*n); % input signal
% y=filter(B,A,x);% filter
% sound(y,fs);



% x=0.2*rand(N,1);
% y=conv(x,fs);
```

```
% sound(x,fs);

% while fs<12
%     fs=fs+1;
%     x=rand(2^fs);
%     N=length(x);
%     X=zeros(N,1);
%     tic
%     for k=0:N-1
%         for n=0:N-1
%             X(k+1)=X(k+1)+x(n+1)*exp(-1i*2*n*k*pi/N);
%         end
%     end
%     sound(X,fs);
%     toc
% end
```

Task Two MATLAB Code:

```
clc
clear
% Reading Data From The Original Song.
[x,fs]=audioread('C:\Users\Haidy Magdy\Downloads\17\17/17.wav');
l=length(x);
freq=0:fs/l:fs-(fs/l);
% Loading The Noisy Signal.
load('C:\Users\Haidy Magdy\Downloads\17\17/FIR_T2_1.mat');
%-------------------------------------------------------------------
% Making The Window with It's Order.
Lorder500=317;
w=hamming(Lorder500);
%Making The Band Stop Filter.
fc1=2255.5;
fc2=2855.5;
n = -(Lorder500-1)/2:1:(Lorder500-1)/2;
bsf = sinc(n)+2*(fc1/fs)sinc(2(fc1/fs)n)-2(fc2/fs)sinc(2(fc2/fs)*n);
% Applying The Window On The Filter.
b= bsf.*(w');
H = freqz(b,1,freq,fs);
filter_signal=filter(b,1,FIR_T2);
%-------------------------------------------------------------------------
----------%
% Plotting Magnitude Response of the filter.
f=2000:1:3000;
HD=freqz(b,1,f,fs);
figure
plot(f,20*log10(abs(HD)))
title('Filter Magnitude Response');
xlabel('Frequency(HZ)');
ylabel('|H(z)|(dB))');
% Plotting The Frequency Domain Of The Three Signals To Compare.
figure
subplot(3,1,1)
plot(freq,abs(fft(x(:,1))))
xlim([2000,3000])
title('Frequency Domain of Original signal');
xlabel('Frequency(HZ)');
```

```
ylabel('X');
subplot(3,1,2)
plot(freq,abs(fft(FIR_T2(:,1))))
xlim([2000,3000])
title('Frequency Domain of Noisy signal');
xlabel('Frequency(HZ)');
ylabel('FIR T2');
subplot(3,1,3)
plot(freq,abs(fft(filter_signal(:,1))))
title('Frequency Domain of Modified signal');
xlabel('Frequency(HZ)');
ylabel('Y2');
xlim([2000,3000])
% Playing the Sounds Throught Matlab To Listen To The Difference.
disp('Second taskbefore filter');
sound(FIR_T2,fs)
pause(30)
disp('Second task after filter');
sound(filter_signal,fs)
% Audio Writing Phase
 audiowrite('Corrected-T2-1001.wav',y2,fs);
```

Task Three MATLAB Code:

```
clear
clc
% Reading the original song file
[x,fs]=audioread('17.wav');
load('IIR_T3_0.mat');%Loading the contaminated file
%sound(IIR_T3,fs);
%Fourier Transform of sound file
Nsamps = length(IIR_T3);%To get no. of samples of the song
%t = (1/fs)*(1:Nsamps); %Preparing the time scale
mag = abs(fft(IIR_T3)); %changing from time to freq --> complex so get mag.
f=0:fs/Nsamps:fs-fs/Nsamps; %Just plotting the +ve scale no need for image
part
%To plot in time domain
%figure
%plot(t,IIR_T3)
%%%To plot in freq domain
figure
plot(f,mag)
xlim([0,200])%Just to limit the scale for a better view
figure
subplot(3,1,1)
plot(f,20*log10(abs(fft(x))))
xlabel('frequency'); ylabel('X'); title('Original Signal'); xlim([0,6000])
subplot(3,1,2)
plot(f,20*log10(abs(fft(IIR_T3))))
xlabel('frequency'); ylabel('IIR T3'); title('Noisy Signal'); xlim([0,6000])
scope = dsp.SpectrumAnalyzer('SampleRate',fs); %spectrum analyzer view
step(scope,IIR_T3)
figure
periodogram(IIR_T3) %%viewing Power Spectral Density Just to make sure of
%the noise place
figure
 periodogram(x)
scope(IIR_T3)
 figure
```

```
spectrogram(x(:,1)) % 3D freq vs time vs amplitude
title('Original')
 figure
spectrogram(IIR_T3(:,1))
title('Noisy')
%%Elliptic Filter function
 [n,wp] = ellipord(0.001,0.01,0.5,50);% From the noisy signal plot:
%We got the stop band from 0 to 50 so ws (beginning of stop band = 50/fs)
%Transition band = 500 so 500 + 50 = 550 is the wp (Beginning of passband =
%550/fs normalized), I set the passband ripples Rp to 0.5 as suggested by
%MATLAB help, and Rs is the min attenduation = 50 dB
%%NOTE: SIMILAR WAY TO GET THE 750 Trans band
[b,a] = ellip(n,0.5,50,wp,'high'); %Takes order gets Transfer func. params
freqz(b,a); %Filter freq response
y = filter(b,a,IIR_T3);
figure
 plot(f,y)
%sound(y,fs)
scope = dsp.SpectrumAnalyzer('SampleRate',fs);
step(scope,y)
title('Filtered Signal');
figure
spectrogram(y(:,1))
title('Filtered Signal');
%%%%For the chebychev filter types 1 and 2
[n2,wp2]=cheb1ord(0.001,0.01,0.5,50);
[b2,a2]=cheby1(n,0.5,wp,'high');
freqz(b2,a2);
 y2=filter(b,a,IIR_T3);
 figure
plot(f,y2)
[n3,ws]=cheb2ord(0.001,0.01,0.5,50);
[b3,a3]=cheby2(n,0.5,ws,'high');
freqz(b3,a3);
 y3=filter(b,a,IIR_T3);
 figure
plot(f,y3)
%%%Detailed Step Analysis For High Pass Butterworth Filter
fpd = 550/fs; %Normalized passband freq
wpd = 2*pi*fpd;
fcd = 50/fs; %Normalized cutoff freq
wcd = 2*pi*fcd;
Ts = 1/fs;
At = 50; %Stopband attenuation in dB
Atratio = power(10,50/20);
%Pre-warping
wpa = (2/Ts)*(tan(wpd*Ts/2)); %Analog angular passband frequency
wca = (2/Ts)*(tan(wcd*Ts/2)); %Analog angular cutoff freq
N = abs(round((log10(Atratio))/(log10(wca/wpa)))); %Order %abs bec HPF
z = exp(complex(0,wca*Ts));
s = (1-(power(z,-1)))/(1+power(z,-1));
HZ = 1/(power((wca/s),2)+((sqrt(2))*(wca/s))+1);
%figure
%plot(f,abs((fft(HZ))));
%Since H(z)= b/a, by calculating H(Z) we now have their values, and we can
%plot them using the freq response function freqz
[n4,wn4]= buttord(0.01,0.001,3,50);
[b4,a4] = butter(n4,wn4,'high');
figure
freqz(b4,a4);
title('Butterworth 500Hz');
y4=filter(b4,a4,IIR_T3);
```

```
[Z,P,K] = besself(3,0.2,'high');
[zd, pd, kd] = bilinear(Z,P,K,fs);
[b5,a5] = zp2tf(zd,pd,kd);
figure
freqz(b5,a5);
title('Bessel 500Hz');
y5=filter(b5,a5,IIR_T3);
%-----------------------------------------------------%
To listen to the sounds
sound(IIR_T3,fs);
pause(15);
sound(y,fs); %Elliptic
pause(15);
sound(y4,fs);%Butterworth
pause(15);
sound(y5,fs);%Bessel
%-----------------------------------------------------%
%Comparison Btween Butterworth 500 Hz & 700 Hz transition bands
[n6,wn6]= buttord(0.017,0.001,3,50);
[b6,a6] = butter(n6,wn6,'high');
figure
freqz(b6,a6);
title('Butterworth 700 Hz');
y6=filter(b6,a6,IIR_T3);
% sound(y4,fs);%Butterworth 500
% pause(15);
% sound(y6,fs);%Butterworth 700
%-----------------------------------------------------%
%To save the signal to my computer
audiowrite('C:\Users\Hassan Ghandoor\Desktop\17\Recursive.wav', y, fs)
```