Alexandria University
Faculty of Engineering
Electronics and Communication Department
Signals and Systems EEC271

جامعة الإسكندرية
كلية الهندسة
قسم الاتصالات والإلكترونيات
الإشارات والأنظمة

Mini Project II
Simple Communication System

الاسم	الرقم الجامعي
مصطفى سيد أحمد طه	19016660
مروان محمد أحمد أبو العلا	19016615
محمد رزق أمين محمد	19016365
هيثم أحمد شعبان أحمد	19016856

Introduction

The aim of the project is to process sound signal and implement simple communication system, as shown in figure 1, then observe each section of this system. Program transmits signal given at transmitter then send it through one of four channels. Program can add noise which is standard normal distribution function. Signal is then received at receiver and filtered so that the output signal is voice sound only.

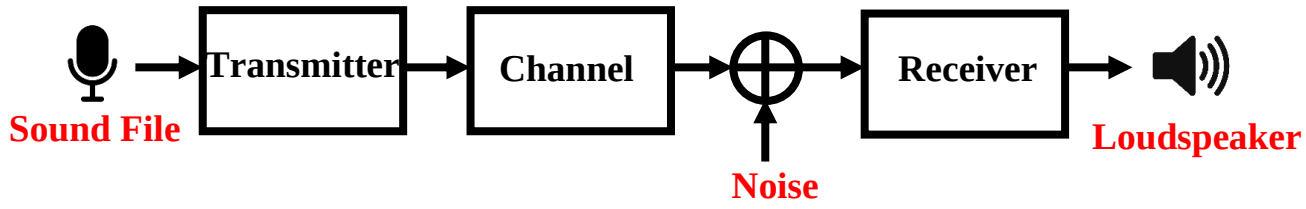
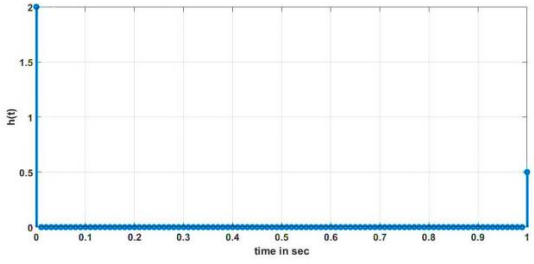


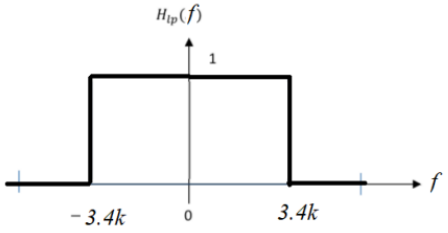
Figure 1

Program Specifications

1. **Transmitter**
   At the first stage, which is called the transmitter. It is required to enter the sound file and prepare it for the transmission over the channel.
2. **Channel**
   The channel has the following impulse response. At this stage, it is needed to pass the sound message over the channel. There are 4 options for the channel impulse response.
   a. Delta function
   b. exp(-2*pi*5000t)
   c. exp(-2pi*1000t)
   d. The channel has the following impulse response shown in figure 2.

   Try the four different impulse responses for the channel and compare the effect of the first three ones on the sound signal.
3. **Noise**
   The program should have the ability to add noise (simply random signal) to the output of the channel The random signal generation is done as following:
   Z(t) = sigma * randn(1, length(x))
   Where x is a vector represents the output of the channel.
   The user should enter the value of the sigma at this stage The output will be a Gaussian distributed noise with zero mean and standard deviation of sigma.
4. **Receiver**
   In order to limit the effect of the noise,
   a. Construct an ideal low pass filter which has a cutoff 3400 KHz. The frequency response of the filter as shown in figure 3.
   b. Pass the noisy sound over the ideal filter.





Algorithm & Code

Transmitter

The program reads audio file using ‘audioread’ function and store samples in x variable and sampling frequency in fs.

The program specifies period of audio file required for testing time_initial and time_final is inputted from user. Then x is cut so that the input signal that will be played is samples specified. If time_final is less than or equal time_initial, the samples will be from 0 to end of audio file Audio file is, then played using sound function.

```
% Transmitter
clear;
clc;
% Import Audio File
[x, fs]=audioread('Imagine_Dragons_-_Thunder.wav');

% Period
time_initial  = input('Enter Starting time in audio file : ');
time_final = input('Enter Ending time in audio file : ');

% If statement used to check if input time_final less than or equal
% time_final and if so the audio file will be played from zero to the end

if( time_initial >= time_final)
    time_initial = 0;
    time_final = length(x)/fs;
end

% Using only time of input Audio file
x=x(1+(time_initial)*fs:(time_final)*fs,1).';

% making time initial to zero so that delta can be applied
        time_final = time_final - time_initial;
        time_initial = 0;

% Playing time of Audio file
  sound(x,fs);
```

t is variable used as x-axis in time representation. Signal in time domain is then plotted.

```
% Time domain Representation
t = linspace(time_initial, time_final, fs * abs(time_final-time_initial));

figure
plot(t, x);
xlabel('Time in sec');
ylabel('Input Signal in time domain x')
title('Input Signal Representation in time domain');
```

Signal is then converted to frequency domain using fft function (fast Fourier transform) and centered at zero using fftshift function.

Fvec is vector used as x-axis to represent signal in frequency domain. The signal is then represented by plotting its magnitude with frequency then plotting its phase with frequency.

```
% Frequency domain Representaion
X = fftshift(fft(x));
Fvec = linspace(-fs/2, fs/2, length(X));

% Magnitude Specturm Ploting
figure
plot(Fvec,abs(X));
xlabel('Frequency in Hz');
ylabel('Magnitude of input signal abs(X)')
title('Magnitude Spectrum');

% Phase Specturm Ploting
figure
plot(Fvec,angle(X));
xlabel('Frequency in Hz');
ylabel('phase of input signal angle(X)')
title('Phase Spectrum');
```

Channel

Using “menu” function, the user can choose channel option as it returns the number of options selected which is then checked using switch case statement.

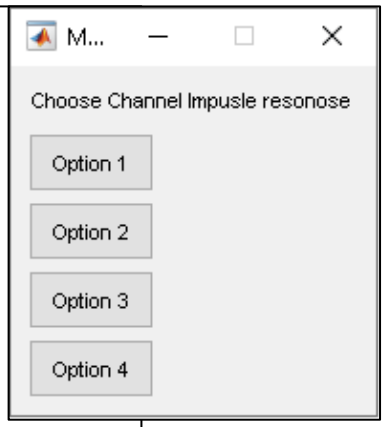
```
% Channel

time = abs(time_final - time_initial);

option = menu('Choose Channel Impusle resonose','Option 1','Option 2','Option 3','Option 4');
```

Implementation of delta function

```
function x = delta(time_initial,time_final,to,t)
time = abs(time_final-time_initial);
x = [zeros(1,round(length(t)*abs(to-time_initial)/time)) 1 zeros(1,round(length(t)*abs(time_final-to)/(time))-1)] ;
end
```

Case 1 is entered if the user chooses option 1.

```
switch (option)

    case 1
      % Impulse response of the channel (delta function)
      % dirac function return signal dc signal from 0 to inf
      % then index is used to check where is the inf in h
      % then this index in h is then assigned to be equal 1
      % then h become the delta function

     h = delta(time_initial,time_final,0,t);

      % Impulse Response of channel Plotting
        figure;
        stem(t, h);
        xlabel('Time in sec');
        ylabel('Impulse Response h');
        title(['Impulse Response of channel ' num2str(option)]);
```



Case 2 is entered if the user chooses option 2. The impulse response of the channel is implemented by the following expression `h = exp(-2*pi*5000*t)`

```
        case 2

          % Impulse response of the channel (exponential function)
          h = exp(-2*pi*5000*t);

          % Impulse Response of channel Plotting
            figure;
            plot(t, h);
            xlabel('Time in sec');
            ylabel('Impulse Response h');
            title(['Impulse Response of channel ' num2str(option)]);
```

Case 3 is entered if the user chooses option 3. The impulse response of the channel is implemented by the following expression `h = exp(-2*pi*1000*t)`

```
        case 3

          % Impulse response of the channel (exponential function)
          h = exp(-2*pi*1000*t);
          % Impulse Response of channel Plotting
            figure;
            plot(t, h);
            xlabel('Time in sec');
            ylabel('Impulse Response h');
            title(['Impulse Response of channel ' num2str(option)]);
```

Case 4 is entered if the user chooses option 4

```
      case 4
        % Impulse response of the channel (two delta functions)
        h = 2*delta(time_initial,time_final,0,t) + 0.5*delta(time_initial,time_final,1,t) ;

        % Impulse Response of channel Plotting
          figure;
          stem(t, h);
          xlabel('Time in sec');
          ylabel('Impulse Response h');
          title(['Impulse Response of channel ' num2str(option)]);
  end
```

Performing convolution between input signal and impulse response to get y signal by multiplying X and H in frequency domain then convert it back to time domain.

```
  % Performing convolution between input signal and impulse respose to get
  % y signal by using conv function
  y = conv(x,h);
  if(option == 4)
      y = y(1:length(x)+fs);

  else
       y = y(1:length(x));
  end
  t_new = linspace(0, length(y)/fs, length(y));
  % Ploting Signal after passing through channel
  figure
  plot(t_new, y);
  xlabel('Time in sec');
  ylabel('Output Signal of the Channel y');
  title(['Output Signal of the Channel ',num2str(option)]);
```

Then signal is transformed to frequency domain, and it is represented by magnitude and phase spectrum.

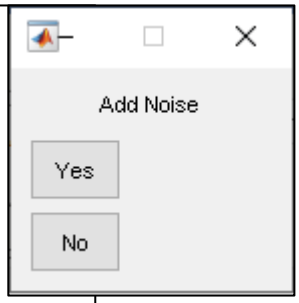
```
% Frequency domain Representaion
Y= fftshift(fft(y));
Fvec = linspace(-fs/2,fs/2,length(Y));

% Magnitude Specturm Ploting
figure
plot(Fvec,abs(Y));
xlabel('Frequency in Hz');
ylabel('Magnitude of input signal abs(X)')
title(['Magnitude Spectrum Output Signal of the Channel ',num2str(option)]);

% Phase Specturm Ploting
figure
plot(Fvec,angle(Y));
xlabel('Frequency in Hz');
ylabel('phase of input signal angle(X)')
title(['Phase Spectrum Output Signal of the Channel ',num2str(option)]);
```

Noise

A menu appears with ‘Add Noise’ title for user to choose whether noise is added or not.

In case ‘Yes’ was chosen, Noise = 1 and the user is asked to enter value of sigma. Sigma is then multiplied by output of randn function which generates vector of number of y elements (y is output of the channel) then noise signal z is added to y to produce signal with Noise y_N.

In case ‘No’ was chosen, Noise = 2 and y_N = y (no noise is added) then signal is played.




```
% Noise

% Adding Noise to output signal of the channel
Noise = menu('Add Noise','Yes','No');

switch (Noise)

    case 1
       % Sigma Input Section
       sigma = input('Enter value of sigma = ');

       % Noise Decleration
       Z = sigma*randn(1,length(y));

       y_N = y + Z;

    case 2

        y_N = y;

end

% Play Sound
sound(y_N,fs);
```

Then the signal with noise y_N is represented in time domain using plot function.

```
% Time domain Representation
figure
plot(t_new,y_N);
xlabel('Time in sec');
ylabel('Output Signal after adding Noise y_N');
title('Output Signal Representaion in Time domain');
```

Signal is then converted to frequency domain and its magnitude is plotted then its phase is plotted to represent Signal in frequency domain.

```
% Frequency domain Representaion
Y_N=fftshift(fft(y_N));
Fvec_new = linspace(-fs/2,fs/2,length(Y_N));

% Magnitude spectrum ploting
figure
plot(Fvec_new,abs(Y_N));
xlabel('Frequency in Hz');
ylabel('Magnitude of output Signal after adding Noise Y_N');
title('Magnitude Spectrum after adding noise');

% Phase spectrum ploting
figure
plot(Fvec_new,angle(Y_N));
xlabel('Frequency in Hz');
ylabel('Phase of output Signal after adding Noise Y_N');
title('Phase Spectrum after adding noise');
```

Receiver

A variable named ‘y_R’ stores the received signal vector. It is the signal after noise y_N. Signal is then converted into frequency domain using fft function and fftshift then Fvec_new is vector used to represent signal in frequency domain.

```
% y_N is signal received after noise is added
% y_R is received signal at Receiver in time domain
y_R = y_N ;

% y_R is converted to frequency domain to apply filtering
Y_R = fftshift(fft(y_R));

% Fvec_new represents all frequencies in signal Y_R
% Note: it is Fvec_new as there is Fvec previously used in reperesenting
% input signal in frequency domain at the transmitter
Fvec_new = linspace(-fs/2,fs/2,length(Y_R));

% sample_per_hertz is calculated which will be used in filtering the signal
sample_per_hertz = (length(y_N)/fs);

% the filter will allow signal of frequency ranging from
% -3.4khz to 3.4khz while prevent the other frequencies from passing so
% samples from start to -3.4 khz and samples from 3.4 khz to end become zeros
% round is used to eliminate the probability of existing fraction index as
% that will result in error.
Y_R (1:round(sample_per_hertz*(fs/2 - 3.4*10^3))) = 0;
```

Sample_per_hertz is used to evaluate number of samples used to represent single hertz. It is rounded to prevent fraction index while filtering signal.

Signal from -fs/2 to -3.4 KHZ and from 3.4 KHz to fs/2 is filtered by putting zeros in index from 1 to sample_per_hertz*(fs/2 -3.4 khz) and index from sample_per_hertz *(fs/2 + 3.4 KHz) +1 to the end of signal.

Signal is represented in frequency domain by its magnitude and Phase. Then it is converted into time domain by using ifftshift to return signal to original shift and ifft(inverse fast Fourier transform) to return signal to time domain then taking real part of output signal.

```
% Representing Received Signal in frequency domain after filtering

% Magnitude Spectrum after filtering
figure
plot(Fvec_new,abs(Y_R));
title('Magnitude Spectrum after filtering')
xlabel('Frequency in (HZ)');
ylabel('Received signal Magnitude  abs(Y_R)');

% Phase Specturm after filtering
figure
plot(Fvec_new,angle(Y_R));
title('phase Spectrum after filtering')
xlabel('Frequency in (HZ)');
ylabel('Received signal Phase angle(Y_R)');

% Returning signal back to time domain
y_R = real(ifft(ifftshift(Y_R)));

% Playing Received signal after filtering
% as it is noticed only the voice is heard as this filter pass frequency of
% voice which is from -3.4khz to 3.4khz and reject the other frequecnies
%sound(y_R,fs);
```

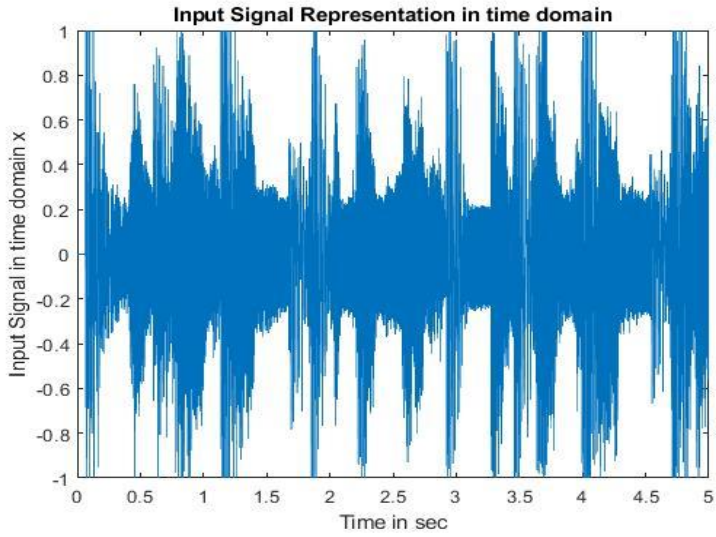
```
% Representing Received Signal in time domain after filtering
figure
plot(t_new,y_R);
title('Received Signal in time domain after filtering')
xlabel('time in (sec)');
ylabel('Received signal y_R');
```

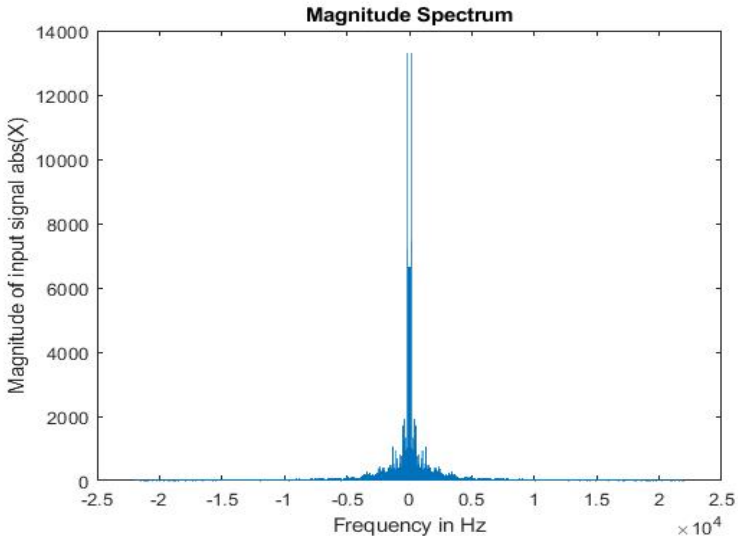
Results

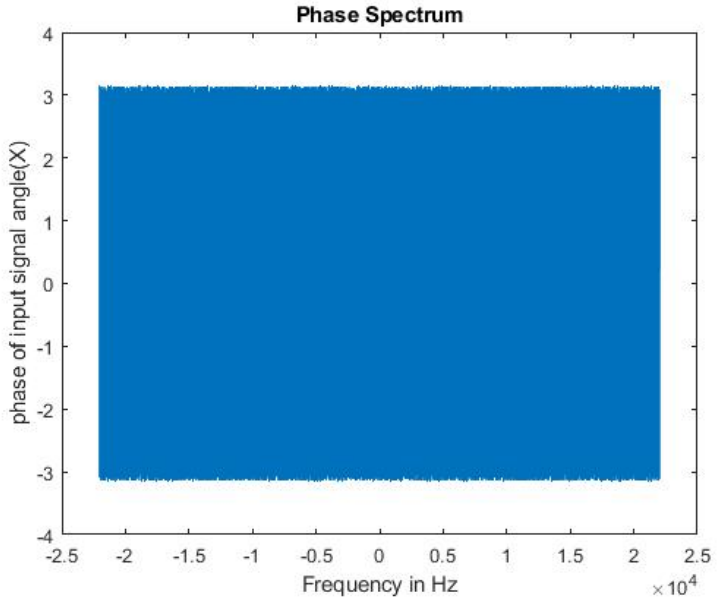
```
Enter Starting time in audio file : 0
Enter Ending time in audio file : 5
```

Transmitter

The input audio file is cut from starting time to ending time entered by user

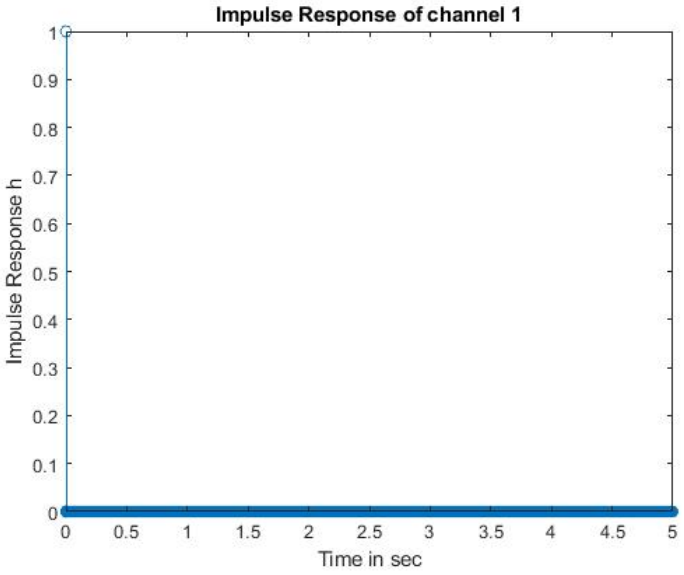


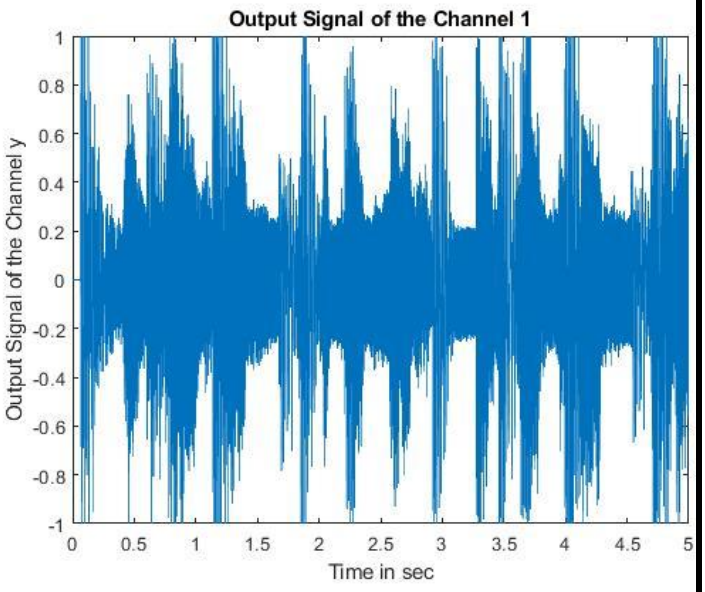




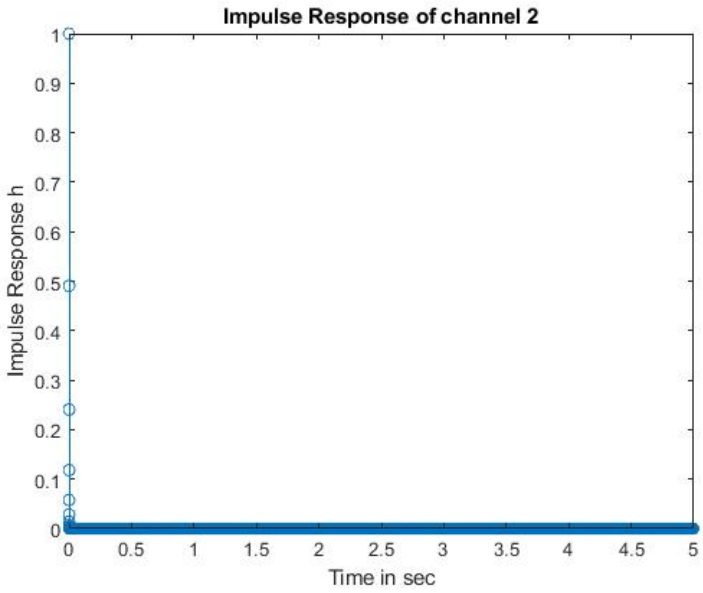
Channel

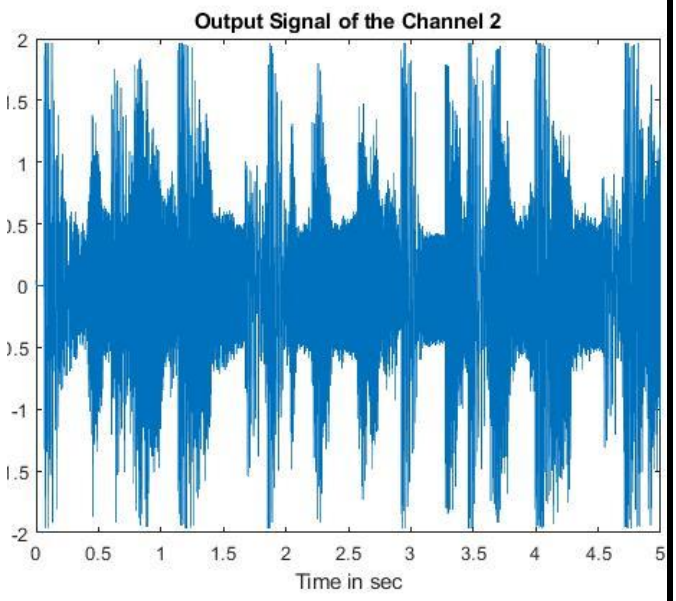
1. Option 1



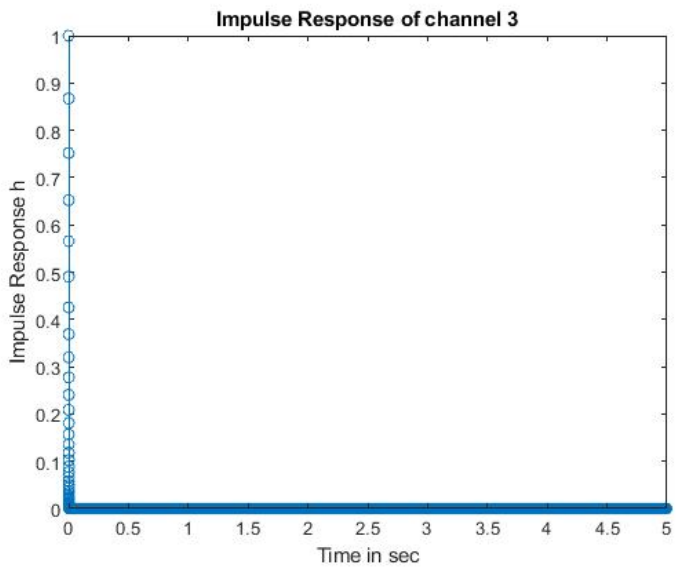


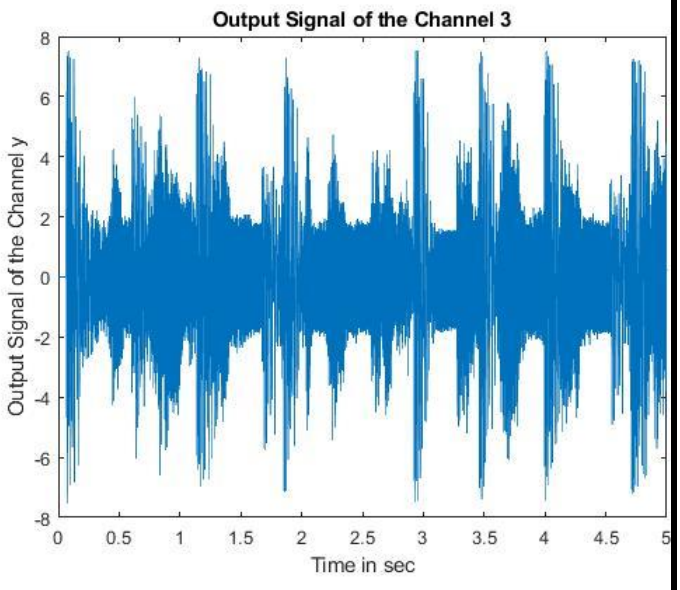
2. Option 2



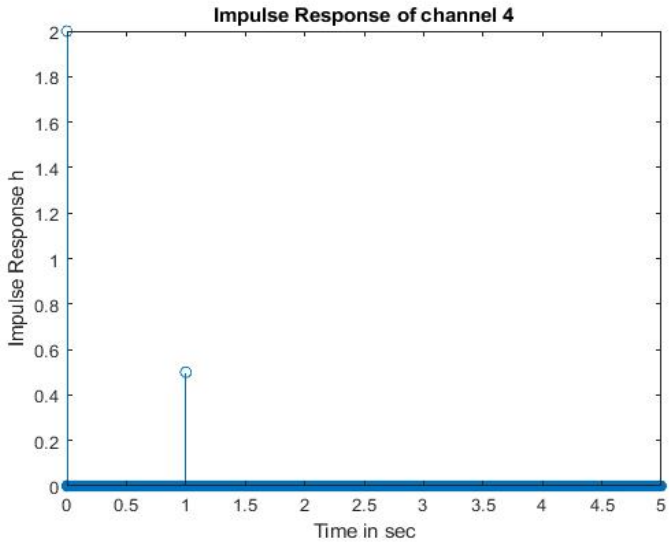


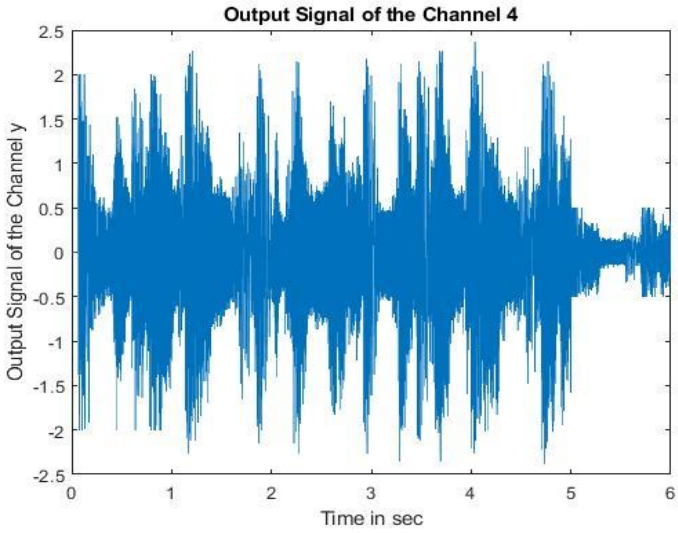
3. Option 3





4. Option 4





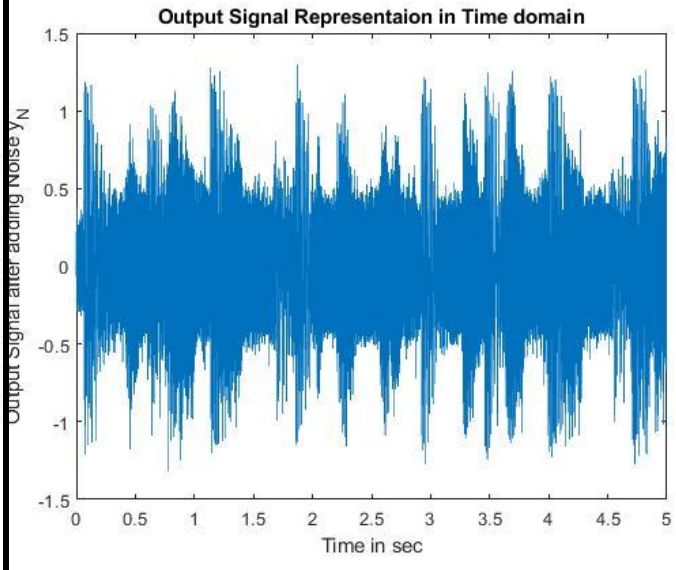
Comment:

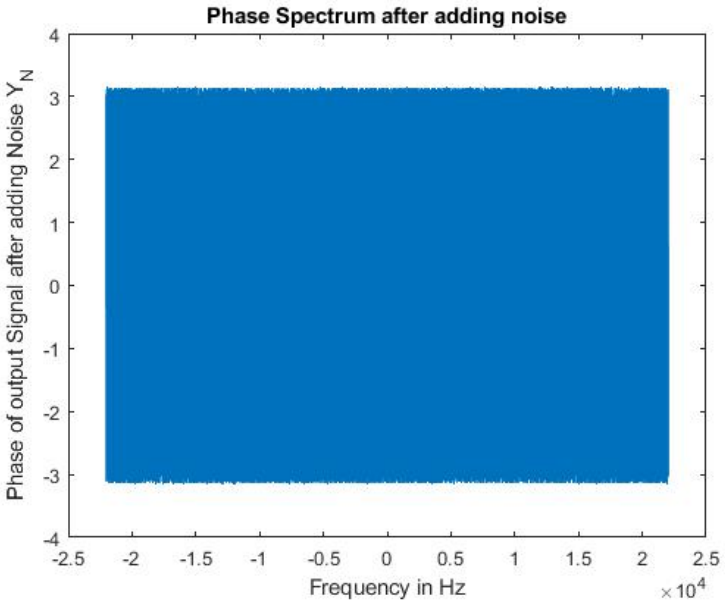
- The first channel produces output signal equal to the input signal as channel impulse response is delta and convolution of any signal with delta is the same input signal.
- The second channel produces signal like first channel but with amplitude scaling = 2 as this channel acts as many delta functions added at t = 0, and their sum = 2. $e^{-2\pi*5000t}$ will have 1 at zero and very small value approximately zero before t=1 and many values less than 1 but greater than zero at time near zero so their sum makes delta function of amplitude = 2. So, the output sound will be much louder and deeper than that from channel 1.
- The third channel produces signal like first channel but with amplitude = 8 as this channel acts as many delta functions added at t = 0, and their sum = 8. $e^{-2\pi*1000t}$ will have 1 at zero and very small value approximately zero before t=1 and even more values less than 1 but greater than zero at time near zero so their sum makes delta function of amplitude = 8. So, the output sound will be much louder and deeper than that from channel 2.

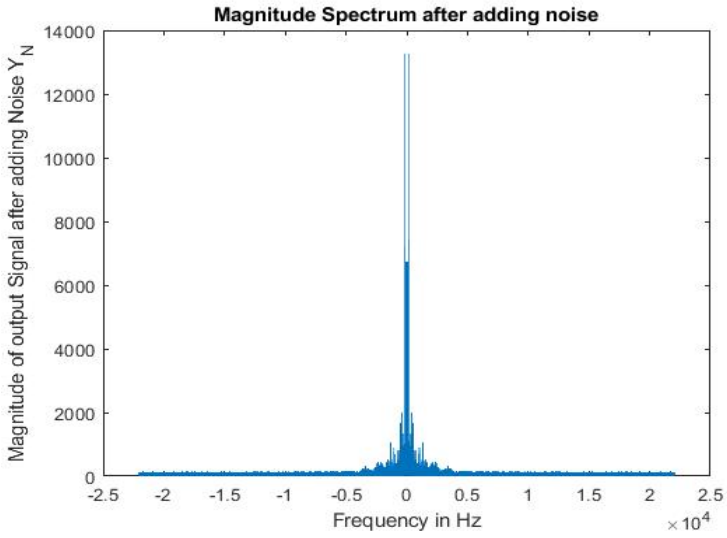
Noise

1. Output signal after adding noise to channel 1

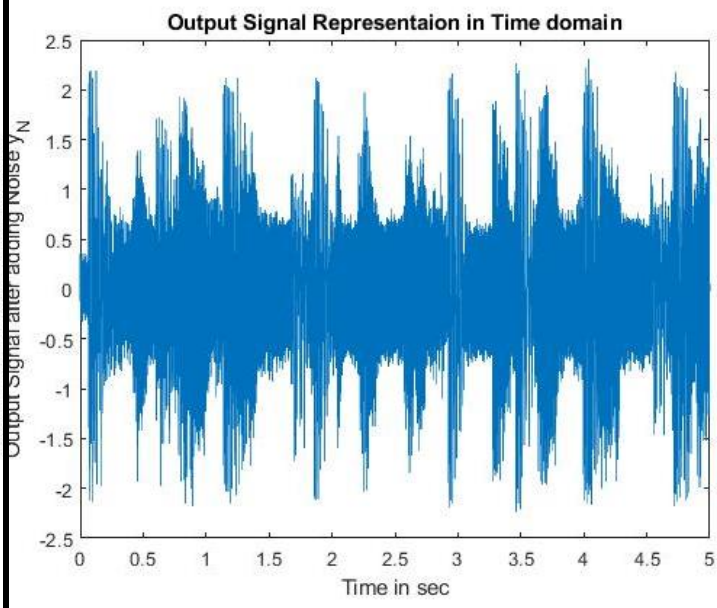
```
Enter value of sigma = 0.5
```

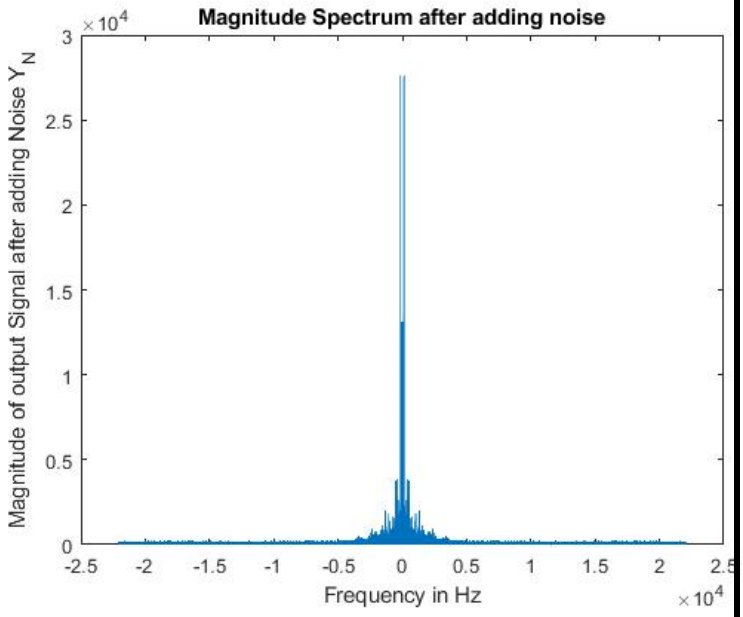


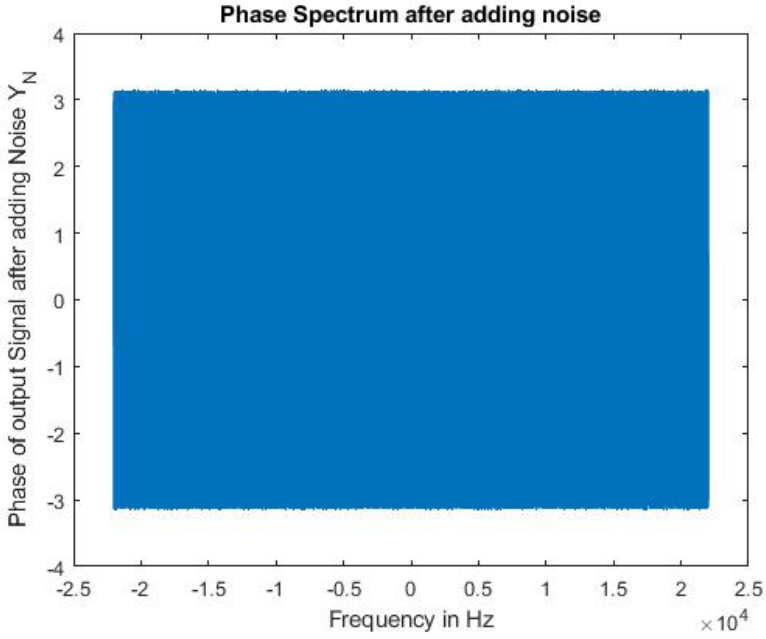




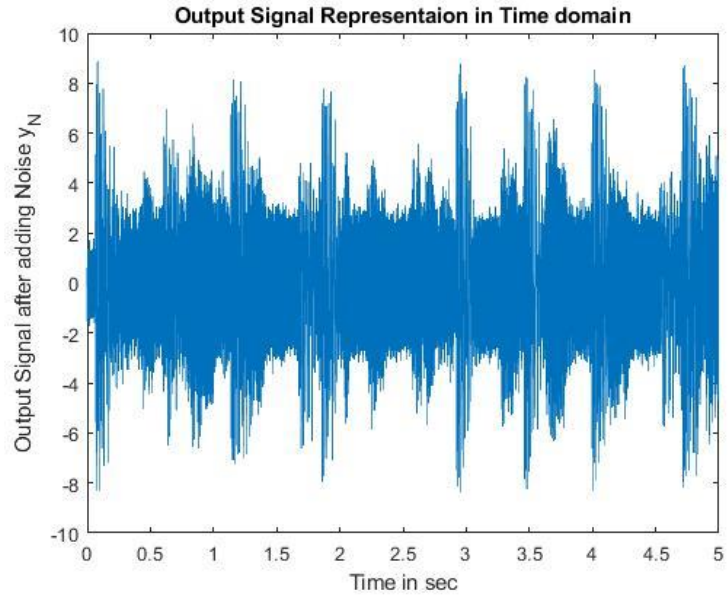
2. Output signal after adding noise to channel 2

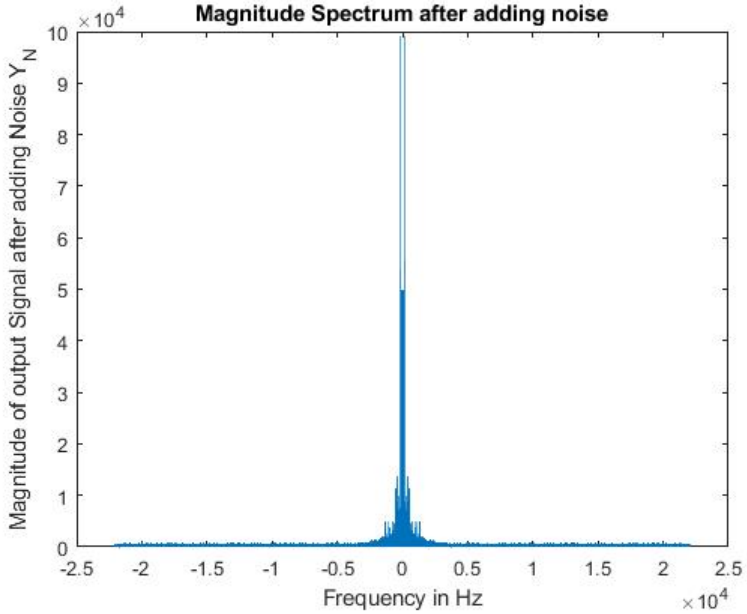


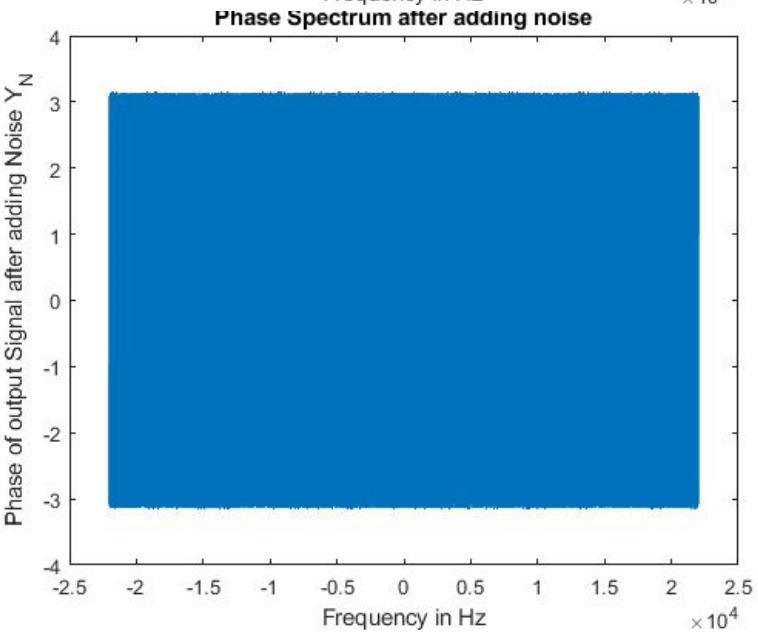




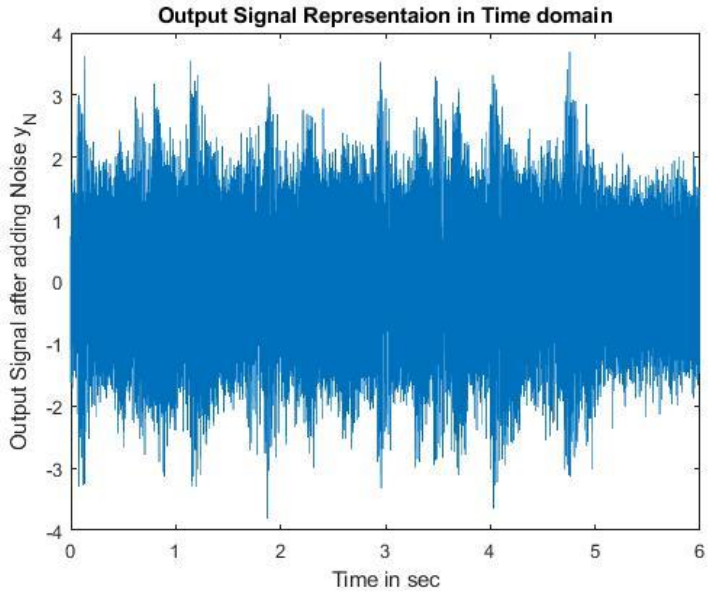
3. Output signal after adding noise to channel 3

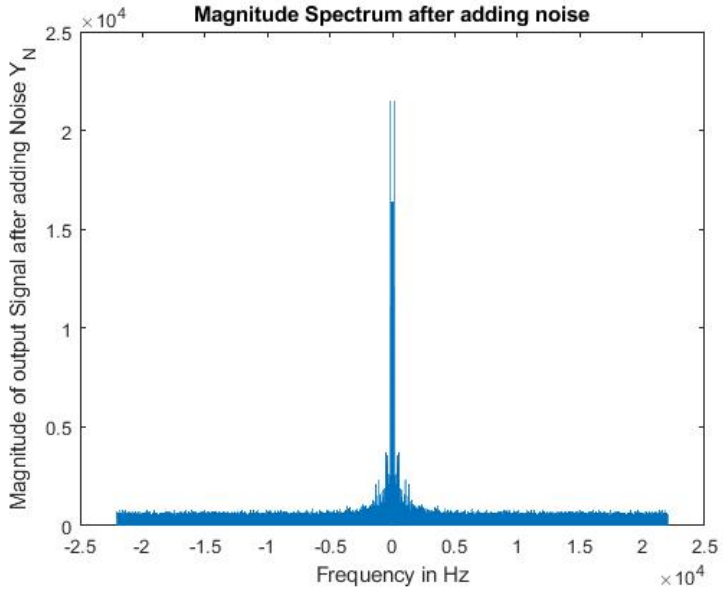


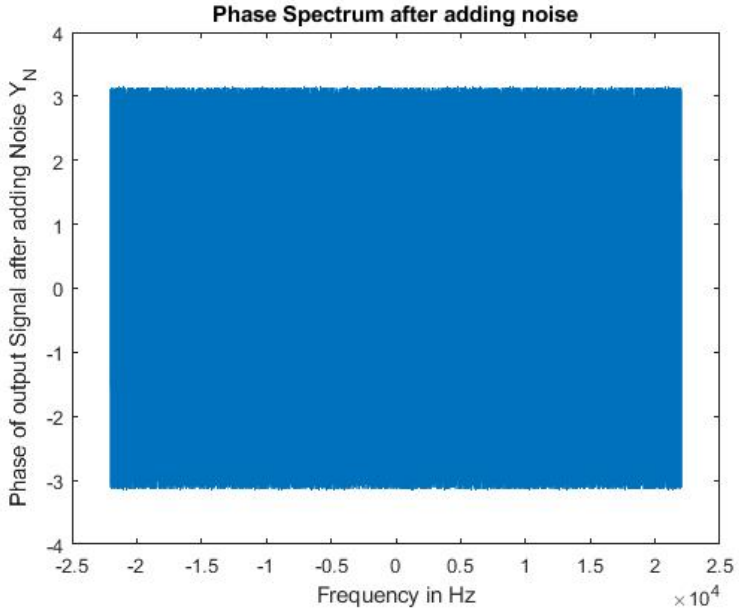




4. Output signal after adding noise to channel 4

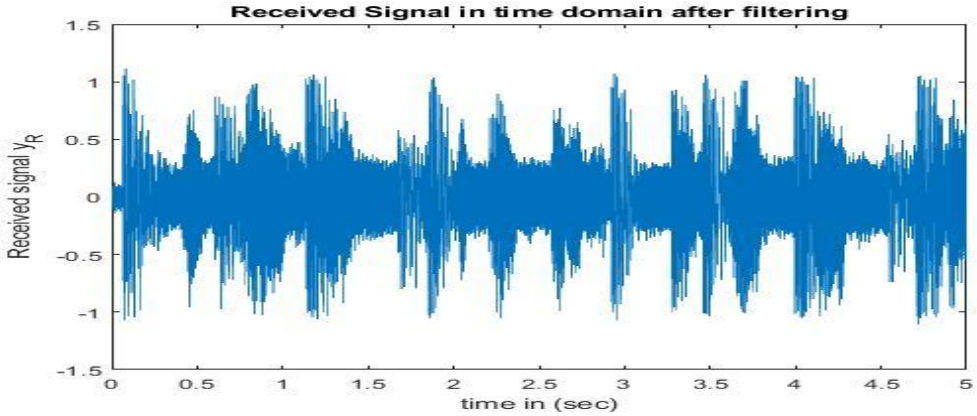




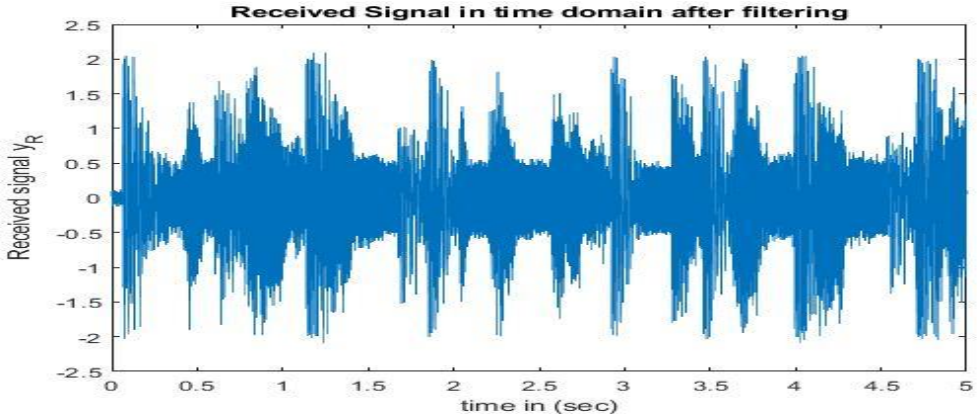


Receiver

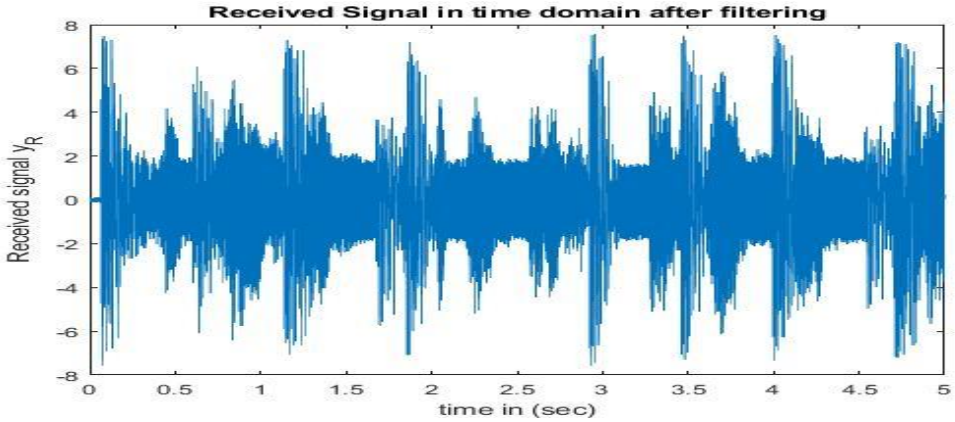
1. The output signal through channel 1



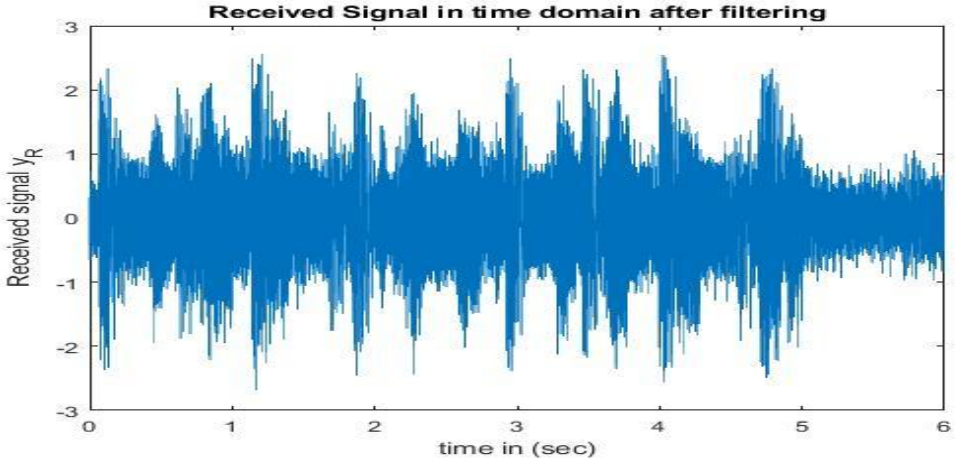
2. The output signal through channel 2



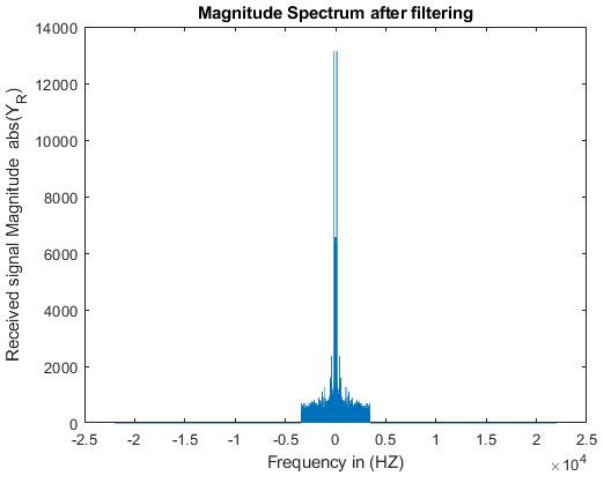
3. The output signal through channel 3

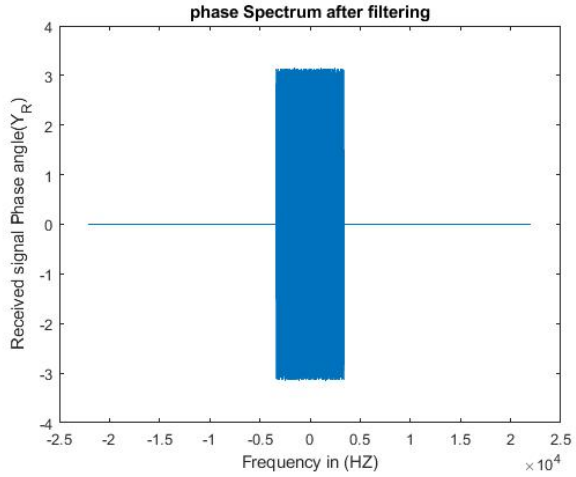


4. Output signal after adding noise to channel 4

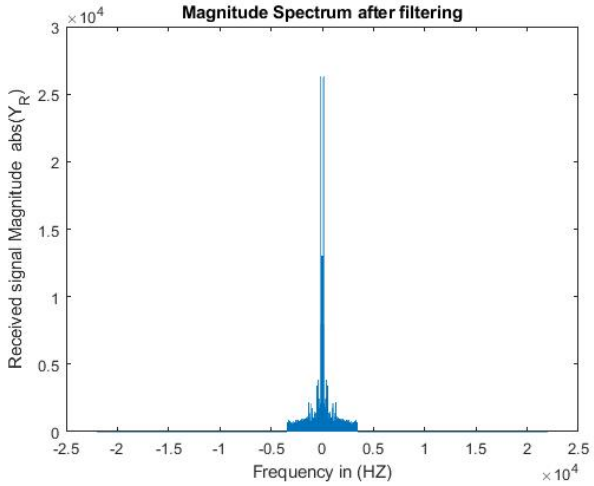


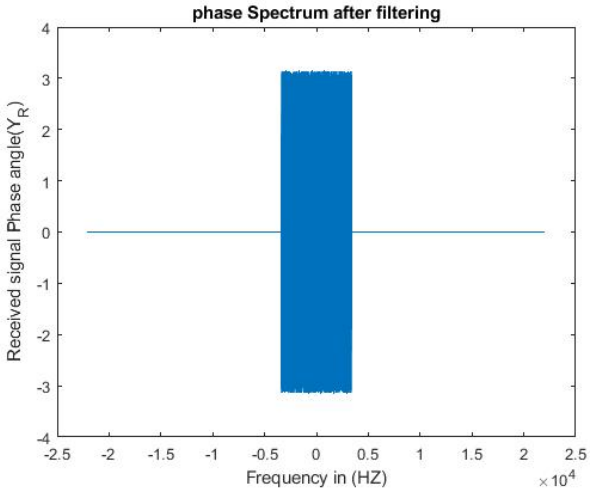
1. The output signal through channel 1 in frequency domain.



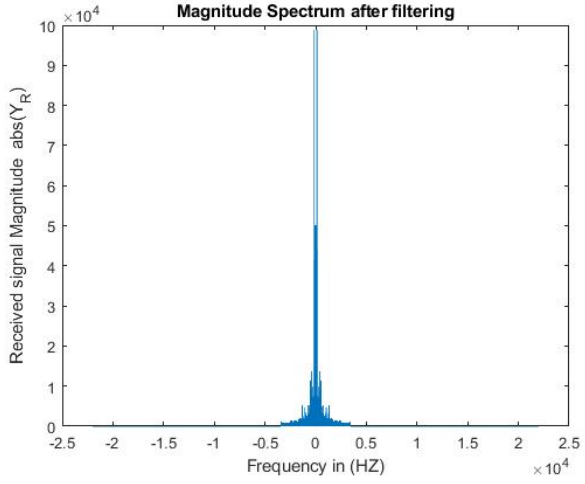


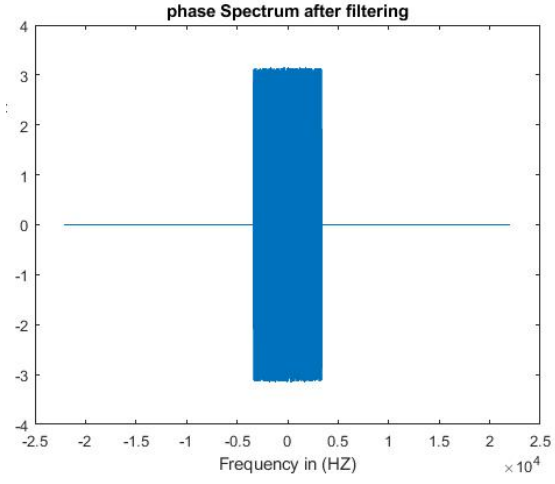
2. The output signal through channel 2 in frequency domain.



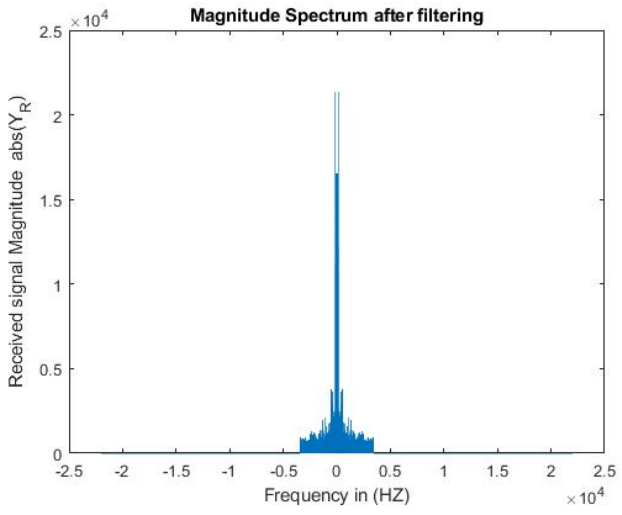


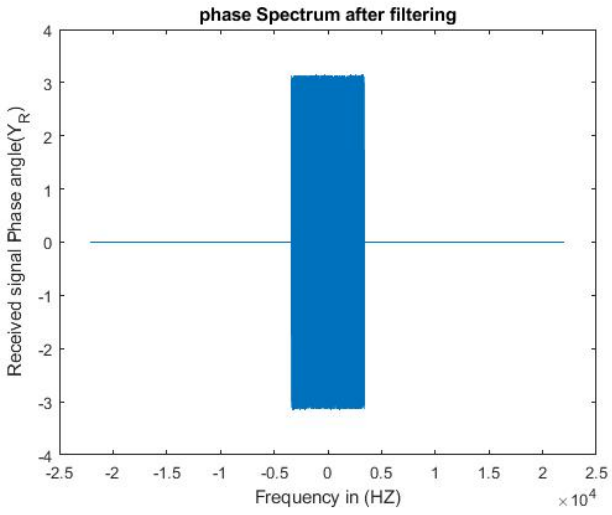
3. The output signal through channel 3 in frequency domain.





4. The output signal through channel 4 in frequency domain.





Comment:

- The output signal from receiver is voice signal only as filter remove any signals with frequencies higher than 3.4k Hz leaving voice signal only with frequency less than 3.4 Hz