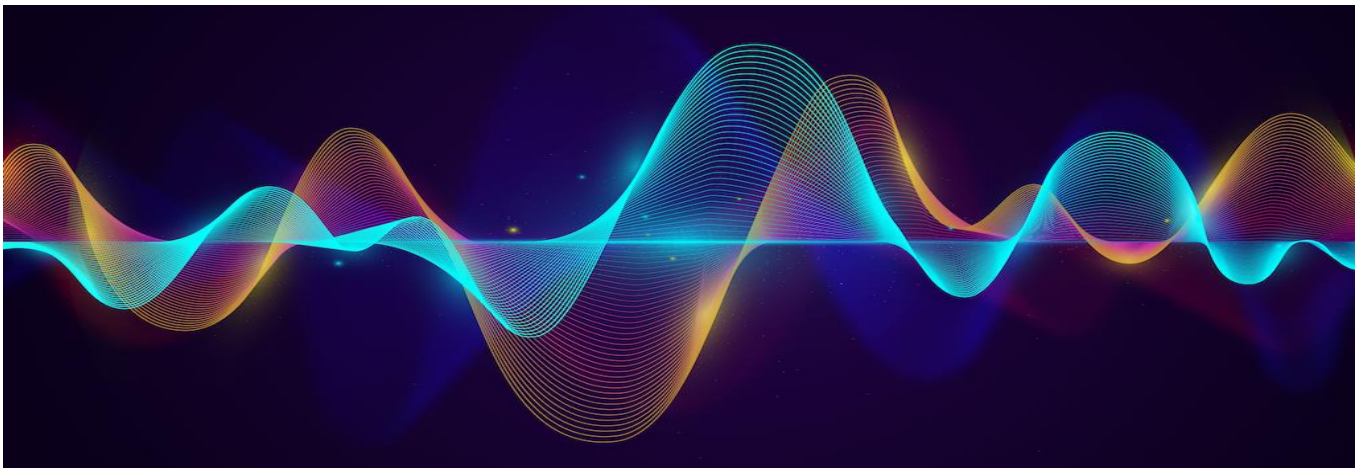
Faculty of Engineering Alexandria University

EEC271 – Signals and Systems

Electronics and Communication Engineering Major

Signal Processing using MATLAB

Term Project II: Communication System



Signals Final project

In this project, you will implement a very simple communication systems as shown in the following figure. Here you try to send a sound file over a communication channel & then try to receiver it.

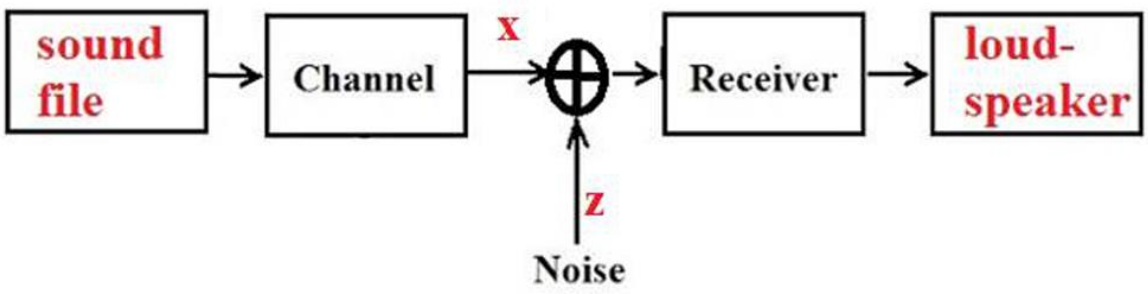


Figure 1Block diagram of a communication system

The project consists of four stages:

First stage transmitter:

1. Transmitter

At the first stage, which is called the transmitter. You will enter your sound file and prepare it for the transmission over the channel.

Requirement:

- Play your sound file through Matlab
- Plot your sound file in time domain and the frequency domain

Here we can browse for any audio file and read it, we take only 5 seconds of the music to reduce processing. As every music file has two components one for the left speaker and another for the right one so we plot both speakers in the time domain.

Then we converted the signal to the frequency domain using Fourier transform and shift Fourier transform and then plot the signals of both speakers (magnitude and phase spectra, real and Imaginary components):

The code:

```
%transmitter
%-------------
%time

fprintf('browse for the audio you want to transmit\n')
[file,path] = uigetfile('*.mp3');%there we can browse for mp3 files and get the file nanme and path
fname = fullfile(path,file);%here fullfile combines the path and file name in one line
[y,fs]=audioread(fname);%reading the data of the sound and getting the sampling frequency
x=y(1:5*fs,:);%taking 5 seconds from the signal as fs is (sample per sec) then it is multiplied by
%number of seconds to get number os samples taken from the sound
% for all columns channels

fprintf('the audio is playing...\n')
sound(x,fs)%playing the 5 sec sound
pause(5)

t=linspace(0,5,5*fs);%creating time axis for the signal % determins by the num of samples
t=t';
% plotting the left and right speaker data

figure(1)

subplot(2,2,[1,2])
plot(t,x(:,1))
title('left speaker sound in time domain')

subplot(2,2,[3,4])
plot(t,x(:,2))
title('right speaker sound in time domain')
%--------------------------------------------------------------------------
```

```
%frequancy

s=linspace(-fs\2,fs\2,5*fs);%creating frequency axis for the signal
ys=fftshift(fft(x));%appling fourier transform and fourier shift transform to get the sygnal in
%frequency domain with its -ve and +ve components

mag_L=abs(ys(:,1));%calculating left speaker signal magnitude
mag_R=abs(ys(:,2));%calculating right speaker signal magnitude

phase_L=angle(ys(:,1));%calculating left speaker signal phase
phase_R=angle(ys(:,2));%calculating right speaker signal phase

real_ys_L=real(ys(:,1));%calculating left speaker signal real part
real_ys_R=real(ys(:,2));%calculating right speaker signal real part

imag_ys_L=imag(ys(:,1));%calculating left speaker signal imaginary part
imag_ys_R=imag(ys(:,2));%calculating right speaker signal imaginary part

```

```
49      %plotting the signal components
50
51 -    figure(2)
52      %magnitude
53 -    subplot(2,2,1)
54 -    plot(s,mag_L)
55 -    title('magnitude of left speaker sound in frequency domain')
56
57 -    subplot(2,2,2)
58 -    plot(s,mag_R)
59 -    title('magnitude of right speaker sound in frequency domain')
60      %----------------------------------------
61      %phase
62 -    subplot(2,2,3)
63 -    plot(s,phase_L,'r')
64 -    title('phase of left speaker sound in frequency domain')
65
66 -    subplot(2,2,4)
67 -    plot(s,phase_R,'r')
68 -    title('phase of right speaker sound in frequency domain')
69      %----------------------------------------
70 -    figure(3)
71      %real part
72 -    subplot(2,2,1)
73 -    plot(s,real_ys_L)
74 -    title('real part of left speaker sound in frequency domain')
75
76 -    subplot(2,2,2)
77 -    plot(s,real_ys_R)
78 -    title('real part of right speaker sound in frequency domain')
79      %----------------------------------------
80      %imaginary part
81 -    subplot(2,2,3)
82 -    plot(s,imag_ys_L,'r')
83 -    title('imaginary part of left speaker sound in frequency domain')
84
85 -    subplot(2,2,4)
86 -    plot(s,imag_ys_R,'r')
87 -    title('imaginary part of right speaker sound in frequency domain')
88
```

Second stage (channel)

2. Channel

The channel has the following impulse response. At this stage, you will need to pass your sound message over the channel

You have 4 options for the channel impulse response.

1. Delta function
2. exp(-2pi*5000t)
3. exp(-2pi*1000t)
4. The channel has the following impulse response

The code:

```
%%
%channel -- Impulse

flag_1=0;
while flag_1==0
channel = input('Select the channel: \n  (1)Delta function.\n (2)exp(-
2pi*5000t).\n (3)exp(-2pi*1000t).\n (4)2*delta(t)+ delta(t-1).\n');
    if channel>=1 & channel<=4
        flag_1=1;
    else
        flag_1=0;
        messagebox6=msgbox("Select the correct channel from (1 to 4) ");
    end
end


if channel == 1 % still the same
    ch = zeros(5*fs,1); %col vector due to same num element wise muti
    ch(1) = 1;
    z = zeros(length(conv(x(:,1), ch)) , 2 );
    z(:,1) = conv(x(:,1), ch);
    z(:,2) = conv(x(:,2), ch);



elseif channel == 2 % inc amp
    ch=exp(-2*pi*5000*t);
    z = zeros(length(conv(x(:,1), ch)) , 2 );
    z(:,1) = conv(x(:,1), ch);
    z(:,2) = conv(x(:,2), ch);

elseif channel == 3 %
    ch=exp(-2*pi*1000*t);
    z = zeros(length(conv(x(:,1), ch)) , 2 );
    z(:,1) = conv(x(:,1), ch);
    z(:,2) = conv(x(:,2), ch);
```

```
elseif channel == 4 %echo
    ch=zeros(5*fs,1);
    ch(1)=2;
    ch(fs +1)=1;
    z = zeros(length(conv(x(:,1), ch)) , 2 );
    z(:,1) = conv(x(:,1), ch);
    z(:,2) = conv(x(:,2), ch);

end

fprintf('the convoluted audio is playing...\n')
sound(z,fs)
pause(5)

t=linspace(0,length(conv(x(:,1), ch))/fs,length(ch)+length(x(:,1))-1);
t=t';
 figure;

subplot(2,1,1)
plot(t,z(:,1))
title('left speaker of convoluted signal')

subplot(2,1,2)
plot(t,z(:,2))
title('right speaker of convoluted signal')
```

First, we check the user input, it must be from 1 to 4, if user enter the false input the program asks the user to enter his choice again.

Then the signal is convoluted with the chosen channel, The length of the output signal = length(channel) + length(signal) - 1

- If the channel 1 is chosen, it is just a delta function, so the signal will be the same.
- If the channel 2 is chosen, the effect of exponential is increasing the amplitude.
- If the channel 3 is chosen, the effect of exponential is increasing the amplitude.
- If the channel 4 is chosen, there are two deltas which makes the echo.

Third stage noise:

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

Requirement:

- Play your sound file after adding noise
- Plot your sound file in time domain and the frequency domain

The code:

```
%noise

fprintf('noise stage: if the value of sigma exceeds 0.1, it will damage the
sound \n please enter the sigma \n')
sigma=input('');
ns=sigma*randn(1,length(z));
ns=ns';

z(:,1)=z(:,1)+ns;
z(:,2)=z(:,2)+ns;

fprintf('the audio with noise is playing ... \n')
sound(z,fs)
pause(10)

figure;

subplot(2,1,1)
plot(t,z(:,1))
title('left speaker of the signal after adding noise in time domain')

subplot(2,1,2)
plot(t,z(:,2))
title('right speaker of the signal after adding noise in time domain')
```

The code:

```
zs=fftshift(fft(z));
s=linspace(-fs\2,fs\2,length(zs));

mag_L=abs(zs(:,1));
mag_R=abs(zs(:,2));

phase_L=angle(zs(:,1));
phase_R=angle(zs(:,2));

figure;

subplot(2,2,1)
plot(s,mag_L)
title('magnitude of left speaker after adding noise in frequency domain')

subplot(2,2,2)
plot(s,mag_R)
title('magnitude of right speaker after adding noise in frequency domain')

subplot(2,2,3)
plot(s,phase_L,'r')
title('phase of left speaker after adding noise in frequency domain')

subplot(2,2,4)
plot(s,phase_R,'r')
title('phase of right speaker after adding noise in frequency domain')
```

In this part of the code, we create a random signal which works as the noise. Then we add it to the output signal of the channel after convolution.

We play the sound and plot its signal in time domain. Then convert it to the frequency domain and plot its magnitude and phase.

Final stage receiver:

4. Receiver

In order to limit the effect of the noise,

1. You will construct an ideal low pass filter which has a cut off of 3400 KHz. The frequency response of the filter as shown in figure.
2. pass the noisy sound over the ideal filter

Requirement:

- Play the sound file after the filter
- Plot the output sound file in time domain and the frequency domain

The code:

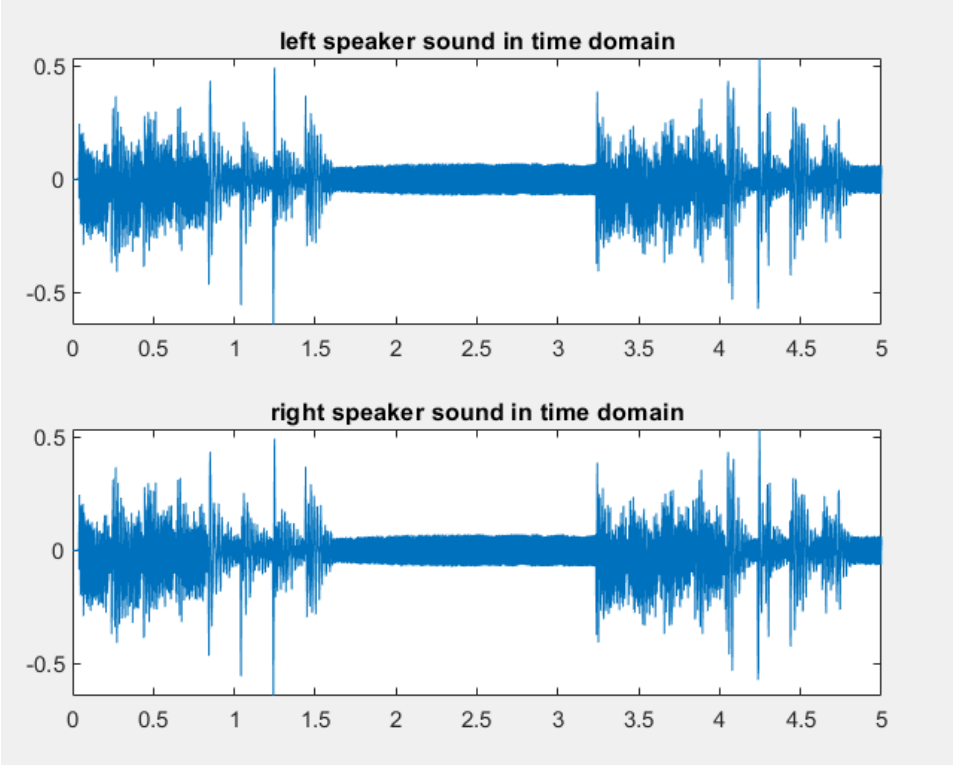
```
%% receiver
time=(length(zs))/fs;

limit1=(fs/2-3400)*time;
limit2=length(zs)-limit1+1;

zs([1:round(limit1) round(limit2):end],1)=0;
zs([1:round(limit1) round(limit2):end],2)=0;

zsnew=zs;

mag_L_new=abs(zsnew(:,1));
mag_R_new=abs(zsnew(:,2));

phase_L_new=angle(zsnew(:,1));
phase_R_new=angle(zsnew(:,2));


figure;

subplot(2,2,1)
plot(s,mag_L_new)
title('magnitude of left speaker after filtering noise in frequency domain')

subplot(2,2,2)
plot(s,mag_R_new)
title('magnitude of right speaker after filtering noise in frequency
domain')

subplot(2,2,3)
plot(s,phase_L_new,'r')
title('phase of left speaker after filtering noise in frequency domain')

subplot(2,2,4)
plot(s,phase_R_new,'r')
title('phase of right speaker after filtering noise in frequency domain')
```

```
% time domain
znew=real(ifft(ifftshift(zsnew)));

figure;

subplot(2,1,1)
plot(t,znew(:,1))
title('left speaker of the signal after filter in time domain')

subplot(2,1,2)
plot(t,znew(:,2))
title('right speaker of the signal after filter in time domain')

fprintf('the audio after filteing is playing ... \n')
sound(znew,fs)
```

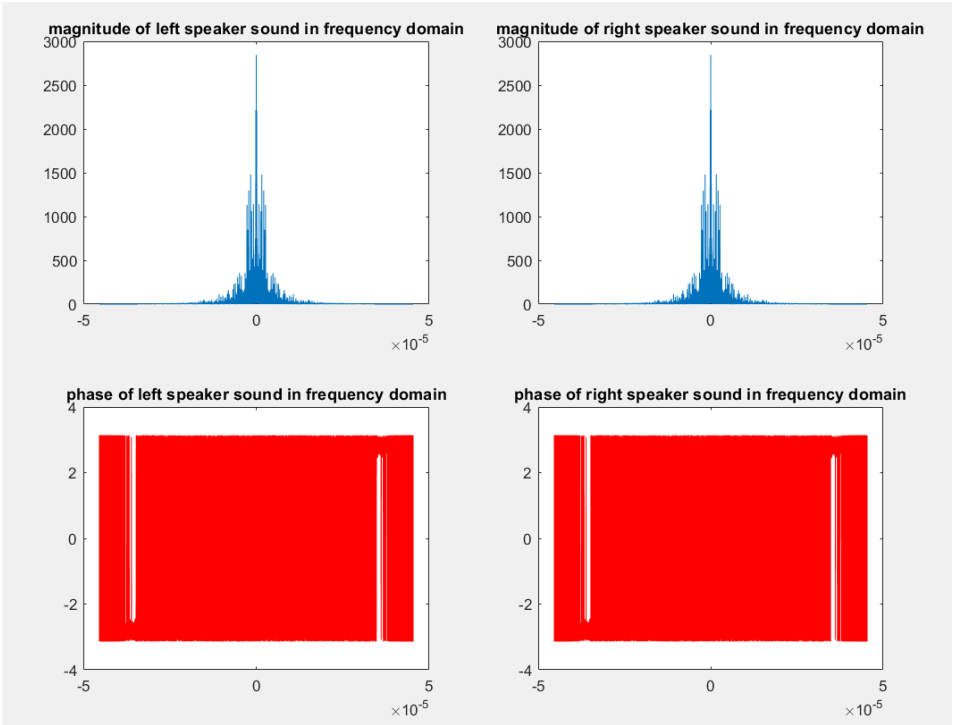
Receiver is needed to filter the noise in the original sound after being transmitted. In this code, the used filter is ideal low pass filter with cut off frequency 3.4 KHz. The sound passes through the filter then the receiver plays the sound after filtering. Also, figures of the sound after filtering in Time and Frequency domain will be shown.

Example (1) the delta channel output figures

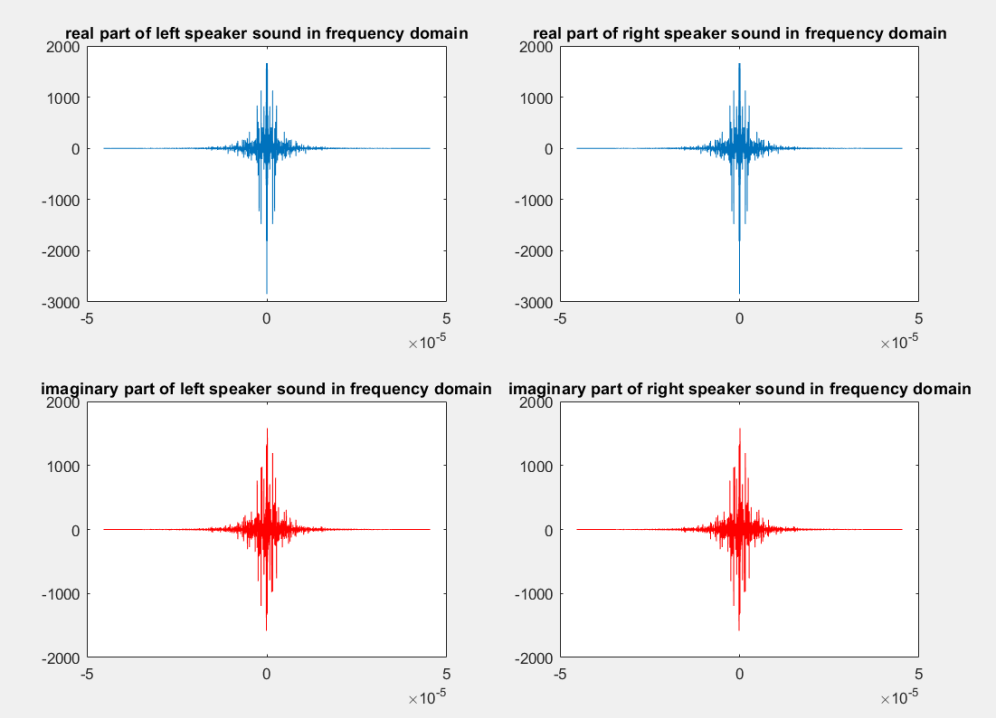
the input signal before Any modification



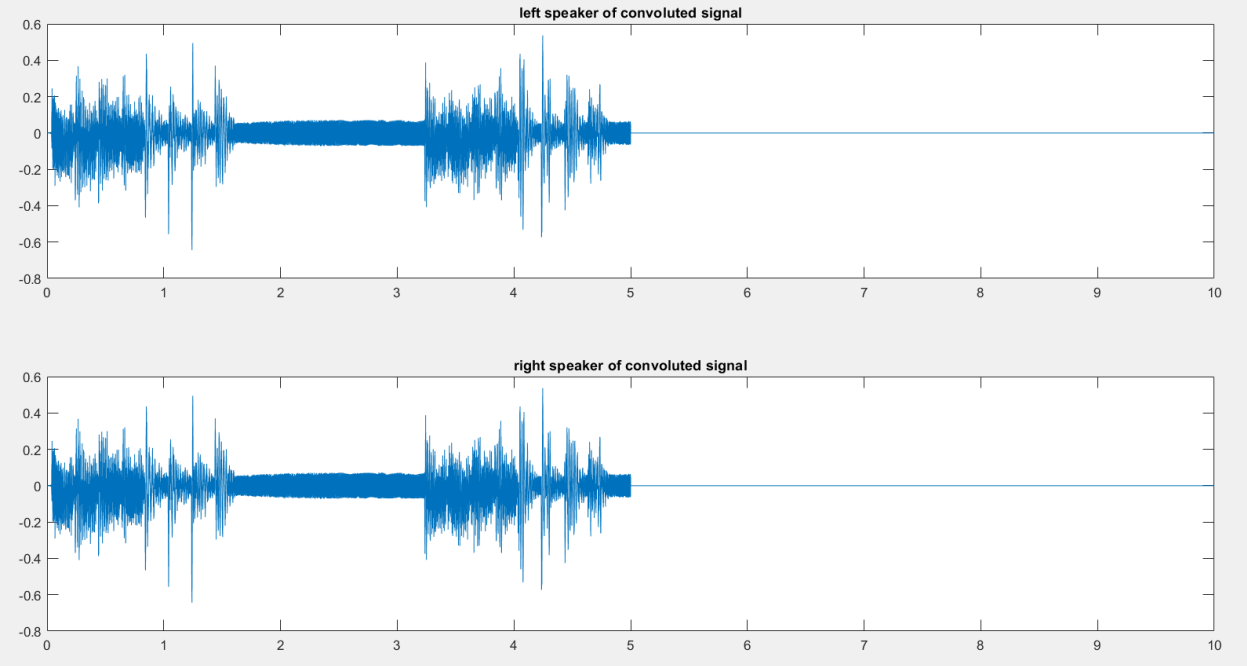
The magnitude and the phase of the signal



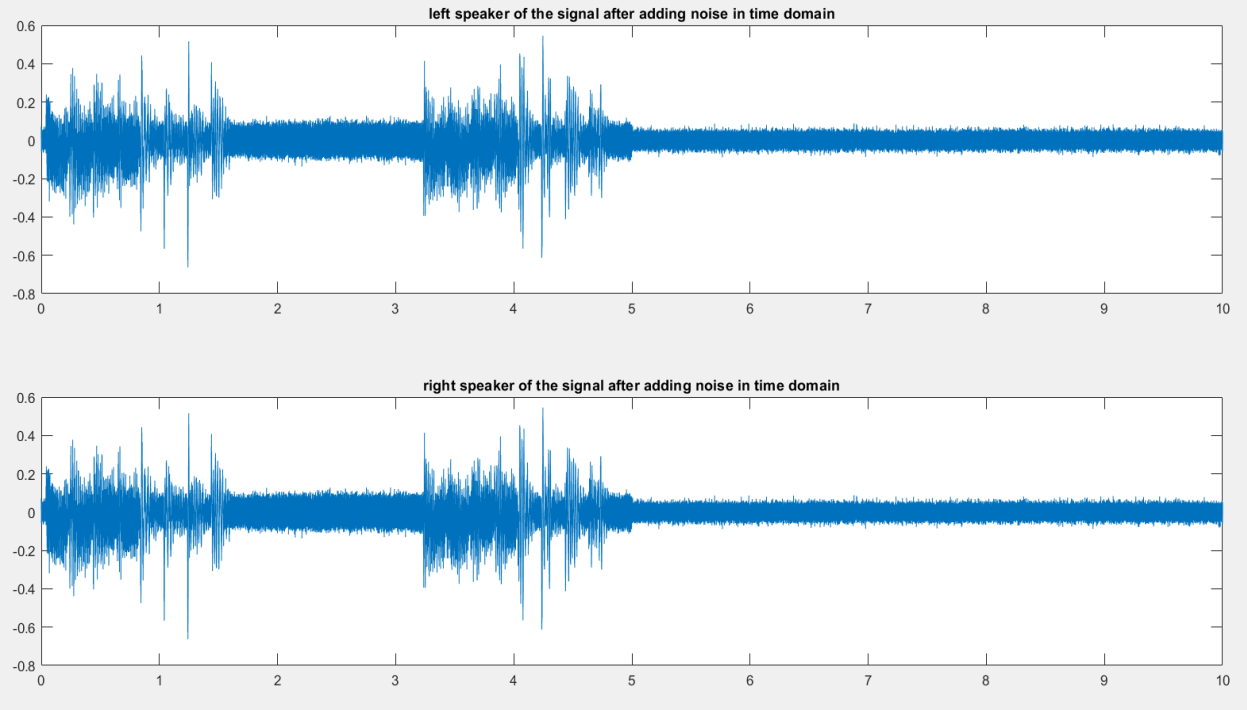
The real and imaginary components of the signal



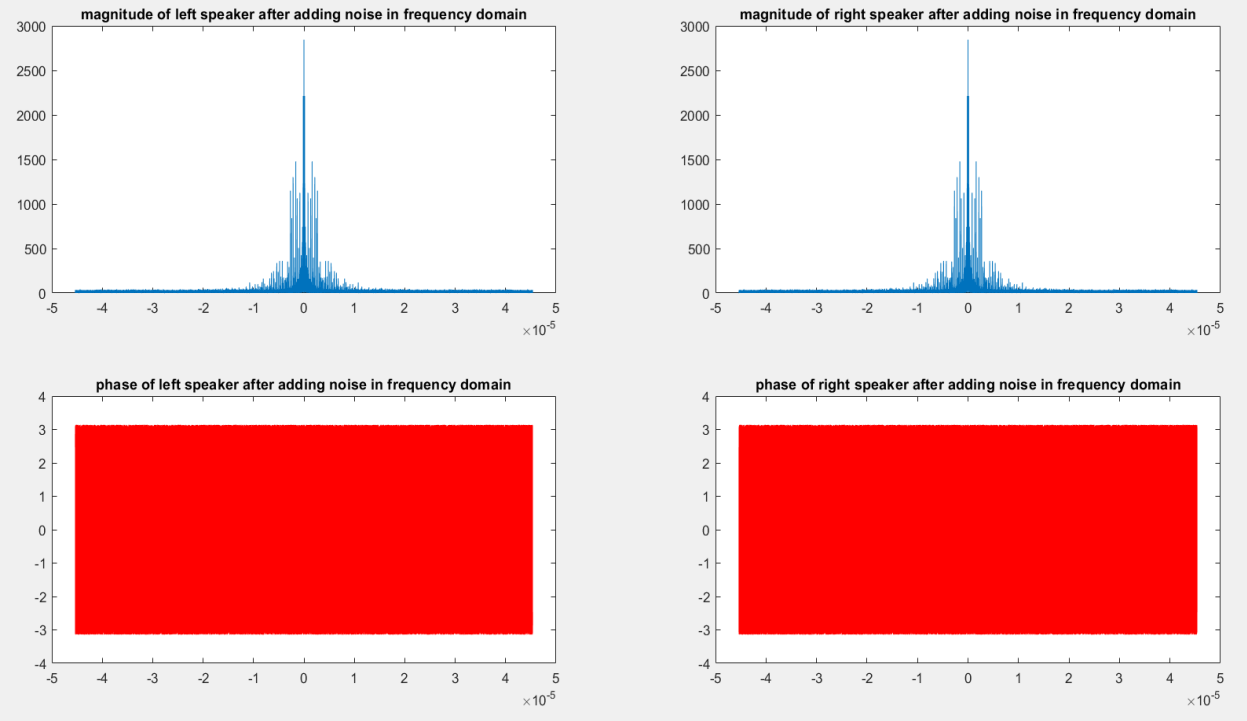
The output after convolution with time



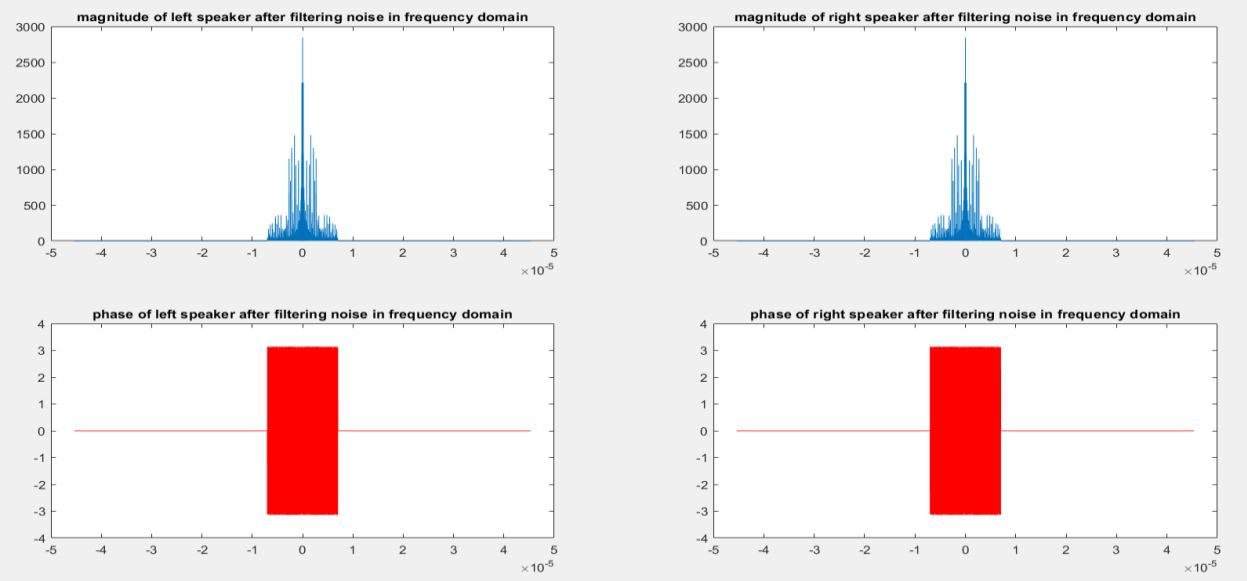
The output after noise with time



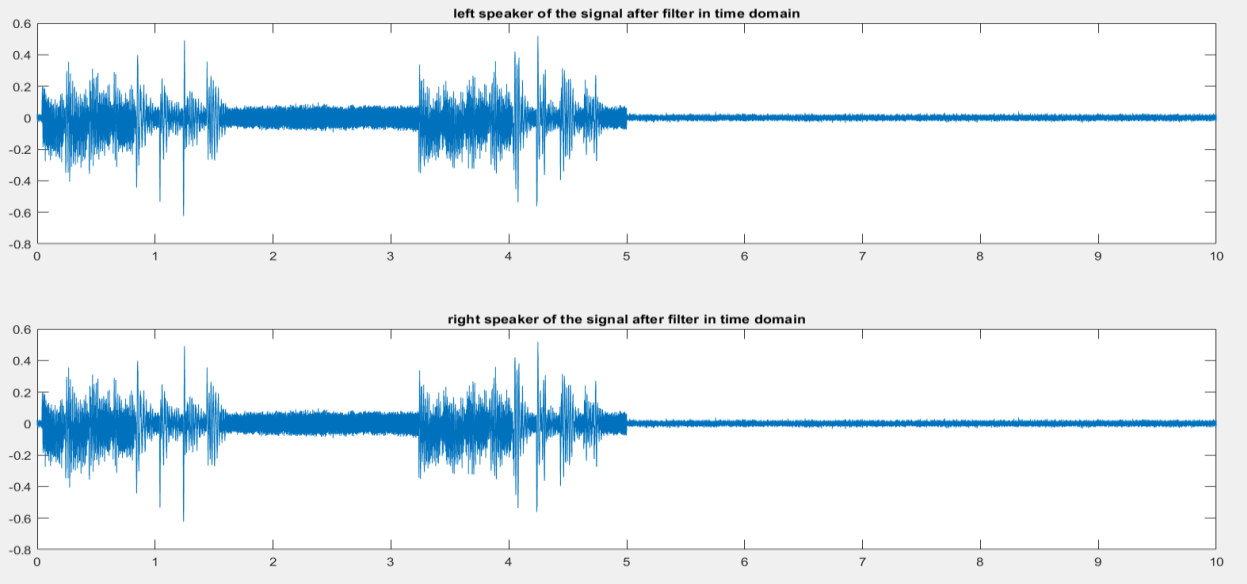
The magnitude and the phase after noise



The magnitude and the phase after filter

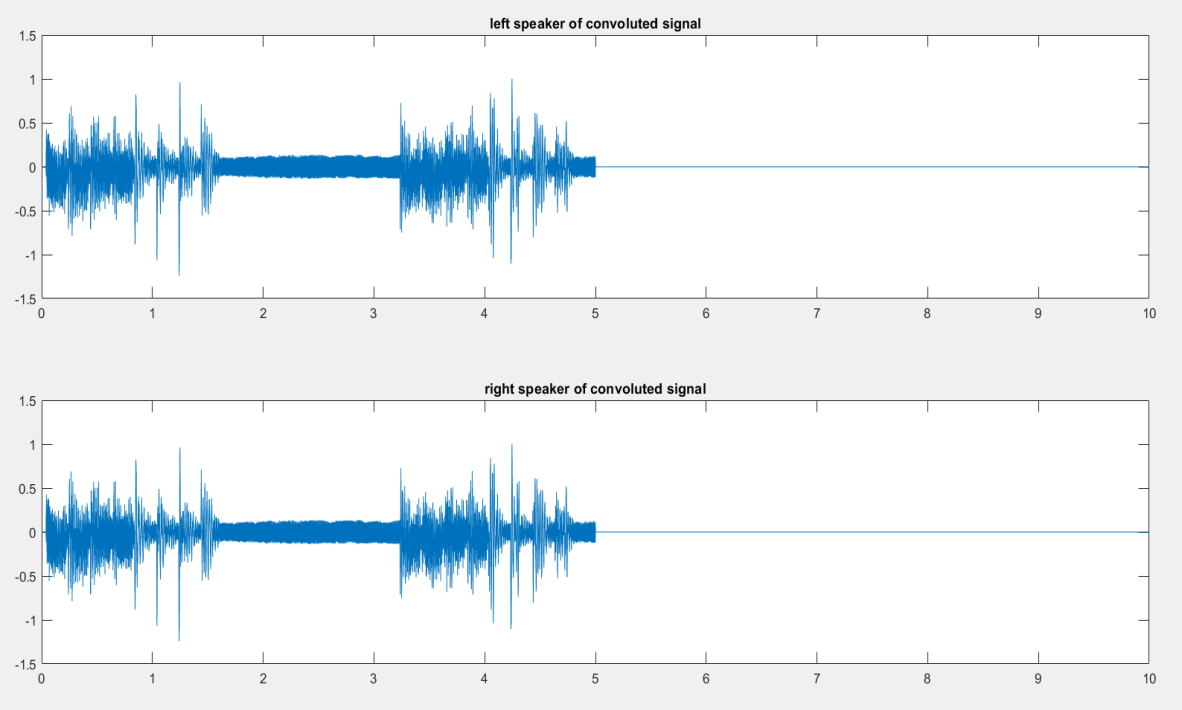


The output after filter with time

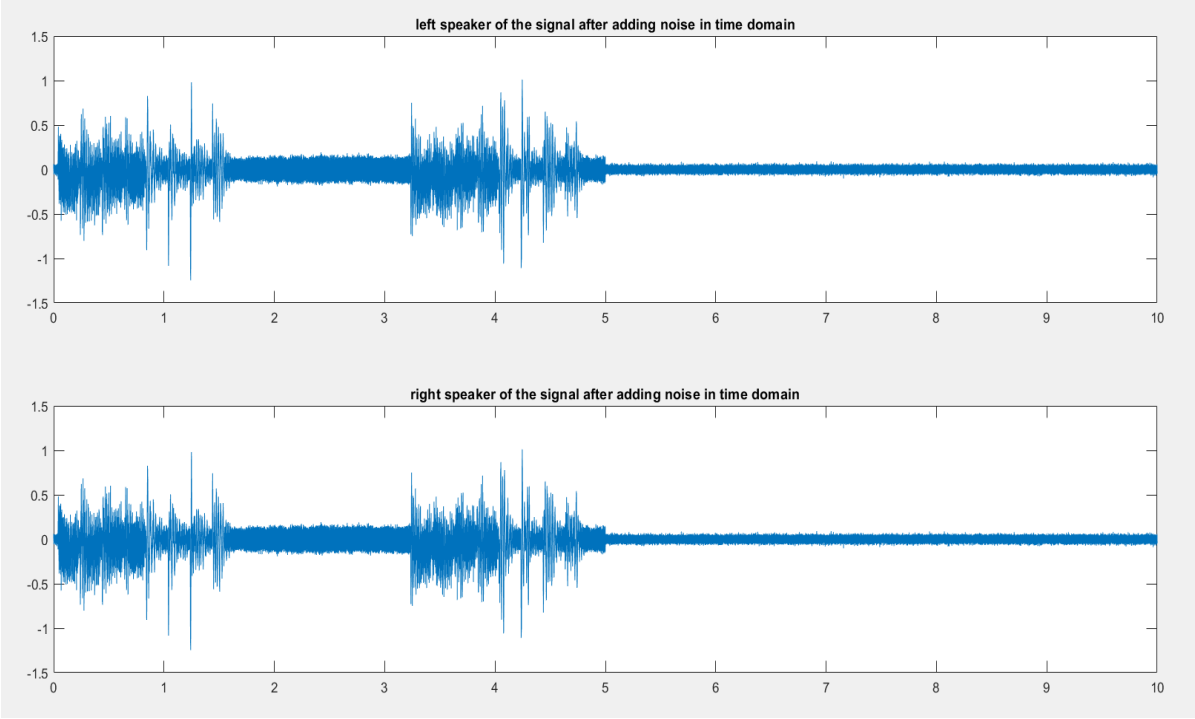


Example (2) the exp(-2pi*5000t) channel output figures

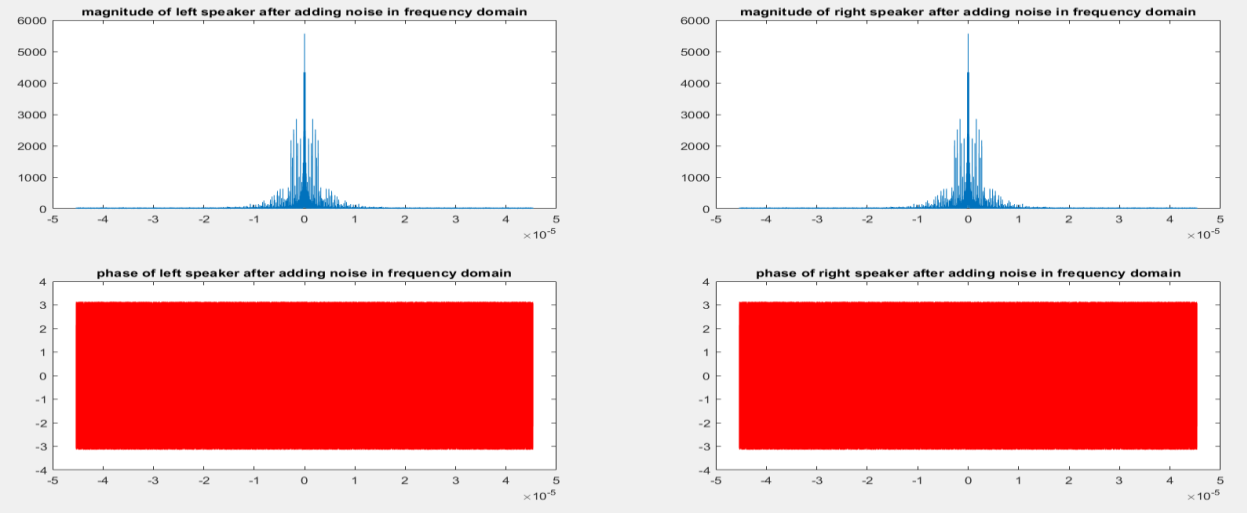
The output after convolution with time



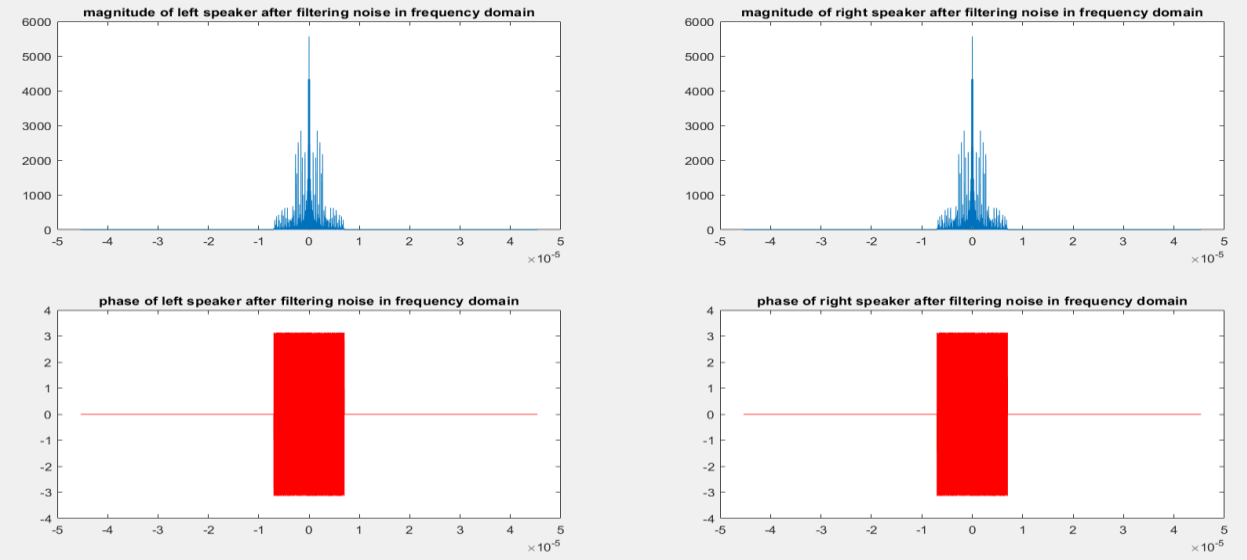
The output after noise with time



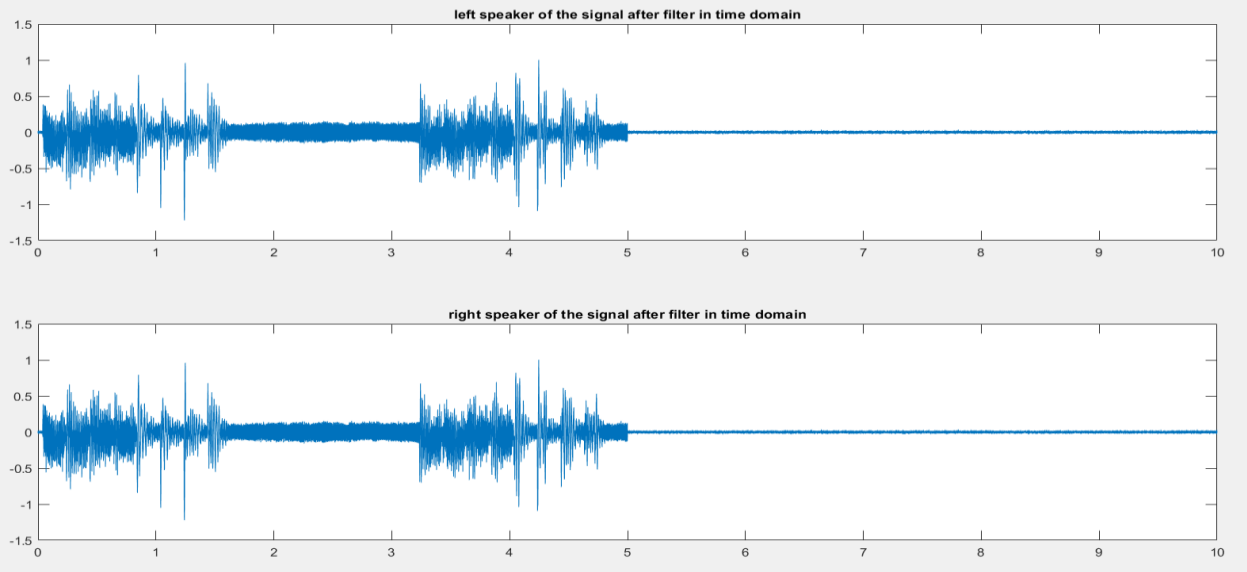
The magnitude and the phase after noise



The magnitude and the phase after filter

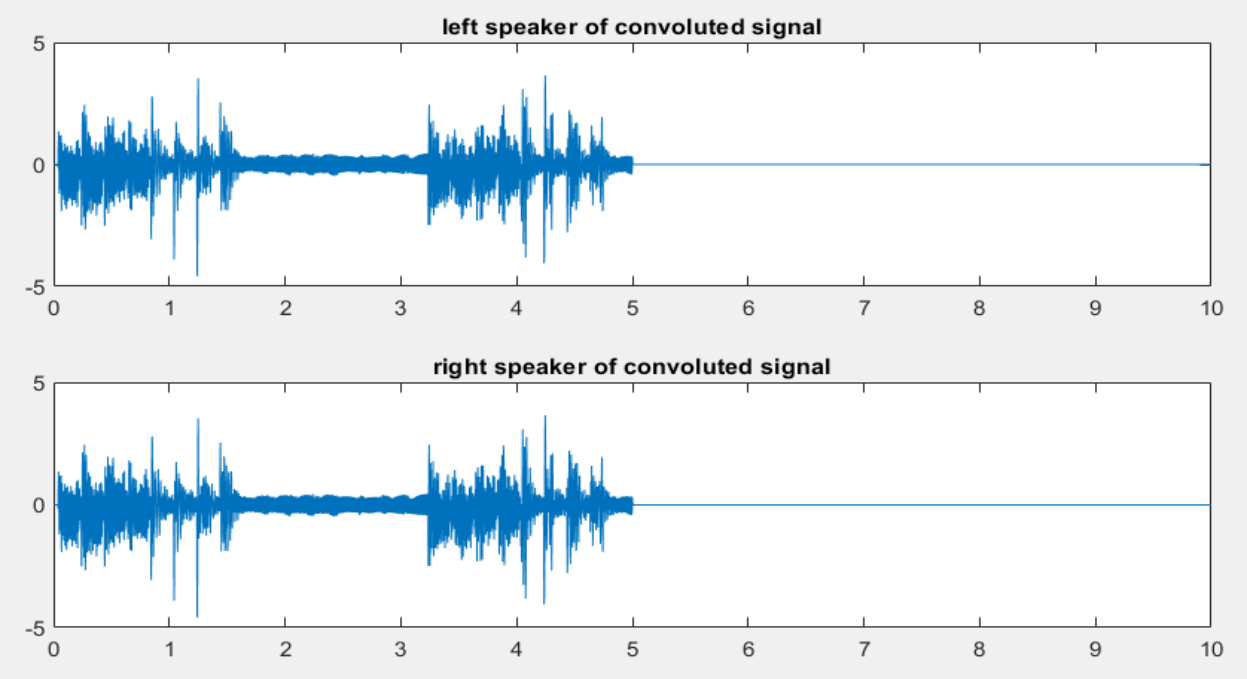


The output after filter with time

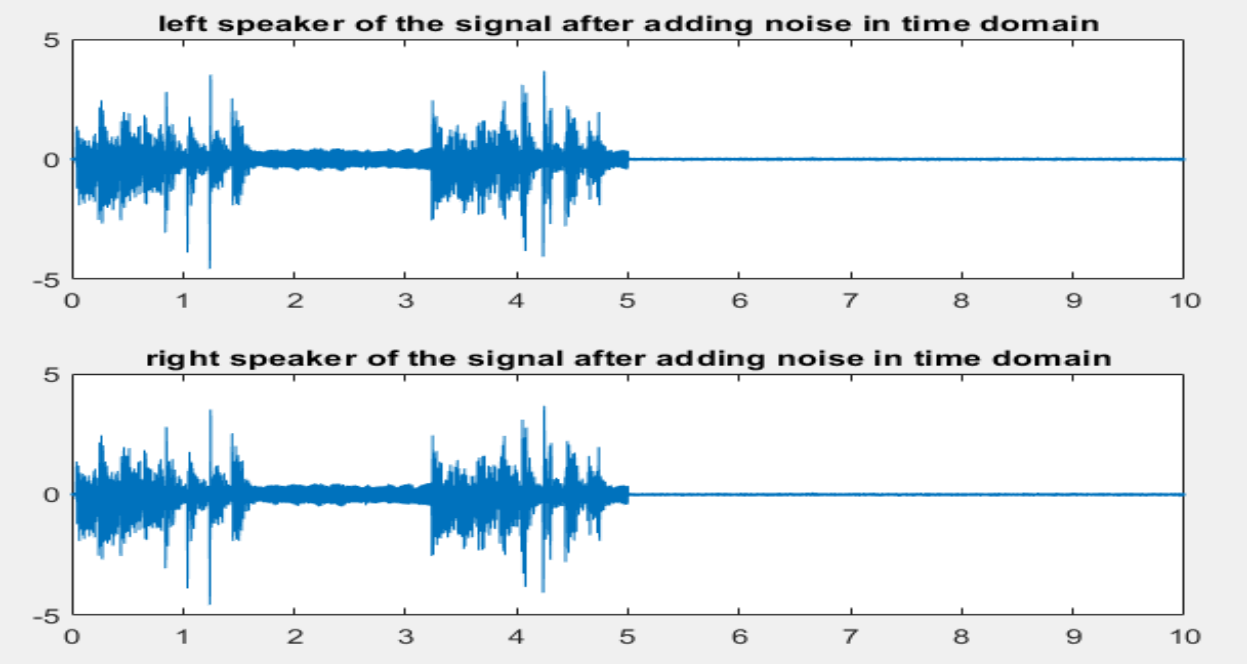


Example (3) exp(-2pi*1000t) channel output figures

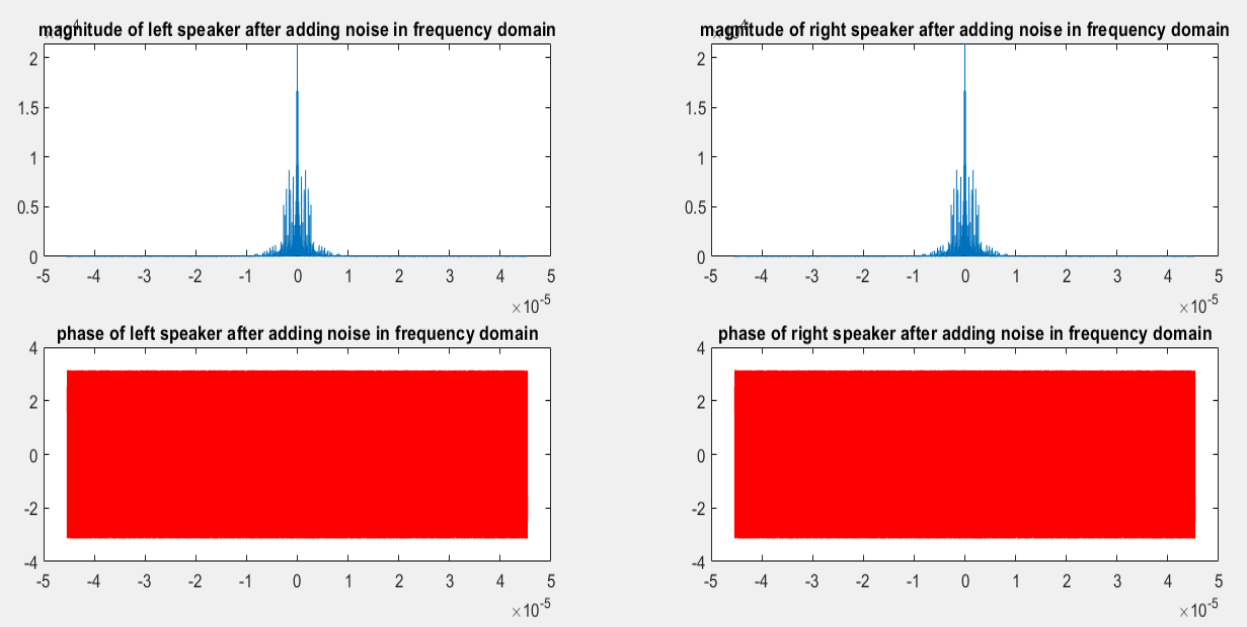
The output after convolution with time



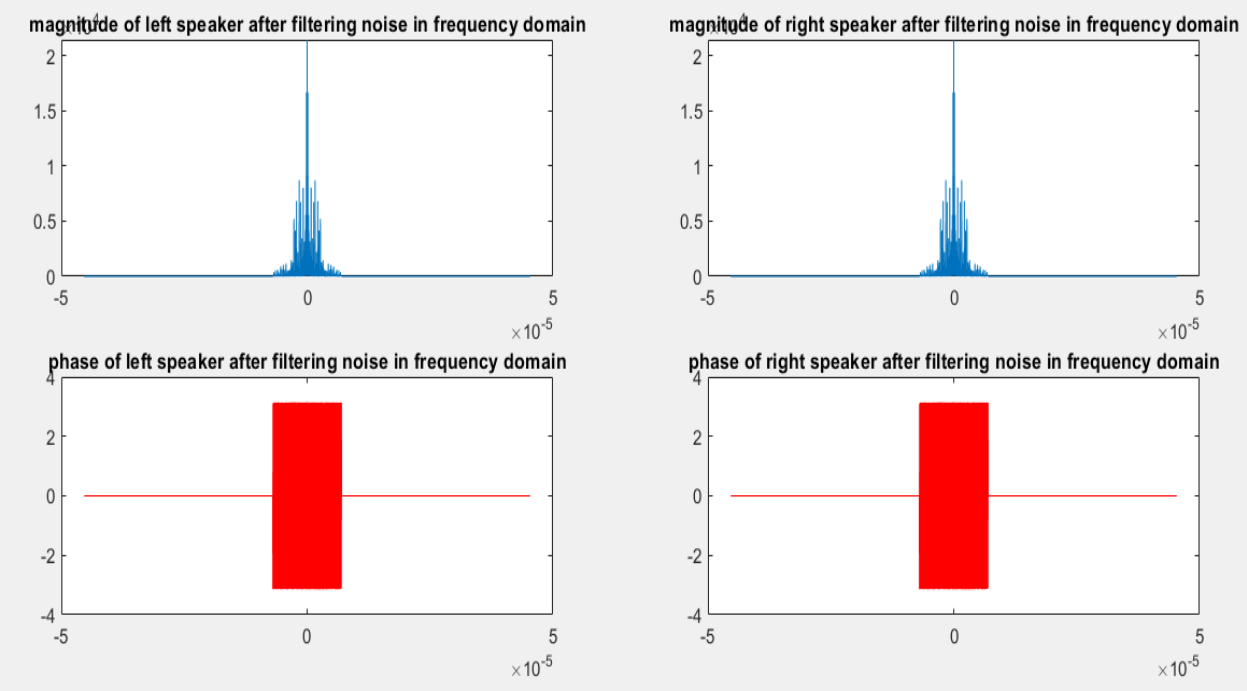
The output after noise with time



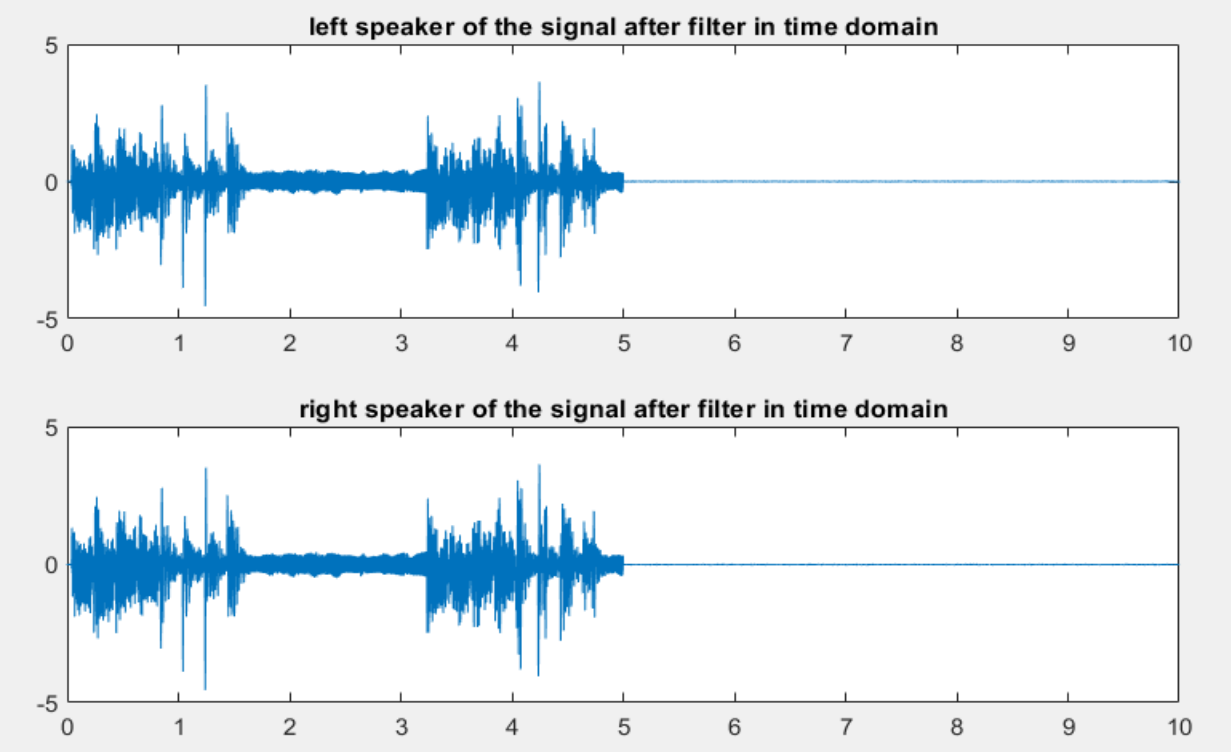
The magnitude and the phase after noise



The magnitude and the phase after filter

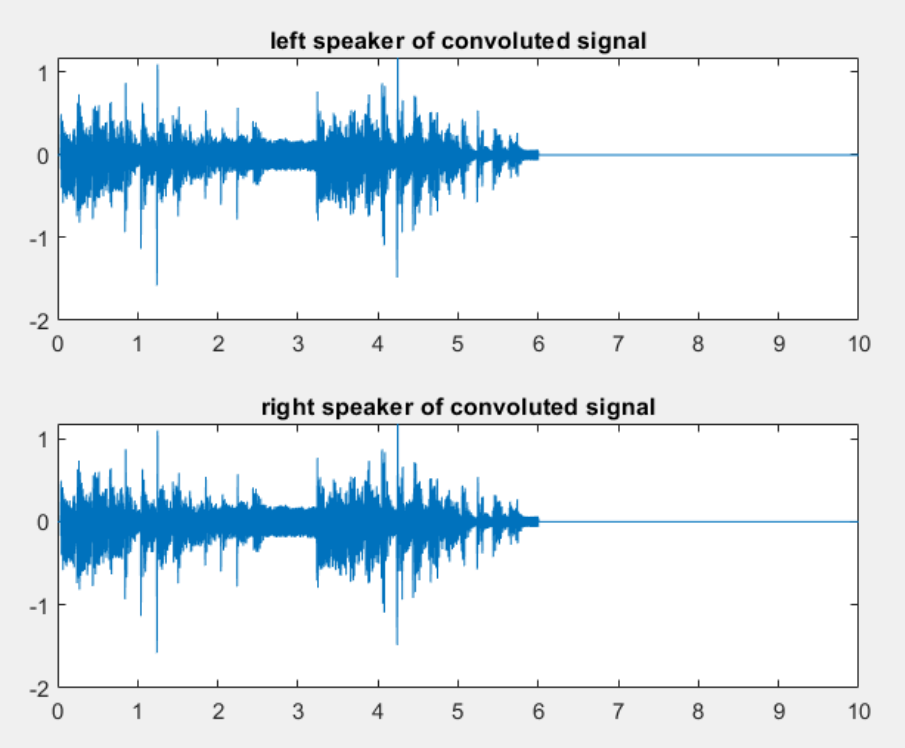


The output after filter with time

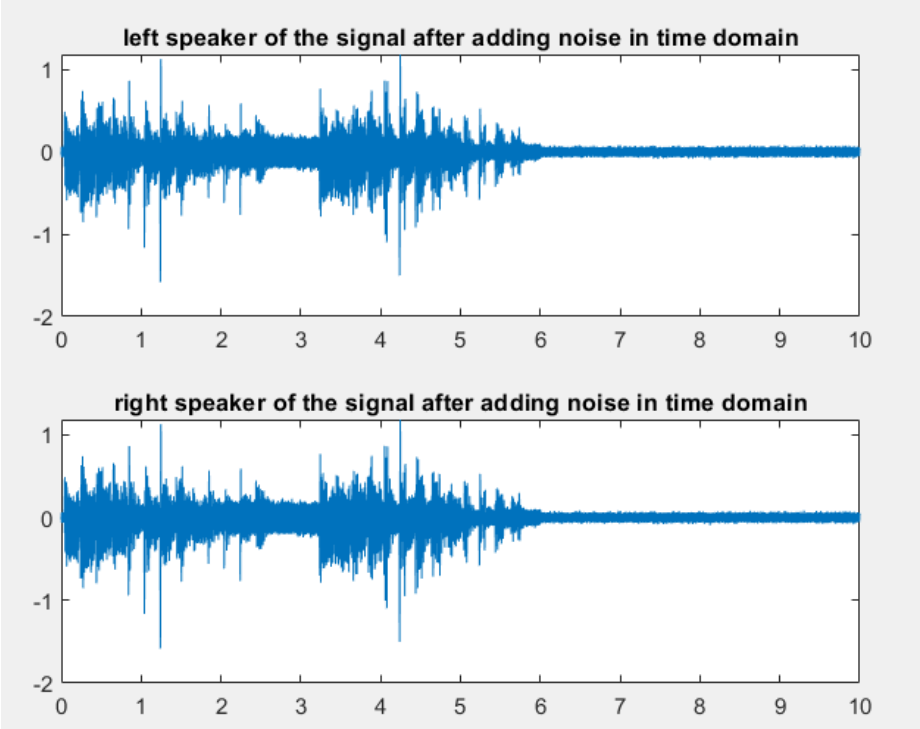


Example (4) the two deltas channel output figures

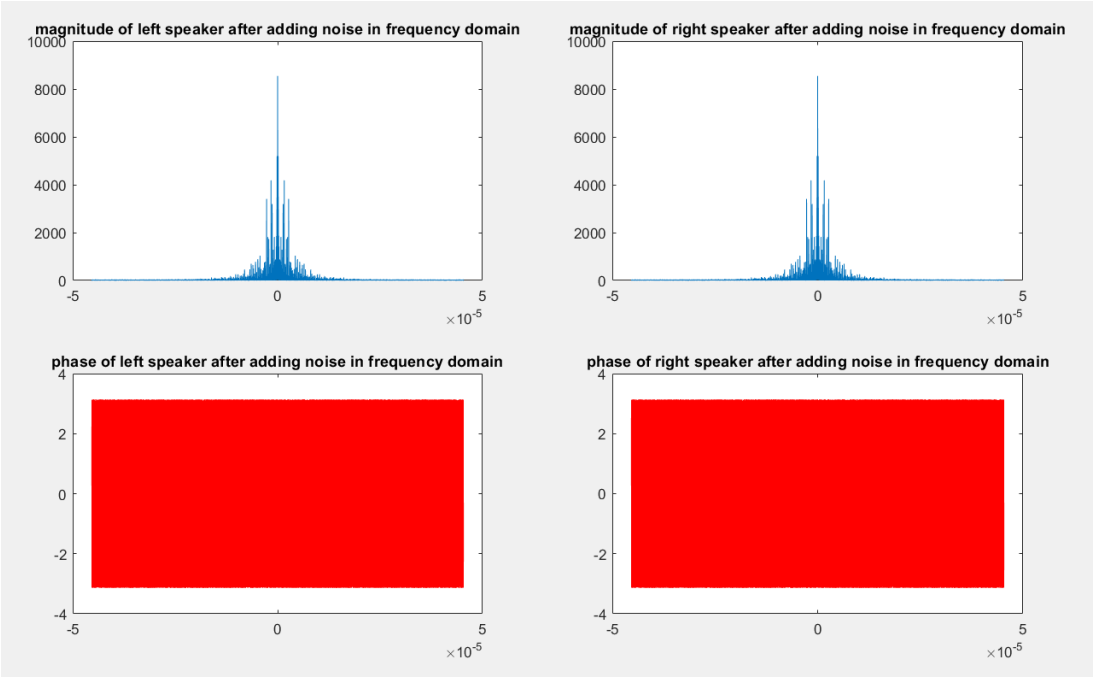
The output after convolution with time



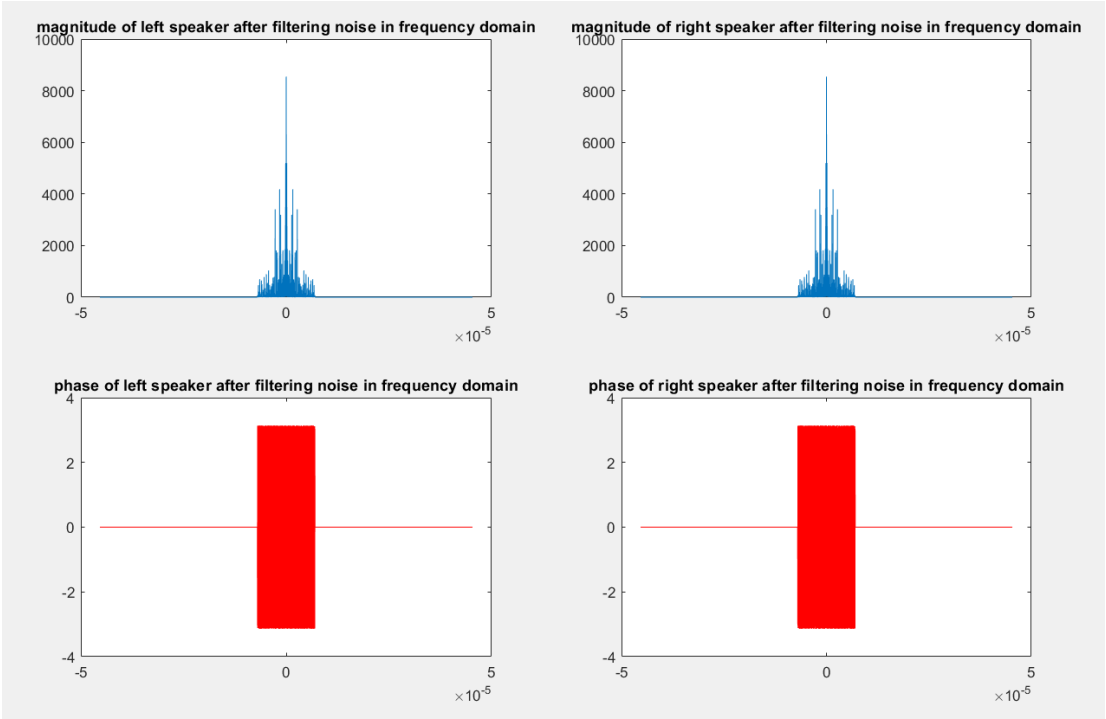
The output after noise with time



The magnitude and the phase after noise



The magnitude and the phase after filter



The output after filter with time