Alexandra University
Faculty of engineering
Communication program
1st term 2022/2023

جامعة الإسكندرية
كلية هندسة
البرنامج اتصالات
الفصل الدراسي الاول

Analog Communication **الاتصالات التناظرية**

(EEC 381)

Project

MATLAB Assignment

Section: 7

Department: Communication and Electronics Engineering

Level: 3

#	الاسم	الرقم الجامعي
1	محمد رزق امين محمد	19016365
2	هيثم احمد شعبان	19016856
3	مروان محمد احمد ابو العلا	19016615
4	ساجد وائل رضوان مرسي	19015744
5	مصطفي سيد احمد طه احمد	١٩٠١٦٦٦٠

Experiment One

EXPERIMENT ONE: DOUBLE SIDEBAND MODULATION

3.1 INTRODUCTION

Double Sideband modulation is the easiest and most direct type of analog modulation. In this scheme, the modulated signal is obtained using a direct multiplication of the modulating signal (i.e. the message) by a cosine carrier. This multiplication results in shifting the entire spectrum of the message to a center frequency defined by the carrier frequency. The modulation is said to be double sideband transmitted carrier (DSB-TC) when the carrier is transmitted along the modulation term. If the carrier term is omitted, the modulation is termed double sideband suppressed carrier (DSB-SC). DSB-TC has a significant advantage in the receiver design (i.e. the envelop detector). Also transmitting the carrier independently enables us to extract useful information such as the carrier frequency which can be helpful for carrier synchronization. However, the DSB-TC loses to the other variant (i.e. the SC) in terms of power efficiency.

3.2 AIM

In this experiment, you're required to achieve the following:

1. Get familiar with the concept of DSB modulation, and its parameters. 2. Study the performance of the DS modulation.

3. Examine different detectors (coherent detector, envelope detector). 4. Study the performance of coherent detection in the presence of frequency or phase mismatch.

3.3 PROCEDURE

1. Use Matlab to read the attached audio file, which has a sampling frequency Fs= 48 KHz. Find the spectrum of this signal (the signal in frequency domain). [audioread, fft , fftshift , plot]

2. Using an ideal Filter, remove all frequencies greater than 4 KHz.

3. Obtain the filtered signal in time domain and frequency domain, this is a band limited signal of BW=4 KHz. [ifftshift ,ifft]

4. sound the filtered audio signal (make sure that there is only a small error in the filtered signal) [sound]

5. Modulate the carrier with the filtered signal you obtained, you are required to generate both types of modulation (DSB-TC and DSB-SC). Choose a carrier frequency of 100 KHz. For the DSB-TC take the DC bias added to message before modulation to be twice the maximum of the message (modulation index =0.5 in this case). Note: You will also need to increase the sampling frequency of the filtered audio signal, the sampling frequency must be at least 2 times the carrier frequency, In this simulation use Fs = 5 Fc . [resample] You have to sketch the modulated signal of both DSB-TC & DSB-SC in frequency domain.

6. For both types of modulations (DSB-SC & DSB-TC), use envelop detector to receive the message (assume no noise). Note: to obtain the envelope you can use the following MATLAB command. $envelope = abs(hilbert(modulated\ signal))$

7. After the reception of both modulation types using envelope detector, sketch the received signal in time domain, and Play the received signal back (Note: to sound signal after demodulation process you have to decrease the sampling frequency again). What observation can you make of this or which type of modulation the envelope detector can be used with? For DSB-SC, perform steps 9-11.

8. Use coherent detection to receive the modulated signal with SNR=0, 10, 30dB then sound the received signals and plot them in both time and frequency domain.

9. Repeat the coherent detection with frequency error, F=100.1 KHz instead of 100 KHz and Find the error. Do you have a name for this phenomenon?

10. Repeat the coherent detection with phase error = 200 .

3.4 USEFUL COMMANDS

audioread,fft, fftshift, ifft, ifftshift , plot, awgn, resample, sound, hilbert, abs, max

Code:

Audio is read using audioread built in function in MATLAB that return two important variables: the message and sampling frequency. Signal is calculated in frequency domain using fft then it is normalized using fftshift. Then we plot it in frequency and time domain.

```
% read audio message
[m, fs] = audioread('eric.wav');
sound(m, fs);

M = fftshift(fft(m));
Fvec = linspace(-fs/2,fs/2,length(M));

% Freq domain representation
figure
plot(Fvec, abs(M));
title('Freq domain representation of message');

% Time domain representation
t=linspace(0, length(m)/fs, length(m));
figure
plot(t, m);
title('Time domain representation of message');
```

Signal is later filtered by zeroing samples that represent frequencies out of -4khz to 4khz as shown in this code:

The filtered signal is then plotted in both time and frequency domain as demanded.

```
% filterring frequencies above 4KHz
sample_per_hertz = (length(m)/fs);
M (1 : round(sample_per_hertz*(fs/2 - 4*10^3))) = 0;
M (round(sample_per_hertz*(fs/2 + 4*10^3))+ 1 : end) = 0;

% Freq domain representation after filtering
figure
plot(Fvec, abs(M));
title('Filtered message in freq domain')

% Time domain representation after filtering
m = real(ifft(ifftshift(M)));
t = linspace(0, length(m)/fs, length(m));
figure
plot(t, m);
title('Filtered message in time domain');
```

We use function DSB_Mod which will Modulate and demodulate the signal, the parameters are the message , the sampling frequency, DC _coeff , and the type whether it's SC or TC.

we generate the carrier frequency as 100000 and the sampling frequency needed to be 5 times the carrier frequency , so we resample the message by the new rate.

```
function DSB_Mod(m, fs, DC_coeff, type)
%------------------------TX DSC-----------------------
fc = 100000;
fs_new = 5*fc;
f m resampled = resample(m, fs new, fs);
```

Then we generate the carrier with new t convinent to the new sampling rate,

DC_coeff is the reciprocal of modulation index which tends to be infinity in DSB_SC , so DC_coeff will be zero, so A will be also zero ,and now we are ready to generate the transmitted message , we multiply the message by the carrier, and make it ready to display in freq domain.

```
t = linspace(0, length(f_m_resampled)/fs_new, length(f_m_resampled)
carrier = cos(2*pi*fc*t);
A = DC_coeff * max(f_m_resampled);
s_tx = (A + f_m_resampled) .* (carrier');

%Freq domain representation of transmitted signal
S_tx = fftshift(fft(s_tx));
Fvec = linspace(-fs_new/2, fs_new/2, length(s_tx));
figure
plot(Fvec, abs(S_tx));
title('DSB-'+type+' transmitted signal in freq domain');
```

We use envelop detector to receive the message (assume no noise). Then we display the envelope in time domain.

```
%------------------------RX DSB-------------------------
envelop = abs(hilbert(s_tx));
% Time domain representation of received signal
figure
plot(t, envelop);
title('DSB-'+type+' received (envelop) signal in time domain');
```

The previous function is applied for the first requirement

```
DC_coeff = 0;
DSB_Mod(m, fs, DC_coeff, "SC");
```

Also we use this function for DSB-TC only the difference is that A is not zero.

```
mod_index = 0.5;
DC_coeff = 1 / mod_index;
DSB_Mod(m, fs, DC_coeff, "TC");
```

Another two functions are used in case of SNR and phase error with DSB-SC .

DSB_Mod_coh & DSB_SC_Coherent

DSB_Mod_coh is used for preparing the carrier signal and transmitted signal as what we do previously.

```
%------------------------TX DSC-SC-------------------------
fc = 100000;
fs_new = 5*fc;
f_m_resampled = resample(m, fs_new, fs);
t = linspace(0, length(f_m_resampled)/fs_new, length(f_m_resampled)
carrier = cos(2*pi*fc*t);
A = DC_coeff * max(f_m_resampled);
s_tx = (A + f_m_resampled) .* (carrier');
```

Then we check if the type of modulation is DSB-SC to go to the next function after adding the phase error to the phase of carrier.

```
%--------------------TX DSC-SC with noise--------------------
if (type == "SC")
    carrier = cos(2*pi*fc*t + phase_error*(pi/180));
    DSB_SC_coherent(s_tx, fs, fs_new, carrier, SNR, fc + freq_error, phase_error);
end
```

We use awgn to add guassian noise to the transmitted signal

Then we obtain the received message after multiplying it by the carrier with new phase error.

```
t = linspace(0, length(s_tx)/fs_new, length(s_tx));
noisy_s_tx = awgn(s_tx, SNR);
s_rx = (noisy_s_tx) .* (carrier');
```

Then we display the RX signal in both time and frequency domain

```
S_rx = fftshift(fft(s_rx));
Fvec = linspace(-fs_new/2, fs_new/2, length(s_rx));
figure
plot(Fvec, abs(S_rx));
title("DSB-SC(coherent)signal in freq with: SNR= "+SNR+", Fc= "+fc+", phase error= "+phase_error);

figure
plot(t, s_rx);
title("DSB-SC(coherent)signal in time with: SNR= "+SNR+", Fc= "+fc+", phase error= "+phase_error);
```

Finally we use LPF that represent frequencies out of -4khz to 4khz.

Then display it in freq domain.

```
% LPF
sample_per_hertz = (length(s_rx)/fs_new);
S_rx (1 : round(sample_per_hertz*(fs_new/2 - 4*10^3))) = 0;
S_rx (round(sample_per_hertz*(fs_new/2 + 4*10^3))+ 1 : end) = 0;

figure
plot(Fvec, abs(S_rx));
title("DSB-SC(coherent)LPF signal in freq with: SNR= "+SNR+", Fc="+fc+", phase error= "+phase_error);

LPF_s_rx = real(ifft(ifftshift(S_rx)));
figure
plot(t, LPF_s_rx);
title("DSB-SC(coherent)LPF signal in freq with: SNR= "+SNR+", Fc="+fc+", phase error= "+phase_error);
```

And this is the requirements for SNR and Phase error.

```
%coherent detector
%changing SNR with freq error = 0 and phase error = 0
freq_error = 0;
phase_error = 0;
DC_coeff = 0;
DSB_Mod_coh(m, fs, DC_coeff, freq_error, 0, phase_error, "SC");
DSB_Mod_coh(m, fs, DC_coeff, freq_error, 10, phase_error, "SC");
DSB_Mod_coh(m, fs, DC_coeff, freq_error, 30, phase_error, "SC");

%changing SNR with freq error = 100 and phase error = 0
freq_error = 100;
phase_error = 0;
DC_coeff = 0;
DSB_Mod_coh(m, fs, DC_coeff, freq_error, 0, phase_error, "SC");
DSB_Mod_coh(m, fs, DC_coeff, freq_error, 10, phase_error, "SC");
DSB_Mod_coh(m, fs, DC_coeff, freq_error, 30, phase_error, "SC");

%changing SNR with freq error = 0 and phase error = 20
freq_error = 0;
phase_error = 20;
DC_coeff = 0;
DSB_Mod_coh(m, fs, DC_coeff, freq_error, 0, phase_error, "SC");
DSB_Mod_coh(m, fs, DC_coeff, freq_error, 10, phase_error, "SC");
DSB_Mod_coh(m, fs, DC_coeff, freq_error, 30, phase_error, "SC");
```

Questions:

-which type of modulation the envelope detector can be used with?

We can use envelope detector in case of under modulation (m<1)
or critical modulation (m=1).

- Do you have a name for the phenomenon of frequency error?

The distortion due to frequency error is called Beat effect.

Results:

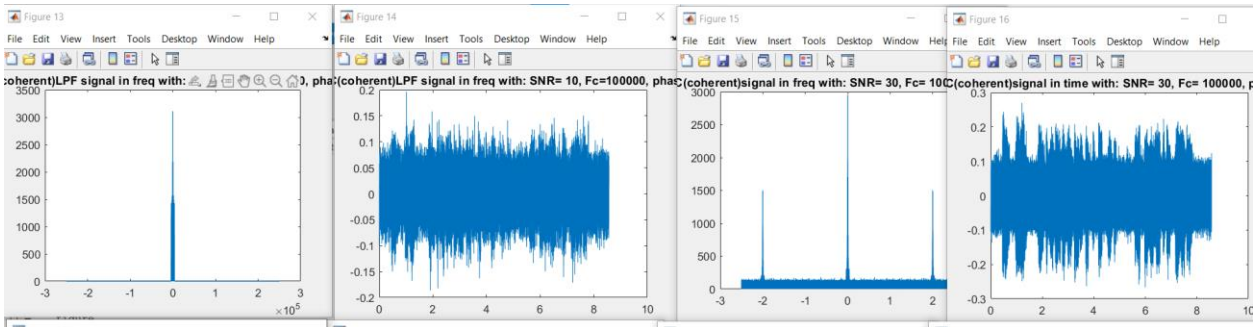


Figure 13
File Edit View Insert Tools Desktop Window Help
Figure 14
File Edit View Insert Tools Desktop Window Help
(coherent)LPF signal in freq with: SNR= 10, Fc=100000, phas
Figure 15
File Edit View Insert Tools Desktop Window Help
(coherent)signal in freq with: SNR= 30, Fc= 10
Figure 16
File Edit View Insert Tools Desktop Window Help
(coherent)signal in time with: SNR= 30, Fc= 100000, p

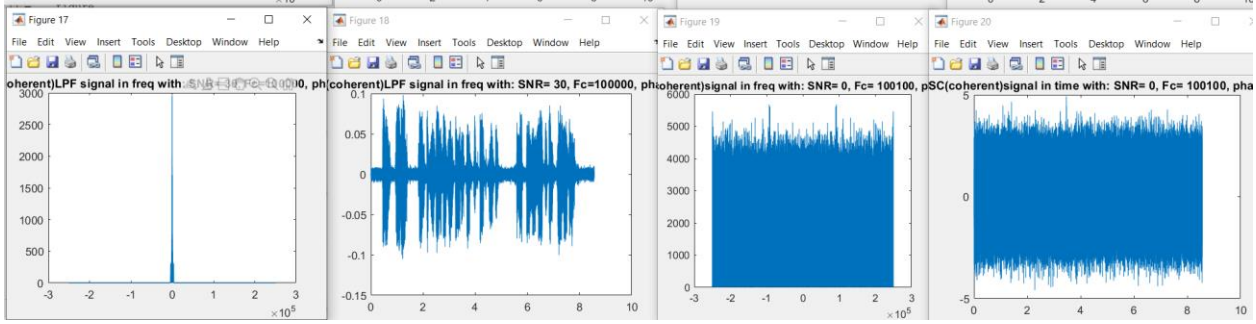


Figure 17
File Edit View Insert Tools Desktop Window Help
Figure 18
File Edit View Insert Tools Desktop Window Help
(coherent)LPF signal in freq with: SNR= 30, Fc=100000, ph
Figure 19
File Edit View Insert Tools Desktop Window Help
oherent)signal in freq with: SNR= 0, Fc= 100100, p
Figure 20
File Edit View Insert Tools Desktop Window Help
SC(coherent)signal in time with: SNR= 0, Fc= 100100, pha

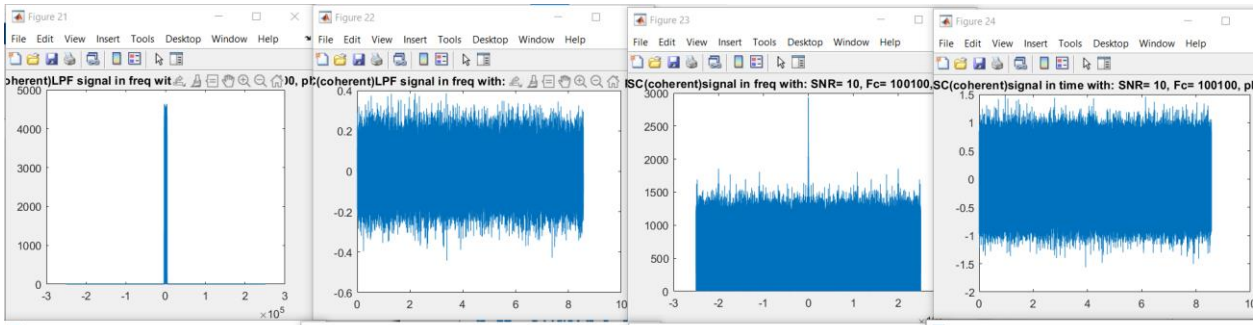


Figure 21
File Edit View Insert Tools Desktop Window Help
Figure 22
File Edit View Insert Tools Desktop Window Help
(coherent)LPF signal in freq with:
Figure 23
File Edit View Insert Tools Desktop Window Help
SC(coherent)signal in freq with: SNR= 10, Fc= 100100,
Figure 24
File Edit View Insert Tools Desktop Window Help
SC(coherent)signal in time with: SNR= 10, Fc= 100100, p

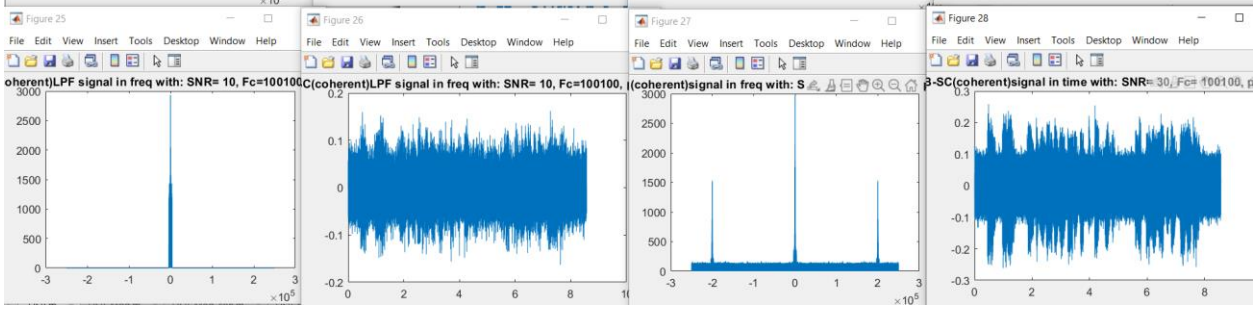


Figure 25
File Edit View Insert Tools Desktop Window Help
Figure 26
File Edit View Insert Tools Desktop Window Help
C(coherent)LPF signal in freq with: SNR= 10, Fc=100100,
Figure 27
File Edit View Insert Tools Desktop Window Help
(coherent)signal in freq with: S
Figure 28
File Edit View Insert Tools Desktop Window Help
-SC(coherent)signal in time with: SNR= 30, Fc= 100100,

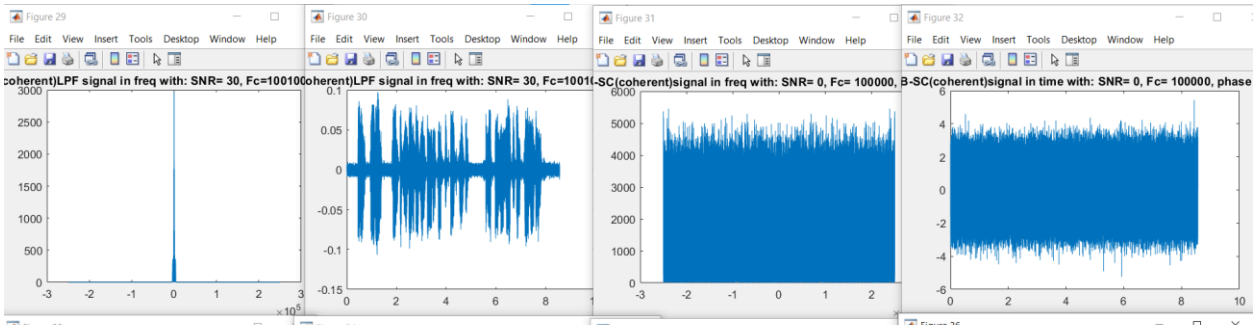


Figure 29
File Edit View Insert Tools Desktop Window Help
Figure 30
File Edit View Insert Tools Desktop Window Help
Figure 31
File Edit View Insert Tools Desktop Window Help
Figure 32
File Edit View Insert Tools Desktop Window Help
-SC(coherent)signal in time with: SNR= 0, Fc= 100000, phase

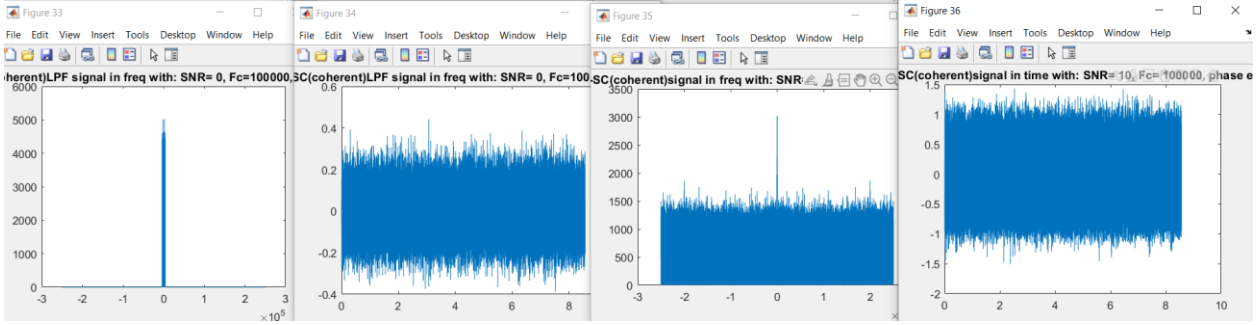
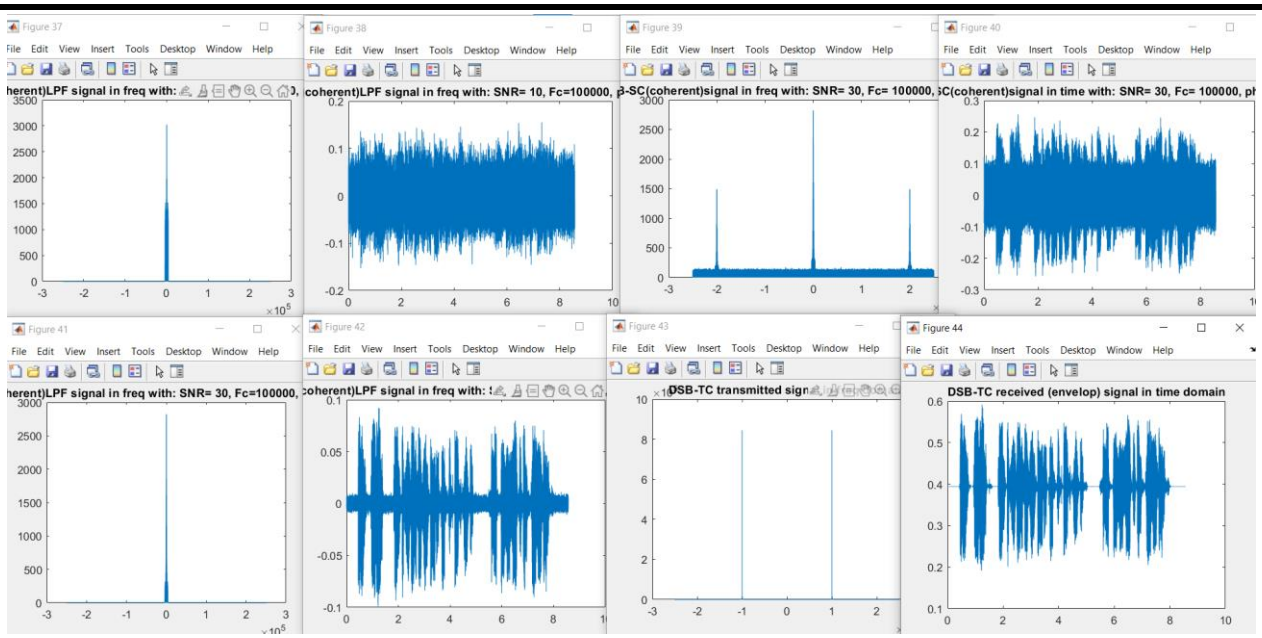


Figure 33
File Edit View Insert Tools Desktop Window Help
Figure 34
File Edit View Insert Tools Desktop Window Help
Figure 35
File Edit View Insert Tools Desktop Window Help
Figure 36
File Edit View Insert Tools Desktop Window Help
SC(coherent)signal in time with: SNR= 10, Fc= 100000, phase e



Experiment Two: Single Sideband Modulation:

INTRODUCTION

The bandwidth inefficiency stemming from the DSB transmission was the main reason why the single sideband (SSB) was developed. In SSB modulation, the bandwidth required for band pass transmission is equal to the bandwidth of that of the baseband. In other words, the band requirement for SSB is halved with respect to that of DSB which requires twice the baseband bandwidth. This reduction in transmission bandwidth is possible since the sideband are replicated twice over the positive and negative frequencies. However, the bandwidth reduction doesn't come entirely free. In fact, SSB suffers from several disadvantages that we hope to cover in the following simulation

PROCEDURE

1. Use Matlab to read the attached audio file which has a sampling frequency Fs= 48 KHz. Find the spectrum of this signal. (same as previous experiment)

2. Using an ideal Filter, remove all frequencies greater than 4 KHz. (same as previous experiment)

3. Obtain the filtered signal in time domain, this is a band limited signal of BW=4KHz. You could play the sound back, to make sure only small distortion was introduced. (same as previous experiment

4. Generate a DSB-SC modulated signal and plot its spectrum. Choose the carrier frequency to be 100kHz. Remember to set the sampling frequency to Five times the carrier frequency Fs = 5Fc. (same as previous experiment)

5. Obtain the SSB by filtering out the USB (we need to get LSB only) of the DSB-SC modulated signal using an ideal filter then Plot the spectrum again.

6. Use coherent detection with no noise interference to get the received signal (to demodulate the SSB-SC) and play the file back also sketch the received waveform and spectrum.

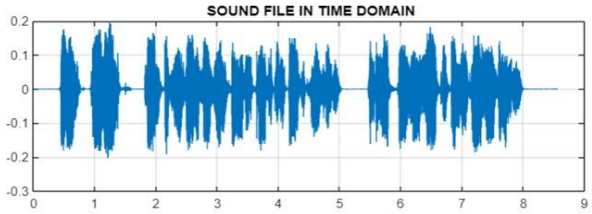
7. Repeat steps 5 and 6, only this time. Use a practical 4th order Butterworth filter. [butter, filter]

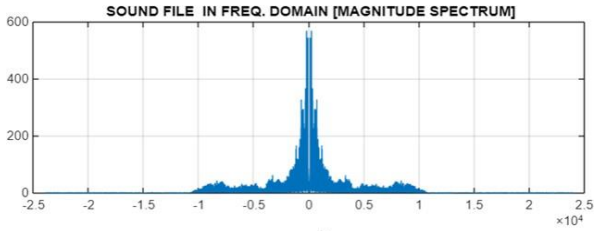
8. For the ideal filter case, get the received signal again but when noise is added to SSB-SC with SNR = 0, 10, and 30 also play the received sound and sketch the received waveform/spectrum in each case. [awgn]

9. For the ideal filter case, generate a SSB-TC. As with experiment one, set the DC bias to twice the maximum of the message. Use envelope detector to demodulate the message (without noise) . Play back the received message and sketch its waveform

- Code with Plots:

```
% 1
[y,fs] = audioread('eric.wav');
% sound(y,fs);
% pause(length(y)/fs); % wait until finishing sound file
% Duration = number of samles[length(y)]/sampling frequency[fs]
t1 l= linspace(0,length(y)/fs,length(y));
yf = fftshift(fft(y));
yfmag = abs(yf);
fvac = linspace(-fs/2,fs/2,length(yf));
figure; subplot(2,1,1)
plot(t1,y);

title('SOUND FILE IN TIME DOMAIN');
grid on
subplot(2,1,2)
plot(fvac,yfmag);
title('SOUND FILE IN FREQ. DOMAIN
[MAGNITUDE SPECTRUM]')
grid on
```



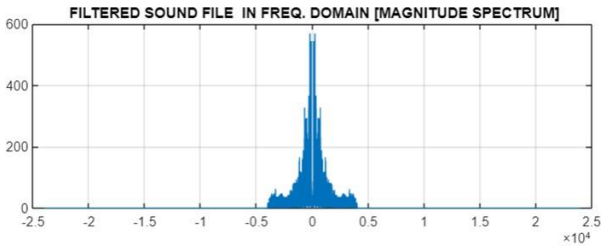


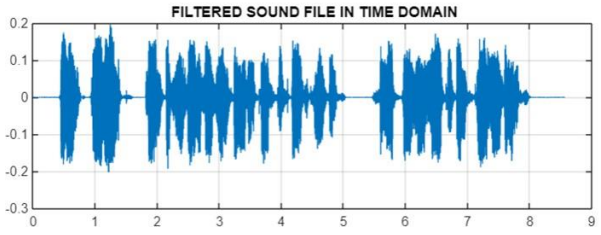
Comment

The first two figures represent the original signal in time domain and frequency domain .

```
% 2
Filter=ones(length(yf),1);
f1=round((-4*1000+fs/2).*(length(yf)/fs));   f2=round((4*1000+fs/2).*(length(yf)/fs));
                                                    Filter([1:f1
f2:end])=0;
yfiltered = yf.* Filter;
yfmagfiltered = abs(yfiltered);
fvacfiltered = linspace(-fs/2,fs/2,length(yfiltered));

% 3
yt = ifft(ifftshift(yfiltered));
t2= linspace(0,length(yt)/fs,length(yt));
figure;
subplot(2,1,1)
plot(fvacfiltered,yfmagfiltered);
title(' FILTERED SOUND FILE IN FREQ. DOMAIN
[MAGNITUDE SPECTRUM]')
grid on
subplot(2,1,2)
plot(t2,yt);
title(' FILTERED SOUND FILE IN TIME DOMAIN');
grid on
yt2 = yt;
ytmag=abs(yt);
%sound(ytmag,fs);
%pause(length(ytmag)/fs);
% wait until finishing sound file
```

The original signal



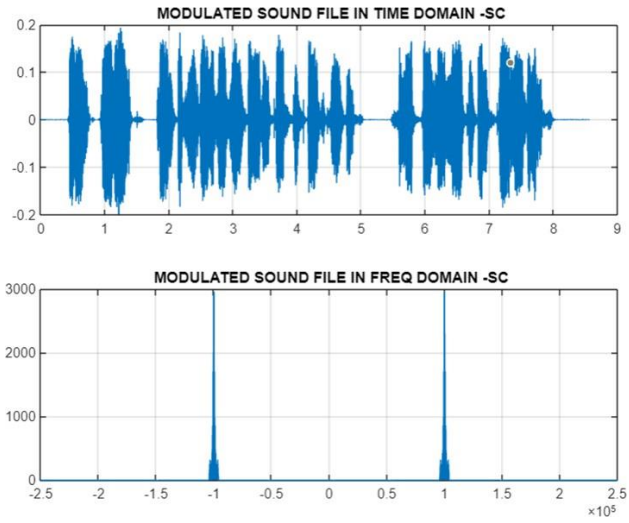


We hear that the sound is having some distortion

Comment

Apply law pass filter to eliminate high
Frequencies to eliminate high frequency using a
Low pass filter from -4khz to 4 khz.

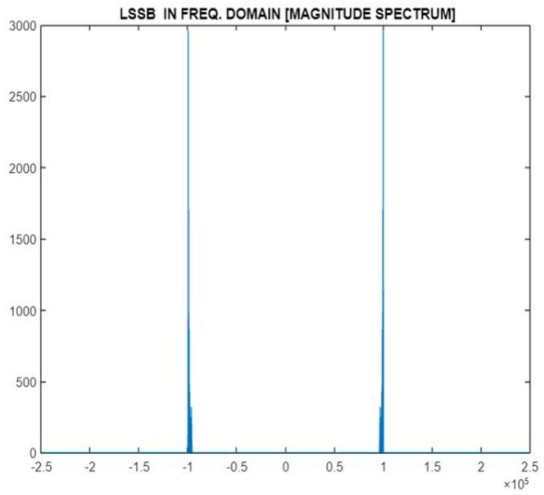
```
% 4
yf2 = fftshift(fft(yt2));
yfmag2 = abs(yf2);
fvac2 = linspace(-fs/2,fs/2,length(yf2));

%      DSB-SC
cs=5*100000;
yt3 = resample(yt2,cs,fs); %resampled message
t3  =  linspace(0,length(yt3)/cs,length(yt3));
carrier = cos(2*pi*100000*t3);
carrier = carrier';
stsc = yt3 .* carrier;
yf3 = fftshift(fft(stsc));
yfmag3 = abs(yf3);
fvac3 = linspace(-cs/2,cs/2,length(yt3));
figure; subplot(2,1,1)
plot(t3,stsc);

title(' MODULATED SOUND FILE IN TIME DOMAIN -SC');
grid on
subplot(2,1,2)
plot(fvac3,yfmag3);
title(' MODULATED SOUND FILE IN FREQ DOMAIN -SC');
grid on
```



Comment

The following figures represent the modulated signal
(Dsb-sc) by multiplying the message by the
Carrier with fc =1000khz so the spectrum of the
Message will be shifted at fc and –fc in this case
It will be shifted at -1000khz and 1000khz

```
% 5
% SSB-SC
lsb_sc_f=yf3;
B = 100000; %filtering the usb
SSBFdem = length(lsb_sc_f)./(cs);
lim = ceil(((cs/2)-B)*SSBFdem);
lsb_sc_f([1:lim  length(lsb_sc_f)-lim+1:length(lsb_sc_f)]
lsb_sc_fmag  = abs(lsb_sc_f) ;
flsbfiltered = linspace(-cs/2,cs/2,length(lsb_sc_f));
figure;
plot(flsbfiltered,lsb_sc_fmag);
title(' LSSB  IN FREQ. DOMAIN [MAGNITUDE SPECTRUM]');
%lsb in time domain
lsb_sc_t = ifft(ifftshift(lsb_sc_f));
```

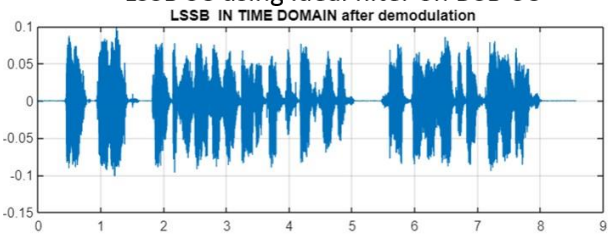


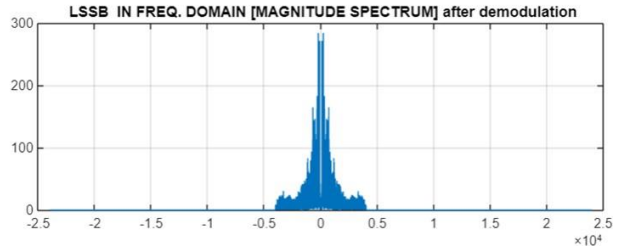
Comment

By using a low pass filter, we can select only Lower sides to reduce the bandwidth

```
% 6
coh_dem=2.*carrier;
lsb_sc_t_dem=lsb_sc_t.*coh_dem; %coherent detection
mfenvtc = fftshift(fft(lsb_sc_t_dem));
B = 4000; %bandwidth of the filter
SSBFdem = length(mfenvtc)./(cs);
lim = ceil(((cs/2)-B)*SSBFdem);
mfenvtc([1:lim  length(mfenvtc)-lim+1:length(mfenvtc)]) = 0;
lsb_sc_t_dem = ifft(ifftshift(mfenvtc));
lsb_sc_t_res = resample(lsb_sc_t_dem,fs,cs); %resample to fs
lsb_sc_f_res = fftshift(fft(lsb_sc_t_res));
flsbdem=linspace(-fs/2,fs/2,length(lsb_sc_f_res));
lsb_sc_f_resmag=abs(lsb_sc_f_res);
lsb_sc_t_resmag = abs(lsb_sc_t_res);
tlsbdem= linspace(0,length(lsb_sc_t_res)/fs,length(lsb_sc_t_res));
figure;
subplot(2,1,1)
plot(tlsbdem,lsb_sc_t_res);
title(' LSSB IN TIME DOMAIN after demodulation');
grid on
subplot(2,1,2)
plot(flsbdem,lsb_sc_f_resmag);
title(' LSSB  IN FREQ. DOMAIN [MAGNITUDE SPECTRUM]
after demodulation');
grid on
% sound(lsb_sc_t_resmag,fs);
% pause(length(lsb_sc_t_resmag)/fs);
% wait until finishing sound file
```

LSSB SC using ideal filter on DSB-SC



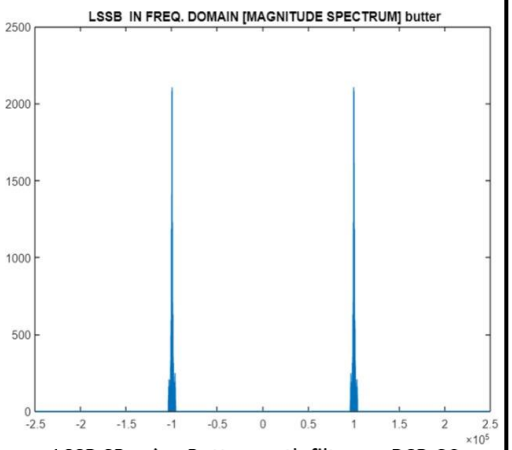


When we play the sound,
we obtain the original signal after filtering.

Comment

Then we use a coherent detector to demodulate the message by multiplying it by same carrier with same frequency
after this we use a low pass filer to select the desired signal.

```
% 7 repeat 5
lsb_sc_f_btr=yf3;
lsb_sc_t_btr = ifft(ifftshift(lsb_sc_f_btr));
lsb_sc_f_btrmag  = abs(lsb_sc_f_btr) ;
[C,B2] = butter(4,100000/(cs/2),'low');
lsb_sc_t_btr_fl=filter(C,B2,lsb_sc_t_btr);
lsb_sc_f_btr_fl= fftshift(fft(lsb_sc_t_btr_fl));
lsb_sc_f_btr_flmag  = abs(lsb_sc_f_btr_fl) ;
flsbfilteredbtr=linspace(-cs/2,cs/2,length(lsb_sc_f_b   tr_fl))
figure;
plot(flsbfilteredbtr,lsb_sc_f_btr_flmag);
title('LSSB  IN FREQ. DOMAIN [MAGNITUDE SPECTRUM] butter');
```

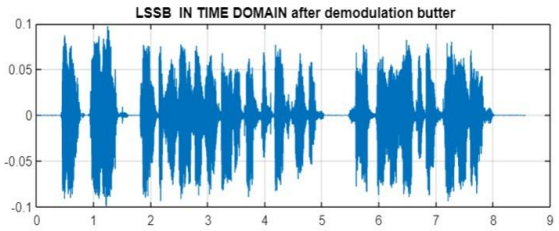


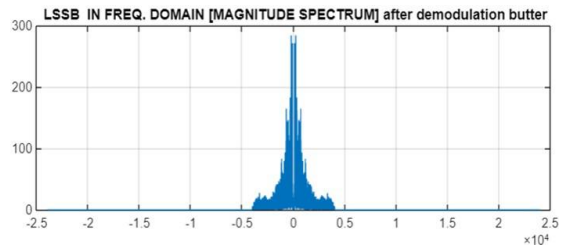
LSSB SB using Butterworth filter on DSB-SC

Comment

we can make a practical 4^{th} order law pass filter (butter worth) we can select the lower side only by making A practical law pass filter with a cut off frequency equal To (fc-fm)the main disadvantage of using a practical filter Is taking a small part from upper side band

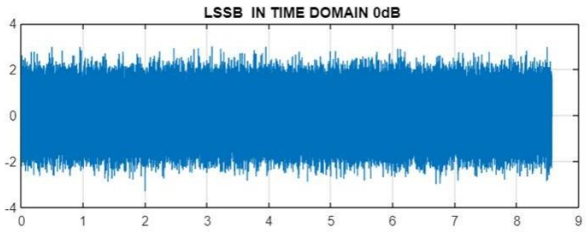
```
% 7 repeat 6
coh_dem=2.*carrier;
lsb_sc_t_dem_btr=lsb_sc_t.*coh_dem;
%lsb_sc_f_dem_btr = fftshift(fft(lsb_sc_t_dem_btr));
[z,f] = butter(4,4000/(cs/2),'low');
lsb_sc_t_dem_btr=filter(z,f,lsb_sc_t_dem_btr);
%lsb_sc_t_dem_btr = ifft(ifftshift(lsb_sc_f_dem_btr));
lsb_sc_t_btr_res = resample(lsb_sc_t_dem_btr,fs,cs);
lsb_sc_f_btr_res = fftshift(fft(lsb_sc_t_btr_res));
flsbdembtr=linspace(-fs/2,fs/2,length(lsb_sc_f_btr_res));
lsb_sc_f_btr_resmag=abs(lsb_sc_f_btr_res);
lsb_sc_t_btr_resmag = abs(lsb_sc_t_btr_res);
tlsbdembtr= linspace(0,length(lsb_sc_t_btr_res)/fs,length(lsb_sc_t_btr_res));
figure;
subplot(2,1,1)
plot(tlsbdembtr,lsb_sc_t_btr_res);
title('LSSB IN TIME DOMAIN after demodulation butter');
grid on
subplot(2,1,2)
plot(flsbdembtr,lsb_sc_f_btr_resmag);
title('LSSB IN FREQ. DOMAIN [MAGNITUDE SPECTRUM]
after demodulation butter');
grid on
% sound(lsb_sc_t_btr_resmag,fs);
% pause(length(lsb_sc_t_btr_resmag)/fs);
% wait until finishing sound file
```

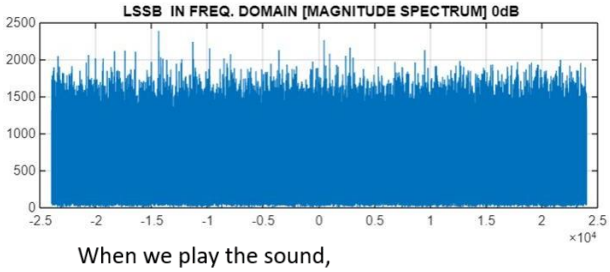




```
% 8

% 0 dB
snr0=awgn(lsb_sc_t_res,0);
snr0f=fftshift(fft(snr0));
snr0fmag=abs(snr0f);
snr0mag=abs(snr0);
figure;
subplot(2,1,1)
plot(tlsbdem,snr0);
title('LSSB IN TIME DOMAIN 0dB');
grid on
subplot(2,1,2)
plot(flsbdem,snr0fmag);
title('LSSB IN FREQ. DOMAIN[MAGNITUDE SPECTRUM] 0dB');
grid on
% sound(snr0mag,fs);
% pause(length(snr0)/fs);

% 10 dB
snr10=awgn(lsb_sc_t_res,10);
snr10f=fftshift(fft(snr10));
snr10fmag=abs(snr10f);
snr10mag=abs(snr10);
```

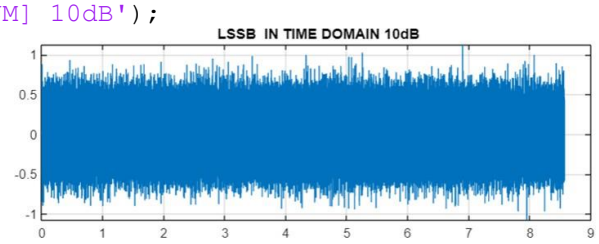
When we play the sound,
we find less distortion than the signal obtained by ideal filter.

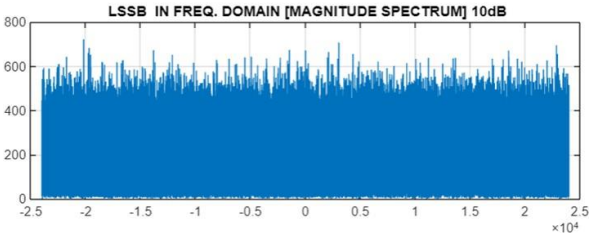


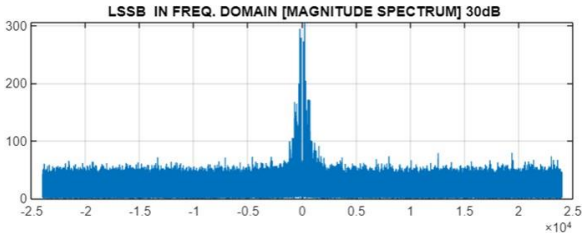


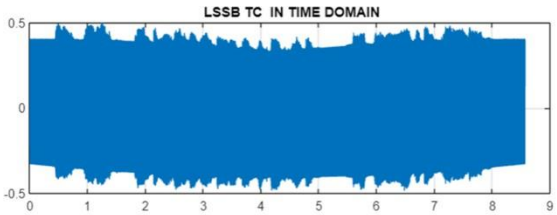
When we play the sound,
we found almost no sound because of the noise.

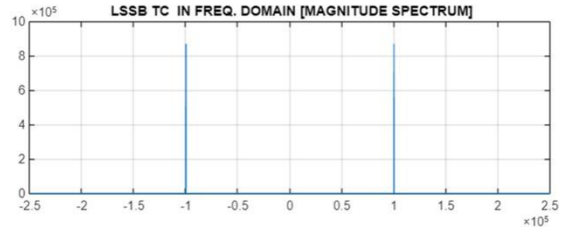
```
figure;
subplot(2,1,1);
plot(tlsbdem,snr10);
title('LSSB IN TIME DOMAIN 10dB');
grid on
subplot(2,1,2)
plot(flsbdem,snr10fmag);
title('LSSB IN FREQ. DOMAIN [MAGNITUDE SPECTRUM] 10dB');
grid on
% sound(snr10mag,fs);
% pause(length(snr10)/fs);

% 30 dB
snr30=awgn(lsb_sc_t_res,30);
snr30f=fftshift(fft(snr30));
snr30fmag=abs(snr30f);
snr30mag=abs(snr30);
figure;
subplot(2,1,1);
plot(tlsbdem,snr30);
title('LSSB IN TIME DOMAIN 30dB');
grid on
subplot(2,1,2)
plot(flsbdem,snr30fmag);
title('LSSB IN FREQ. DOMAIN [MAGNITUDE SPECTRUM] 30dB');
grid on
% sound(snr30mag,fs);
% pause(length(snr30)/fs);

% 9
am=max(abs(y)); Ac=am/0.5;
mtlssbtc=lsb_sc_t+Ac.*carrier;
mflssbtc=fftshift(fft(mtlssbtc));
mflssbtcmag=abs(mflssbtc);
figure;
subplot(2,1,1)
plot(t3,mtlssbtc);
title('LSSB TC IN TIME DOMAIN');
grid on
subplot(2,1,2)
plot(flsbfiltered,mflssbtcmag);
title('LSSB TC IN FREQ. DOMAIN [MAGNITUDE SPECTRUM]');
grid on
```







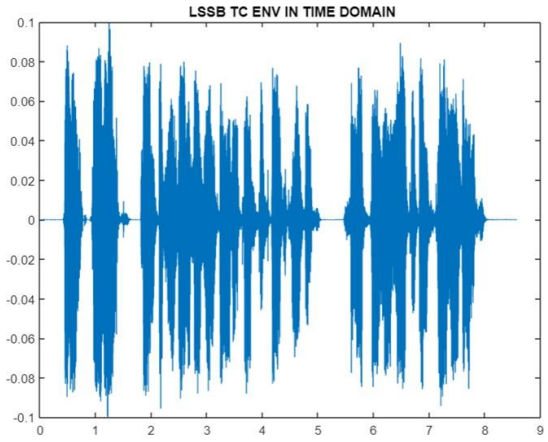




Comment

The following figure represent the (tc) version of ssb
By increase carrier level.

```
% Envelope Detector
mtenvtc =
abs(hilbert(mtlssbtc
)) - Ac;mfenvtc =
fftshift(fft(mtenvtc
));
B = 4000; %
bandwidth of the
filterSSBF_low =
length(mfenvtc)./
(cs);
lim =
ceil(((cs/2)-
B)*SSBF_low);
mfenvtc([1:lim  length(mfenvtc)-
lim+1:length(mfenvtc)])=0;mtenvtcfl =
ifft(ifftshift(mfenvtc));
tenv = linspace(0, length(mtenvtc)/cs, length(mtenvtc));
figure;
plot(tenv,mtenvtcfl);
title(' LSSB TC ENV IN TIME DOMAIN');
envtcres = resample(mtenvtcfl,fs,cs);
% sound(envtcres,fs);
% pause(length(envtcres)/fs);
```



Comment

When we generate a ssb-tc we can use envelope detector
to
Perform demodulation

Experiment Three

3 -EXPERIMENT THREE: FREQUENCY MODULATION

٣,١INTRODUCTION

Frequency modulation (FM) is a modulation type in which the instantaneous frequency of the carrier is changed according to the message amplitude. The motive behind the frequency modulation was to develop a scheme with inherent ability to combat noise. The noise, being usually modeled as additive, has a negative effect on the amplitude by introducing unavoidable random variations which are superimposed on the desired signal. Unlike the amplitude, frequency has a latent immunity against noise. Since it resides “away” from the amplitude, any changes in the amplitude would be completely irrelevant to the frequency. In other words, there is no direct correlation between the variation in amplitude and frequency, thus making FM a better candidate over AM with respect to noise immunity. However, what FM gains in noise immunity lacks in bandwidth efficiency. Since FM usually occupies larger bandwidth, AM is considered more bandwidth wise.

٣,٢AIM

In this experiment, we investigate the narrowband frequency modulation when the SNR is varied.

Students are expected to:

1. Develop an appreciation of FM ability to counteract noise.

2. Be able to simulate the generation and the demodulation of NBFM using MATLAB.

3. To be able to tell the similarities and differences between AM and NBFM.

٣,٣PROCEDURE

1. Repeat steps 1 through four in experiment 1,2.

2. Generate the NBFM signal. Use a carrier frequency of 100kHz and a sampling frequency of $Fs = {}^{\circ}Fc$. Plot the resulting spectrum. What can you make out of the resulting plot?

3. what is the condition we needed to achieve NBFM.

4. Demodulate the NBFM signal using a differentiator and an ED. For the differentiator, you can use the following command: diff. Assume no noise is introduced.

٣,٤USEFUL COMMANDS

audioread, fft, fftshift, ifft,ifftshift, awgn, upsample,resample, sound,hilbert, abs,mean,cumsum,diff

Code:

Audio is read using audioread built in function in MATLAB that return two important variables: the message and sampling frequency. Message is transposed to be able to process it easily with variables that will be generated later in the program. Signal is calculated in frequency domain using fft then it is normalized using fftshift.

```
% This code simulate Experiment 3 about Narrow Band Frequency Modulation

% Clear is used to clear variables in workspace so that each time we run
% the program it will process as it was the first time also to prevent
% the confusion of variables from the previous experiments

% clc is used to clear commands in Command Window

% close all is used to close all previous figures
clear
clc
close all

% Step One : reading audio file attached
% m_t will store message sample vector and fs will store sampling rate of
% messege
[m_t, fs] = audioread("eric.wav");

% Since Messege is column vector, we have to convert it into row vector to
% be able to process it with other variables
%(we can transpose the other variables instead of this one but it will be
% tedious task)
m_t = m_t';

% Evaluating Message Specturm
M_F = fftshift(fft(m_t));
```

endTime variable is used to evaluate the last second in signal to be able to generate time variable that will be used to plot signal in time domain. Freq variable will be used to plot signal in frequency domain.

```
% Ploting Message in Time domain and Frequency domain

% endTime will store period of audio without knowing it in advance
endTime = length(m_t) / fs ;

% time vector will be used to plot signal in time domain
time = linspace(0,endTime,length(m_t));

% while freq will be used to plot signal in frequency domain
% according to Nyquist: Sampling frequency is greater than maximum
% frequency in messege that is why we divide sampling frequency and plot
% from negative half of sampling frequency to half of sampling frequency
freq = linspace(-fs/2,fs/2,length(m_t));

% Ploting signal in time domain
figure
plot(time,m_t);
title("Signal In Time Domain");

% Ploting signal in frequency domain
figure
plot(freq,abs(M_F));
title("Signal In Frequency Domain");

% Step Two : use ideal filter to remove frequencies above 4kHz
% Factor is calculated which will be used in filtering the signal
factor = (length(M_F)/fs);
```

Signal is later filtered by zeroing samples that represent frequencies out of -4khz to 4khz as shown in this code:

The filtered signal is then plotted in both time and frequency domain as demanded.

```
% the filter will allow signal of frequency ranging from
% -4khz to 4khz while prevent the other frequencies from passing so
% samples from start to -4 khz and samples from 4 khz to end become zeros
% round is used to eliminate the probability of existing fraction index as
% that will result in error.

f_F = M_F;
f_F (1:round(factor*(fs/2 - 4*10^3))) = 0;
f_F (round(factor*(fs/2 + 4*10^3))+ 1:end)=0;

% Returning Filtered signal to time domain
f_t = real(ifft(ifftshift(f_F)));

% Ploting filtered signal in time domain
figure
plot(time,f_t);
title("Filtered Signal In Time Domain");

% Step Three :
% Ploting signal in frequency domain
figure
plot(freq,abs(f_F));
title("Filtered Signal In Frequency Domain");

% Sounding the filtered signal
sound(f_t,fs);
```

Signal is upsampled by 5 as it is required to upsample signal by 5 times before modulating it. DSB-SC of signal is evaluated according to equation:

$x(t) = m(t) \cos(2*pi*fc*t)$

```
% Step Four : Generating DSB - SC Modulated Signal
% Carrier Properties
fc = 100 * 10^3;
fm = 5*fc;

% e_t is upsampled version of f_t
e_t = resample(f_t, fm, fs);
e_F = fftshift(fft(e_t));

% due to upsampling, old time vector can't be used as well as old freq
% vector. so we create new vector that will be used to plot the signal
% later in the program
time_new = linspace(0,endTime,length(e_t));
freq_new = linspace(-fm/2,fm/2,length(e_t));

% Ploting resampled filtered signal in time domain
figure
plot(time_new,e_t);
title("Resampled Filtered Signal In Time Domain");

% Ploting resampled filtered signal in frequency domain
figure
plot(freq_new,abs(e_F));
title("Resampled Filtered Signal In Frequency Domain");

% DSB-SC modulated Signal
x_SC =  e_t.*cos(2*pi*fc*time_new);
X_SC = fftshift(fft(x_SC));
```

Phi variable will hold integrated verison of signal multiplied by frequency senitivity (K_f). since MATLAB store signal as samples so instead of integrating signals, program will perfrom summation of samples using cumsum built in function in MATLAB.

```
% Ploting DSB-SC modulated Signal in time domain
figure
plot(time_new,x_SC);
title("DSB-SC modulated Signal In Time Domain");

% Ploting resampled filtered signal in frequency domain
figure
plot(freq_new,abs(X_SC));
title("DSB-SC modulated Signal In Frequency Domain");

% Modulated Signal

% Modulation Properties
kf = 0.8;
f_maxdev = kf*max(e_t)/(2*pi);
A = 1;

% Phi is angle deviation in modulated signal
% cumsum is summation of signal which is equivalent to integration for
% continous signal. since MATLAB store values as samples we use summation
% instead of intgration.
Phi = kf.* cumsum(e_t);

% Applying signal to FM formula
s_t =  A.*cos(2*pi*fc*time_new + Phi);
s_F = fftshift(fft(s_t));
```

Question : What can you make out of the resulting plot?

Answer : The Modulated signal is nearly cosine signal with various frequencies that increase with increase of message amplitude and decreases with decrease of message amplitude .maximum frequency deviation = (max(m_t)*kf)/(2*pi) = 0.0253. so the transmitted power is constant (independent of message) = 0.5 watt

```
% Ploting Modulated Signal in time domain
figure
plot(time_new,s_t);
title("Modulated Signal In Time Domain");

% Ploting Modulated Signal in frequency domain
figure
plot(freq_new,abs(s_F));
title("Modulated Signal In Frequency Domain");

% Step 2 Question Answer
% The Modulated signal is nearly cosine signal with various frequencies
% that increase with increase of message amplitude and decreases with
% decrease of message amplitude.
% maximum frequency deviation = (max(m_t)*kf)/(2*pi) = 0.0253

% Step 3 Question Answer
% it is required that phase deviation to be less than 30 degree (pi/6 rad)
% and modulation index or frequency deviation constant very small compared
% to one. so that bandwidth of transmitted signal become nearly 2*fm in
% this case 8 kHZ (Similar to DSB-TC)
```

Question : What is the condition we needed to achieve NBFM ?

Answer : it is required that phase deviation to be less than 30 degree (pi/6 rad) and modulation index or frequency deviation constant very small compared to one. so that bandwidth of transmitted signal become nearly 2*fm in this case 8 kHZ (Similar to DSB-TC)

Demodulation is done by passing Signal through differentiatior so that the message become at envolope then I can be recovered by detecting envolope.

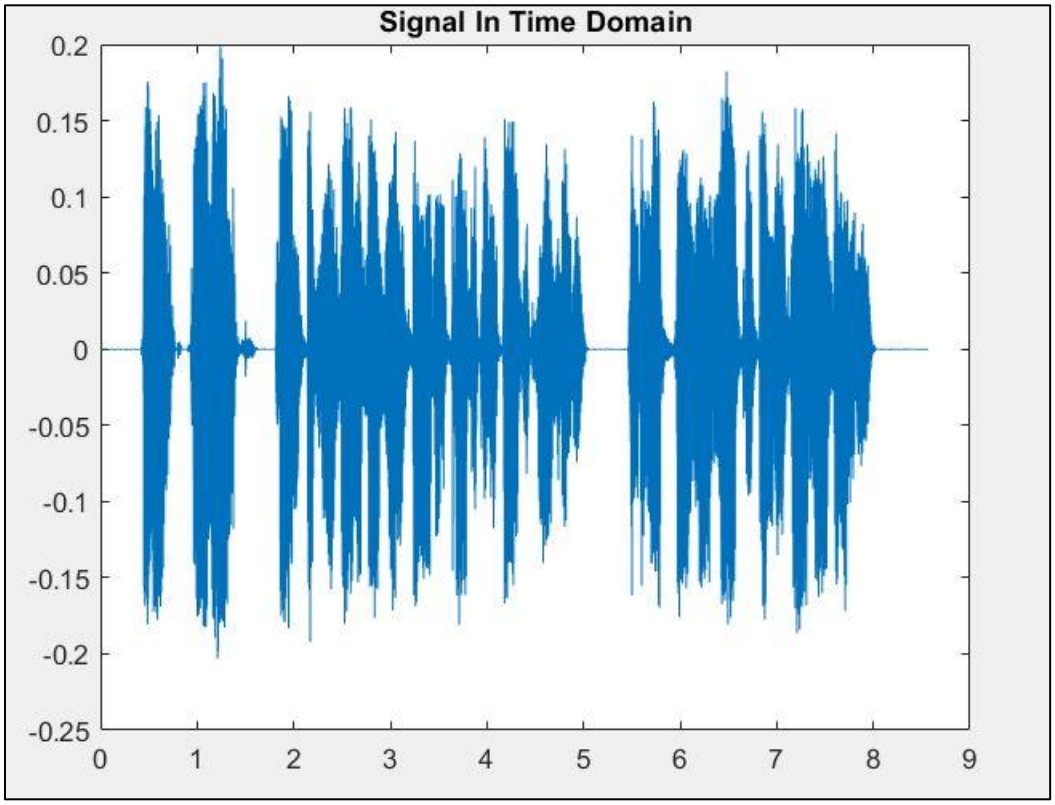
Differentiation is done using diff function that calculate difference of each sample and the sucessive one (similar to integration, difference was used instead of differentiation as signals are samples in MATLAB)

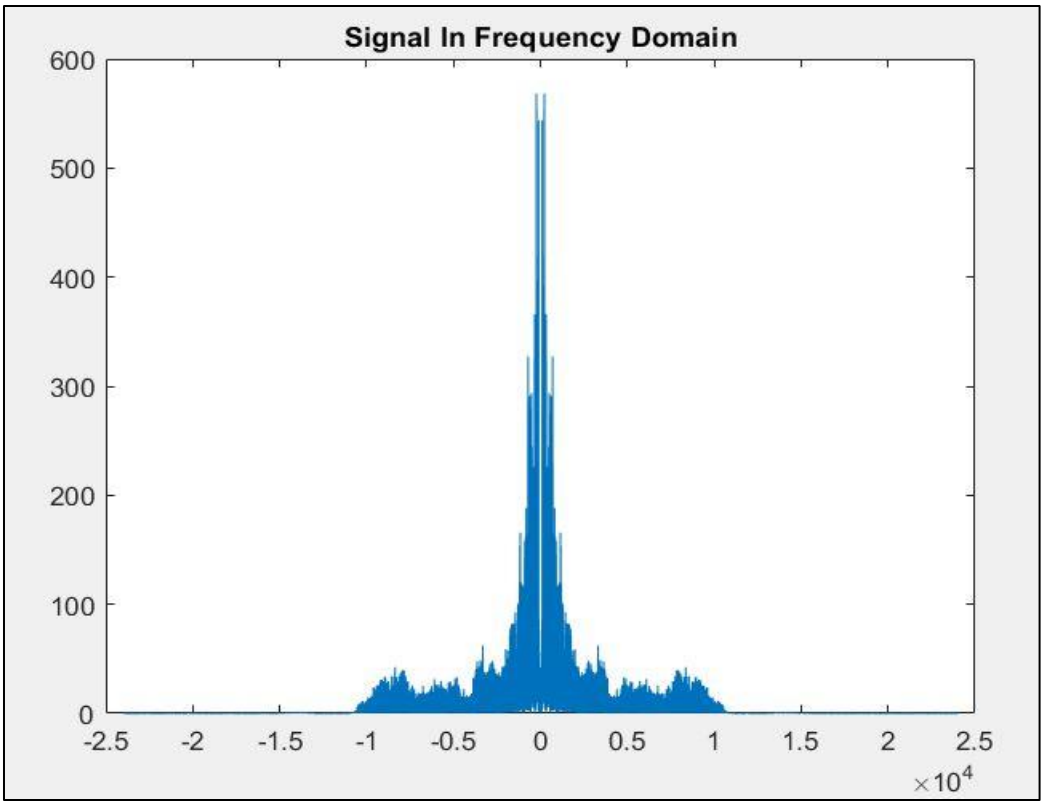
Envolope detection in matlab is carried out by using abs(hilbert(tranmitted_message)).

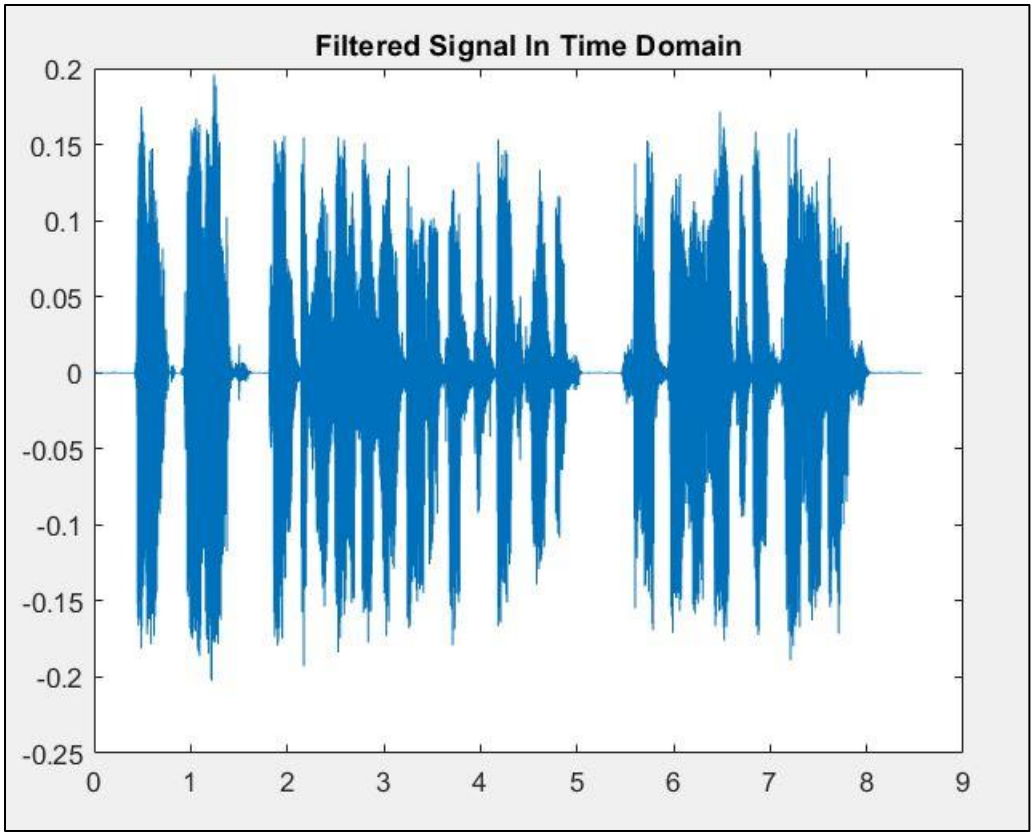
The resulted vector from diff function will be less than the transmitted by one sample that is why the last sample in time_new and freq_new is truncated to be able to plot output signal with time and frequency.

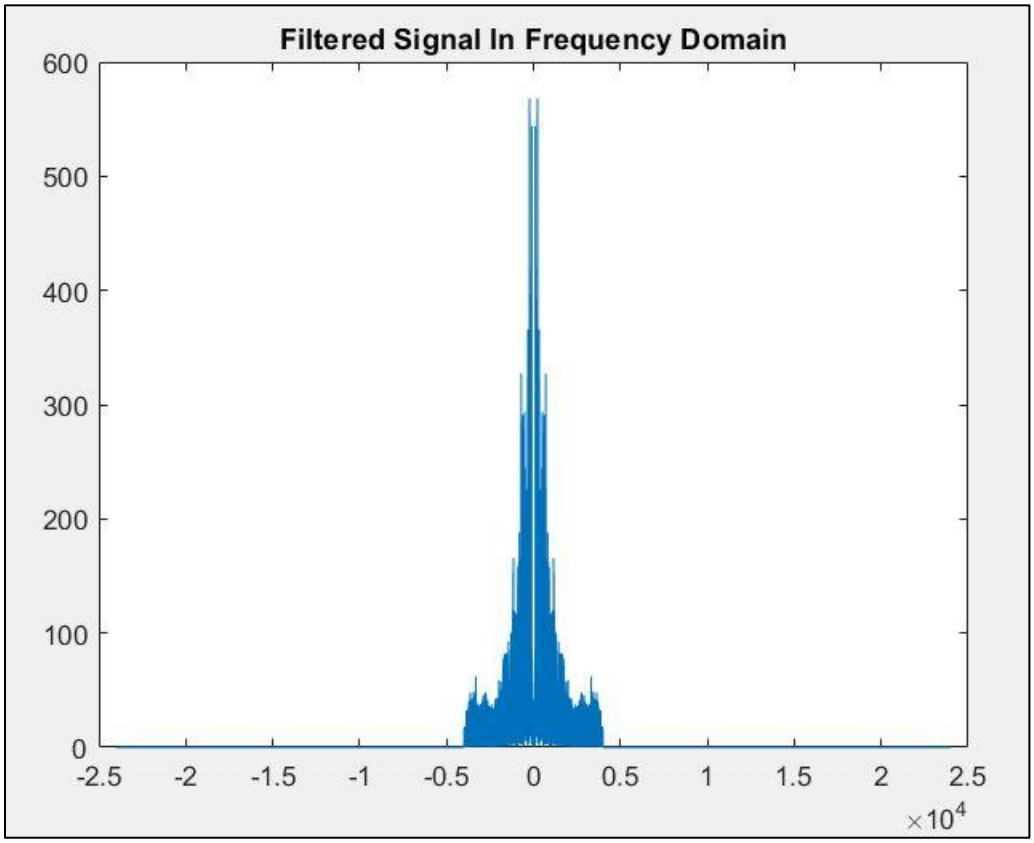
```
% Demodulation

% r is received signal
% demodulation require differentiator then result pass through envolope
% detector to detect envolope of output signal
% diff is used to calculate difference of vector which is equivalent to
% differentiation for continous signal but again as MATLAB store signal as
% samples. differencing samples is equivalent to differentiating the signal
diff_t = diff(s_t);
diff_F = fftshift(fft(diff_t));

% abs(hilbert()) is equivlent to envolope of signal
r_t = abs(hilbert(diff_t));
r_t = r_t -mean(r_t);
r_t = r_t/kf;
r_F = fftshift(fft(r_t));

% since the output signal is of length less than original by 1 so the
% time_new vector and freq_new vector must be reduced by one sample too
time_new = time_new(1:end-1);
freq_new = freq_new(1:end-1);

% Ploting Demodulated Signal in time domain
figure
plot(time_new,r_t);
title("Demodulated Signal In Time Domain");


% Ploting Demodulated Signal in frequency domain
figure
plot(freq_new,abs(r_F));
title("Demodulated Signal In Frequency Domain");

% to playback the sound, we need to return it to original rate
r_t = resample(r_t,fs,fm);
sound(r_t,fs);
```

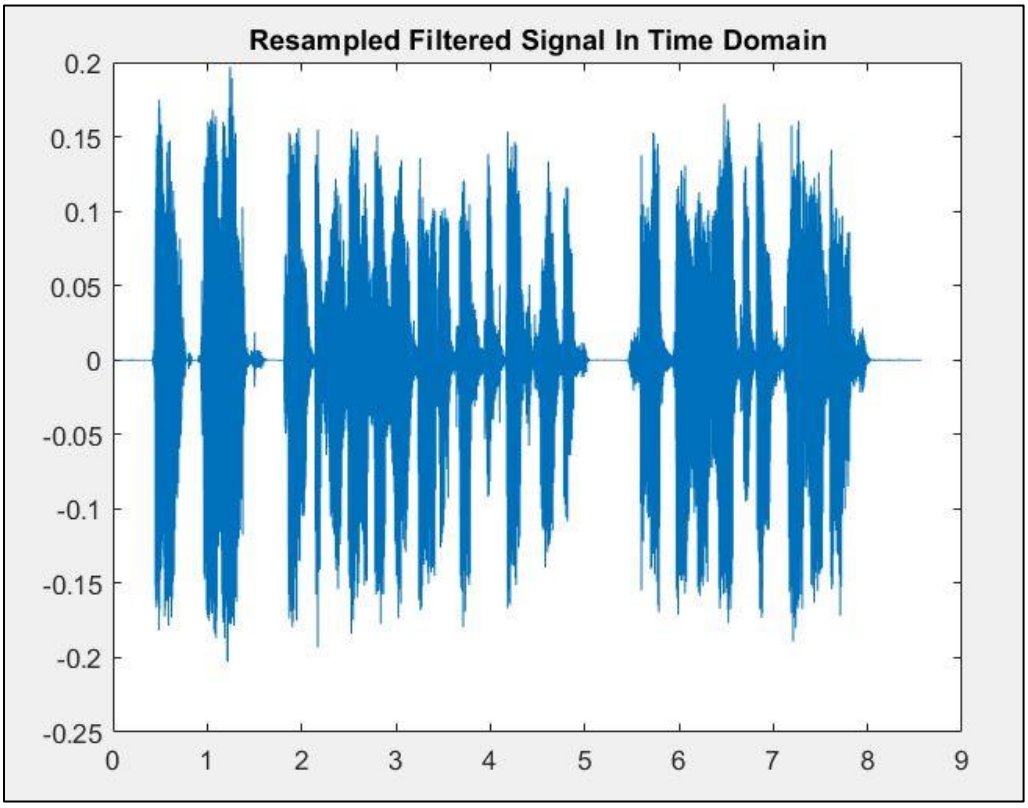
Result:

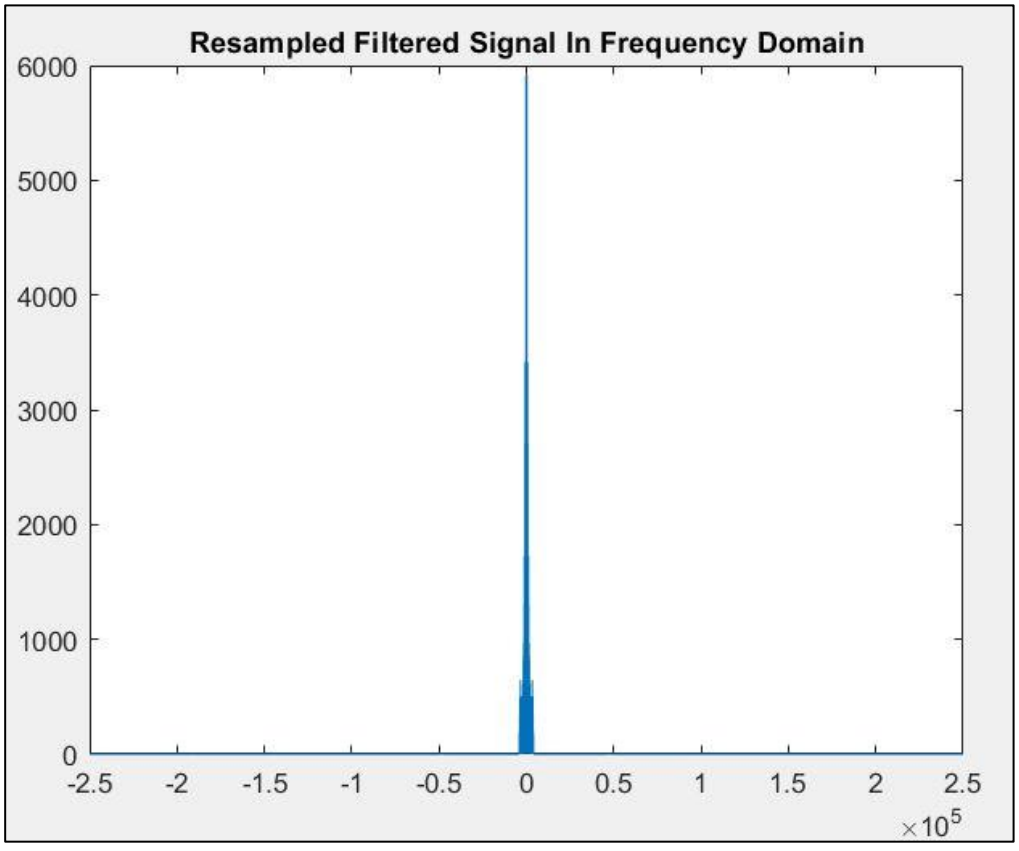

Signal In Time Domain
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
0
1
2
3
4
5
6
7
8
9

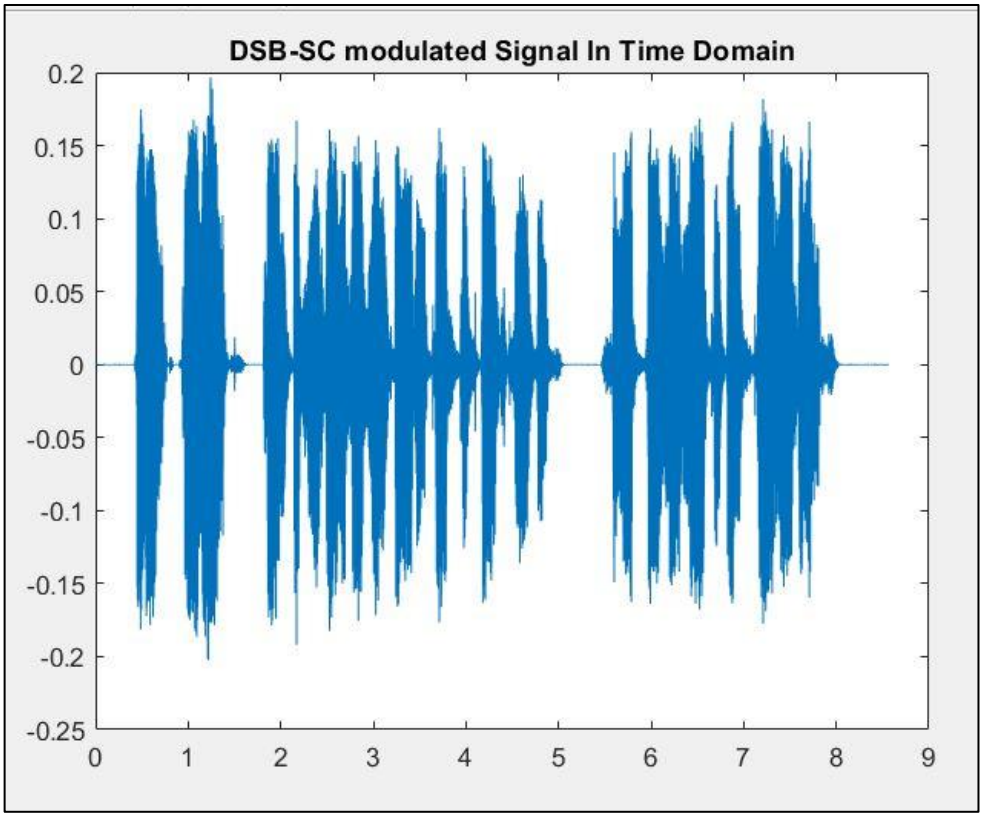

Signal In Frequency Domain
600
500
400
300
200
100
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
×10^4

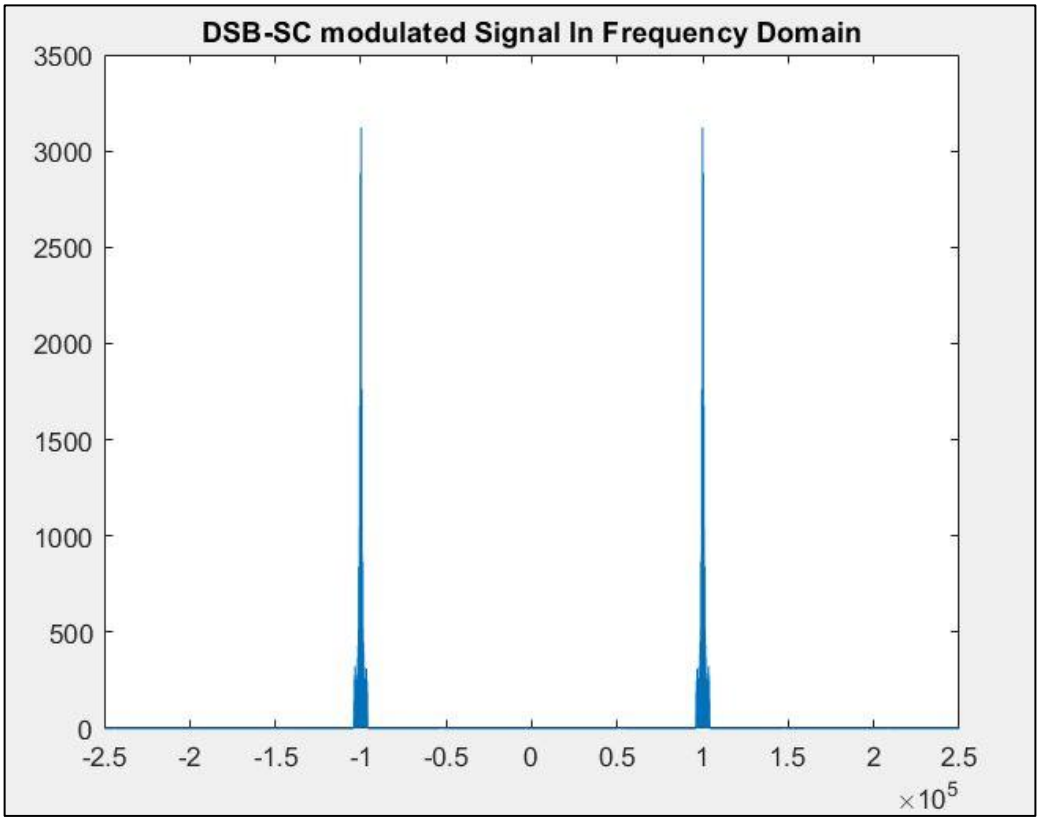

Filtered Signal In Time Domain
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
0
1
2
3
4
5
6
7
8
9

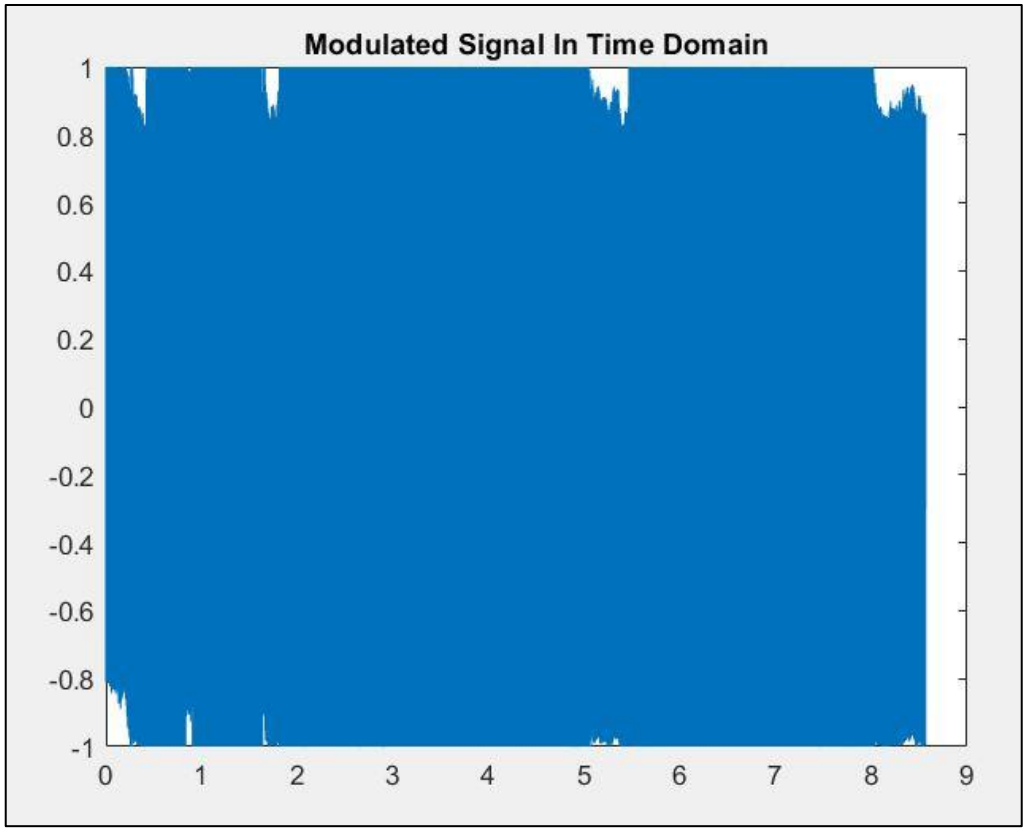

Filtered Signal In Frequency Domain
600
500
400
300
200
100
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
×10^4

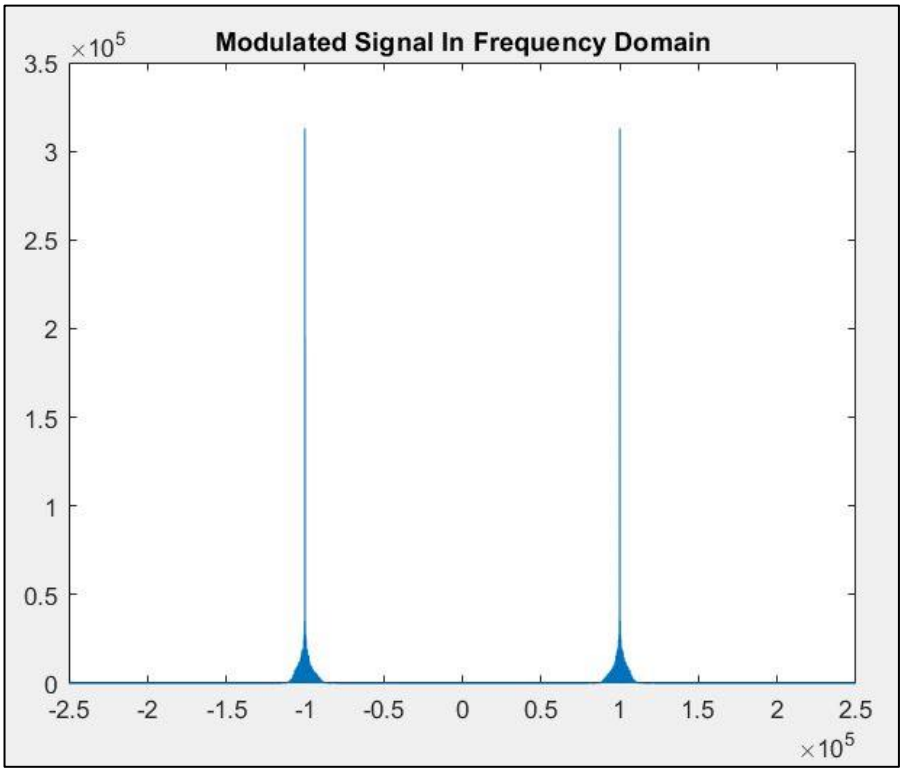

Resampled Filtered Signal In Time Domain
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
0
1
2
3
4
5
6
7
8
9

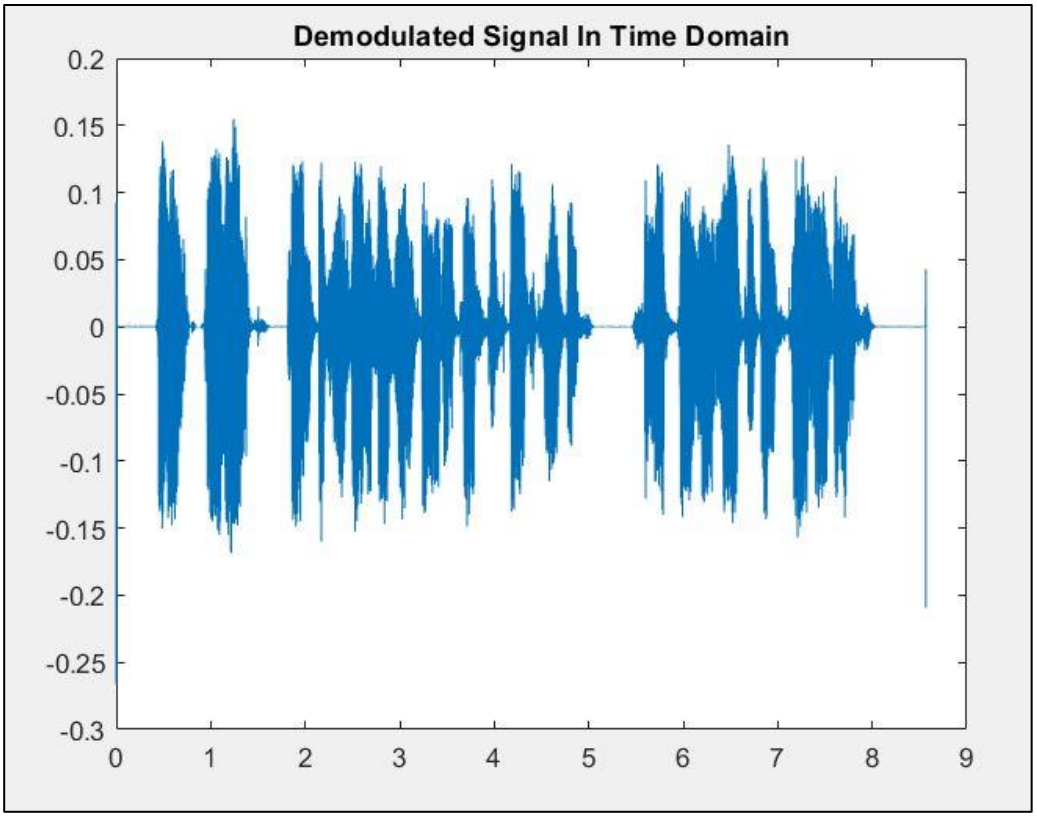

Resampled Filtered Signal In Frequency Domain
6000
5000
4000
3000
2000
1000
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
×10^5

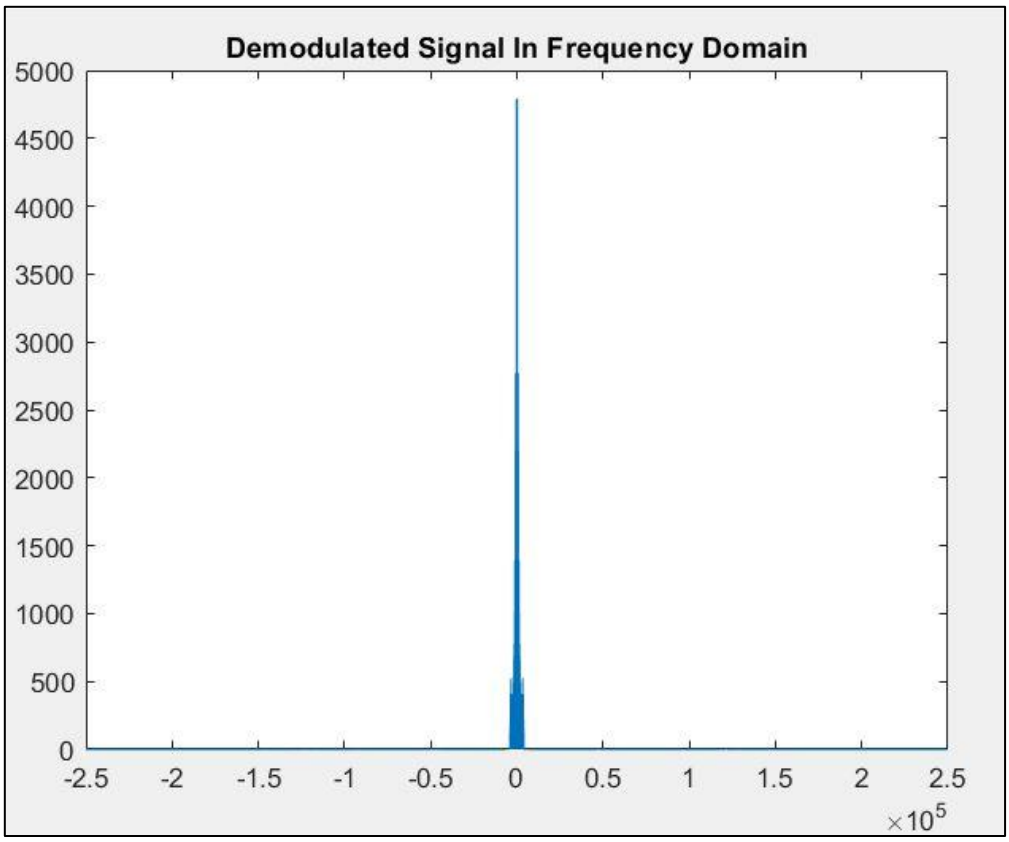

DSB-SC modulated Signal In Time Domain
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
0
1
2
3
4
5
6
7
8
9


DSB-SC modulated Signal In Frequency Domain
3500
3000
2500
2000
1500
1000
500
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
×10^5


Modulated Signal In Time Domain
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7
8
9


Modulated Signal In Frequency Domain
×10^5
3.5
3
2.5
2
1.5
1
0.5
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
×10^5


Demodulated Signal In Time Domain
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
0
1
2
3
4
5
6
7
8
9


Demodulated Signal In Frequency Domain
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
×10^5

Conclusion:

Signal is transmitted using Narrow band frequency modulation with very little noise and with small bandwidth of about double frequency of message.