Project Report

Digital Filter Design

Objective

The objective of this project is to design a filter to remove noise from voice signals. To remove the noise from the audio signal we need to analyze the characteristics of the Voice signal, noise, and the voice with the noise signal. To remove the noise, we need to select an appropriate filter based on the analysis, taking into account lower implementation cost, faster response time, and an implementation structure with the fixed precision algorithm.

Observation

We have collected two voice samples from the same person for our analysis.

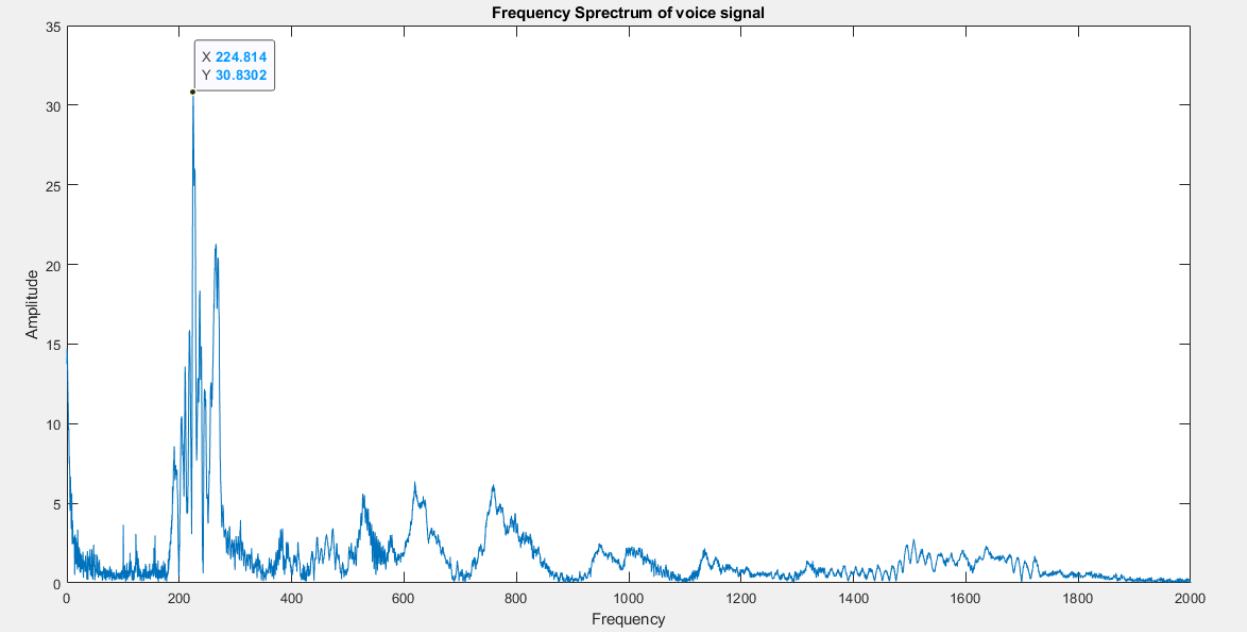

Figure 1: Frequency spectrum of Voice signal 1.

From Figure 1, we can observe that most of the frequency components of the voice signal are available at above 200 Hz.

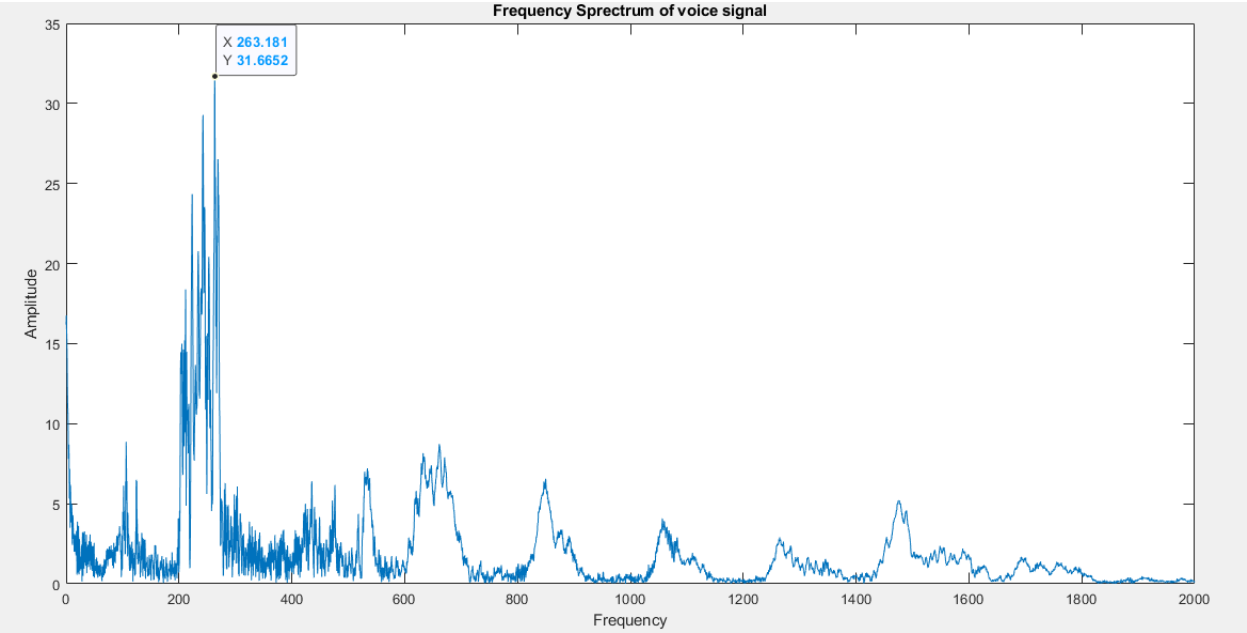

Figure 2: Frequency spectrum of Voice signal 2.

From Figure 2, we can observe that most of the frequency components of this voice signal are available at above 200 Hz.

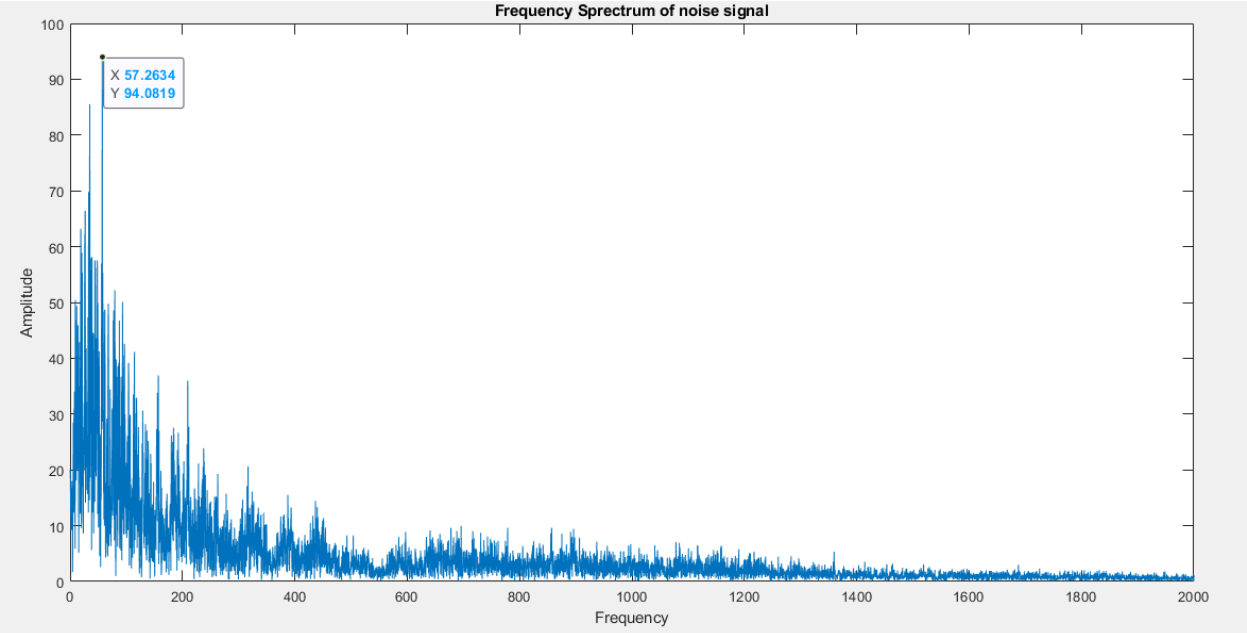

Figure 3: Frequency spectrum of the Noise signal.

From Figure 3, we can observe that the frequency components of the noise signal are available from 0 to 1200 Hz, but higher amplitudes are found underneath 200 Hz.

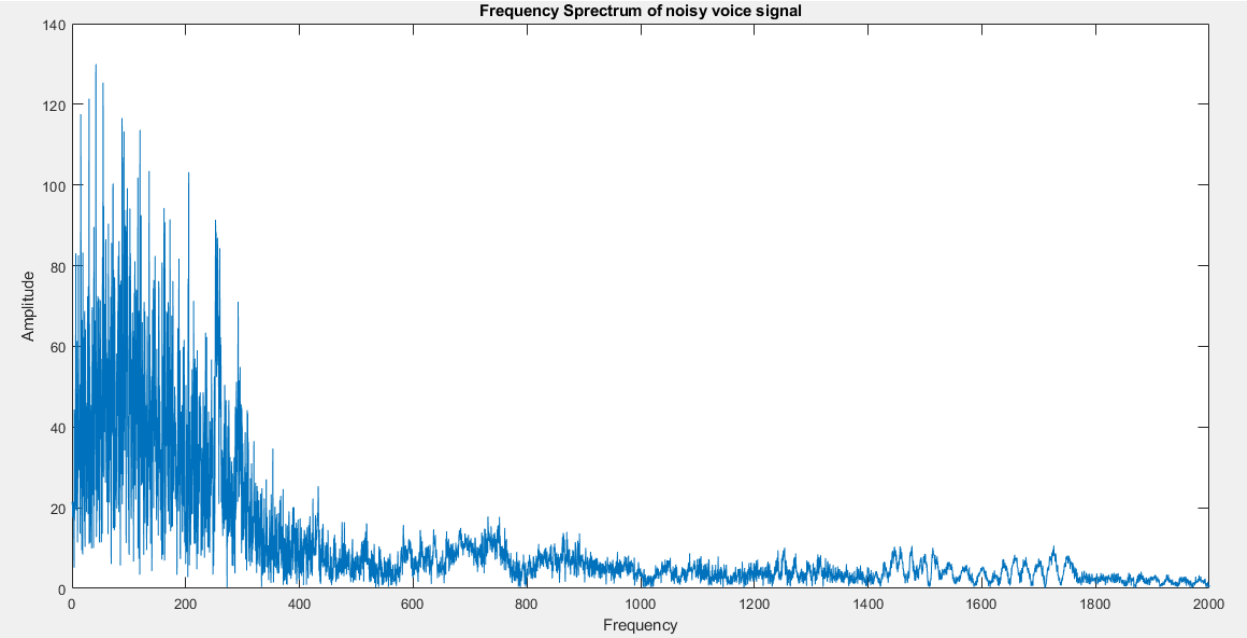

Figure 4: Frequency spectrum of noisy voice signal 1.

From Figure 4, we can observe that the frequency components of the noisy voice signal are available from 0 to 2000 Hz, but higher amplitudes are found underneath 200 Hz.

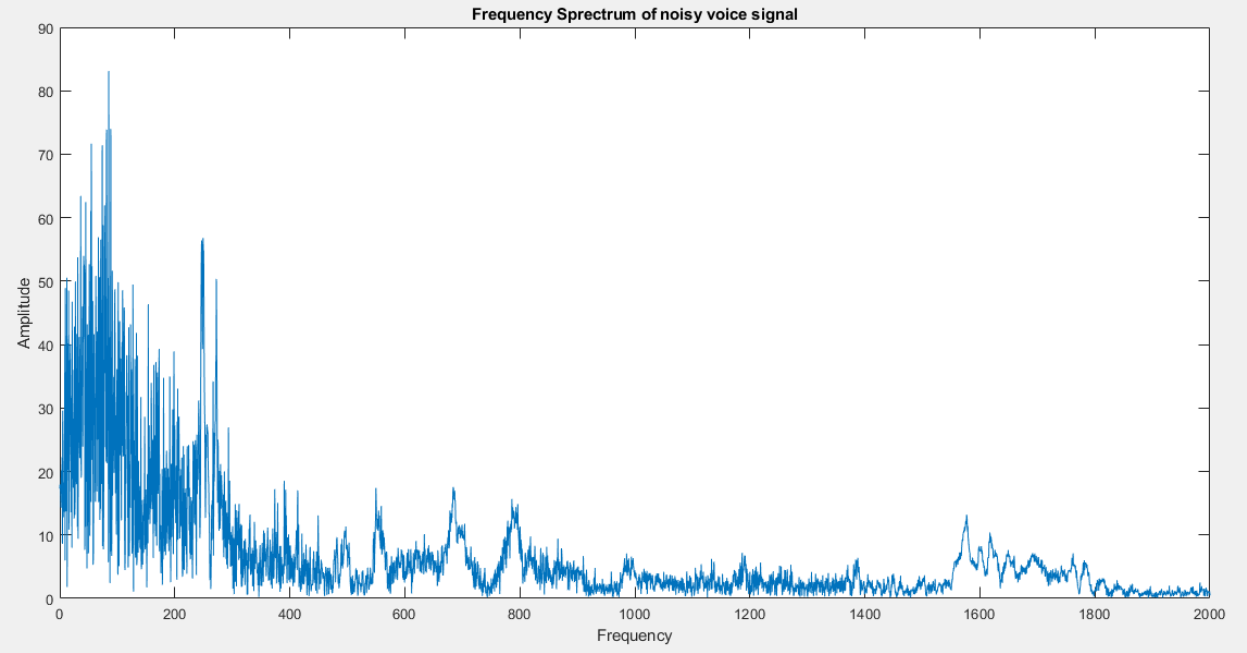

Figure 5: Frequency spectrum of noisy voice signal 2.

From Figure 4, we can observe that the frequency components of the noisy voice signal are available from 0 to 2000 Hz, but higher amplitudes are found underneath 200 Hz.

By observing the signals, we can say that to minimize the effect of noise from the noisy voice signals, we need to filter out the higher magnitude frequency components of the noise signal which is underneath 200 Hz.

Filter Selection

There are two types of digital filters available, which are: Finite Impulse Response (FIR) and Infinite Impulse Response (IIR). There are many differences between FIR filter and IIR filter,

1. FIR filter generates an output using the present and past input values. An IIR filter, on the other hand, generates the current output by combining present and past input values as well as the past output values. As the present output has no relationship with the past output, FIR filters are more stable. IIR filters, on the other hand, are less stable because they use past output samples as well.

2. FIR filters do not use a feedback method, while IIR filters make use of a feedback loop to connect past output with present input. Due to the absence of the feedback loop, the implementation of FIR filters in a system is quite easy in comparison to IIR filters.

3. FIR filters support linear phase filtering, however, IIR filters do not support linear phase filtering.

4. The transfer function of the system having FIR filter contains only zeros. While in the case of IIR filters the transfer, the function contains both poles and zeros.

5. FIR filters show non-recursive behavior. But IIR filters show recursive behavior.

6. The delay in responding in the case of FIR filters is more than that of the IIR filter.

Linear-phase describes the response of a filter. When a signal goes through a filter, it experiences a time delay or phase shift. In a perfect filter, all frequencies should experience the same time delay which preserves the wave shape as much as possible. All filters have a phase shift. To preserve the waveshape, we need to design a filter where the phase shift is linear concerning frequency. A system that accurately preserves such relationships is said to be phase linear. For voice signals, we need linear phase filtering. So that we must choose the FIR filter to avoid phase shift.
Though the IIR filter needs less memory, it does not support linear phase filtering. So that we can not choose IIR filter in our case.

From the observations, we have seen that the voice signals frequency is more than 200 Hz. But in the case of the noise signal, the higher magnitudes of the signals are underneath 200 Hz. To clear the noise, we need to use a high pass filter that would pass the signals over 200 Hz. Using this method, we cannot completely cancel out the noise from the noisy voice signal. But it will minimize the effect of noise.

Calculations

We have selected to use a highpass FIR filter. In FIR filters the memory requirement is high. It is depended on the sampling frequency of the signal, transition bandwidth, and passband and stopband edge ripples. We have reduced the sampling frequency of our audio signal to 8000 Hz so that our filter can have less memory. We have selected the following parameters to design our filter,

Sampling Frequency, Fs = 8000 Hz
Transition bandwidth, ΔF = 50 Hz
Cut off Frequency, Fc = 250 Hz [as per the observations]

So,
Passband edge frequency, Fp = Fc + ΔF /2
= 275 Hz

Stopband edge frequency, Fp = Fc - ΔF /2
= 225 Hz

Let,
Passband ripple, δp = 0.001
Stopband ripple, δs = 0.001

stopband attenuation, A = −20log(δs) = −20log(0.001) = 60 dB.

According to the stopband attenuation, we can use Blackman and Kaiser windows for our desired filter.

Blackman Window
Transition width, Δf = 50 Hz
Normalized Δf = 50/8000 = 0.00625 [Fs = 8000 Hz]

For Blackman, $\Delta f = 5.5/M+1$
$\Rightarrow 0.0063 = 5.5/M+1$
$\Rightarrow M+1 = 5.5/0.00625$
$\Rightarrow M = 879$

Using the Blackman window, the filter needs 879 memories.

Kaiser Window

Passband ripple, δp = 0.001
Stopband ripple, δs = 0.001
The Kaiser filter must be designed to meet the smaller of the two ripple constraints:
Both the ripple parameters are the same.
So,δ = 0.001

Now, A = −20log(δ)
= −20log(0.001)
= 60 dB

For A>50
Kaiser Parameter,$\beta = 0.1102(A - 8.7)$
= 5.65326

Normaized transition bandwidth, Δf = 0.00625

Now, Δw= Δf × 2π = 0.0393

Window length, $M + 1 \geq [1 + \frac{A-8}{2.285 \times \Delta\omega}]$

$$M = 579.5 \approx 580$$

We can see that for Blackman window, we need more memory than the Kaiser window. So, we choose Kaiser window method to design our highpass filter.

Filter Design

We Have designed the filter using Filter designer application from Matlab.

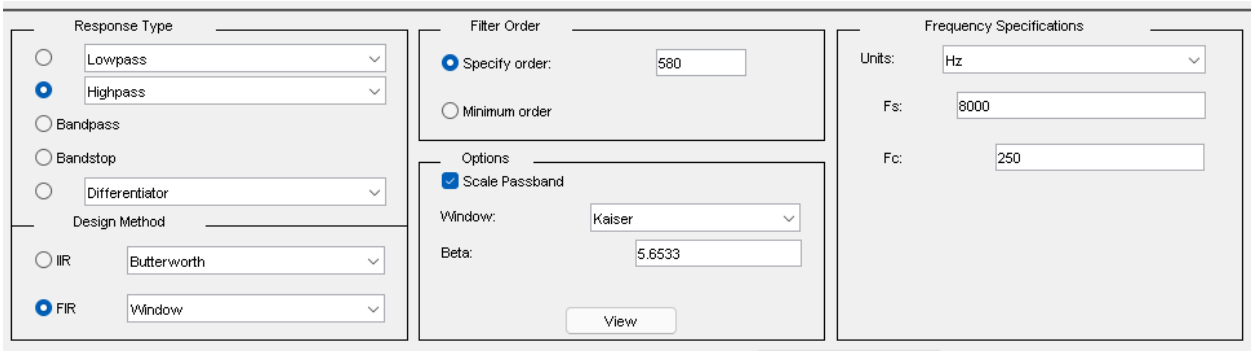

Figure 6: Filter parameters.

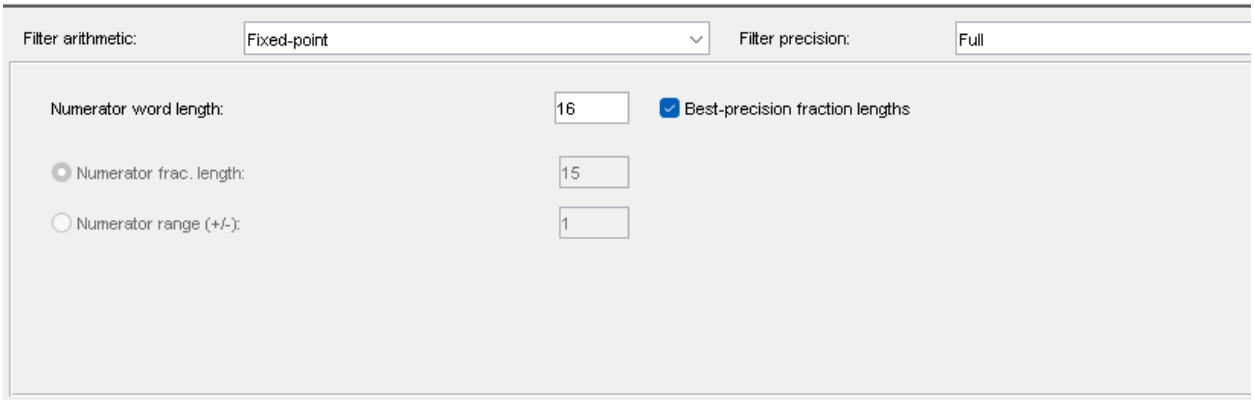

Figure 7: 16 bit fixed point precision.

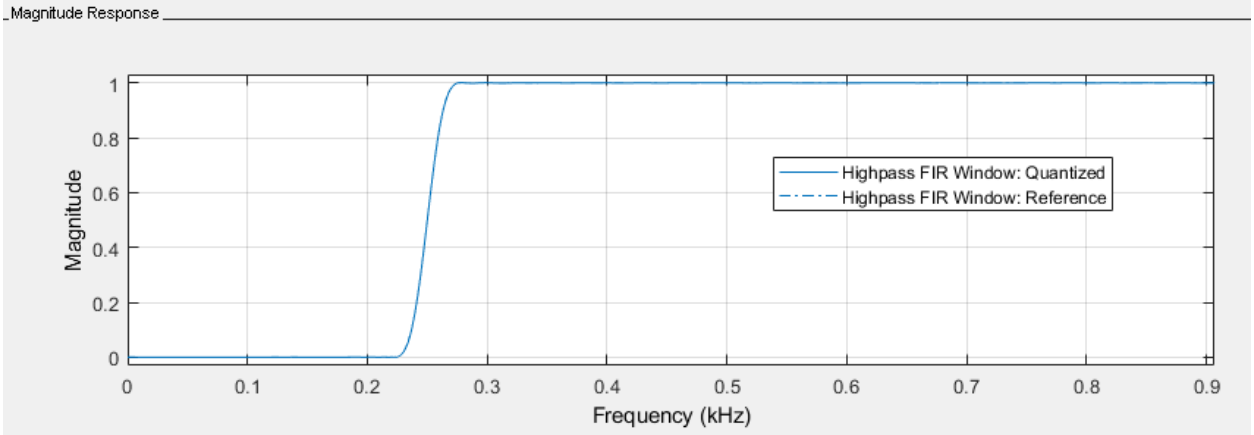

Figure 8: Magnitude Response of the filter.

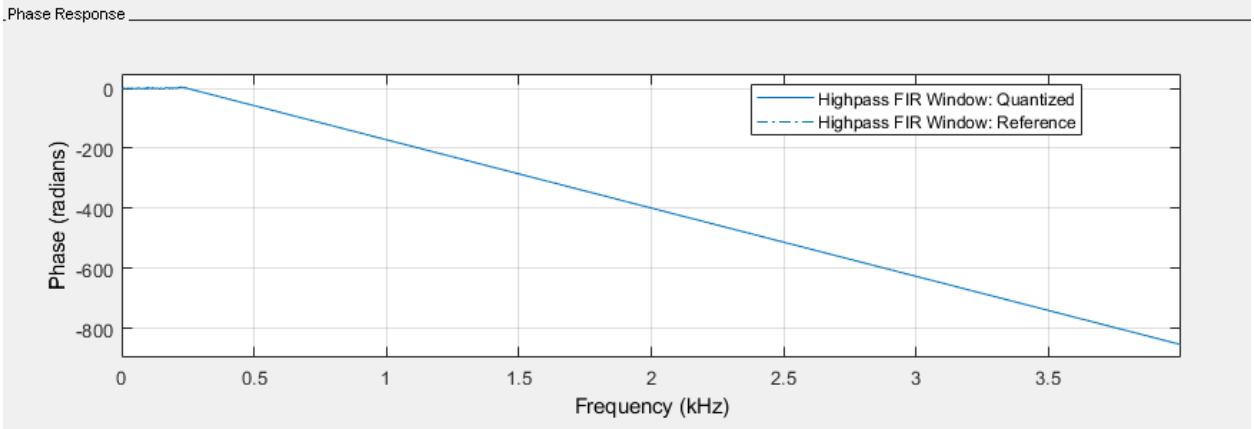

Figure 9: Phase Response of the filter.

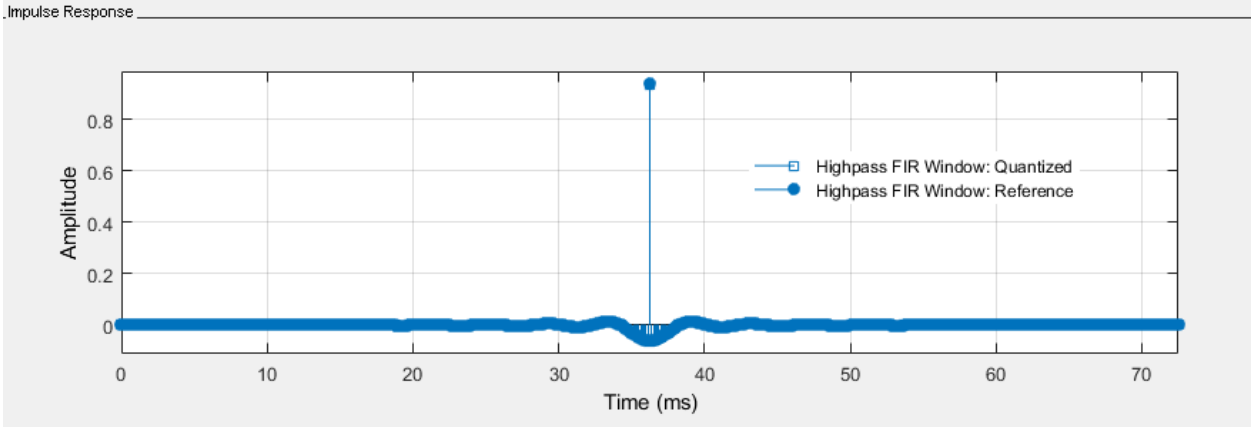

Figure 10: Impulse Response of the filter.

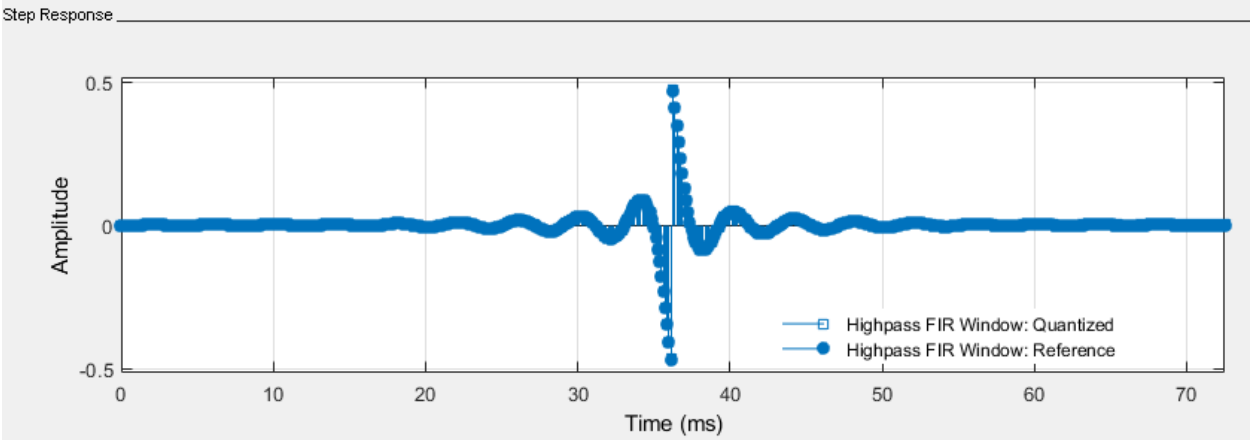

Figure 11: Step Response of the filter.

```
Discrete-Time FIR Filter (real)
-------------------------------
Filter Structure  : Direct-Form FIR
Filter Length     : 581
Stable            : Yes
Linear Phase      : Yes (Type 1)
Arithmetic        : fixed
Numerator         : s16,15 -> [-1 1)
Input             : s16,15 -> [-1 1)
Filter Internals  : Full Precision
  Output          : s33,30 -> [-4 4)   (auto determined)
  Product         : s31,30 -> [-1 1)   (auto determined)
  Accumulator     : s33,30 -> [-4 4)   (auto determined)
  Round Mode      : No rounding
  Overflow Mode   : No overflow
Implementation Cost
Number of Multipliers            : 581
Number of Adders                 : 580
Number of States                 : 580
Multiplications per Input Sample : 581
Additions per Input Sample       : 580
```

Figure 12: Filter Information

Performance Analysis

To analyze the performance of our designed filter, we have generated the matlab function for the filter from filter design application. Then we have analyzed it in matlab by passing 2 noisy voice signals through it and observed the output.

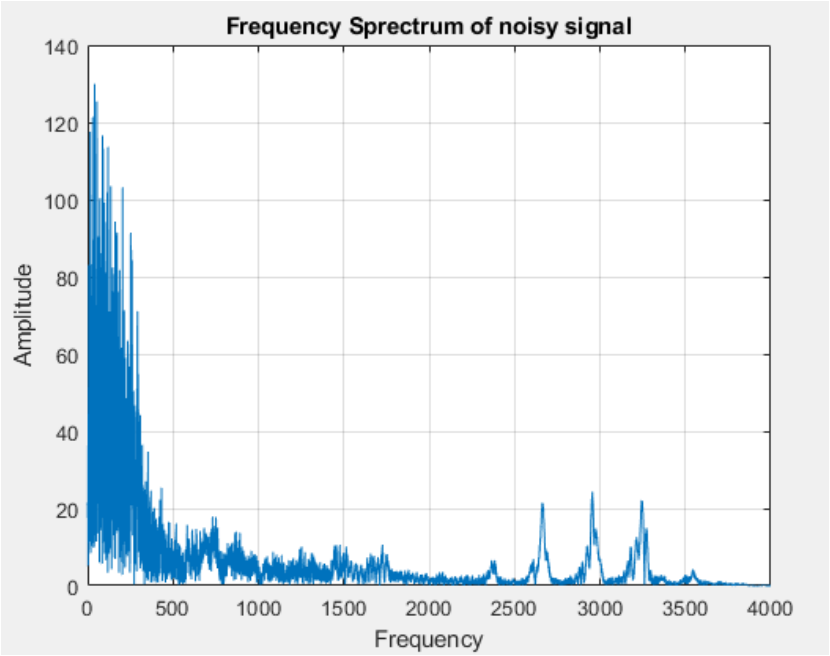

Figure 13: Frequency spectrum of noisy voice signal 1.

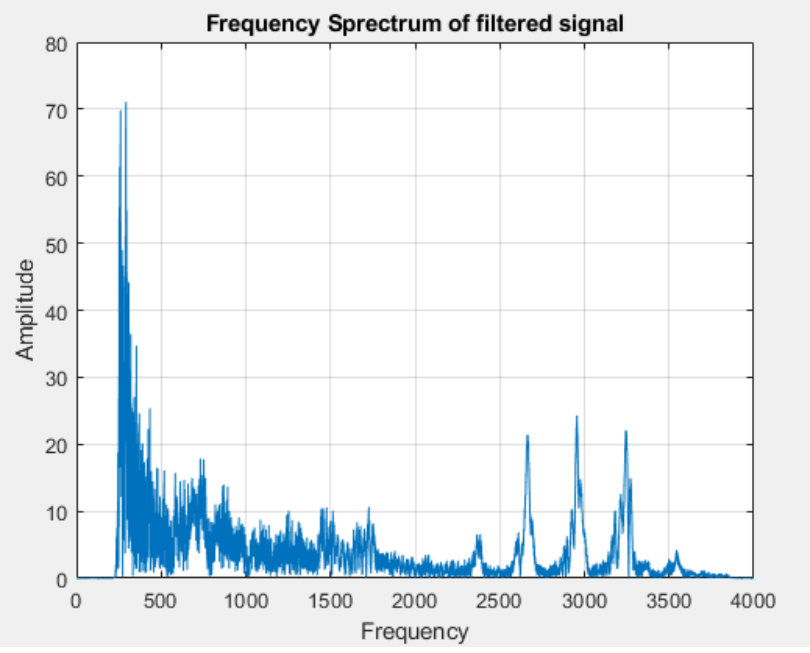

Figure 14: Frequency spectrum of filtered voice signal 1.

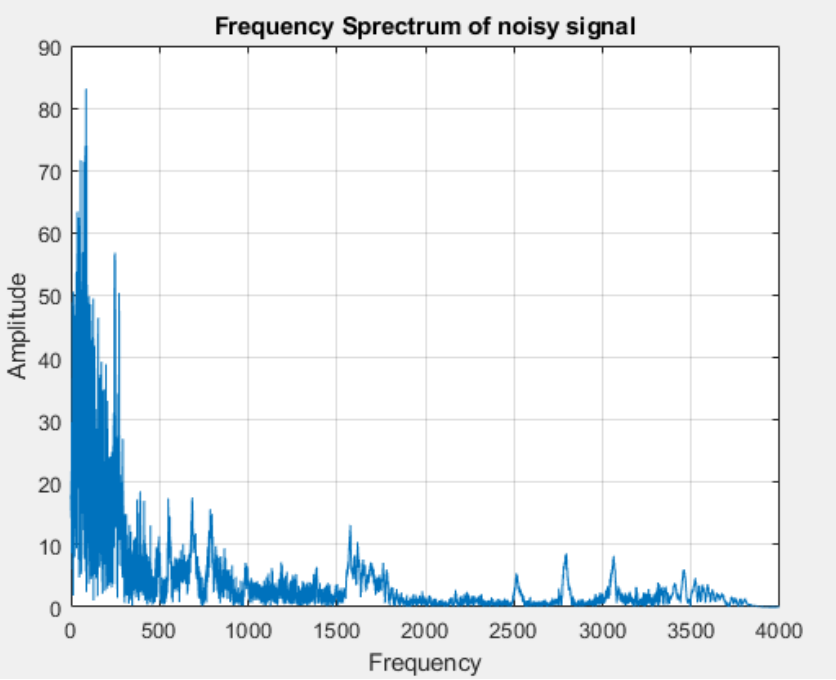

Figure 15: Frequency spectrum of noisy voice signal 2.

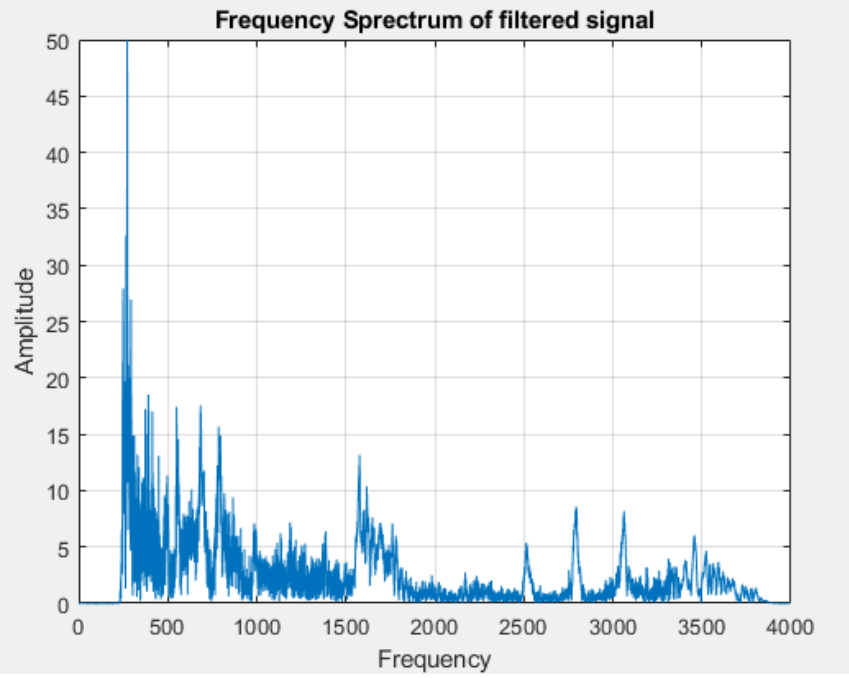

Figure 16: Frequency spectrum of filtered voice signal 2.

From the above figures we can observe that, after filtering the noisy signals, the output has canceled out the signals from 0 to 250 Hz approximately. We can say that the noise signal has been suppressed but it could not be removed completely.

To export the filtered audio file, we have used Simulink.

Figure 17: Simulink model to export audio.

Discussion

We have successfully designed an FIR highpass filter as per our design. We were able to reduce the noise from our recorded noisy signal by using this filter. We could not completely eliminate the noise with our designed filter. But it can be observed that the filtered signal has less noise in comparison to the recorded noisy signal. We have tried to meet the requirements without compromising the filter design. For the minimum cost, we need to reduce the number of multiplies or adders. To reduce the memory requirement, we have reduced the sampling frequency of our signal, we did not increase the ripples too much to reduce the memory. We took a decent transition bandwidth as well. We have used the Kaiser window method over Blackman because it has less memory requirement. The noise can be suppressed more if the cut-off frequency is taken a little far. But in this case, some of the original audio frequencies may get lost. The ripples can be increased but it will change the shape of the filter.

While exporting the audio we faced some issues, we could not export the audio either in matlab or Simulink. Then we have tried different approaches and changed the fixed point precision to double-precision floating point. After that, we could export the audio.

Conclusion

In this project, we have learned to design digital filters. We have learned how to change filter parameters to get desired output. We also have learned different MATLAB codes. We have learned to analyze the audio signals and the effect of noise and distorted audio signals. We have learned to use filter designing application. Finally, we have designed a filter considering different specification and constraints.

Appendix

Signal Characteristics

```
clc
close all
clear all

[y fs] = audioread('Audio signal location\audio.mp3');
L=length(y);
yl=1:1:L;
magnitude=y(yl,1);
ys=abs(fft(magnitude,L));
a=ys(1:L/2);
F=((0:1/L:1-1/L)*fs).';
f=F(1:L/2);


plot(f,a)
 xlim([0,2000]);
title ('Frequency Sprectrum of signal');
xlabel ('Frequency');
ylabel ('Amplitude');

%frequency at which the amplitude is maximum
A=a(2:1:L/2);
amax=max(A);
B=find(a==amax(1));
frequency=f(B(1))
```

Filter Function

```
function Hd = Highpass
%HIGHPASS Returns a discrete-time filter object.

% MATLAB Code
% Generated by MATLAB(R) 9.10 and DSP System Toolbox 9.12.
% Generated on: 14-Jan-2022 22:20:55

% FIR Window Highpass filter designed using the FIR1 function.

% All frequency values are in Hz.
Fs = 8000;  % Sampling Frequency

N    = 580;       % Order
Fc   = 250;       % Cutoff Frequency
flag = 'scale';  % Sampling Flag
Beta = 5.6533;   % Window Parameter

% Create the window vector for the design algorithm.
win = kaiser(N+1, Beta);

% Calculate the coefficients using the FIR1 function.
b  = fir1(N, Fc/(Fs/2), 'high', win, flag);
Hd = dfilt.dffir(b);
% Set the arithmetic property.
set(Hd, 'Arithmetic', 'fixed', ...
    'CoeffWordLength', 16, ...
    'CoeffAutoScale', true, ...
    'Signed',         true, ...
    'InputWordLength', 16, ...
    'inputFracLength', 15, ...
    'FilterInternals',  'FullPrecision');
denormalize(Hd);


% [EOF]
```

Performance check

```
clc
clear
close all

%reading recorded audio file
[audio,fs] = audioread('F Audio signal location\audio.mp3');
y=audio(:,1);
y1 = audioplayer(y, fs);
% play(y1)
L1= length(y);
m = abs(fft(y));
m = m(1:L1/2);
f = ((0:1/L1:1-1/L1)*fs)';
f=f(1:L1/2);
figure(1)
plot(f,m)
title ('Frequency Sprectrum of noisy signal');
xlabel ('Frequency');
ylabel ('Amplitude');
grid on
%filtering the audio
fvtool(Highpass); %loading filter info from filter designing app
f_audio= filter(Highpass ,y); %filtering the recorded audio signal


hplayer = audioplayer(f_audio, fs);
% play(hplayer)

L2= length(f_audio);
magnitude = abs(fft(f_audio));
magnitude = magnitude(1:L2/2);
freq = ((0:1/L2:1-1/L2)*fs)';
freq=freq(1:L2/2);
figure(2)
plot(freq,magnitude)
title ('Frequency Sprectrum of filtered signal');
xlabel ('Frequency');
ylabel ('Amplitude');
grid on
```