Implementation of filters and Noise removal by using filters

A MAJOR PROJECT REPORT

Submitted by

CH.EN.U4AIE20001	ANBAZHAGAN E
CH.EN.U4AIE20053	RAMYA POLAKI
CH.EN.U4AIE20069	TRINAYA KODAVATI
CH.EN.U4AIE20031	SMITHIN REDDY K
CH.EN.U4AIE20035	ABHIRAM KUNCHAPU

Under the Guidance of

Dr. Jayaram K P

In partial fulfilment for the award of the degree

Of

BACHELORS OF TECHNOLOGY

In COMPUTER SCIENCE ENGINEERING

AMRITA SCHOOL OF ENGINEERING, CHENNAI

AMRITA VISHWA VIDYAPEETHAM

CHENNAI – 601103, TAMIL NADU

DECEMBER -2021

Acknowledgement:

We offer our sincere pranams at the lotus feet of Universal guru, MATA AMRITANANDAMAYI DEVI who blessed us with her grace to make this a successful major project.

We express our deep sense of gratitude to Dr. P Shankar, Principal, Amrita School of Engineering, Chennai for his kind support. We would like to extend my gratitude and heartfelt thanks to Dr. A Manikandan, Chairperson, Department of Electrical and Electronics Engineering, for his constant help, suggestions and inspiring guidance. I am grateful to my guide Dr. Jayaram K P, for his invaluable support and guidance during the course of the major project work. I am also indebted to Dr. Prasanna Kumar, Chairperson of Department of AI, ASE, Chennai for his guidance.

We are also thankful to all other staff members of the Department of Computer Science Engineering for their valuable help, all my classmates who have always been a source of strength, for always being there and extending their valuable helps to the successful completion of this work

Table of contents: **Pg No:**

Abstract:

Filters are a basic component of all signal processing and telecommunication systems. The primary functions of a filter are one or more of the followings:

(a) to confine a signal into a prescribed frequency band or channel for example as in anti-aliasing filter or a radio/tv channel selector,

(b) to decompose a signal into two or more sub-band signals for sub-band signal processing, for example in music coding,

(c) to modify the frequency spectrum of a signal, for example in audio graphic equalizers, and

(d) to model the input-output relation of a system such as a mobile communication channel, voice production, musical instruments, telephone line echo, and room acoustics.

In this we introduce the general form of the equation for a linear time-invariant filter and consider the various methods of description of a filter in time and frequency domains. We study different filter forms and structures and the design of low-pass filters, band-pass filters, band-stop filters.

In digital control system, interference, which is mixed in the input signal, has a great influence on the performance of the system. Therefore, processing of input signal has to be done to get useful signal. Finite impulse response (FIR) filter plays an important role in the processing of digital signal. Designing the FIR filter by MATLAB can simplify the complicated computation in simulation and improve the performance. By using the methods of window function, the design of FIR filter has been processed by MATLAB.

A Digital filter finds applications in digital signal processing and can be implemented using either infinite impulse response (IIR) or finite impulse response (FIR) methods. The designing of FIR filter is chosen as they have several advantages

To remove the presence of large oscillations in both passband and stopband we have chosen Kaiser window function that contains a taper and decays towards zero gradually.

CHAPTER 1:

Introduction:

The field of signal processing includes filtering as a basic and very essential process. It is a linear system which is used for the removal of noise and all other unwanted components from the signal and gets the desired signal in the output. Using the filters, the desired amplitude phase and frequency of a signal can be obtained from the original signal. Both the digital and analog filters are a part of filtering. The digital filters are more preferable as compared to the analog filters in many fields as it is efficient for detecting and filtering the noise signals.

For the digital filtering the input analog signal is converted into digital signal using sampling and then it is processed and converted back to the analog form and received as the output. The digital filters are classified into many various types based on several factors and characteristics.

The IIR filters and the FIR filters have been explained. The FIR Filters and their related functions have been explained in the following sections

Types of Filters:

Low-pass filter

Low-pass filters allow the frequencies below the selected cut-off frequency level and cut the frequencies above the cut-off range.

High-pass filter

A high-pass filter is the opposite of a low-pass filter. It filters and passes the frequency, which is higher than the cut-off frequency range and attenuates the frequency lower than the cut-off range.

Bandpass Filter

After resampling of signals, band pass filter is applied to remove the extra noise and be considered the most ideal filter in signal processing. It attenuates the frequencies which are higher or lower than the cut off frequencies range and only passes the frequencies which fall within the cut-off range.

Band-rejection/stop filter

It is also known as a notch filter and opposite of band-pass filter. It leaves most of the frequencies unaltered and attenuates those within a specified range to very low levels.

Objectives:

- Implement various filters in MATLAB
- Record Audio Using Microphone/Or Input from the system.
- Saving the recording audio in the computer.
- Results of the filters techniques (Frequency vs dB)
- Adding Noises to the Audio and removing the noise by using filter techniques
- And Implementation in GNU radio

Chapter 2:

Methodology:

Filtering

Filters are considered the most basic circuit in any signal processing used in almost every process. It removes the unwanted noise, echo, distortion, and allows the filtered data to pass through it. We will be discussing pass filters that allow only specific frequencies while rejecting others.

There are Low-pass filter, High-pass filter, Bandpass Filter Band-reject

The other classification of filters is based on the time domain. They are:

1) Infinite Impulse Response (IIR) Filters
2) Finite Impulse Response (FIR) Filters

Introduction of digital filter:

The digital filter is a discrete system, and it can do a series of mathematic processing to the input signal, and therefore obtain the desired information from the input signal. The transfer function for a linear, time-invariant, digital filter is usually expressed as

$$H(z) = \frac{\sum_{j=0}^{M} b_j z^{-j}}{1+\sum_{i=1}^{N} a_i z^{-i}} \ ,$$

where ai and bi are coefficients of the filter in Z-transform. There are many kinds of digital filters, and also many different ways to classify them. According their function, the FIR filters can be classified into four categories, which are lowpass filter, highpass filter, bandpass filter, and band stop filter. According to the impulse response, there are usually two types of digital filters, which are finite impulse response (FIR) filters and infinite impulse response (IIR) filters. According to the formula above, if ai is always zero, then it is a FIR filter, otherwise, if there is at least one none-zero ai, then it is an IIR filter. Usually, we need three basic arithmetic units to design a digital filter, which are the adder, the delay, and the multiplier.

The following are several steps of designing a digital filter:

1. Make sure of the property of a digital filter according to the given requirements.

2. Use a discrete linear time-invariant system function to approach to the properties.

3. Make use of algorithms to design the system function.

4. Use a computer simulation or hardware to achieve it.

FIR filter:

The finite impulse response (FIR) filter is one of the most basic elements in a digital signal processing system, and it can guarantee a strict linear phase frequency characteristic with any kind of amplitude frequency characteristic. Besides, the unit impulse response is finite; therefore, FIR filters are stable system. The FIR filter has a broad application in many fields, such as telecommunication, image processing, and so on.

The system function of FIR filter is

$$H(z) = \sum_{n=0}^{L-1} h[n]\, z^{-n},$$

where L is the length of the filter, and h[n] is the impulse response.

The design methods of FIR Filters are based on ideal filter approximation. Using this approximation, the filter designed is of a higher order, due to which it

becomes complex to implement. The transfer function of FIR filters is as shown in equation below.

$$H(z) = \frac{Y(Z)}{X(z)} = \sum_{k=0}^{N} b_k z^{-k}$$

The designing process takes into consideration the required characteristics and specifications. The FIR filters are designed using various windowing techniques as shown in the figure below.

Name of Window function	Transition Width (Hz) (Normalized)	Pass band ripple (dB)	Main Lobe Relative to Side Lobe (dB)	Stopband attenuation (dB) (Maximum)	Window Function w(n), |n|≤(N-1)/2
Rectangular	0.9/N	0.7416	13	21	1
Hanning	3.1/N	0.0546	31	44	$0.5 + 0.5cos\left(\frac{2\pi n}{N}\right)$
Hamming	3.3/N	0.0194	41	53	$0.54 + 0.46cos\left(\frac{2\pi n}{N}\right)$
Blackman	5.5/N	0.0017	57	75	$0.42 + 0.5cos\left(\frac{2\pi n}{N-1}\right) + 0.8cos\left(\frac{4\pi n}{N-1}\right)$
Kaiser	2.93/N (β=4.54)	0.0274	-	50	* ([ ]) + ( )
	4.32/N (β=6.76)	0.00275		70	
	5.71/N (β=8.96)	0.000275		90	

IIR filter:

IIR Filters are the digital filters that have an infinite impulse response. They are also known as recursive filters as they have feedback and hence, they produce a better frequency response. The IIR filters do not possess linear phase characteristics. This being their limitation, they cannot be preferred for a linear

phase system. The IIR filters acquire less memory and also include fewer calculations.

(1) The IIR filters can be realized by Butterworth, Chebyshev-I and Elliptic filters, of which the Butterworth and Chebyshev-I filters

Comparison of FIR and IIR:

(1) Under the same conditions as in the technical indicators, output of the IIR filter has feedback to input, so it can meet the requirements better than FIR. The storage units are less than those of IIR, the number of calculations is also less, and it's more economical.

(2) The phase of FIR filter is strictly linear, while the IIR filter is not. The better the selectivity of IIR filter is, the more serious the nonlinearity of the phase will be.

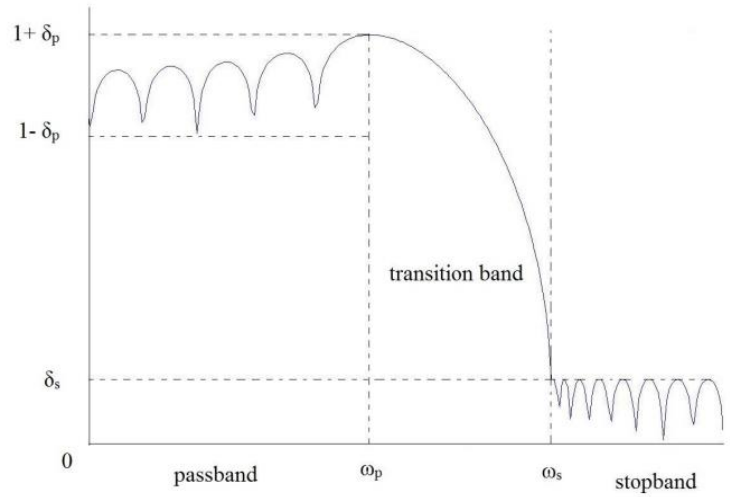
(3) The FIR filter is non-recursive structure; finite precision arithmetic error is very small. While IIR filter is recursive structure, and parasitic oscillation may occur in the operation of IIR filter.

(4) Fast Fourier Transformation can be used in FIR filter, while IIR cannot.

(5) The IIR filter can make use of the formulas, data and tables of the analog filter, and only a small amount of calculation. While FIR filter design may always make use of the computer to calculate, and the order of FIR filter could be large to meet the design specifications.

Design of FIR Filters:

Design of FIR filter It is necessary to specify passband(s), stopband(s), and transition band(s) when designing a frequency-selective filter. In passband(s), frequencies are needed to be passed unattenuated. In stopband(s), frequencies need to be passed attenuated. Transition band(s) contain(s) frequencies which are lying between the passband(s) and stopband(s). Therefore, the entire frequency range is split into one or even more passbands, stopbands, and transition bands. In practical, the magnitude is not necessary to be constant in the passband of a filter. A small amount of ripple is usually allowed in the passband. Similarly, the filter response does not to be zero in the stopband. A small, nonzero value is also tolerable in the stopband. We can see some ripples in the following picture.



As the figure shows, the transition band of the filter is between the passband and the stopband. The frequency ω_p denotes the edge of the passband, and the band-edge frequency ωs defines the edge of the stopband. So, the difference of ω_s and ω_p is the width of the transition band,

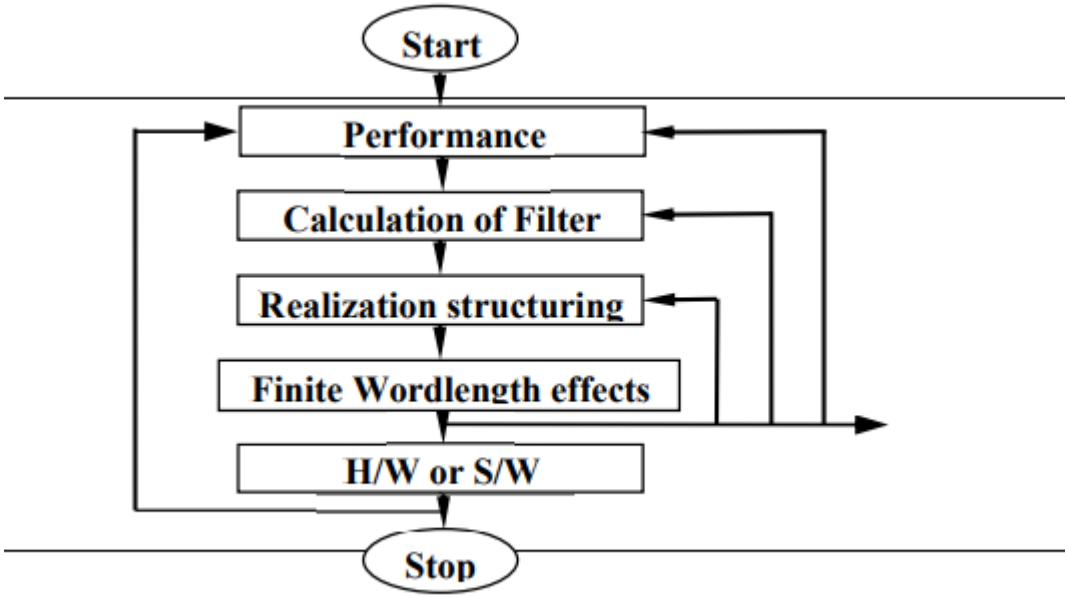
i.e. $\omega_t = \omega_s - \omega_p$.

The ripple in the passband of the filter is denoted as δ_p, and the magnitude of the filter varies from 1-δ_p to 1+ δ_p.

δ_s is the ripple in the stopband. Usually, we use a logarithmic scale to show the frequency response, hence, the ripple in the passband is $20\log 10\delta_p$ dB, and the ripple in the stopband is $20\log 10\delta_s$ dB

CONCEPT OF FILTER DESIGN:

The fundamental concept of FIR filter design is that the filter frequency response is determined by the impulse response & quantized impulse response & the filter coefficients are identical. The input to FIR filter is an impulse, and as the impulse propagates through the delay elements, the filter output is identical to the filter coefficients. The FIR filter design process therefore consists of determining the impulse response & then quantizing the impulse response to generate the filter coefficients. The design of a digital filter involves five steps - Filter specification ii) Filter coefficient calculation, iii) Realization, iv) Analysis of finite word length effects and v) Implementation This involves producing the software code and/or hardware and performing the actual filtering. These five inter related steps are summarized by flow chart:



In the passband, the magnitude response has a peak deviation of δ_p and in the stopband, it as a maximum deviation of δ_s . The width of transition band determines how sharp the filter is. The magnitude response decreases monotonically from the passband to stopband in this region.

The following are the key parameters of interest:

δ_p peak passband deviation (or ripples)

δ_s stopband deviation.

f stopband edge frequency.

f_p passband edge frequency.

f_s sampling frequency.

The edge frequencies are often given n the normalized form, that is as the fraction of the sampling frequency (f_s / F). Passband and stopband deviation may be expressed in decibels. When they specify the passband ripples and minimum stopband attenuation respectively.

Thus, the minimum stopband attenuation,

A_s and the peak passband ripple, A_p in decibels are given as:

A_s (Stopband attenuation) = -20 log10 δ_s

A_p (Passband ripple) = 20 log10 (1 $+\delta_p$)

The difference between δ_s and δ_p is the transition width of the filter. Another important parameter is the filter length, N, which defines the number of filters.

Window function:

Window function in this method, a truncated ideal lowpass filter with a certain bandwidth is generated, and then we use a chosen window to get certain stopband attenuation. The length of filter L can be adjusted to meet a specified roll-off rate in the transition band. Let's start with windowed, truncated lowpass filters, then other kind of filters, like highpass, bandpass, and band stop filters can also be achieved by several technique.

Any finite-length of the ideal lowpass impulse response may be considered as the product of the infinite-length lowpass impulse response and a window function W, which has a finite number of contiguous nonzero-valued samples

$$b = \frac{\sin(\omega_c[n-M])}{\pi[n-M]} W_L[n-M],$$

where L is the window length, M=(L-1)/2, 0≤ n ≤ L-1, and WL[n] is generally a function FE[n] which has even symmetry about M defined as

$$W_L[n] = \begin{cases} F_E[n] & 0 \ll n \ll L-1 \\ 0 & otherwise \end{cases}.$$

The result is a finite-length or truncated lowpass filter

KAISER WINDOW:

The width of the main lobe and attenuation of side lobes depends only upon length 'M', i.e, length of the window. Kaiser window allows separate control of width of the main lobe and attenuation of the sidelobe.

A simple approximation of the window using Bessel functions, as discovered by Jim Kaiser can be given by the formula:

$$w(n) = \frac{I_0\left(\pi\alpha\sqrt{1-\left(\frac{2n}{N-1}-1\right)^2}\right)}{I_0(\pi\alpha)} \quad ; \quad \textbf{for } 0 \le n \le M$$

$$0 \quad ; \quad \textbf{elsewhere}$$

There are two design formulas that can help you design FIR filters to meet a set of filter specifications using a Kaiser window. To achieve a relative sidelobe attenuation of $-\alpha$ dB, the β (`beta`) parameter is

$$\beta = \begin{cases} 0.1102(\alpha - 8.7), & \alpha > 50, \\ 0.5842(\alpha - 21)^{0.4} + 0.07886(\alpha - 21), & 50 \ge \alpha \ge 21, \\ 0, & \alpha < 21. \end{cases}$$

For a transition width of $\Delta\omega$ rad/sample, use the length

$$n = \frac{\alpha - 8}{2.285\Delta\omega} + 1.$$

Filters designed using these heuristics will meet the specifications approximately, but you should verify this. To design a lowpass filter with cutoff frequency 0.5π rad/sample, transition width 0.2π rad/sample, and 40 dB of attenuation in the stopband, try

```
[n,wn,beta] = kaiserord([0.4 0.6]*pi,[1 0],[0.01 0.01],2*pi);
h = fir1(n,wn,kaiser(n+1,beta),'noscale');
```

The kaiserord function estimates the filter order, cutoff frequency, and Kaiser window beta parameter needed to meet a given set of frequency domain specifications.

The ripple in the passband is roughly the same as the ripple in the stopband. As you can see from the frequency response, this filter nearly meets the specifications:

The FIR filter design based on MATLAB:

The realization of window function method by MATLAB:

We often use command firl and kaiserord in the realization of window function in Matlab. Following are some definitions of these two functions.

Function firl: b=firl(n,Wn, 'ftype', window)

where n is the order of filter; Wn is cutoff frequency, normalized frequency of 0 and 1, where 1 corresponds to the Nyquist frequency. If Wn is a two-element vector, Wn = [w1 w2], then it returns to a bandpass filter with a passband from w1 to w2. If Wn is a multi-element vector, Wn = [w1 w2 w3 … wn], it returns an order of n multiband filter with bands from 0 to w1, w1 to w2, …, wn to 1;

ftype is the type of filter, for example, ftype='high', it represents a highpass filter; ftype='stop', it represents a stopband filter. The default type is lowpass filter when there is no indication;

Window refers to required window, Matlab provides other types of windows they are boxcar(n) is rectangular window, hanning(n) is Hanning window, hamming(n) is Hamming window, blackman(n) is Blackman window, kaiserord(n, beta) is Kaiser window, and the default is Hamming window when there is no indication.

Function kaiserord: [n, Wn, beta, ftype]=kaiserord(f, a, dev, fs)

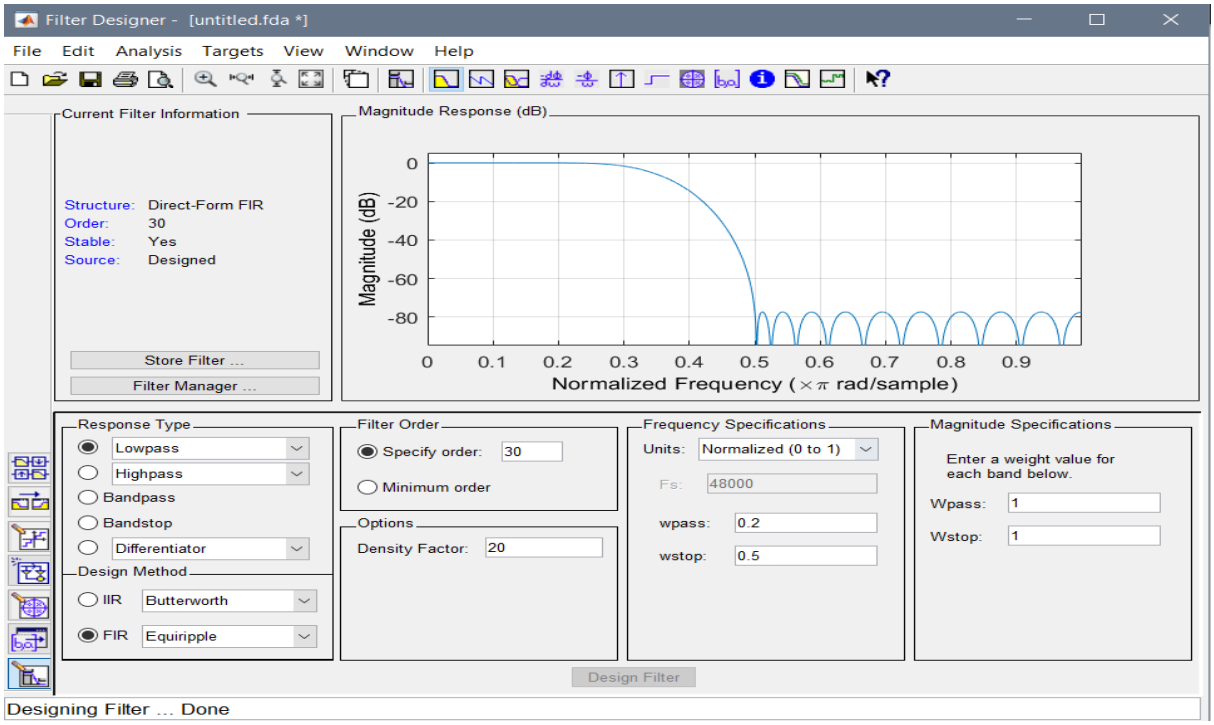
where f is a vector, stands for the start and ending point of filter's transition band; a is a vector, stands for the amplitude of specified frequency; dev is a vector, the same length with a, represents for the maximum amplitude error of each passband and stopband; n is the minimum order of filter that can meet the requirements; Wn is the cutoff frequency of filter; ftype is the type of filter

Chapter 3:

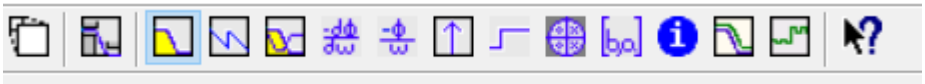
Analysis:

Filter Design and Analysis using FDATool of MATLAB SIMULINK:

Designing a low pass filter:



After setting the design specifications, click the Design Filter button at the bottom of the GUI to design the filter. The magnitude response of the filter is displayed in the Filter Analysis area after the coefficients are computed.



This Filter designing tool also provides allows us to observe several types of responses like order from left to right, the buttons are

- Magnitude response
- Phase response
- Magnitude and Phase responses
- Group delay response
- Phase delay response
- Impulse response
- Step response
- Pole-zero plot
- Filter Coefficients
- Filter Information.

We can select frequency specifications of the filter (pass band, stop band frequencies) and we can select attenuation specifications also to design a filter. Units are selected accordingly like units are normalized to to 0 to 1 .

By using Minimum order MATLAB will minimize the design with minimize possible order.

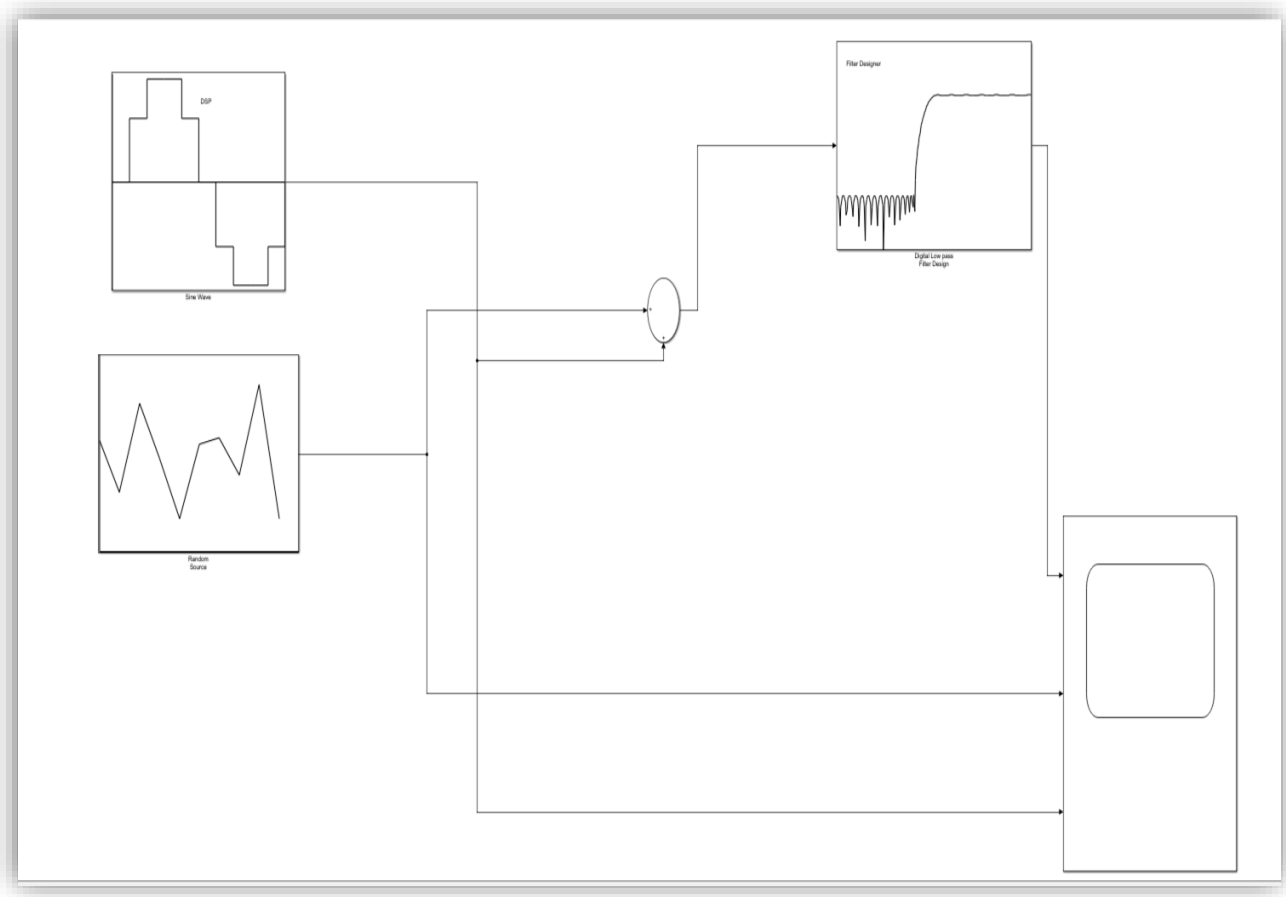
If exporting to the MATLAB workspace, you can export as coefficients or as an object by selecting from the Export from the pulldown menu.

Select Export from the File menu then we can export into MATLAB workspace.

And we can ark the points, optimize the design and we can use the different filter structure, like Filters can be converted to the following representations State-Space

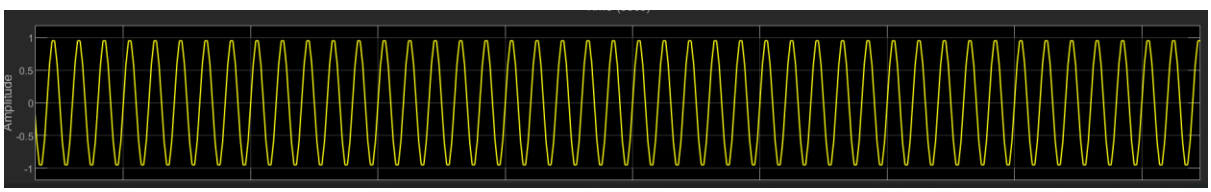
- Direct-Form FIR
- Direct-Form FIR Transposed
- Direct-Form Symmetric FIR.

Similarly, we can implement High, Band, Notch filters

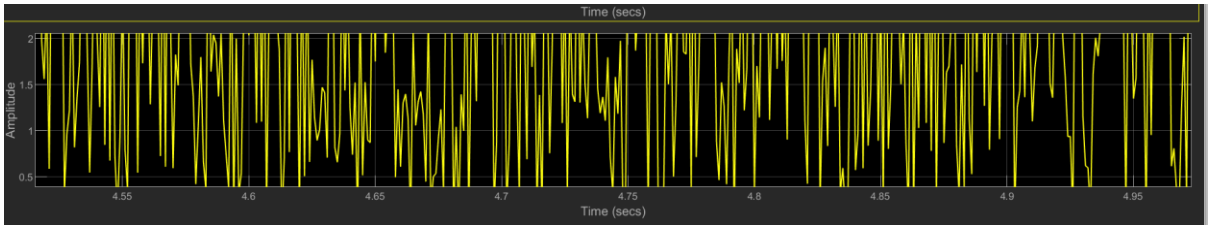


OUTPUT:

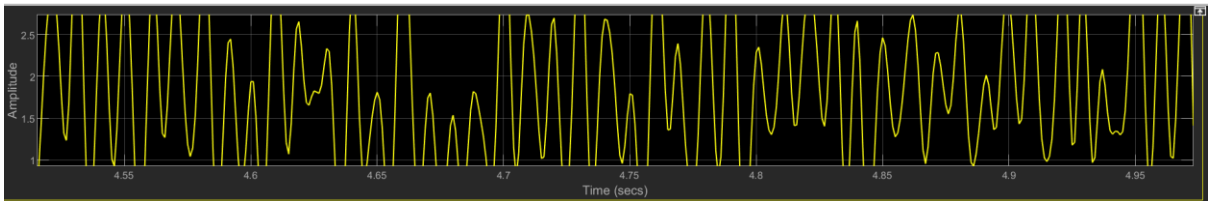
Sinewave:



Random source:

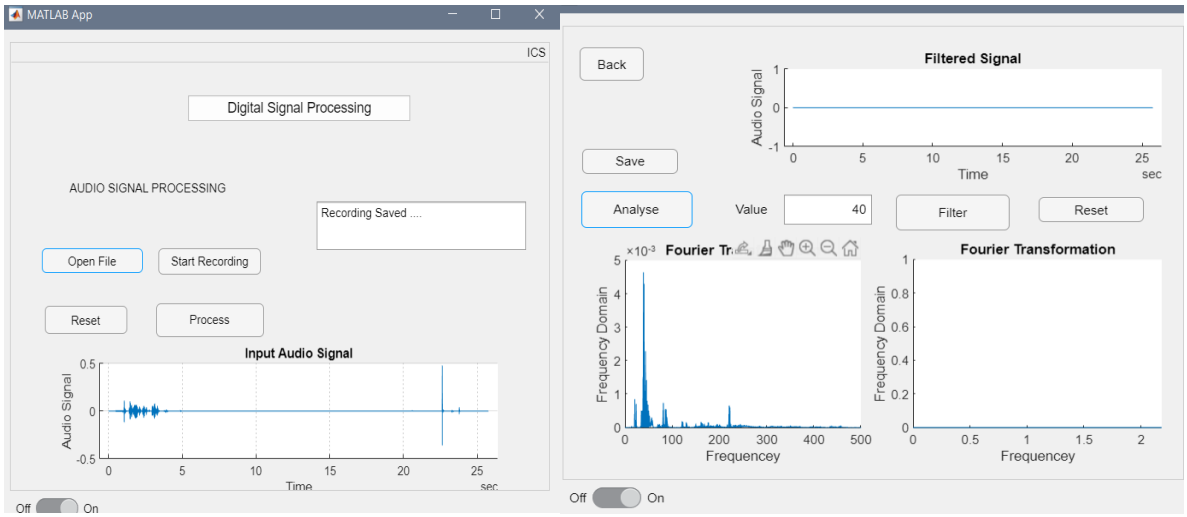


Filtered signal:



By changing the FDA Block parameters, we can change the type of filters.

MATLAB GUI:



A Graphical User Interface (GUI) is a pictorial interface that allows the user to use an application without understanding the language.

This is done by providing intuitive controls. The controls are the buttons that the user clicks to obtain a determined output. GUI is an event-driven program. This is because it acquires input at any given time and uses the call back functions to

execute the program and give results. call back function of the components that make it possible for them to execute the controls.

Buttons in GUI:

Open File: Opens the audio file from the system and it only takes .mp3 or .wav files respectively

Reset: Reset the audio file and record input.

Process: Gives the audio in time domain representation and takes to another interface where we can analyse and save the audio signal and analyse

Analyse: only works when we set the value of the filter and it will convert the input audio signal and removes noises by using low pass filter and also it will give representation of input audio signal in frequency domain, Gives the Fourier transformation of filtered signal.

We have used the low pass filter to remove noises from the input audio.

Analysis and Results in GNU radio:

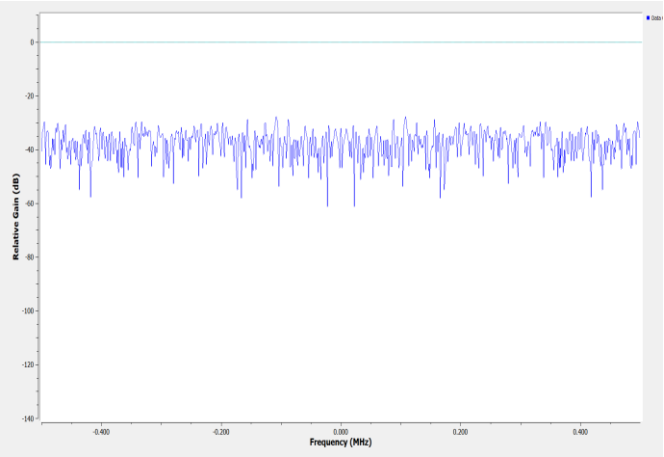


Fig: Input Signal Source

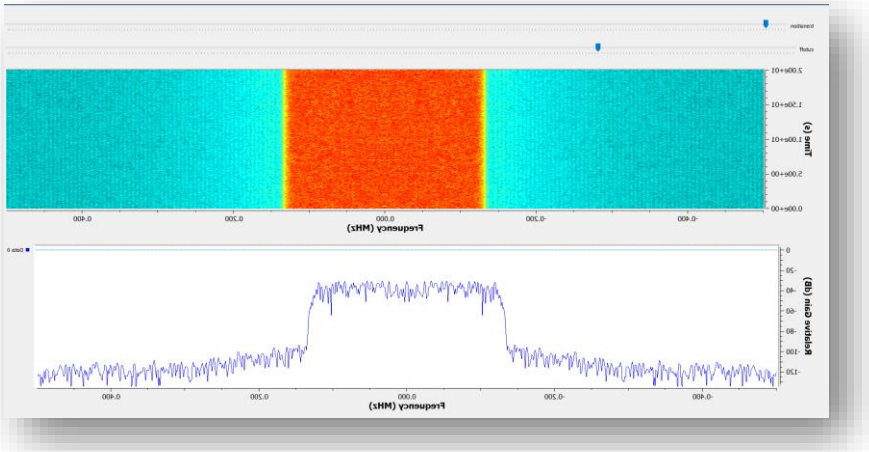


Fig: Low pass filter

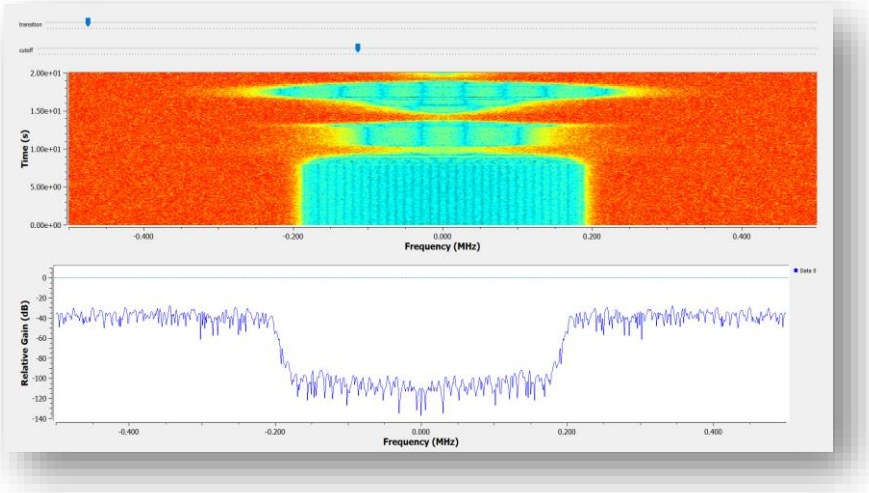


Fig: High pass filter

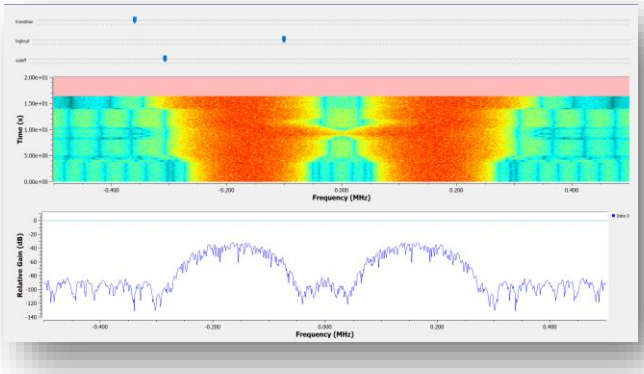


Fig: Band pass filter

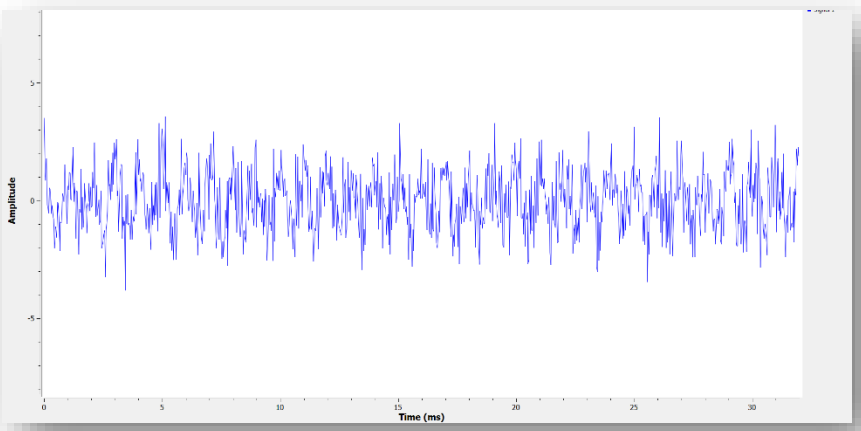
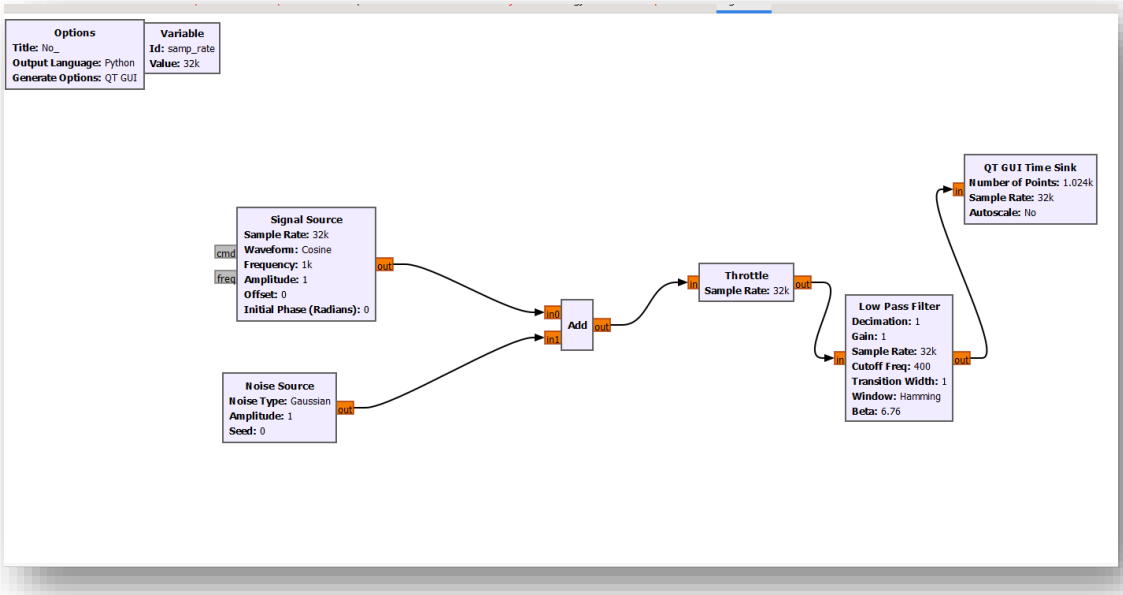


Fig: Signal source with Noise source



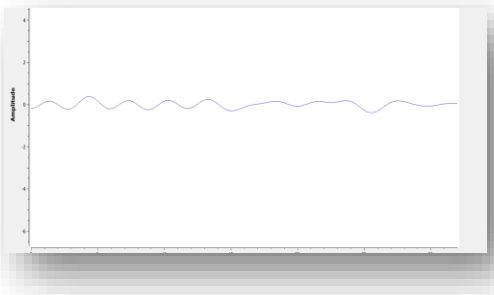
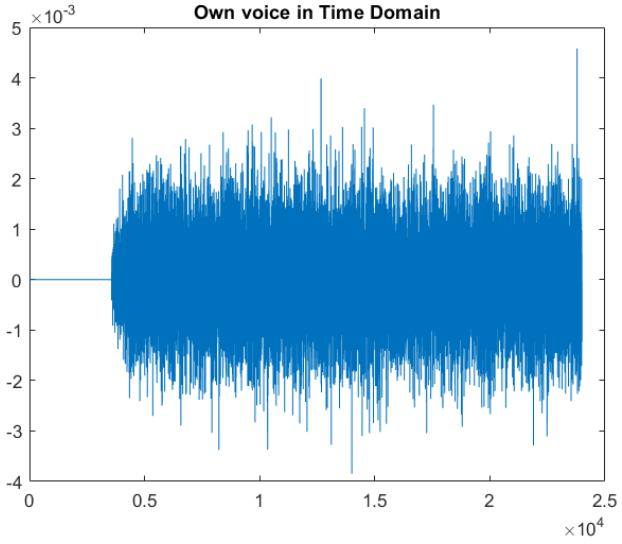
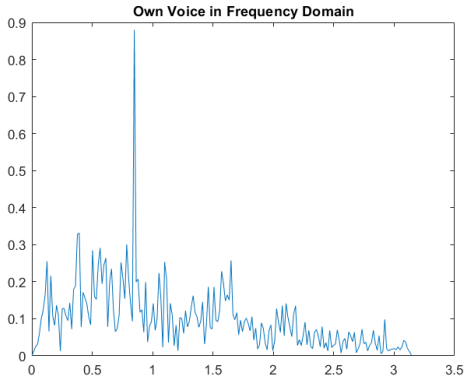


Fig: The filtered Signal using low pass filter

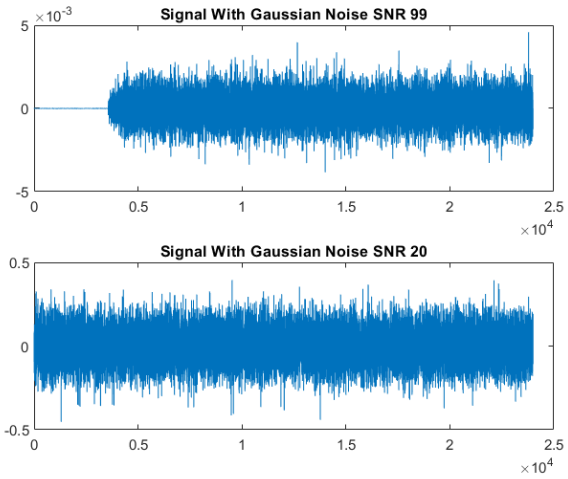
FIR FILTER OUTPUTS:



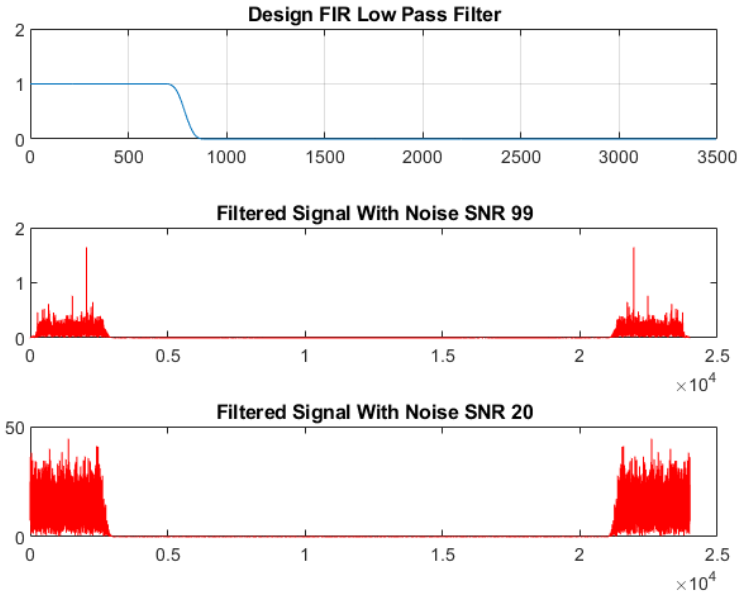
Voice in time domain



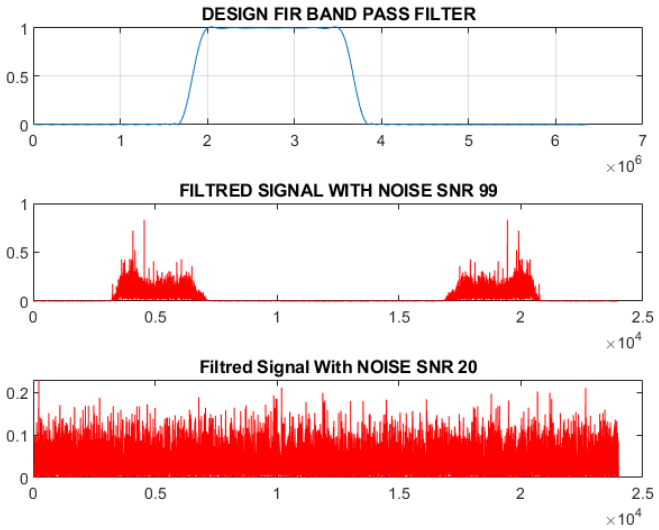
Voice in frequency domain



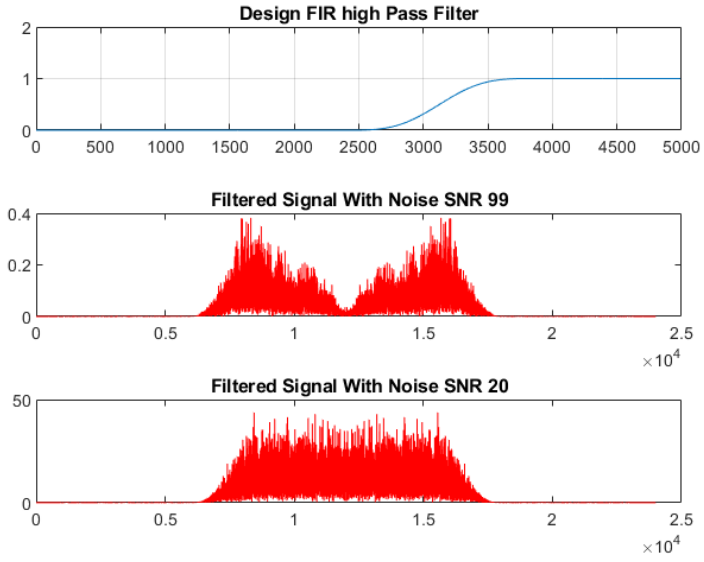
Signal with Noise SNR99 and SNR20



Low pass filtering



Band pass filtering



High pass Filtering

Steps in Audio signal processing:

- Own voice signal into time domain
- Convert into frequency domain
- Adding noise using 'awgn' function
- Audio signal processingOwn voice signal into time domain
- Convert into frequency domain
- Adding noise using 'awgn' function.
- Pass the noisy signal using filters

Inferences:

Filtering method depends on the type of noises in audio signal. In some signals the noise level is very high and it is not possible to recognize it by single recording, it is important to gain a good understanding of the signal processes involved before one attempt to filter preprocess a signal. Some audio signal is very sensitive in nature, and even if small noise mixed with original signal the characteristics of the signal changes. Data corrupted with noise must either filtered or discarded, filtering is important issue for Communication Systems

Future Scope:

Kaiser window is superior to other window techniques. The only disadvantage with the Kaiser window is that we design for the same ripple size in both passband and stopband. So, there would be some overdesign in either of the bands.

Filter is considered to be the suitable choice in wireless communication systems. This, however, is only the beginning. There are an infinite number of applications of digital filters that can be studied to use these foundations

Future work might involve a real time implementation of the system so that the maximum noise is reduced form the audio signals. In the future anybody can extent the order of the different filters and works on higher amplitude signals. They can calculate the efficiency of the filters that they have to implement.

Research can be extended in design of FIR filter using various optimization techniques ACO, PSO etc.,

In further work FIR filters can be design using evolutionary algorithms etc,..

And the optimal design of FIR filter can be achieved by using Weighted least square error method.

Noise removal in ECG signal using Kaiser, rectangular, hamming windows can be implemented.

when Kaiser window is compared with optimal filter design method, equiripple filter design found to be most suitable and optimized method to meet given specification

Conclusions:

Advances in digital audio technology have enabled us to create speech processing algorithms that are both efficient and high-quality. These algorithms are used during the recording, storage, and transmission of audio data. Audio content contains a lot of unwanted echo, interference, and distortions, all of which must be removed in order to get the required audio quality. It operates on the principle of converting analogue to digital audio signals, altering frequency ranges, removing undesirable noise, and adding audio effects to provide a smooth and perfect voice quality.

The major advantage of window technique is its simplicity. The availability of well-defined equations for calculating window coefficient has made this method preferable. But it offers very little design flexibility especially in low pass filter design. Kaiser window offers very low order to meet given specification. The best digital filter design results comes for using the Kaiser window from the windowing design technique, which has parameter s that allows adjustment of the compromise between the overshoot reduction and transition region width spreading

References:

https://www.mathworks.com/help/signal/ug/kaiser-window.html

https://www.mathworks.com/help/signal/ref/kaiserord.html

https://vikramlearning.com/jntuh/notes/digital-signal-processing-lab/design-of-fir-filters-of-%C2%A0low-pass-and-high-pass-filter-%C2%A0using-matlab-commands/374

https://youtu.be/_lwR1cvF4Io

APPENDIX(MATLAB CODE)

Implementation of filters in MATLAB:

```
myvoice=audiorecorder(8000,24,1);
disp('Start to speak')
recordblocking(myvoice,3)
disp('End of recording')
play(myvoice)
voice_vector=getaudiodata(myvoice);
figure(1)
plot(voice_vector)
title('Own voice in Time Domain');


%x = fft(voice_vector);
%figure(2)
%plot(x)
w=0:pi/200:pi;
[h w]=freqz(voice_vector,1,w);
figure(2)
plot(w,abs(h))
title('Own Voice in Frequency Domain')


with_noise1=awgn(voice_vector,99);% Noice
snr = 99
with_noise2=awgn(voice_vector,20);% Noise
snr = 20
figure(3)
subplot(211)
plot(with_noise1)
title('Signal With Gaussian Noise SNR 99')
subplot(212)
plot(with_noise2)
title('Signal With Gaussian Noise SNR 20')
```

LOW PASS FILTER:

```
F1=8000;
F2=800;
F3=1000;
r1=2;
r2=60;
p1=1-10.^(-r1/20);
s1=10.^(-r2/20);
FF=[F2 F3];
ma=[1 0];
```

```
v=[p1 s1];
[A21,wA21,bt,Yp]=kaiserord(FF,ma,v,F1);
lowpassfilter=fir1(A21,wA21,kaiser(A21+1,bt));
[h,w]=freqz(lowpassfilter,1);
figure(4);
subplot(311)
plot(w*7000*0.5/pi,abs(h));
ylim([0,2])
title('Design FIR Low Pass Filter','fontweight','bold');grid;
Signal_1=fftfilt(lowpassfilter,with_noise1);
sound(Signal_1,F1);
sound(Signal_1);
signal1=fft(Signal_1);
subplot(312)
plot(abs(signal1),'r');
title('Filtered Signal With Noise SNR 99');
Signal_2=fftfilt(lowpassfilter,with_noise2);
sound(Signal_2,F1);
sound(Signal_2);
signal2=fft(Signal_2);
subplot(313)
plot(abs(signal2),'r');
title('Filtered Signal With Noise SNR 20');
```

BAND PASS FILTER:

```
F1=8000;
F=[1000 1300 2210 2410]
ma=[0 1 0];
devs=[0.01 0.05 0.01];
[n,wA23,bt,Yp]=kaiserord(F,ma,devs,F1);
n=n+rem(n,2)
bandpassfilter=fir1(n,wA23,'bandpass',kaiser(n+1,bt),'noscale');
[h,w]=freqz(bandpassfilter,1,1024,F1);
figure(6)
subplot(311)
plot(w*10000*0.5/pi,abs(h));
title('DESIGN FIR BAND PASS FILTER','fontWeight','bold');
grid;
signal_5=fftfilt(bandpassfilter,with_noise1)
sound(signal_5,F1);
sound(signal_5);
signal_5=fft(signal_5)
subplot(312)
plot(abs(signal_5),'r');
title('FILTRED SIGNAL WITH NOISE SNR 99');
signal_6=fftfilt(bandpassfilter,with_noise2);
sound(signal_6);
subplot(313)
plot(abs(signal_6),'r');
title('Filtred Signal With NOISE SNR 20');
```

HIGH PASS FILTER

```
F1=8000;
F2=3000;
F3=2000;
r1=2;
r2=60;
p=1-10.^(-r1/20);
s=10.^(-r2/20);
FF=[F3 F2];
ma=[1 0];
v=[p s];
[A23,wA23,bt,Yp]=kaiserord(FF,ma,v,F1);
highpassfilter=fir1(A23,wA23,'high',kaiser(A23+1,bt));
[h,w]=freqz(highpassfilter,1);
figure(5);
subplot(311)
plot(w*10000*0.5/pi,abs(h));
title('Design FIR high Pass Filter','fontweight','bold');
grid;
Signal_3=fftfilt(highpassfilter,with_noise1);
sound(Signal_3,F1);
sound(Signal_3);
Signal3=fft(Signal_3);
subplot(312)
plot(abs(Signal3),'r');
title('Filtered Signal With Noise SNR 99');
Signal_4=fftfilt(highpassfilter,with_noise2);
sound(Signal_4,F1);
sound(Signal_4);
signal4=fft(Signal_4);
subplot(313)
plot(abs(signal4),'r');
title('Filtered Signal With Noise SNR 20');


subplot(3,1,1)
xlim([0 5000])
ylim([0.00 2.00])
```

MATLAB GUI CODE:

```
       AudFile=0;

    end

    methods (Access = public)


        function AudioPath = InputAudio(app)
            [filename ,pathname] = uigetfile( '*.wav','Select your Audio ');
     filewithpath=strcat(pathname,filename) ;
     AudioPath=filewithpath;
     app.AudPath=AudioPath;
     app.AudFile=filename;
        end


        function [T,Freq,PSD1,PSD2,yFilter] = AudioProcess(app)
         app.filter=app.ValueEditField.Value;
          if(app.AudPath==0)

          else
            [y,fs] = audioread(app.AudPath);

           info=audioinfo(app.AudPath);
           t = 0:seconds(1/fs):seconds(info.Duration);
           t = t(1:end-1);
           T=t;
N=length(t);
yin=fft(y,N);
PSD = yin.*conj(yin)/N;
freq=1/(0.001*N)*(0:N);
L=1:floor(N/2);
Freq=freq(L);
PSD1=PSD(L);
ind=PSD>app.filter;
PSD=PSD.*ind;
PSD2=PSD(L);


yin=ind.*yin;
yfilt=ifft(yin);
yFilter=yfilt;
end
end
end


% Callbacks that handle component events
methods (Access = private)
```

```
% Code that executes after component creation
function startupFcn(app)
app.Inputaudioaxe.Visible="off";
app.Switch.Value="Off";
app.MainFrame.Enable="off";
end


% Button pushed function: ProcessButton
function ProcessButtonPushed(app, event)
if(app.FilteringTypeDropDown.Value=="Noise Flitering")
app.MainFrame.Visible="off";
app.Noiseframe.Visible="on";
else
end
end


% Button pushed function: BackButton_2
function BackButton_2Pushed(app, event)
app.MainFrame.Visible="on";
app.Noiseframe.Visible="off";
app.ImageFrame.Visible="off";
end


% Callback function
function BackButtonPushed(app, event)
app.MainFrame.Visible="on";
app.Noiseframe.Visible="off";
end


% Value changed function: Switch
function SwitchValueChanged(app, event)
value = app.Switch.Value;
if(value=="Off")
app.MainFrame.Enable="off";
elseif(value=="On")
app.MainFrame.Enable="on";
end
end


% Button pushed function: OpenFileButton
function OpenFileButtonPushed(app, event)
if(app.FilteringTypeDropDown.Value=="Noise Flitering")
path =app.InputAudio;
[y,fs]=audioread(path);
info = audioinfo(path);
t = 0:seconds(1/fs):seconds(info.Duration);
t = t(1:end-1);
plot(t,y,"Parent",app.Inputaudioaxe)
app.Inputaudioaxe.XLabel.String= "Time";
```

```
app.Inputaudioaxe.YLabel.String="Audio Signal";
app.Inputaudioaxe.Title.String="Input Audio Signal";
app.Inputaudioaxe.Visible="on";
sound(y,fs);
else
end
end


% Button pushed function: AnalyseNoiseButton
function AnalyseNoiseButtonPushed(app, event)
if(app.AudPath==0)
app.MainFrame.Visible="on";
app.Noiseframe.Visible="off";
app.PathTextArea.Value="Audio isnt Selected or Path doesnt Exist";
else
[T,Freq,PSD1,PSD2,yFilter] = AudioProcess(app);
pause(1);
plot(Freq,PSD1,"Parent",app.Fre1);
app.Fre1.XLabel.String = "Frequencey";
app.Fre1.YLabel.String = "Frequency Domain";
app.Fre1.Title.String = "Fourier Transformation";
end
end


% Button pushed function: FilterButton
function FilterButtonPushed(app, event)
if(app.AudPath==0)
app.MainFrame.Visible="on";
app.Noiseframe.Visible="off";
app.PathTextArea.Value="Audio isnt Selected or Path doesnt Exist";
else
[T,Freq,PSD1,PSD2,yFilter] = AudioProcess(app);
pause(1);
plot(Freq,PSD2,"Parent",app.Fre2);
app.Fre2.XLabel.String = "Frequencey";
app.Fre2.YLabel.String = "Frequency Domain";
app.Fre2.Title.String = "Fourier Transformation";
pause(1);
plot(T,yFilter,"Parent",app.OutputA);
app.OutputA.XLabel.String = "Time";
app.OutputA.YLabel.String = "Audio Signal";
app.OutputA.Title.String = "Filtered Signal";
end
end


% Button pushed function: SaveButton
function SaveButtonPushed(app, event)
if(app.AudPath==0)
app.MainFrame.Visible="on";
app.Noiseframe.Visible="off";
app.PathTextArea.Value="Audio isnt Selected or Path doesnt Exist";
```

```
else
[y,fs]=audioread(app.AudPath);
app.AudFile=strrep(app.AudFile,".","Filterd.");
audiowrite(app.AudFile,y,fs);
sound(y,fs);
end
end


% Button pushed function: StartRecordingButton
function StartRecordingButtonPushed(app, event)
while(true)
if(app.StartRecordingButton.Text=="Start Recording")
global recObj
recObj = audiorecorder ; %create object
record(recObj); %start Recording
app.PathTextArea.Value="Speak now .... ";
app.StartRecordingButton.Text="Stop Recording";
break;
end
if(app.StartRecordingButton.Text=="Stop Recording")
global recObj
stop(recObj) % Stop
app.PathTextArea.Value="Recording Stopped .... ";
app.StartRecordingButton.Text = "Save Audio";
break;
end
if(app.StartRecordingButton.Text=="Save Audio")
global recObj
y=getaudiodata(recObj);
Fs=8000;
prompt={'Save As : '};
dlg_title=' Audio File';
num_lines = [1 50] ;
def={''};
out=inputdlg(prompt,dlg_title,num_lines,def);
filename=[out{1} '.wav'];
audiowrite(filename , y,Fs);
app.PathTextArea.Value="Recording Saved .... ";
app.StartRecordingButton.Text="Start Recording";
break;
end
end
end


% Button pushed function: ResetButton
function ResetButtonPushed(app, event)
cla(app.Inputaudioaxe);
cla(app.Fre1);
cla(app.Fre2);
cla(app.OutputA);
app.PathTextArea.Value="";
app.ValueEditField.Value=0;
```

```
app.Inputaudioaxe.Visible="off";
app.fft.Visible="off";
app.fftshift.Visible="off";
app.Path = 0;
app.AudPath = 0;
app. filter=0;
app. AudFile=0;
end


% Callback function
function ResetButton_2Pushed(app, event)
app.fft.Visible="off";
app.fftshift.Visible="off";
app.outputimg.Visible="off";
app.DensityEditField.Value=20;
end


% Button pushed function: ResetButton_3
function ResetButton_3Pushed(app, event)
cla(app.Fre1);
cla(app.Fre2);
cla(app.OutputA);
app.ValueEditField.Value=0;
end
end


% Component initialization
methods (Access = private)


% Create UIFigure and components
function createComponents(app)


% Create UIFigure and hide until all components are created
app.UIFigure = uifigure('Visible', 'off');
app.UIFigure.Position = [100 100 603 484];
app.UIFigure.Name = 'MATLAB App';


% Create Switch
app.Switch = uiswitch(app.UIFigure, 'slider');
app.Switch.ValueChangedFcn = createCallbackFcn(app, @SwitchValueChanged,
true);
app.Switch.Position = [38 9 45 20];


% Create Noiseframe
app.Noiseframe = uipanel(app.UIFigure);
app.Noiseframe.Position = [10 36 581 437];
```

```
% Create OutputA
app.OutputA = uiaxes(app.Noiseframe);
title(app.OutputA, 'Title')
xlabel(app.OutputA, 'X')
ylabel(app.OutputA, 'Y')
zlabel(app.OutputA, 'Z')
app.OutputA.Position = [163 286 402 130];


% Create Fre1
app.Fre1 = uiaxes(app.Noiseframe);
title(app.Fre1, 'Title')
xlabel(app.Fre1, 'X')
ylabel(app.Fre1, 'Y')
zlabel(app.Fre1, 'Z')
app.Fre1.Position = [28 15 261 217];


% Create Fre2
app.Fre2 = uiaxes(app.Noiseframe);
title(app.Fre2, 'Title')
xlabel(app.Fre2, 'X')
ylabel(app.Fre2, 'Y')
zlabel(app.Fre2, 'Z')
app.Fre2.Position = [288 15 277 216];


% Create BackButton_2
app.BackButton_2 = uibutton(app.Noiseframe, 'push');
app.BackButton_2.ButtonPushedFcn = createCallbackFcn(app, @BackButton_2Pushed,
true);
app.BackButton_2.Position = [16 384 60 32];
app.BackButton_2.Text = 'Back';


% Create AnalyseNoiseButton
app.AnalyseNoiseButton = uibutton(app.Noiseframe, 'push');
app.AnalyseNoiseButton.ButtonPushedFcn = createCallbackFcn(app,
@AnalyseNoiseButtonPushed, true);
app.AnalyseNoiseButton.Position = [17 243 104 35];
app.AnalyseNoiseButton.Text = 'Analyse';


% Create ValueEditFieldLabel
app.ValueEditFieldLabel = uilabel(app.Noiseframe);
app.ValueEditFieldLabel.HorizontalAlignment = 'right';
app.ValueEditFieldLabel.Position = [137 250 55 22];
app.ValueEditFieldLabel.Text = 'Value';


% Create ValueEditField
app.ValueEditField = uieditfield(app.Noiseframe, 'numeric');
app.ValueEditField.Position = [207 246 82 29];
```

```
% Create FilterButton
app.FilterButton = uibutton(app.Noiseframe, 'push');
app.FilterButton.ButtonPushedFcn = createCallbackFcn(app, @FilterButtonPushed,
true);
app.FilterButton.Position = [312 240 106 35];
app.FilterButton.Text = 'Filter';


% Create SaveButton
app.SaveButton = uibutton(app.Noiseframe, 'push');
app.SaveButton.ButtonPushedFcn = createCallbackFcn(app, @SaveButtonPushed,
true);
app.SaveButton.Position = [18 295 90 24];
app.SaveButton.Text = 'Save';


% Create ResetButton_3
app.ResetButton_3 = uibutton(app.Noiseframe, 'push');
app.ResetButton_3.ButtonPushedFcn = createCallbackFcn(app,
@ResetButton_3Pushed, true);
app.ResetButton_3.Position = [445 248 99 24];
app.ResetButton_3.Text = 'Reset';


% Create MainFrame
app.MainFrame = uipanel(app.UIFigure);
app.MainFrame.Position = [10 36 581 437];


% Create Inputaudioaxe
app.Inputaudioaxe = uiaxes(app.MainFrame);
title(app.Inputaudioaxe, 'Audio Signal')
zlabel(app.Inputaudioaxe, 'Z')
app.Inputaudioaxe.XMinorGrid = 'on';
app.Inputaudioaxe.YGrid = 'on';
app.Inputaudioaxe.Position = [52 15 478 147];


% Create TextArea
app.TextArea = uitextarea(app.MainFrame);
app.TextArea.Editable = 'off';
app.TextArea.HorizontalAlignment = 'center';
app.TextArea.FontSize = 14;
app.TextArea.Position = [160 391 236 22];
app.TextArea.Value = {'Digital Signal Processing'};


% Create FilteringTypeDropDownLabel
app.FilteringTypeDropDownLabel = uilabel(app.MainFrame);
app.FilteringTypeDropDownLabel.HorizontalAlignment = 'right';
app.FilteringTypeDropDownLabel.Position = [33 293 77 22];
app.FilteringTypeDropDownLabel.Text = 'Filtering Type';
```

```
% Create FilteringTypeDropDown
app.FilteringTypeDropDown = uidropdown(app.MainFrame);
app.FilteringTypeDropDown.Items = {'Image Processing', 'Noise Flitering'};
app.FilteringTypeDropDown.ItemsData = {'Image Processing', 'Noise Flitering'};
app.FilteringTypeDropDown.Position = [125 289 150 30];
app.FilteringTypeDropDown.Value = 'Image Processing';


% Create OpenFileButton
app.OpenFileButton = uibutton(app.MainFrame, 'push');
app.OpenFileButton.ButtonPushedFcn = createCallbackFcn(app,
@OpenFileButtonPushed, true);
app.OpenFileButton.Position = [34 231 109 24];
app.OpenFileButton.Text = 'Open File';


% Create ProcessButton
app.ProcessButton = uibutton(app.MainFrame, 'push');
app.ProcessButton.ButtonPushedFcn = createCallbackFcn(app,
@ProcessButtonPushed, true);
app.ProcessButton.Position = [157 169 116 33];
app.ProcessButton.Text = 'Process';


% Create StartRecordingButton
app.StartRecordingButton = uibutton(app.MainFrame, 'push');
app.StartRecordingButton.ButtonPushedFcn = createCallbackFcn(app,
@StartRecordingButtonPushed, true);
app.StartRecordingButton.Position = [160 230 110 25];
app.StartRecordingButton.Text = 'Start Recording';


% Create PathTextArea
app.PathTextArea = uitextarea(app.MainFrame);
app.PathTextArea.Position = [330 254 226 47];


% Create ResetButton
app.ResetButton = uibutton(app.MainFrame, 'push');
app.ResetButton.ButtonPushedFcn = createCallbackFcn(app, @ResetButtonPushed,
true);
app.ResetButton.Position = [37 172 89 27];
app.ResetButton.Text = 'Reset';


% Show the figure after all components are created
app.UIFigure.Visible = 'on';
end
end


% App creation and deletion
methods (Access = public)
```

```
% Construct app
function app = ICS


% Create UIFigure and components
createComponents(app)


% Register the app with App Designer
registerApp(app, app.UIFigure)


% Execute the startup function
runStartupFcn(app, @startupFcn)


if nargout == 0
clear app
end
end


% Code that executes before app deletion
function delete(app)


% Delete UIFigure when app is deleted
delete(app.UIFigure)
end
end
end
```

END

.