Contents

1. INTRODUCTION:

Digital filters are a necessary component of digital signal processing. In fact, DSP's outstanding performance is one of the main factors boosting its popularity. Two uses for filters are signal separation and signal restoration. Signal separation is necessary when a signal has been contaminated by interference, noise, or other signals. Signal restoration is utilized when a signal has been distorted in some way.

Before analog signals are sent to other components to be processed, they must first pass through physical hardware called analog filtering. Analog data is transmitted to a processor for digital filtering by the processor's software. The filters' function is to fully block some frequencies while allowing others to flow through unmodified. The frequencies that are passed are referred to as the passband, whereas the frequencies that are blocked are found in the stopband. Between is the transitional band. A quick roll-off indicates a relatively narrow transition band. The cutoff frequency designates the boundary between the passband and the transition band.

The four primary types of filters include the low-pass filter, the high-pass filter, the band-pass filter, and the notch filter (or the band-reject or band-stop filter).

A digital filter uses a digital processor to perform numerical calculations on sampled values of the signal. The processor may be a general-purpose computer such as a PC, or a specialized DSP (Digital Signal Processor) chip.

A low-pass filter (LPF) is a filter that allows signals below a cutoff frequency (known as the passband) and attenuates signals above the cutoff frequency (known as the stopband). Low-pass filters are often used to clean up signals, remove noise, create a smoothing effect, perform data averaging, and design decimators and interpolators. Low-pass filters produce slow changes in output values to make it easier to see trends and boost the overall signal-to-noise-ratio with minimal signal degradation.

High-pass, band-pass and band-reject filters are designed by starting with a low-pass filter, and then converting it into the desired response.

A high-pass filter (HPF) is an electronic filter that passes signals with a frequency higher than a certain cutoff frequency and attenuates signals with frequencies lower than the cutoff frequency.

The amount of attenuation for each frequency depends on the filter design. A high-pass filter is usually modeled as a linear time-invariant system. It is sometimes called a low-cut filter or bass-cut filter in the context of audio engineering. High-pass filters have many uses, such as blocking DC from circuitry sensitive to non-zero average voltages or radio frequency devices. They can also be used in conjunction with a low-pass filter to produce a bandpass filter.

A band-pass filter or bandpass filter (BPF) is a device that passes frequencies within a certain range and rejects (attenuates) frequencies outside that range.

In signal processing, a band-stop filter or band-rejection filter is a filter that passes most frequencies unaltered, but attenuates those in a specific range to very low levels. It is the opposite of a band-pass filter. A notch filter is a band-stop filter with a narrow stopband (high Q factor).

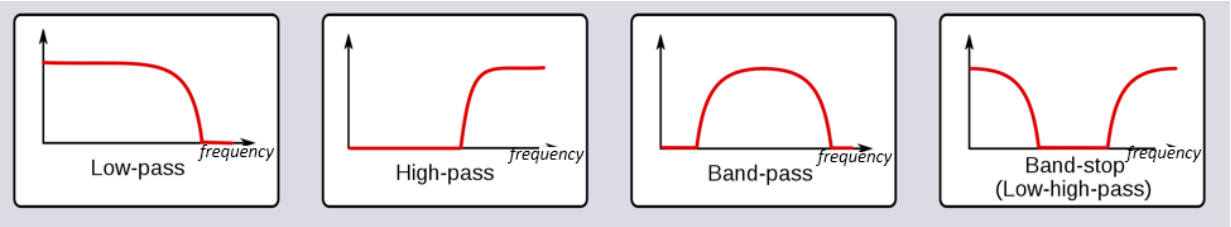
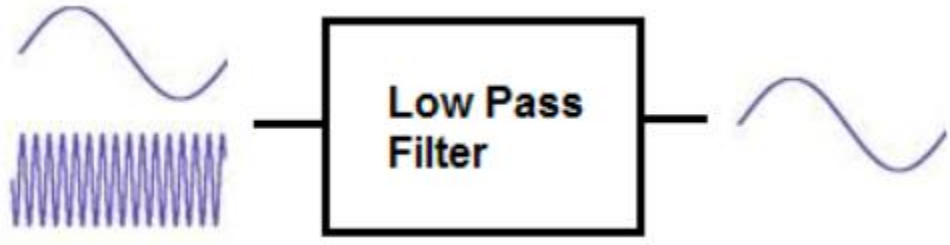


Fig 1.1



Input signal with noise **Output signal**

Fig 1.2

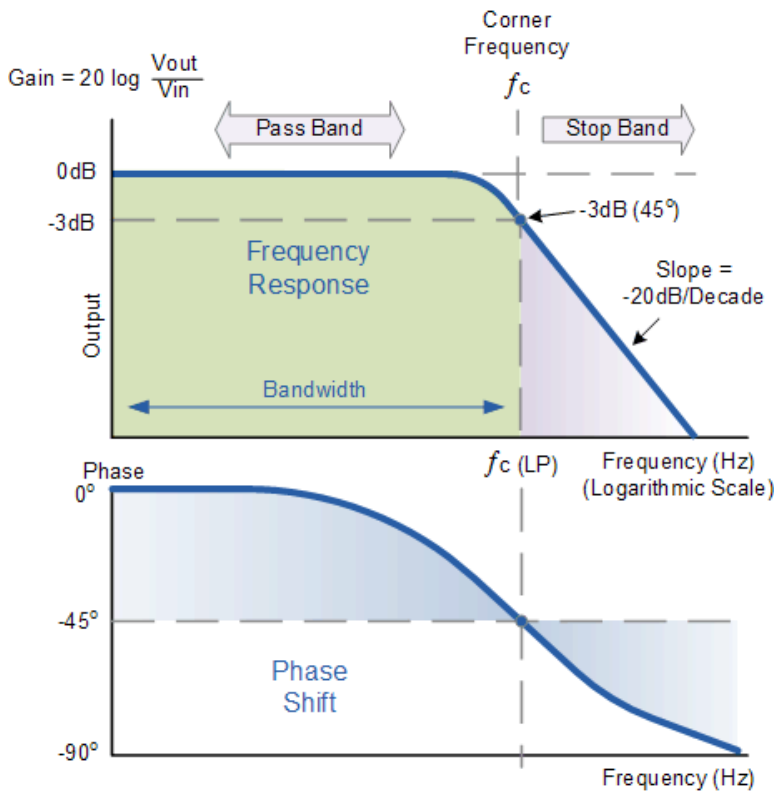


Fig 1.3

Fig 1.3 shows the Frequency Response of the filter to be nearly flat for low frequencies and all of the input signal is passed directly to the output, resulting in a gain of nearly 1, called unity, until it reaches its Cut-off Frequency point (fc)

2. OBJECTIVES :

a) To design a Low Pass Filter by giving cutoff frequencies.

A low pass filter has been designed by giving the input sample frequency and a cutoff frequency, Order of the filter, its magnitude spectrum, its phase spectrum, pole-zero plot have been obtained and also its stability nature, hence we can design a filter according to necessary requirements.

b) To get the results of a low pass filter by eliminating high frequency components from noisy audio.

An input audio file which has high frequency values in it is taken. Here we have designed a low pass filter which stops the high frequencies and the output of the signal is clear audio with less noise in it. This can also be observed well by using sound function to play the input and output audio files.

c) To compare the graphs of input, output signal, obtained from time and frequency response.

After Implementation of each step, the graph is to be observed, at first frequency response of the input signal, noisy signal and output signal, then time domain response of input and output signal, followed by power spectrum of input signal, noisy signal and output signal in this way a comparison has to be made between these graphs of input, noisy and output results.

3. THEORETICAL BACKGROUND :

An ideal low-pass filter completely eliminates all frequencies above the cut-off frequency while passing those below unchanged; its frequency response is a rectangular function and is a brick-wall filter. The transition region present in practical filters does not exist in an ideal filter. An ideal low-pass filter can be realized mathematically (theoretically) by multiplying a signal by the rectangular function in the frequency domain or, equivalently, convolution with its impulse response, a sine function, in the time domain.

However, the ideal filter is impossible to realize without also having signals of infinite extent in time, and so generally needs to be approximated for real ongoing signals, because the sine function's support region extends to all past and future times. The filter would therefore need to have infinite delay, or knowledge of the infinite future and past, in order to perform the convolution. It is effectively realizable for pre-recorded digital signals by assuming extensions of zero into the past and future, or more typically by making the signal repetitive and using Fourier analysis.

Real filters for real-time applications approximate the ideal filter by truncating and windowing the infinite impulse response to make a finite impulse response; applying that filter requires delaying the signal for a moderate period of time, allowing the computation to "see" a little bit into the future. This delay is manifested as phase shift. Greater accuracy in approximation requires a longer delay.

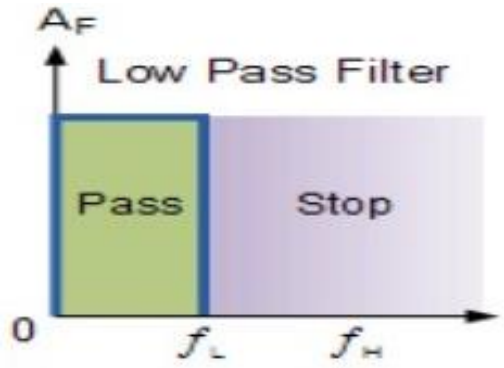


Fig 3.1

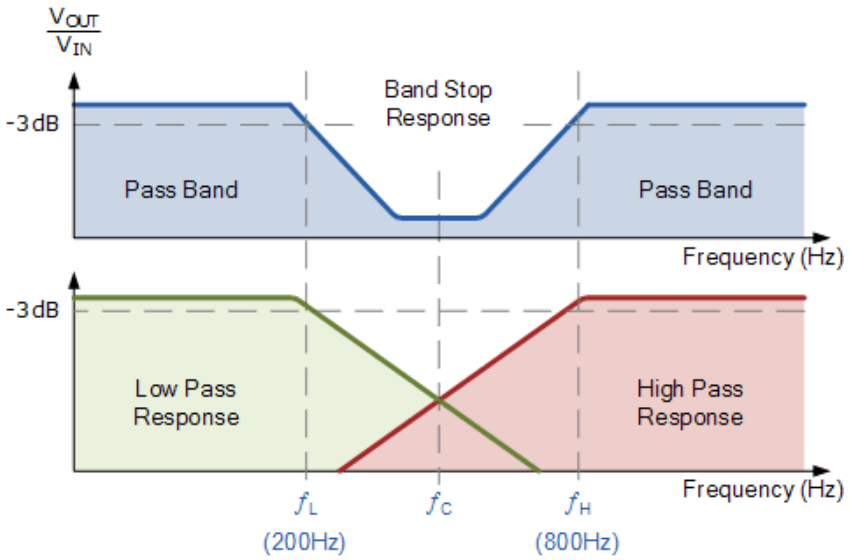


Fig 3.2

Basis for Comparison	FIR Filter	IIR Filter
Stands for	Finite Impluse Response	Infinite Impulse Response
Nature	Non-recursive	Recursive
Computational Efficiency	Less	Comparatively more
Usage	Difficult	Quite easy
Feedback	Absent	Present
Stability	More	Less
Requirement to generate current output	Present and past samples of input.	Present and past samples of input along with past output.
Delay offered	High	Comparatively lower
Transfer function	Only zeros are present.	Both poles and zeros are present.

Fig 3.3

Windows

The simplest method of FIR filter design is called the window method. Windows are meant to focus on a limited range of values (in time, space or frequencies), with more weight in the center. By doing this, windows concentrate on more stationary or consistent parts of the data (because they are related at close distance).

Kaiser Window

The Kaiser window is an approximation to the prolate spheroidal window, for which the ratio of the mainlobe energy to the sidelobe energy is maximized. For a Kaiser window of a particular length, the parameter β controls the relative sidelobe attenuation. For a given β, the relative sidelobe attenuation is fixed with respect to window length. The statement kaiser(n,beta) computes a length n Kaiser window with parameter beta.

As β increases, the relative sidelobe attenuation decreases and the mainlobe width increases. This screen shot shows how the relative sidelobe attenuation stays approximately the same for a fixed β parameter as the length is varied.

Advantages of Kaiser Window

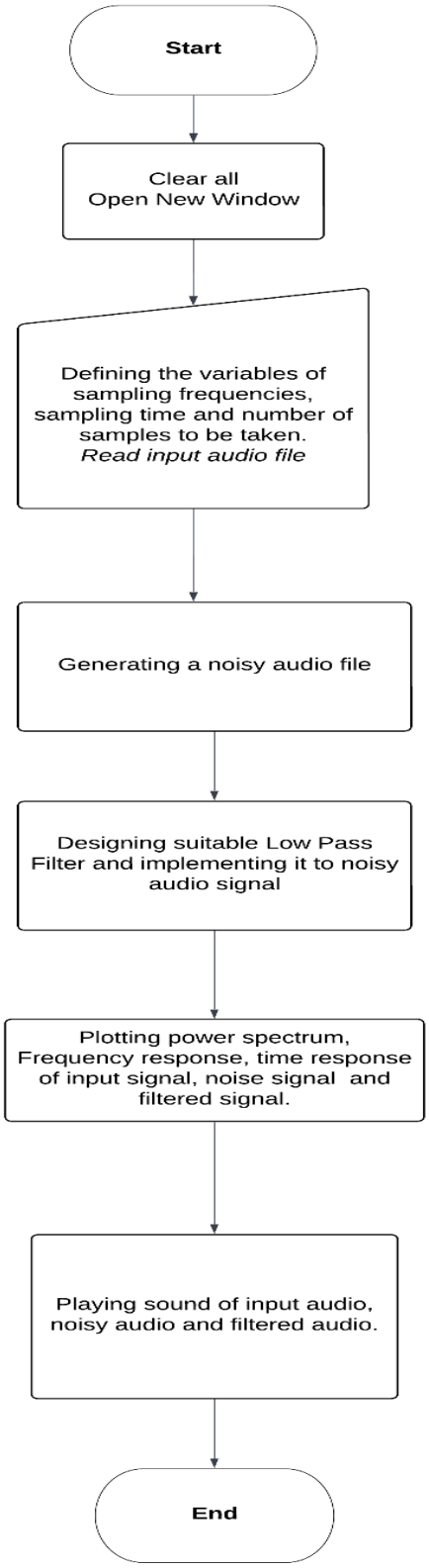
If we talk about the advantages of the Kaiser window then in the Kaiser window the passband and stopband ripple size are not much affected by the change in the window length in comparison to the other windows. It is affected by the coefficient roll off that is also known as the shape factor. The length of the window mainly affects the transition band.

By keeping the window length constant, we can adjust the shape factor to the design for the passband and stopband ripples and it is a huge advantage over other windows in which the window length, ripple size and transition bandwidth have a three-way tradeoff.

Kaiser window has an adjustable shape parameter that allows the window to achieve any desired value of ripple or attenuation. The Kaiser window is unique that it has near optimum performance, in the sense of minimizing the sidelobe energy of the window as well as having the simplest implementation

Hence, these are the advantages of the Kaiser window.

4. FLOWCHART:


Start
Clear all
Open New Window
Defining the variables of sampling frequencies, sampling time and number of samples to be taken.
Read input audio file
Generating a noisy audio file
Designing suitable Low Pass Filter and implementing it to noisy audio signal
Plotting power spectrum, Frequency response, time response of input signal, noise signal and filtered signal.
Playing sound of input audio, noisy audio and filtered audio.
End

5. CODE :

```
clearvars;
clc
close all;

[x,Fs] = audioread('Input_audio.wav');
N = size(x,1);
f = Fs*(0:N-1)*N;
T = 1/Fs;
L = N;
t = (0:L-1)*T;
A = 0.04;

xnew = transpose(x); % xnew=x';
xn = xnew + A*randn(1,length(x));

audiowrite('Noisy-Signal.wav',xn,Fs);

load('Num.mat')

filtered_signal_2 = filter(Num,1,xn);

audiowrite('Output.wav',filtered_signal_2,Fs);

Order = filtord(Num);

%Graphs

[audio_in,audio_freq_sampl]=audioread('Input_audio.wav');
Length_audio=length(audio_in);
df=audio_freq_sampl/Length_audio;
frequency_audio=-audio_freq_sampl/2:df:audio_freq_sampl/2-df;
pow = (abs(x).^2)/Length_audio^2;

figure
subplot(3,1,1)
plot((frequency_audio),pow*4);
```

```
xlabel('Frequency');
ylabel('Power Spectral');
title('Power spectrum of Input signal');

[audio_in,audio_freq_sampl]=audioread('Noisy-Signal.wav');
Length_audio=length(audio_in);
df=audio_freq_sampl/Length_audio;
frequency_audio=-audio_freq_sampl/2:df:audio_freq_sampl/2-df;
pow = (abs(xn).^2)/Length_audio^2;

subplot(3,1,2)
plot((frequency_audio),pow*4);
xlabel('Frequency');
ylabel('Power Spectral');
title('Power spectrum of Noisy signal');

[audio_in_2,audio_freq_sampl_2]=audioread('Output.wav');
Length_audio_2=length(audio_in_2);
df_2=audio_freq_sampl_2/Length_audio_2;
frequency_audio_2=-audio_freq_sampl_2/2:df_2:audio_freq_sampl_2/2-df_2;
pow_2 = (abs(filtered_signal_2).^2)/Length_audio_2^2;

subplot(3,1,3)
plot((frequency_audio_2),pow_2*4);
xlabel('Frequency');
ylabel('Power Spectral');
title('Power spectrum of Output signal');

[audio_in,audio_freq_sampl]=audioread('Input_audio.wav');
Length_audio=length(audio_in);
df=audio_freq_sampl/Length_audio;
frequency_audio=-audio_freq_sampl/2:df:audio_freq_sampl/2-df;

figure
subplot(3,1,1)
FFT_audio_in=fftshift(fft(audio_in))/length(fft(audio_in));

plot(frequency_audio,abs(FFT_audio_in));
```

```
title('FFT of Input Audio');
xlabel('Frequency(Hz)');
ylabel('Amplitude');

[audio_in,audio_freq_sampl]=audioread('Noisy-Signal.wav');
Length_audio=length(audio_in);
df=audio_freq_sampl/Length_audio;
frequency_audio=-audio_freq_sampl/2:df:audio_freq_sampl/2-df;

subplot(3,1,2)

FFT_audio_in=fftshift(fft(audio_in))/length(fft(audio_in));
plot(frequency_audio,abs(FFT_audio_in));
title('FFT of Noisy signal');
xlabel('Frequency(Hz)');
ylabel('Amplitude');

[audio_in,audio_freq_sampl]=audioread('Output.wav');
Length_audio=length(audio_in);
df=audio_freq_sampl/Length_audio;
frequency_audio=-audio_freq_sampl/2:df:audio_freq_sampl/2-df;

subplot(3,1,3)

FFT_audio_in=fftshift(fft(audio_in))/length(fft(audio_in));
plot(frequency_audio,abs(FFT_audio_in));
title('FFT of Output Audio');
xlabel('Frequency(Hz)');
ylabel('Amplitude');

figure
subplot(3,1,1)
plot(t , xnew);
title('Original: Time-domain');
xlabel('time(seconds)');
ylabel('Amplitude');
subplot(3,1,2)
plot(t , xn , 'r');
```

```
title('Noisy: Time-domain');
xlabel('time(seconds)');
ylabel('Amplitude');
subplot(3,1,3)
plot(t , filtered_signal_2 , 'g');
title('After processing Noisy: Time-domain');
xlabel('time(seconds)');
ylabel('Amplitude');

figure
title('Pole zero Plot')
zplane(Num,1)
figure
title('Frequency Response')
freqz(Num,1,N)
figure
title('Impulse Response')
impz(Num,1)

%%
sound(x,Fs)
pause
%%
sound(xn,Fs)
pause
%%
sound(filtered_signal_2,Fs)
```

6. RESULTS :

Filter Parameters

- In the filter specifications the sampling frequency is set to 8000 Hz and β is set to 5, cutoff frequency is set to 2500 Hz and its order is 40.

Low Pass Filter Characteristics of Magnitude Response and Phase Response :

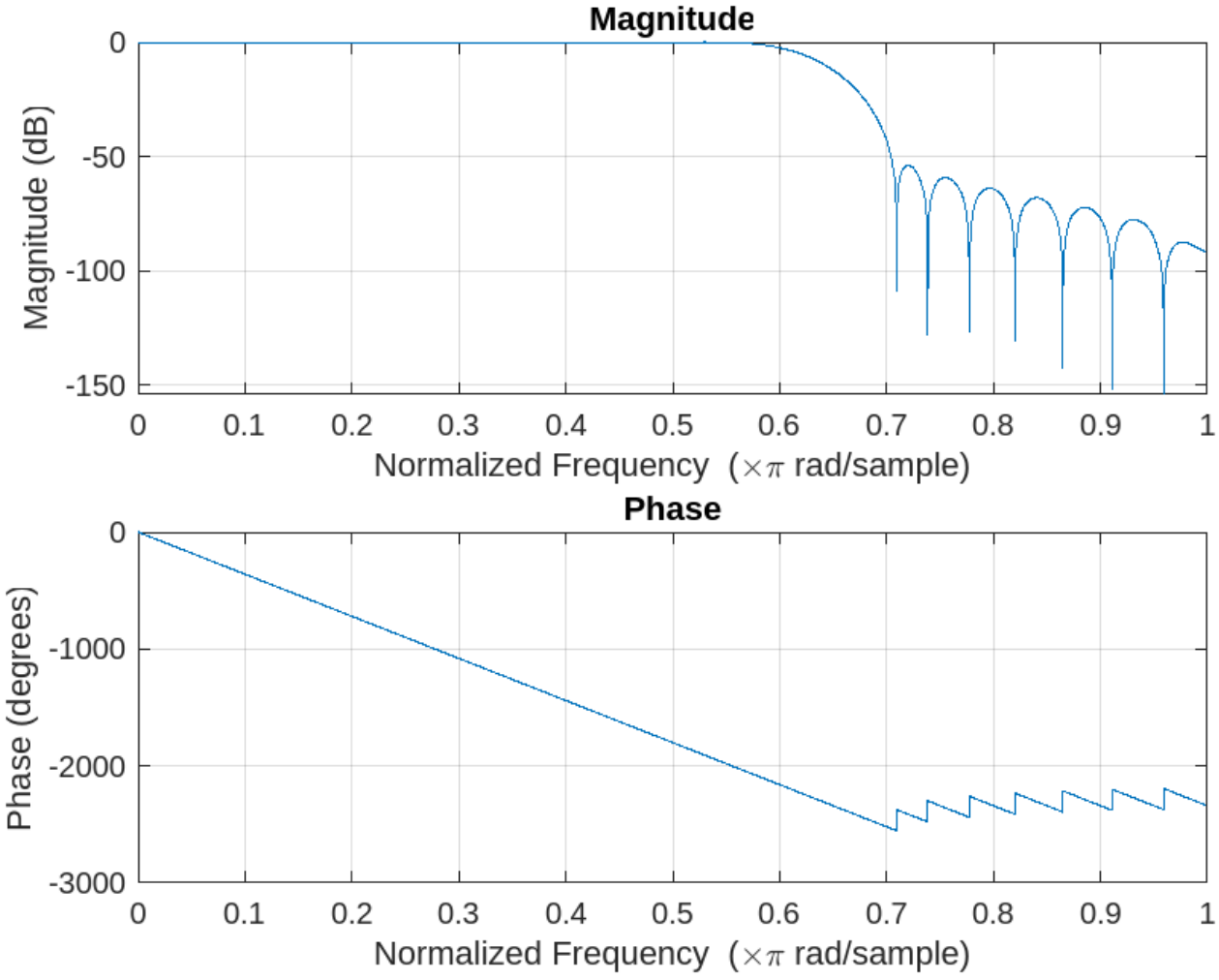


Fig 6.1

Fig 6.1 Shows the magnitude and phase response of the output filtered signal.

Pole-Zero Plot of the Transfer Function of LPF Filter

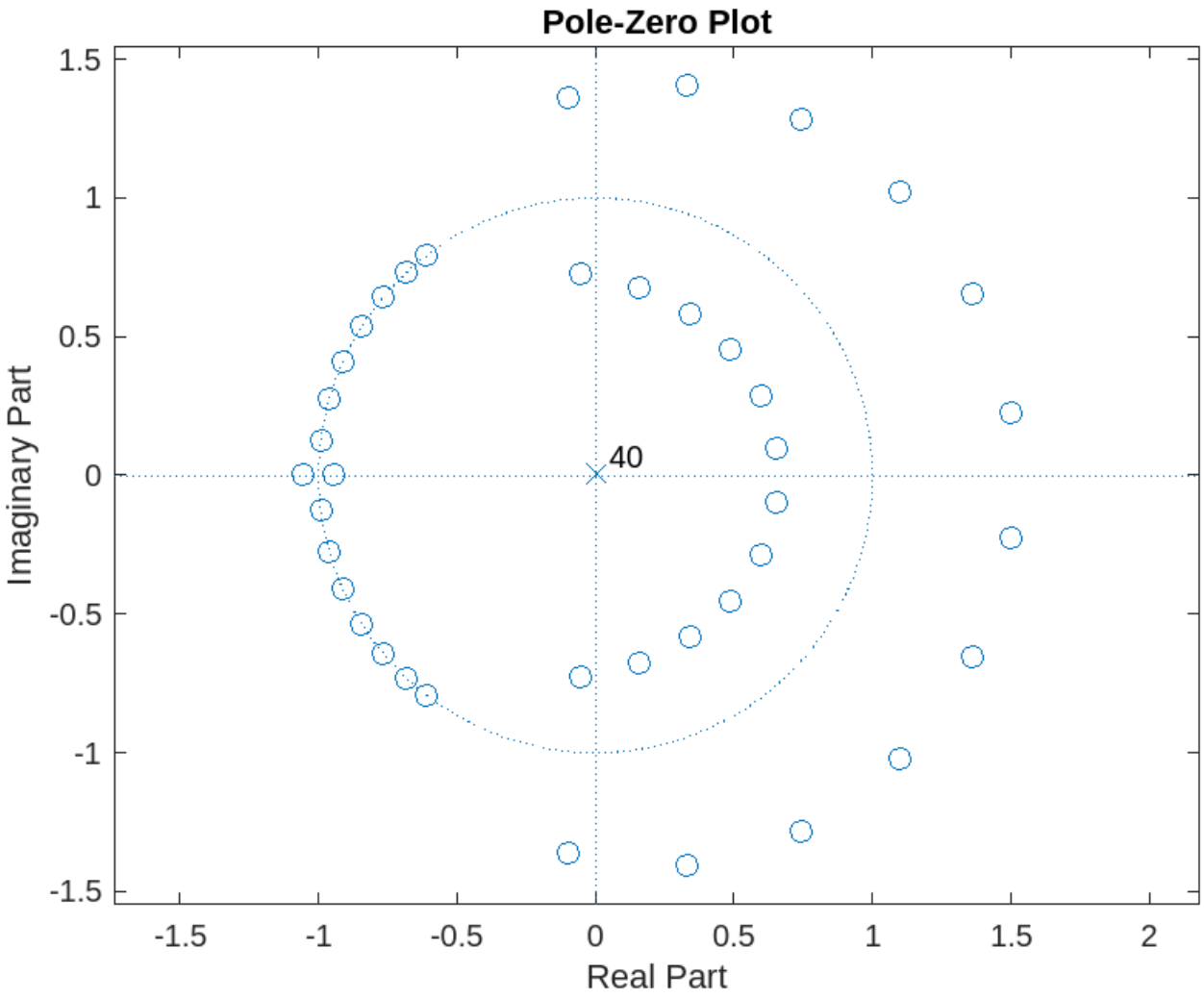


Fig 6.2

Fig 6.2 Shows the order of the filter is 40, in the above pole zero plot 40 zero locations can be observed with hollow circles and pole location at the center of the circle.

FFT Input audio, Noisy audio, Output(Filtered) audio in Frequency domain

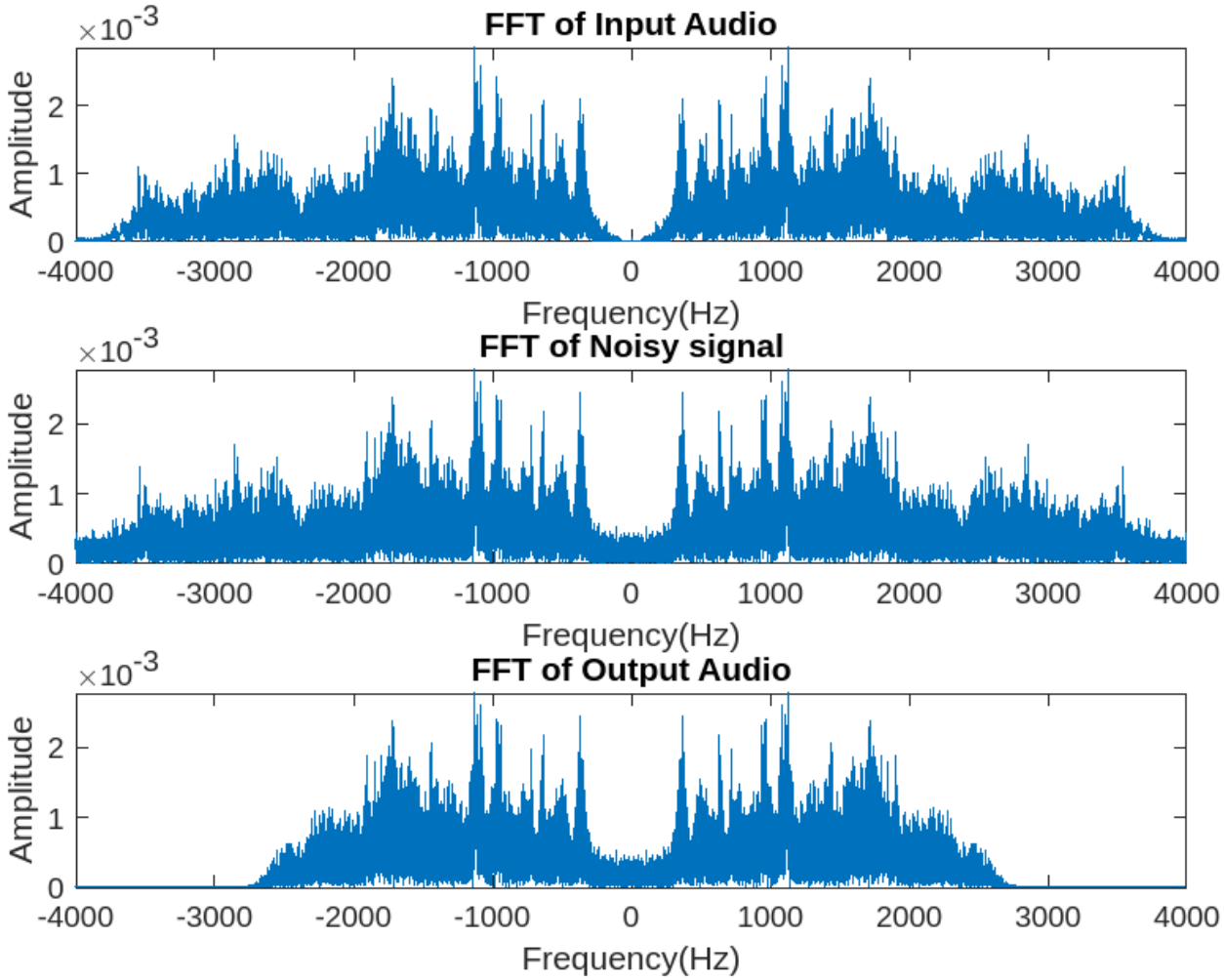


Fig 6.3

The Fig 6.3 shows the frequency domain analysis of the input, noisy and output signals.

Time domain of Input audio, Noisy audio, Output(Filtered) audio

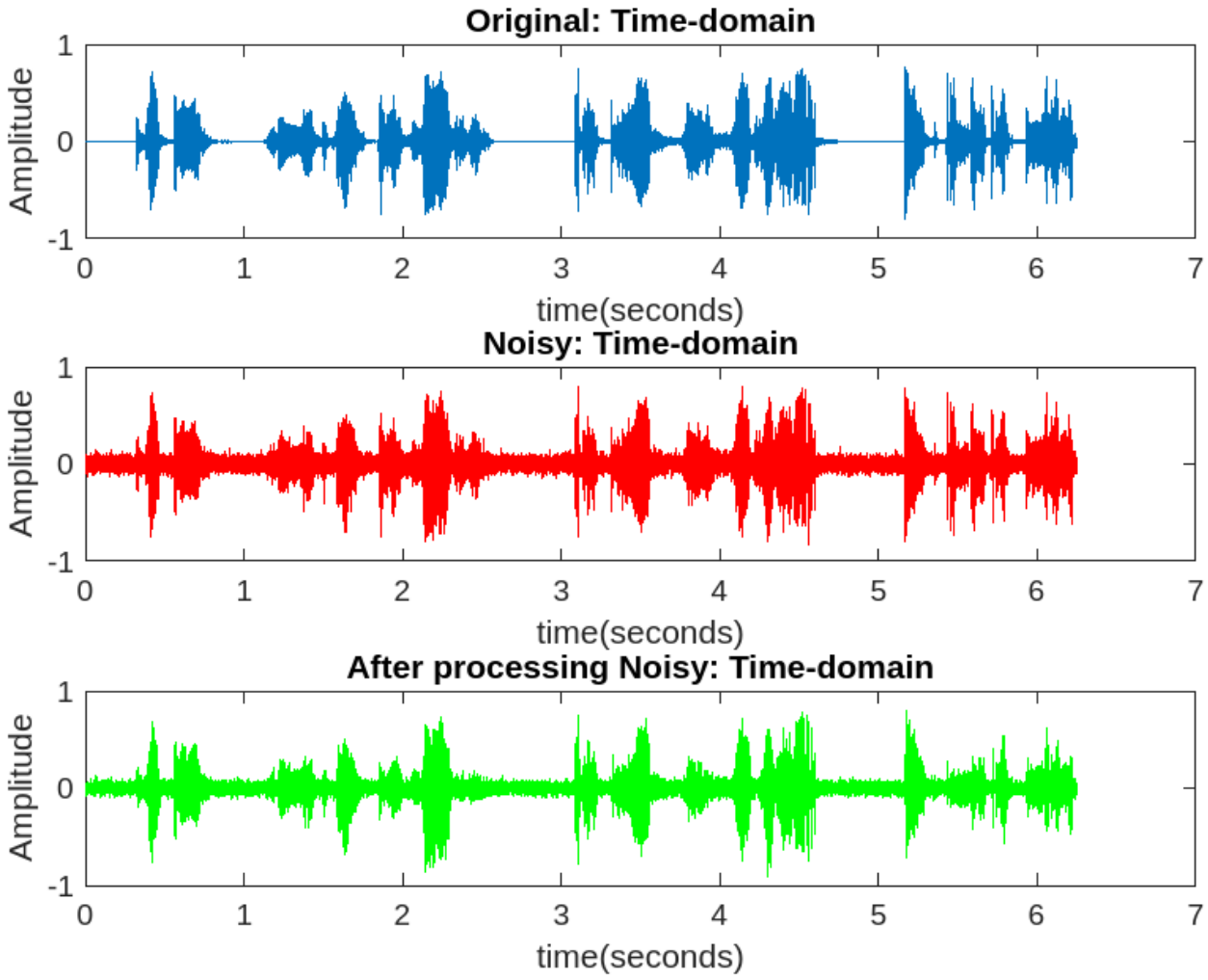


Fig 6.4

The Fig 6.4 shows the time domain analysis of the input, noisy and output signal.

Power Spectrum of Input signal, Noisy signal and Output(Filtered) Signal

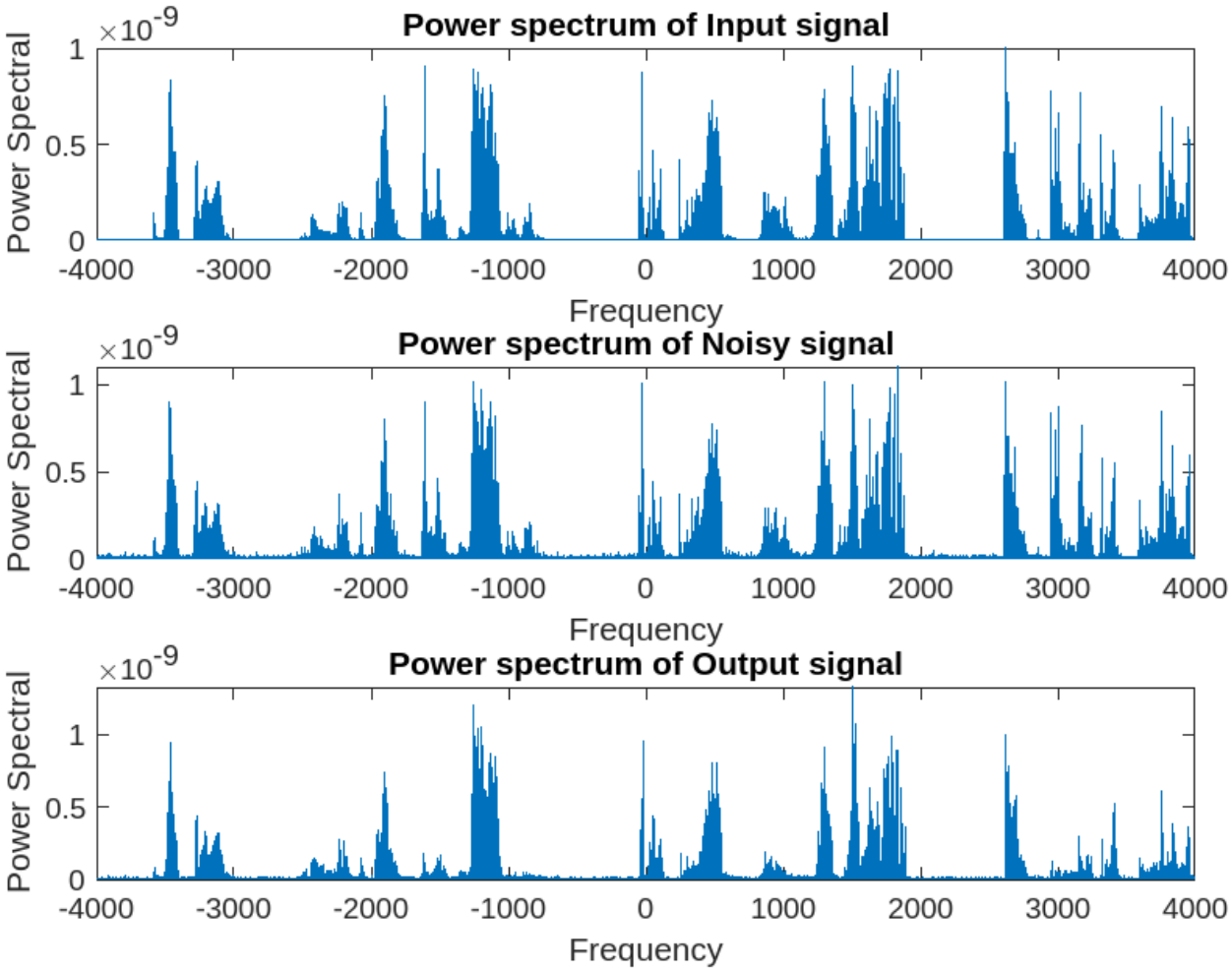


Fig 6.5

The Fig 6.5 shows the comparison between the power spectrums of the input, noisy and output signal.

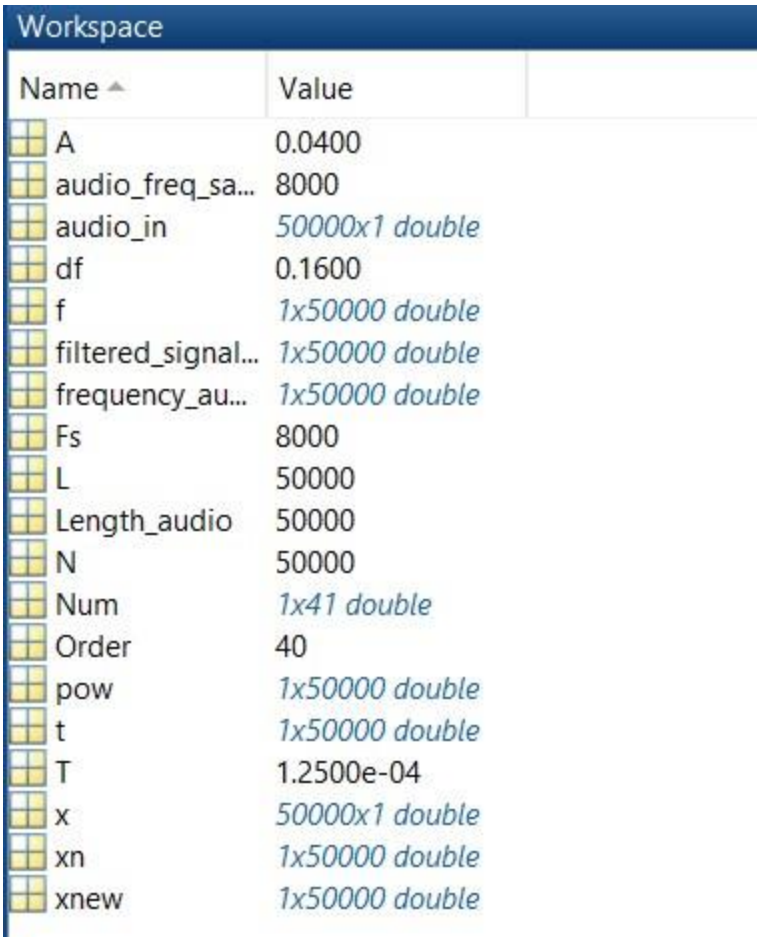


Fig 6.6

In Fig 6.6 workspace specifies the values of the variables that are used in the filter code.

Conclusion:

In this, a Low pass filter has been designed which stops the high frequencies after the given cutoff frequency. Basically a low pass filter has been designed based on its application of removal of noise.The noisy audio high frequencies file is used as the input to the filter. After setting the cutoff frequency of FIR kaiser window of low pass filter, the filtered audio file with less noise compared to noisy audio has been obtained. Hence the designed filter has removed the high frequency audio which can be even recognised as output audio is filtered. Hence a low pass filter has been successfully designed which is used for removal of unwanted noise from the signal.

7. APPLICATION:

- The low pass filter applications include the following.
- Electronic low-pass filters are used on inputs to subwoofers and other types of loudspeakers, to block high pitches that they cannot efficiently reproduce.
- Radio transmitters use low-pass filters to block harmonic emissions that might interfere with other communications.
- Used to remove the noise of high-frequency signals.
- Used in audio applications.
- Used in biomedical applications.
- Used in electronic applications like loudspeakers, subwoofers, etc.
- Used in digital to analog converters.
- Used as anti-analyzing filters.
- Used in wave analyzers, audio amplifiers, and equalizers.

8. LIMITATIONS AND FUTURE SCOPE:

Limitations:

- If a scenario exists where the high frequency noise is added at random positions anywhere in the signal, then in such scenarios applying a low pass filter may not be applicable.
- As far as practical scenarios are concerned , we can never apply low pass filters in audio files which are corrupted by high amplitude random noise or gaussian noise and in which noise is randomly distributed all over the signal.
- To get the ideal filter characteristics of a Low pass filter with minimum order is one of the limitations of the designed filter.

Future Scope:

- This Low pass filter can be used to stop a wide range of high frequencies if we use the specify order option in the filter designer step. As order increases no doubt the hardware increases but the characteristics will reach ideal characteristics with less ripples in bands. Here we have taken monotonic signal , in future we can implement stereo type signal also.

9. REFERENCES:

- https://en.wikipedia.org/wiki/Low-pass_filter#:~:text=Electronic%20low%2Dpass%20filters%20are,might%20interfere%20with%20other%20communications.
- https://in.mathworks.com/matlabcentral/answers/68771-adding-noise-to-a-wav-file
- https://in.mathworks.com/help/dsp/ug/lowpass-filter-design.html
- https://in.mathworks.com/matlabcentral/answers/217711-how-to-add-random-noise-to-a-signal#:~:text=noisy_signal%20%3D%20rand(1%2C%20100,to%20create%20a%20noisy%20signal.