Group - 02

Complex Engineering Assignment

De-noising of Speech Signal Using Digital Filters

Report

Submitted to: Abul Barkat Mollah Sayeed Ud Doulah

Assistant Professor, EEE Department, ULAB

Prepared by

Student's Name	ID
Mohammad Navid Nayyem	172014003
Sabiha Sumaiya	172014007
Md. Asraf Ali	172014008
Abdulla Al Numan	172014013

Course Title: Digital Signal Processing

Course Code: ETE315

Section: 01

Date of Submission: 30 - 09 - 2020

Contents

Table of Figures

Objective: -

In this assignment we were asked to apply different types signal processing method to plot and remove noise from an audio file. We designed a range of filters and codes in MATLAB and figured out four best output. All the working process has been discussed below.

Instruments: -

i. Desktop/Laptop.
ii. MATLAB R2018a Software.

Working Procedures: -

1. Play the audio and show the magnitude plot of audio signal.

Our first task is to play the audio file, so we declared audioread method to read the data from filename and return the data in audio_data and Fs is the sample rate of the data. This audio read function support OGG, FLAC, AU, MP3, MPEG-4, AAC, WAV file. Next, audioplayer this is an object for y signal which is play the audio. By using this audioplay object we can pause the audio or we can identify the recall audio.Play is a function and player is a variable name where we assign the audioplayer variable. FFT stand for Fast Fourier Transform and it converts the original state for Assertion the frequency domain and vice-versa. Nfft where we assign the length of the audio signal. Plot basically create a meaningful frequency that we can plot magnitude as a frequency HZ. x-label is a frequency HZ and y-label is a signal magnitude(dB).

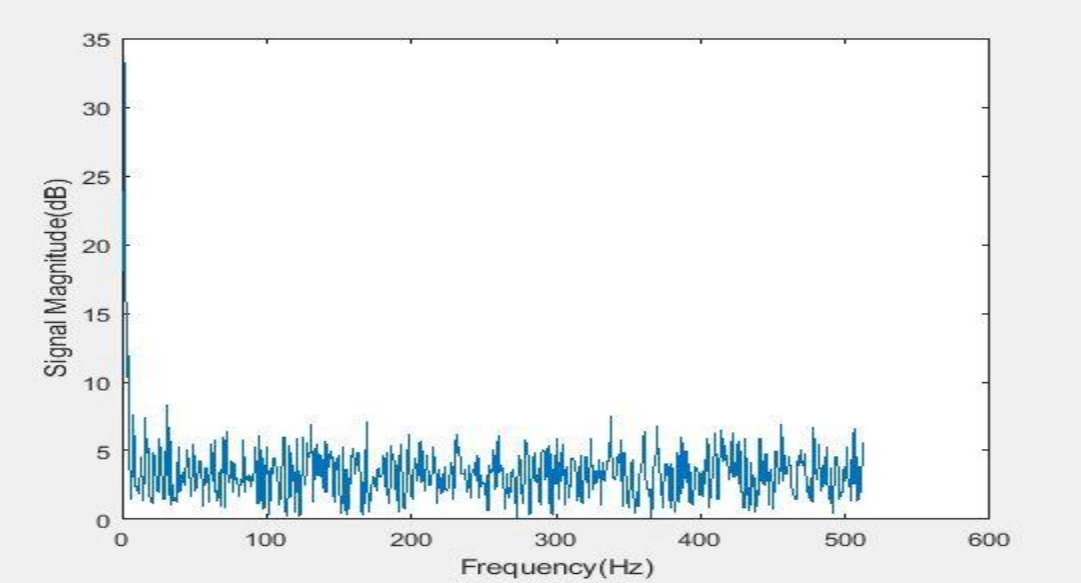

Figure 1: *Magnitude Plot of Audio Signal*

2. Find out the periodicity of the audio signal (if any).

Our second task is to find out the periodicity of an audio signal. Here the audio is read by using the audioread() function. [audio_data, fs] reads data from the file and returns sampled data, audio_data, and a sample rate for that data, fs. **periodogram(audio_data, hamming(length(audio_data)), length(audio_data), fs)** returns the two-sided periodogram estimates at the frequencies specified in the vector. Hamming is the window type. f contains the length of audio signal which is length(audio_data). The frequencies in f are in cycles per unit time. The sample rate, fs, is the number of samples per unit time. If the unit of time is seconds, then f is in cycles/second (Hz). In signal processing, a periodogram is an estimate of the spectral density of a signal. Now, we find out the peaks of the signal and their locations using **findpeaks(X,f)** function where X is the peak of the signal and f contains the values of locs at the peak indices. At last, we find out the time period of the signal using **max(diff(locs))** to specify the maximum peak location by the differences between adjacent elements of locs. Here locs is the location.

3. Process of Filter Designing and Plotting Magnitude Response, Phase and the pole-zero Diagram

Filter Design is a graphical user interface for designing and analyzing filters. It allows to design digital FIR or IIR filters by setting filter performance specifications, by importing filters from our MATLAB workspace or by adding, moving, or deleting poles and zeros. It also provides tools for analyzing filters, such as magnitude and phase response plots and pole-zero plots.

For building a filter, we have to go to Apps option on top of the menu bar section and click on the dropdown arrow and click on Filter Designer in Signal Processing and Communication section of MATLAB Software. Then, a graphical user interface will display with a default filter. The GUI has three main regions: The Current Filter Information region, The Filter Display region and the Design panel. The upper half of the GUI displays information on filter specifications and responses for the current filter. The Current Filter Information region, in the upper left, displays filter properties, namely the filter structure, order, number of sections used and whether the filter is stable or not. It also provides access to the Filter manager for working with multiple filters.

We will design low pass filter with finite impulse response so we have to select lowpass in the Response Type and select Window in FIR of Design Method. Then after selecting FIR-Window, we have the choose the window type. The examples of Window types are Hamming, Hann, Kaiser, Triangular, Rectangular, etc. For each window type we have to put the value of passband-edge frequency (Fc) and sampling frequency (Fs) which is in Hertz or Fs, Fpass and Fstop for some window in Frequency Specification. Specially, for Kaiser window type we have to give the value of beta. If we change the Response Type or Design Method, the filter parameters and Filter Display region will automatically be updated. Then, we have to click Design Filter button at the bottom of

the graphical user interface to design the filter. At last, we have to export the filter as an object and save the filter design file which file extension is (.fda). The object will save in the workspace which extension is (.mat). In the code, we have to call the filter as an object. For each filter, the magnitude response, phase response and pole/zero plot will be automatically plotted in the graphical user interface.

We have designed four low pass FIR Filters and the chart is given below: -

Types of Filter	Design Method	Types of Window	Beta Value	Fs	Fc
Lowpass	FIR - Window	Hamming		42000	4000
Lowpass	FIR - Window	Kaiser	3.5	100	1
Lowpass	FIR - Window	Kaiser	2.5	22000	0.2
Lowpass	FIR - Window	Triangle		44100	5000

Filter 01: LowPassHammingFs42000Fc4000

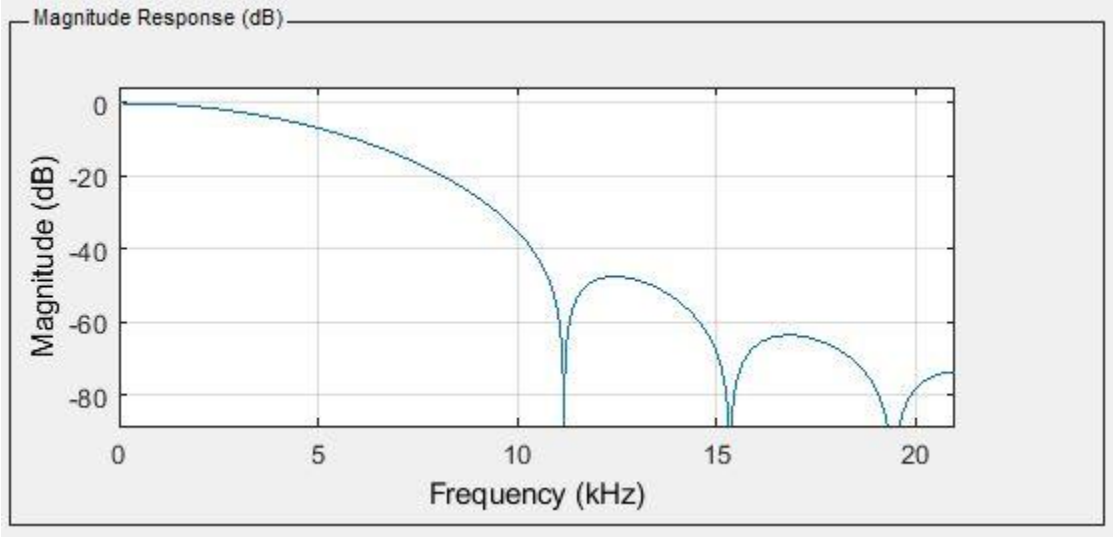

Figure 2: *Magnitude Response (dB) of Lowpass Hamming Filter*

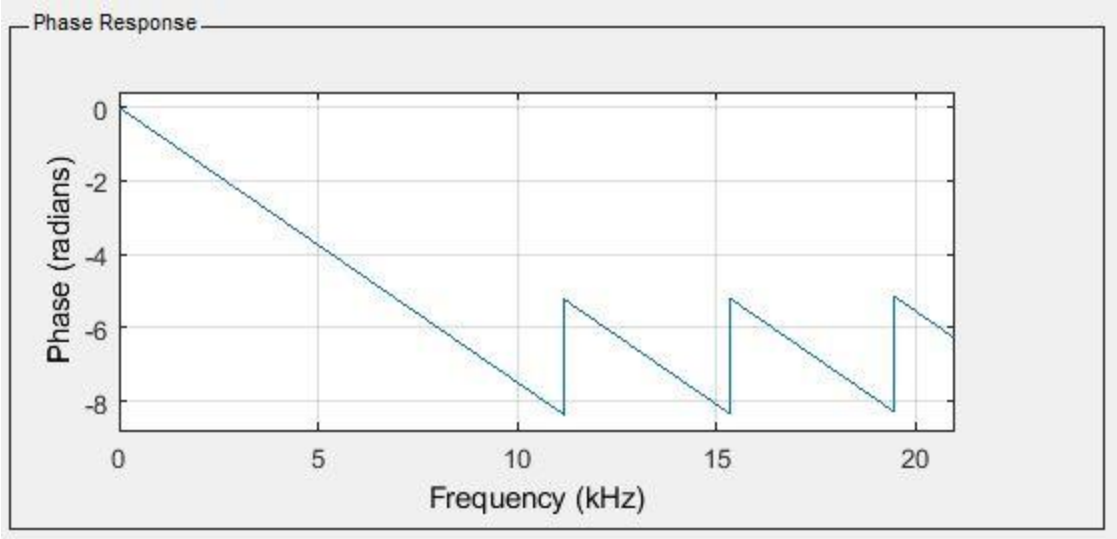

Figure 3: *Phase Response of Lowpass Hamming Filter*

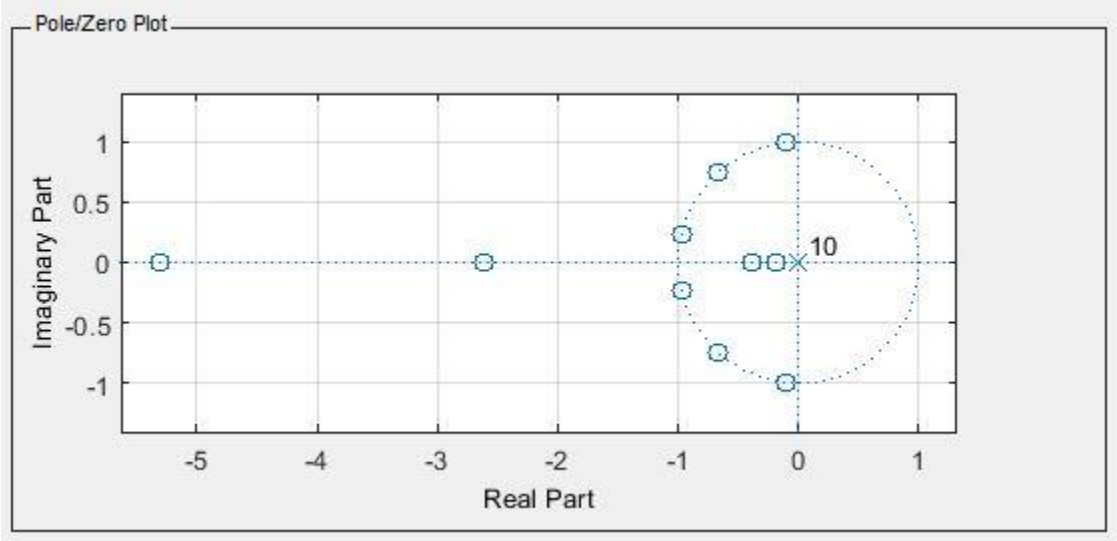

Figure 4: *Pole-Zero Diagram of Lowpass Hamming Filter*

Filter 02: LowPassKaiserFs100Fc1

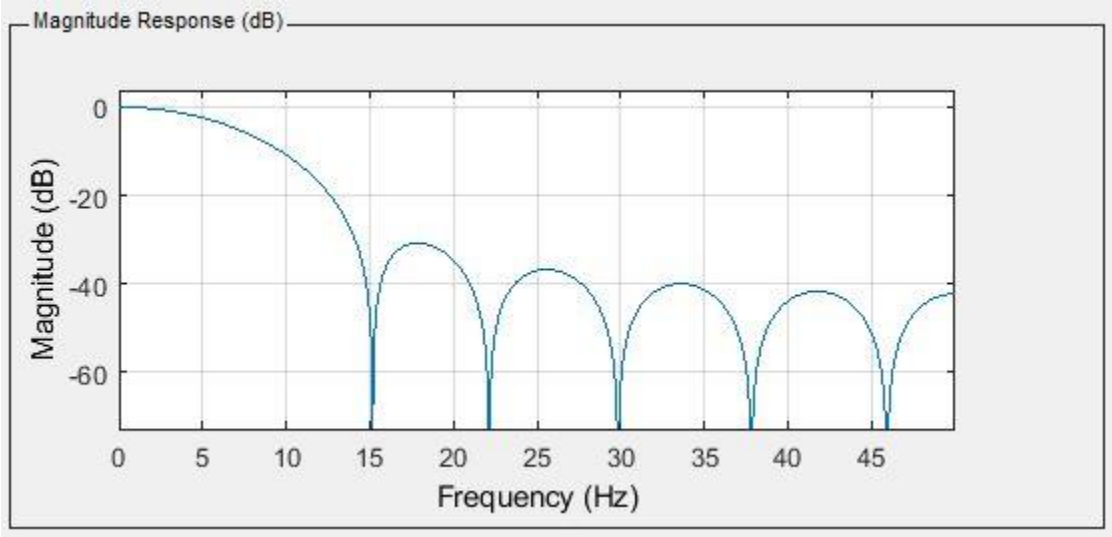

Figure 5: *Magnitude Response (dB) of Lowpass KaiserFc100 Filter*

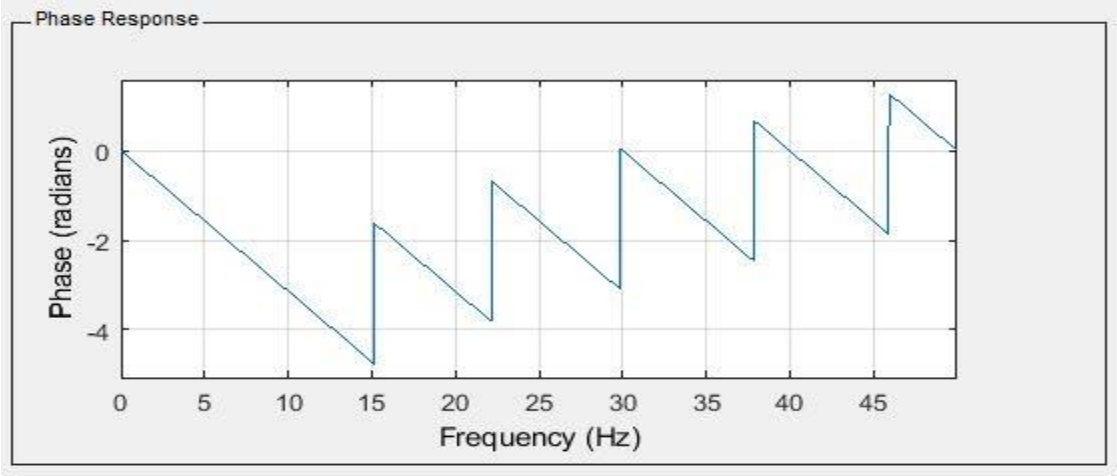

Figure 6: *Phase Response of Lowpass KaiserFc100 Filter*

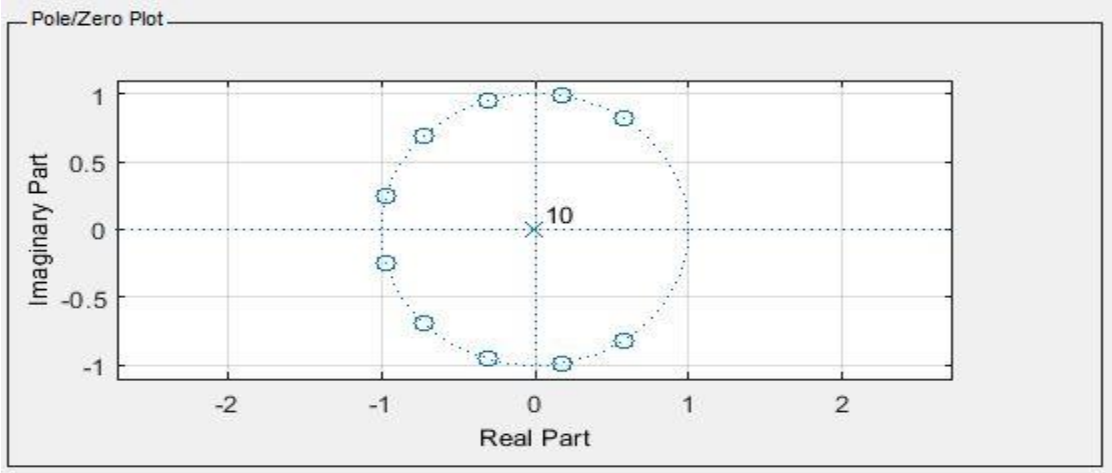

Figure 7: *Pole-Zero Diagram of Lowpass KaiserFc100 Filter*

Filter 03: LowPassKaiserFs22000Fc0point2

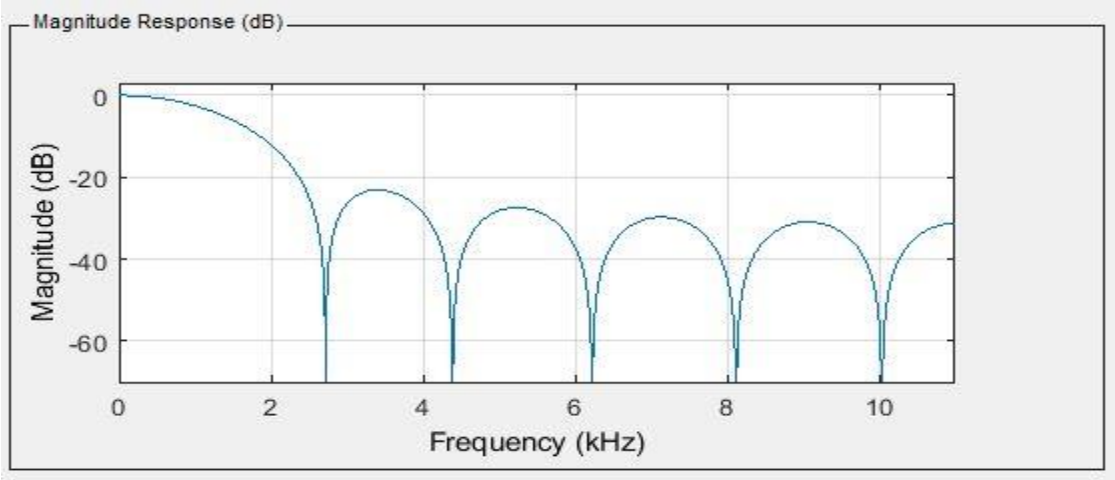

Figure 8: *Magnitude Response (dB) of Lowpass KaiserFc22000 Filter*

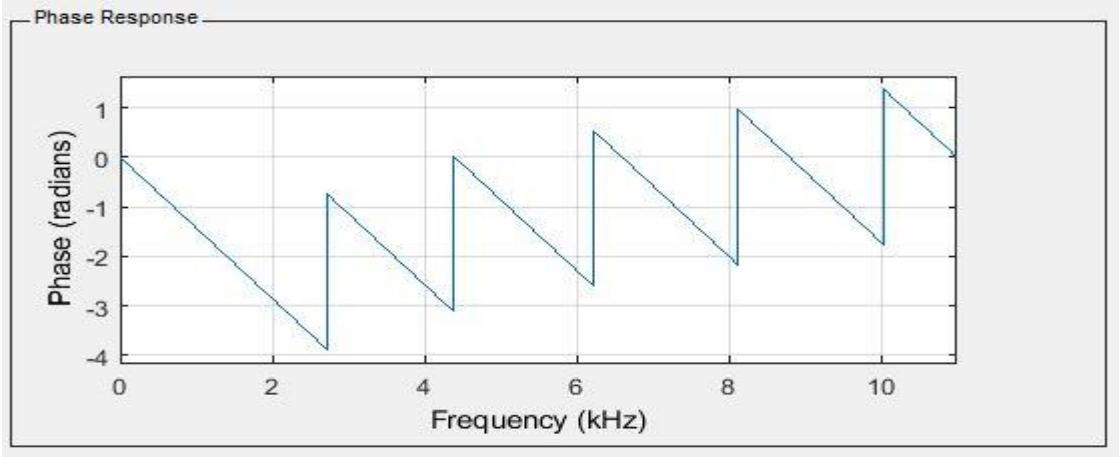

Figure 9: *Phase Response of Lowpass KaiserFc22000 Filter*

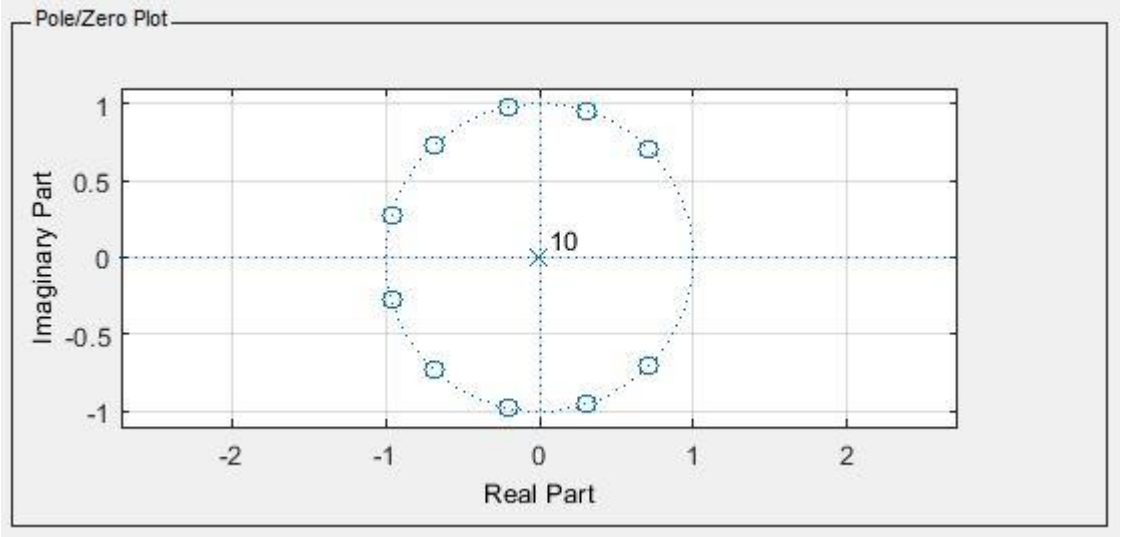

Figure 10: *Pole-Zero Diagram of Lowpass KaiserFc22000 Filter*

Filter 04: LowPassTriangularFs44100Fc5000

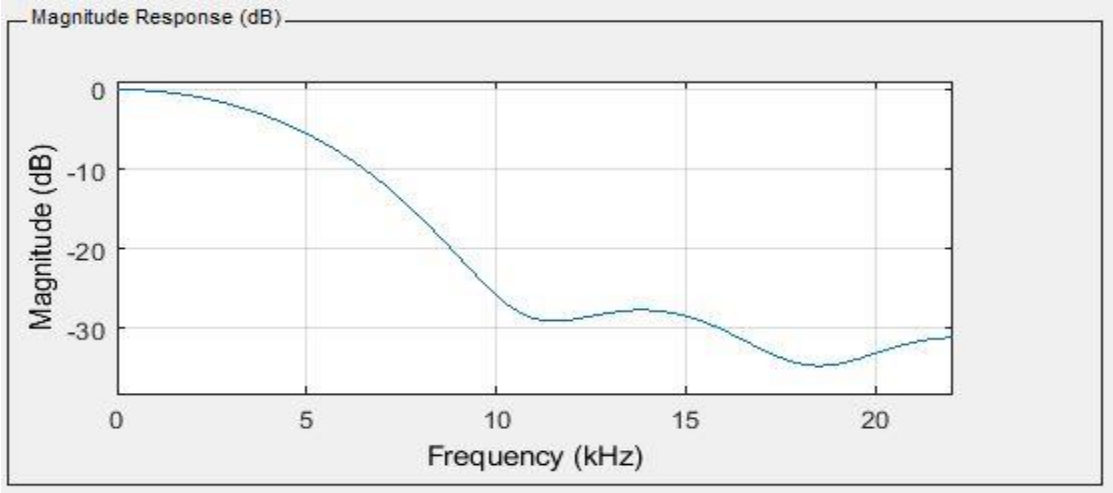

Figure 11: *Magnitude Response (dB) of Lowpass Triangular Filter*

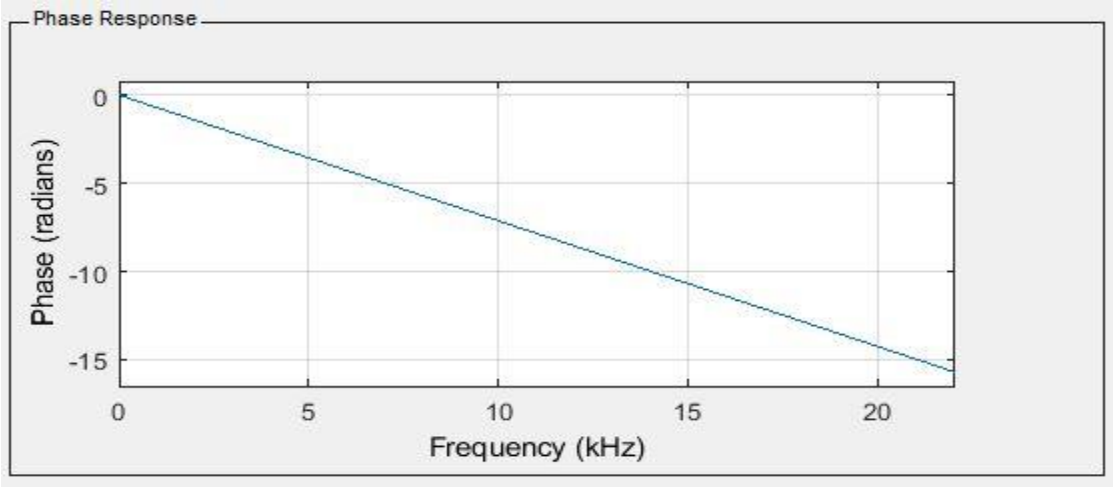

Figure 12: *Phase Response of Lowpass Triangular Filter*

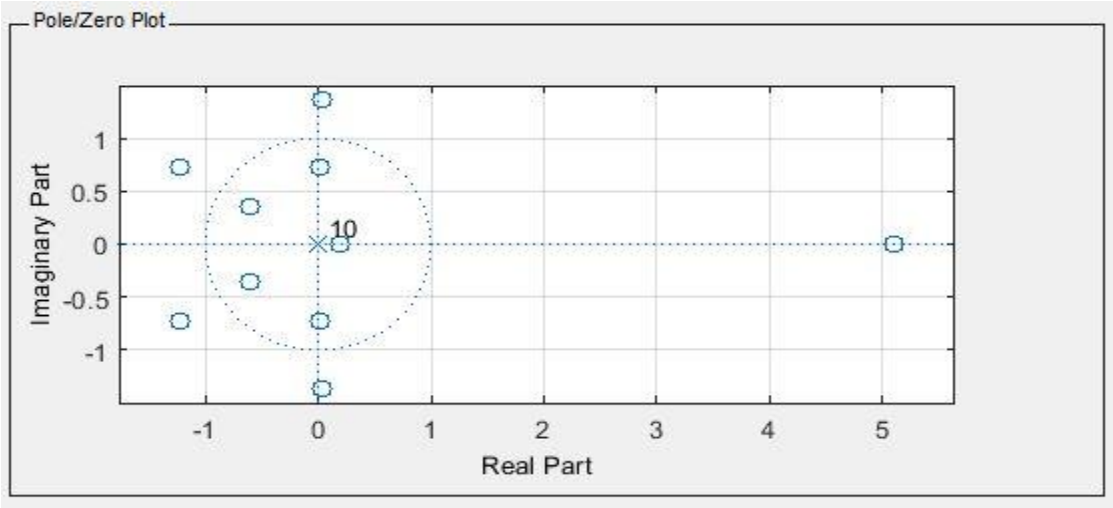

Figure 13: *Pole-Zero Diagram of Lowpass Triangular Filter*

4. Filter noisy signal using your de-noising filter.

For filtering the data from this audio file at first, we called **audioread** method, here **audioread(noisy_audio.wav)** reads data from the file, and returns sampled data, audio_data, and a sample rate for that data, Fs. Then we have to transform the data into frequency domain signal by Fourier transform because in time domain signal we have to perform complex convolution to apply filters but in frequency domain we only have to do multiplication to implement filters. Later, we declared a variable Nfft=1024 to define the length of our Fourier signal. Now, called the **fft (fast Fourier transform) method** and parameterized audio_data and Nfft to transform and plot the data into frequency domain. So, we stored the new frequency domain data in a variable FourierTransformOfOriginalSignal. Next, comes filter applying steps, here we called **filter method** as a value of a variable and parameterized two values in this filter method. One is the filter we designed and exported in our MATLAB work space such as Lowpass Hamming window Fs 42000 Fc is a filter and other value is the audio_data we extracted from the audio file. Then we stored the filtered signal and played the filtered signal by declaring play function. At last, we plotted the figures of Fourier transform of original signal and filtered signal into frequency (Hz) and magnitude(dB) domain.

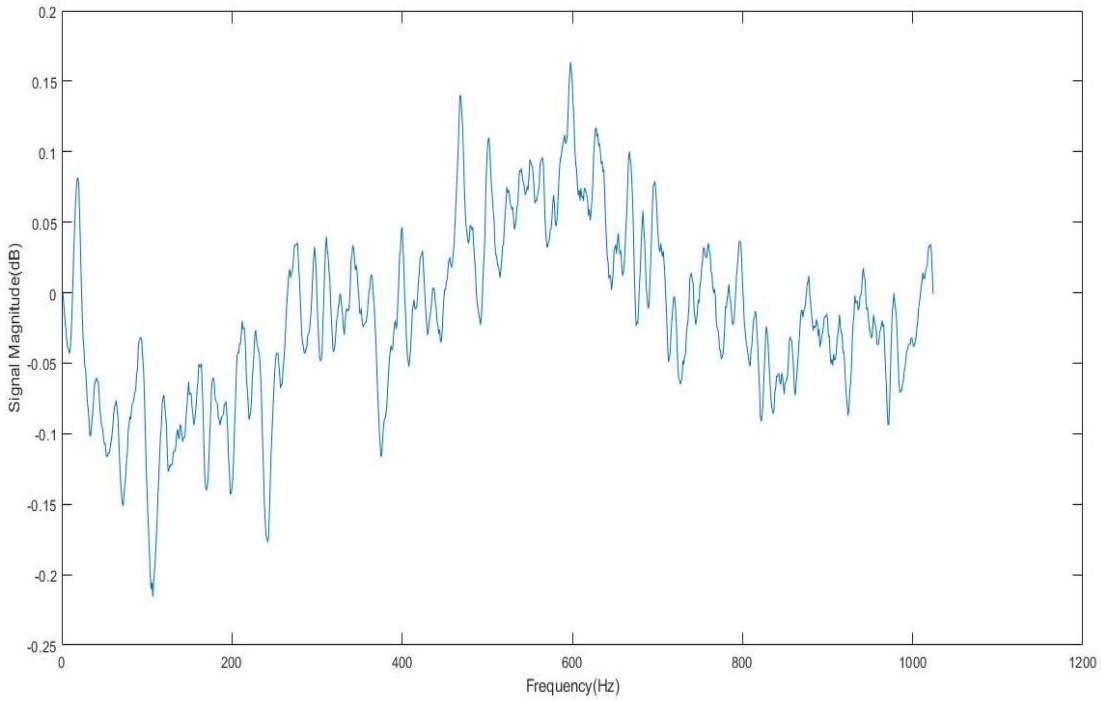

Figure 14: *Filtered Signal*

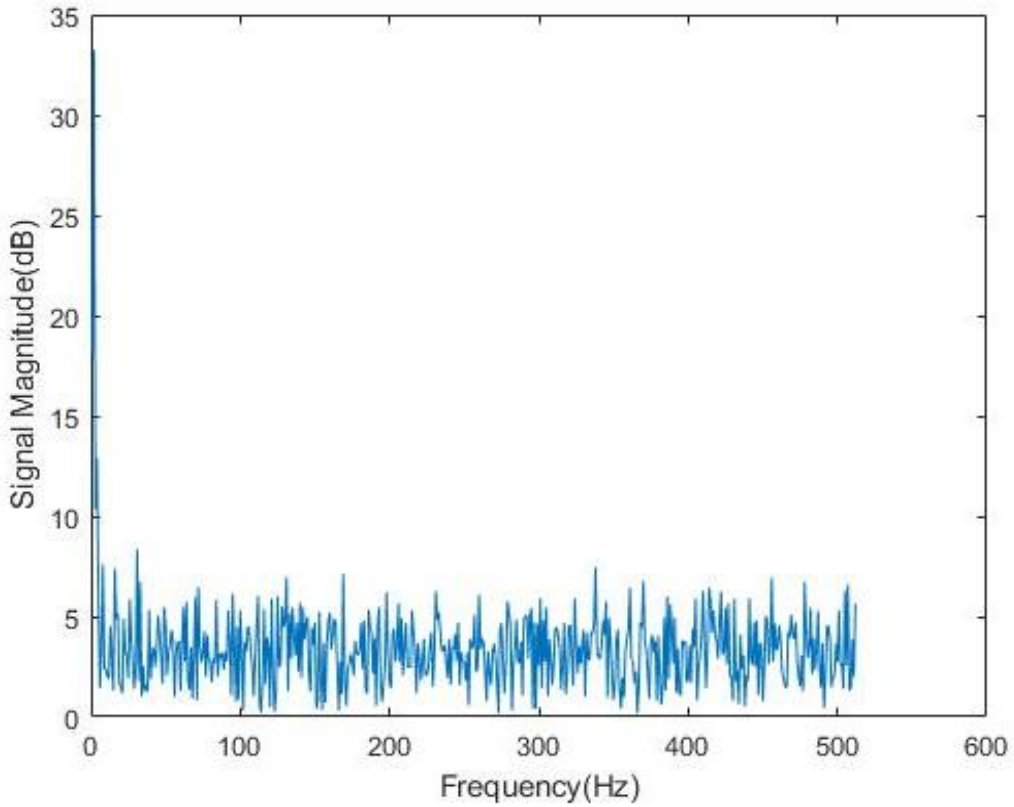

Figure 15: *Fourier Transform of Original Signal*

Comment on Best Filter and Comparison

In this whole process to remove noise we designed a range of filters and finally selected four best output. During the designing stage we noticed Low pass filter worked better than other filters for this given audio signal. In addition, we selected Kaiser, Hamming and Triangular window of FIR (finite impulse response). Then, we examined by applying those filters by changing different values of Fs (Sampling frequency) and Fc(Pass Band Edge Frequency) to remove noise. Finally, we decided a filter Lowpass Kaiser Fs 100 Fc 1 provided a better output comparing others. So, we experienced that different window works better on different frequency level because attenuation of frequency is the key point to remove noise from any signal and following our inspection Lowpass Kaiser Fs 100 Fc 1 works better in frequency attenuation.

References

1. https://www.mathworks.com/
2. https://www.youtube.com/watch?v=VFt3UVw7VrE