**Course Project Report: CS-341**

# Contribution of the Group Members:

1. 2018283: Muhammad Hassan Mirza

**Contribution:** I was responsible for task 1. So i designed a number which is complex using frequency of 44100 by adding real numbers and imaginary numbers. To equalize the size of lines we use zero function in Matlab. So complex numbers are the addition of real numbers and imaginary numbers. And if there is no real number then we use 0 instead of any other real number and add it with imaginary number to make complex numbers

1. 2018195: Mian Maziyar Rehan

**Contribution:** I was responsible for doing task 2. In this task, I mapped frequency ranging from [0, 2𝜋] rad/sample to range [−𝜋, 𝜋] rad/sample by using fftshift (built-in command. This was done by utilizing periodicity property of discrete time signals. After that the resulted frequency response was visualized as magnitude and phase response range which was calculated using powershift and phase functions. In the end noise was identified which was at the start of spectrum (from the figure).

1. 2018202: Muhammad Khizer Adnan

**Contribution:** I was responsible for doing task 3 & 4. We designed filter to filter out high frequency tone. Me and one of my group mate Mateen (2018300) designed filters using the FDA tool in MATLAB. Frequency was selected by hit and try method which came out to be 3500Hz. We filtered it using low pass filter, which we made as mentioned from FDA tool in MATLAB.

1. 2018300: Muhammad Mateen Qazi

**Contribution:** I was responsible for doing task 4 & 3. We designed the filters using FDA tool, and used the standardized 0.45 value for filter. After saving the filtered results, its phase was calculated and compared with sin graphs to highlight the similarities. At the end I used the SNR built in function in MATLAB to find the signal to noise ratio of respective signal (They are displayed in workspace).

# Problem Statement:

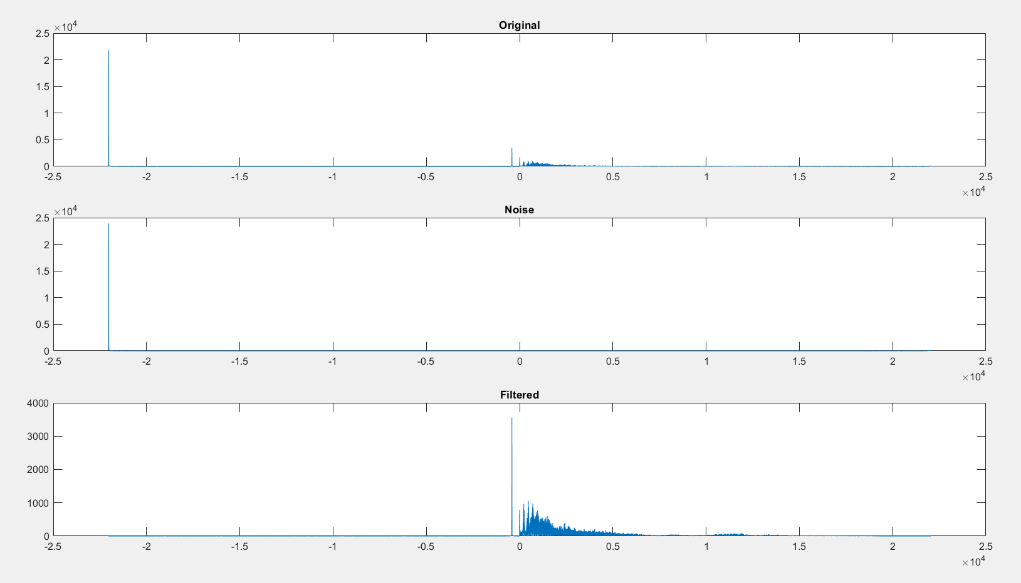
A frequency response is given which is a signal consisting of different tones and noise. From this frequency response we have to separate the noise and each tone.

# Proposed Solution:

The frequency response can be converted to time domain and filters can be applied to this signal to separate out the speech, noise and different tones

# Filter Designing Steps:

1. The magnitude spectrum of frequency response was studied. It showed peaks for noise and where tones at those frequencies. So filters according to those frequencies were made.



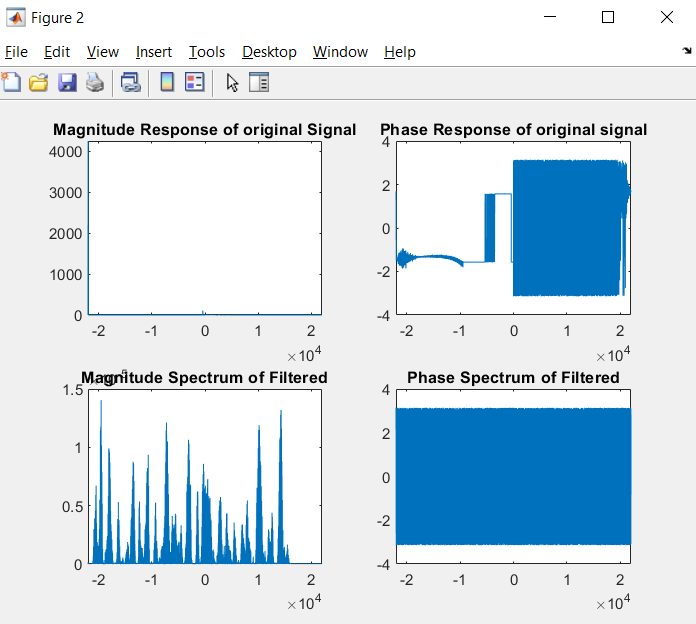
From this image, we can see that noise is at the start of spectrum and main signal is in the middle, So a low pass filter was used to get noise and a high pass filter with same cutoff (3500Hz) used to get main signal

1. To filter tones from the speech (which is the main signal), its spectrum is studied and tones are recognized. Then IIR low pass and high pass filters were made to separate the 2 tones
2. Table for filter parameters

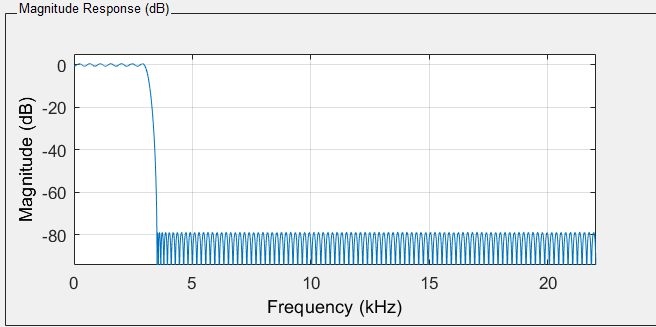
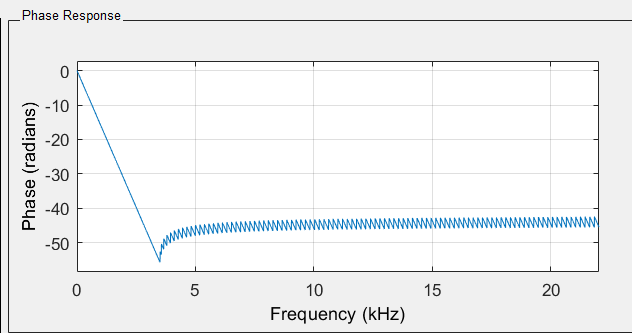
|  |  |  |
| --- | --- | --- |
| Filter Type | Purpose | Cutoff |
| Low pass FIR | To get noise | 3500 Hz |
| High pass FIR | To get speech | 3500 Hz |
| Low pass IIR | To get tone 1 | Normalized: 0.45 |
| High pass IIR | To get tone 2 | Normalized: 0.45 |

1. Magnitude and phase responses

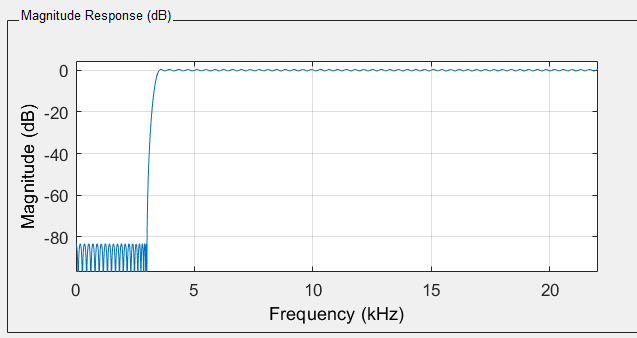
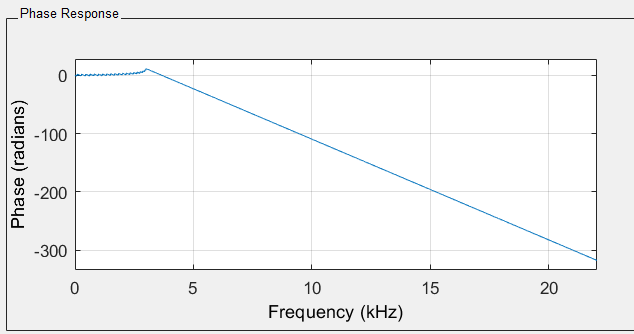
Filtered and Unfiltered:



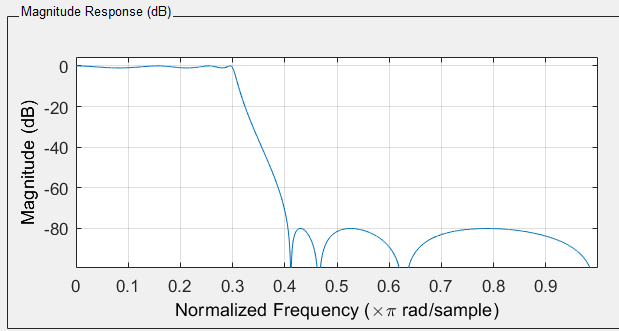
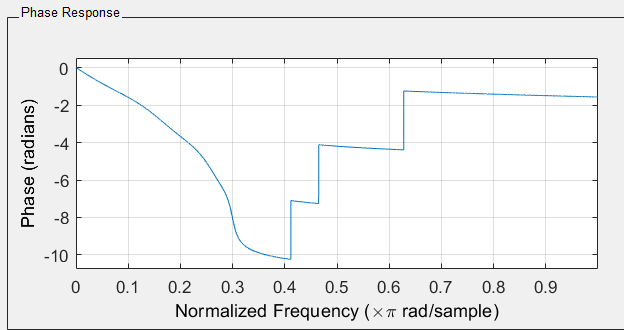
Low Pass FIR:

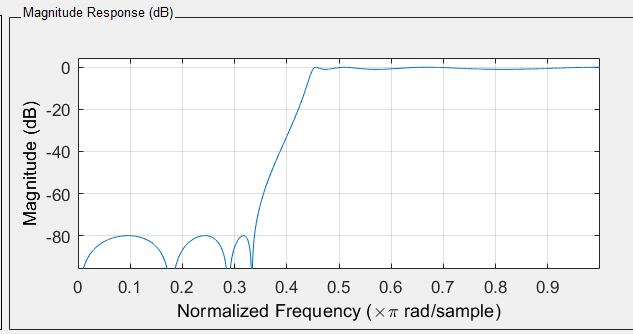
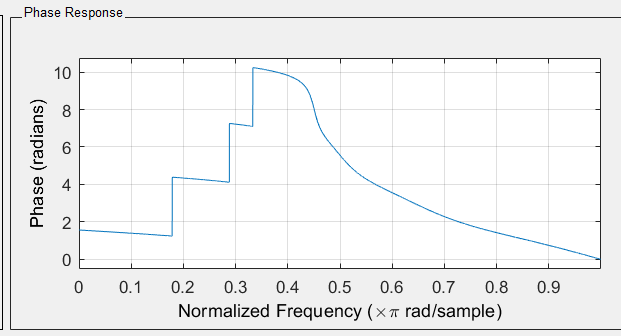
High Pass FIR:

Low Pass IIR:

High Pass IIR:

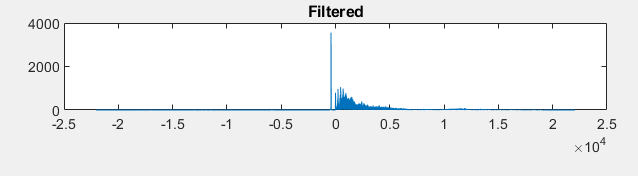
 

|  |  |
| --- | --- |
| \_\_\_1.5\_\_ rad/sample | \_\_\_69.237 \_\_ kHz |

# Results:

Observed bandwidth of speech in rad/sample as well as in kHz:

Speech Signal:

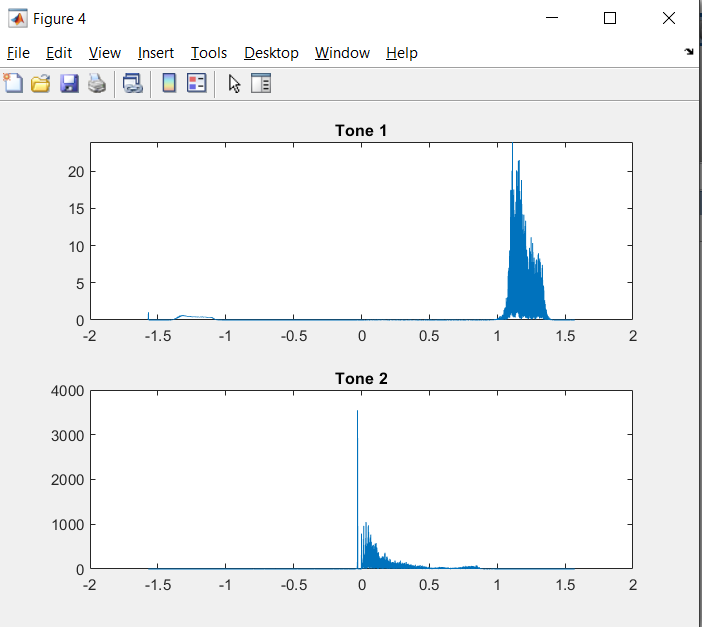


|  |
| --- |
| 2 |

Number of tones you observed:

Frequency, magnitude & phase of each tone

Tones:

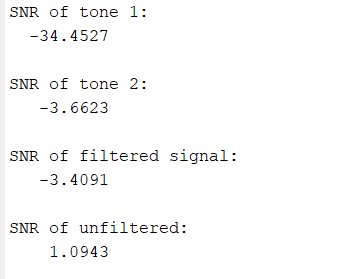


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tone number | Frequency (pi rad/s) | Frequency (KHz) | Magnitude | Phase |
| 1 | 0.34 | 55.125 | 20 |  |
| 2 | 0.8 | 35.28 | 1000 |  |

Conversion process from pi rad/s to KHz:

Frequency in pi rad/s is multiplied by fs to convert into KHz

SNRs:

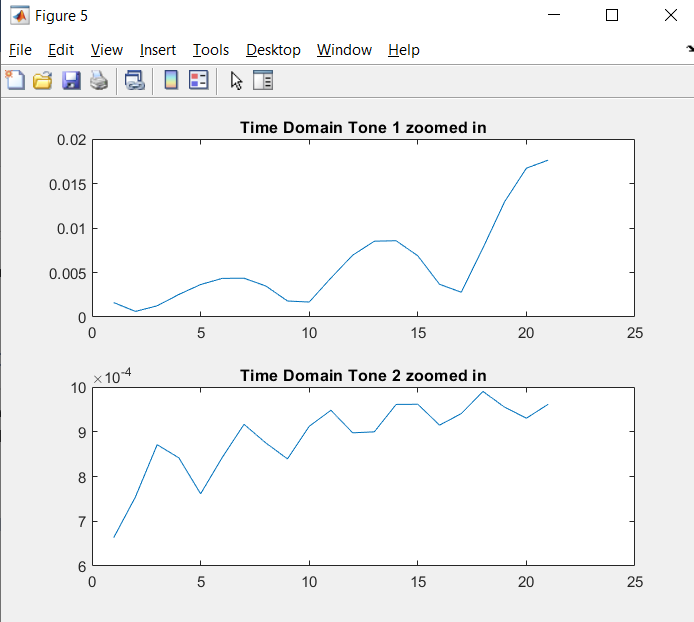


Report the **sentence** you heard in the speech after removing noise.

The sentence was spoken very fast, so could not hear it properly

# Conclusion:

The noise and speech and tones were separated successfully. Difficulty in separating the tones from speech signal as simple FIR filter could not separate it. IIR filter with normalized frequency was used to solve this problem. The tones were to be sin waves at closer look, and the following was observed:



It is closer to sin waves which was supposed to be

# References:

1. Thyagarajan, K., 2019. Introduction To Digital Signal Processing Using MATLAB With Application To Digital Communications. Cham: Springer International Publishing.
2. Hayes, M. and Hayes, M., 2012. Digital Signal Processing. New York: McGraw-Hill.
3. Hahn, B. and Valentine, D., n.d. Essential MATLAB For Engineers And Scientists

# Appendix:

clc;clear

%Task 1

real = table2array(readtable('Real(cvs).csv'));

img = table2array(readtable('Imaginary(cvs).csv'));

fs=44100;

len\_img = length(img);

len\_real = length(real);

freq\_response = zeros(1,len\_img);

for i=1:len\_img

if i<=len\_real

freq\_response(i)=complex(real(i),img(i));

else

freq\_response(i)=complex(0,img(i)); %For no values of real in csv, 0 supposed

end

end

%Importing the filter variables

Num = load('Num.mat'); %Lowpass filter with cutoff 3500 Hz

Num = Num.Num;

Num2 = load('Num2.mat'); %Highpass filter with cutoff 3500 HZ

Num2 = Num2.Num2;

G = load('G.mat'); %Low pass IIR Filter with normalised freq cutoff = 0.35

G = G.G;

G2 = load('G2.mat'); %High pass IIR Filter with normalised freq cutoff = 0.35

G2=G2.G2;

SOS = load('SOS.mat');

SOS=SOS.SOS; %This is done as variable loaded is in struct data type

SOS2 = load('SOS2.mat');

SOS2=SOS2.SOS2;

%Task 2

%Shifting the frequency response to -pi to pi

n = length(freq\_response);

f = (0:n-1)\*fs/n;

power\_ori = abs(freq\_response).^2/n; %For original

n=length(freq\_response);

y = fftshift(freq\_response);

fshift = (-n/2:n/2-1)\*pi/n;

powershift = abs(y).^2/n;

figure

subplot(211)

plot(fshift,powershift)

title('Magnitude Response of original Signal')

phase = angle(y);

subplot(212)

plot(fshift,phase)

title('Phase Response of original signal')

%Task 3

y\_inv = ifft(y); %Data in time domain

figure

subplot(311)

plot(fshift,abs(fft(y\_inv)))

title('Original')

noise = filter(Num,1,y\_inv); %Num is lowpass filtered variable

subplot(312)

plot(fshift,abs(fft(noise)))

title('Noise')

filtered = filter(Num2,1,y\_inv); %Num1 is highpass filtered value

subplot(313)

plot(fshift,abs(fft(filtered)))

title('Filtered')

%Storing signals

audiowrite('noisy\_signal\_6.wav',abs(noise),fs);

audiowrite('filtered\_speech\_6.wav',abs(filtered),fs);

%Plotting magnitude and phase responses of filtered and unfiltered signal

figure

subplot(221)

plot(fshift,powershift)

title('Magnitude Response of original Signal')

phase = angle(y);

subplot(222)

plot(fshift,phase)

title('Phase Response of original signal')

power\_filtered = abs(filtered).^2/n;

phase\_filtered = angle(filtered);

subplot(223)

plot(fshift,power\_filtered)

title('Magnitude Spectrum of Filtered')

subplot(224)

plot(fshift,phase\_filtered)

title('Phase Spectrum of Filtered')

%Task 4

[b,a]=sos2tf(SOS,G); %Converting these variables so they can be passed to filter function

[b2,a2]=sos2tf(SOS2,G2);

tone1 = filter(b,a,filtered);

tone2 = filter(b2,a2,filtered);

figure

subplot(211)

plot(fshift,abs(fft(tone1)))

title('Tone 1')

subplot(212)

plot(fshift,abs(fft(tone2)))

title('Tone 2')

%Storing tones

audiowrite('filtered\_tone\_1\_6.wav',abs(tone1),fs);

audiowrite('filtered\_tone\_2\_6.wav',abs(tone2),fs);

%Phase graphs of tones

subplot(211)

plot(fshift,angle(tone1))

title('Phase of tone 1')

subplot(212)

plot(fshift,angle(tone2))

title('Phase of tone 2')

%Proving tones are sine waves

figure

subplot(211)

plot(abs(tone1(20:40)))

title('Time Domain Tone 1 zoomed in')

subplot(212)

plot(abs(tone2(20:40)))

title('Time Domain Tone 2 zoomed in')

%Reporting SNR

disp('SNR of tone 1:')

disp(snr(tone1,noise))

disp('SNR of tone 2:')

disp(snr(tone2,noise))

disp('SNR of filtered signal:')

disp(snr(filtered,noise))

disp('SNR of unfiltered:')

disp(snr(y\_inv,noise))