BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Group Project Report

Course No: EEE 310

Course Name: Communication Systems Laboratory

Project Title:

Modeling AM Modulation and Demodulation Techniques Using Matlab and Simulink

Submitted to:

Dr. Md. Forkan Uddin
Professor, Department of EEE, BUET

Tashfiq Ahmed
Lecturer, Department of EEE, BUET

Submitted By

MD. Humyom Hossain	1706049
Nazmus Salehin	1706052
Imtiaz Ahmed	1706062
Tamim Ahmed	1706063

Introduction

In this project, we have implemented an AM transmission system in matlab code and simulink block diagrams and illustrated the working principles of the AM transmission system. Amplitude modulation is used in the real world for transmission of message signals over long distances with minimum distortion. For long distance transmission, a major drawback is the message signals transmitted, which are usually audio and video signals, have a low frequency. Such signals lose strength over long distances and thus reconstruction at the receiving end becomes quite hard, at times impossible. Higher the frequency of the transmitted signal, lower the attenuation. Thus, in order to minimize attenuation, the message signal is modulated into a high frequency carrier signal (usually a sine wave). Thus, the attenuation reduces by a great extent and the signal can be reconstructed after demodulation to get almost the perfect signal. This is what we have implemented in our project in a basic level, where as a message signal we
have used sine, triangular and a complex sinc function wave of known frequency. Later we also implemented the system with voice message signals and used Frequency Division Multiplexing Technique (FDM) to transmit our message signal. In this report, we have done a thorough performance analysis of our whole process.

Theory

Amplitude modulation (AM) is a modulation technique in which the amplitude of a high-
frequency carrier signal(sine wave) is varied in direct proportion to that of a modulating signal. The modulating signal carries the required information. For long distance data transfer, the first condition of the signal power has to be strong.The voice signal bandwidth is lower than 3400 Hz signal which has not good enough power to transmit long distance. That's why we need to do modulation so that we can transfer our baseband signal into passband and transmit over a long distance.

Amplitude Modulation is of 4 types:

1. Double Sideband (DSB) modulation
    a. Double Sideband with carrier (DSB-WC)
    b. Double Sideband with suppressed carrier (DSB -SC)
    c. Double Sideband reduced carrier (DSB-RC)

2. Single Sideband (SSB) modulation
    a. Single Sideband with carrier (SSB-WC)
    b. Single Sideband with suppressed carrier (SSB-SC)
3. Vestigial Sideband (VSB) modulation
4. Quadrature Amplitude Modulation (QAM)

A detailed explanation of these modulation techniques is presented below:

1 . Double Sideband Modulation(DSB)

In general, modulation is the process of multiplying a message signal with a carrier signal in the time domain. In the frequency domain, this becomes convolution between carrier and message signal. As the carrier signal is pure sinusoid, it becomes two impulses at its frequency and due to modulation the message signal gets duplicated along with it in both positive and negative frequency side. In the positive frequency side the side of the message signal towards the right is called the upper sideband and the other one is called the lower sideband. As we compute the bandwidth of the message signal by taking in consideration only the positive frequency part, the extra lower sideband part from the original negative frequency part now also comes into play. As a result the bandwidth gets doubled than the original signal. If we use this modulated signal as it is with two sidebands for transmission, this is called double sideband modulation. This modulation technique has some disadvantages which gave rise to other opinions for AM modulations. Such as-

1. Wasteful of transmitted power
2. Wasteful of Bandwidth(2x of message signal required)
3. Easy to be affected by Noise.

But even though these problems remain, this system is very simple and less expensive and has a wide range of uses in analog signal wireless transmission.

Modulation and Demodulation principle:

(In practical system AM modulation is carried out by systems like switching modulators, but in our project we implemented modulation by multiplying carrier and message signal directly)

(The whole process is demonstrated using a sinusoid as the message signal)

Carrier signal, c(t) = Ac*cos(wc*t)
Message signal, m(t) = Am*sin(wm*t)

AM modulated signal, $\boldsymbol{\Phi}$(t) = m(t) * c(t) + c(t) [for DSB-WC, for DSB-SC carrier won't be added]

= [Ac + m(t)] *cos(wc*t) = Ac*[1+m(t)/Ac] * cos(wc*t)
= Ac*[1+u*cos(wm*t)]*cos(wc*t) = [1 + u*cos(wm*t)]*c(t)

Here u is the modulation index of the signal.

This is the format in time domain, in frequency domain it becomes,

$\boldsymbol{\Phi}$(f) = 0.5*Ac*[del(f-fc) + del(f+fc)] + 0.5*[M(f-fc)+M(f+fc)]

Which is just the message signal getting copied around the two impulses of the carrier signal in the frequency domain.

For demodulation, there is different types of techniques like-

1. Envelope detection
   (Works for u<=1 and only when carrier is also transmitted)

2. Rectifier demodulator
   (works for only u<=1)

3. Synchronous/ Coherent/ Homodyne Detection:
   (works for almost all cases and all modulation technique but the system is complex)

(In our project, we have used synchronous detection for demodulation, that's why we'll proceed with it.)
In synchronous detection, we multiply the modulated signal again with the same carrier signal used for modulation.

p(t) = $\boldsymbol{\Phi}$(t) * c(t)
= 0.5 * [Ac + m(t)] (1 + cos(2*wc*t))
= 0.5 * Ac + 0.5 * m(t) + 0.5 * (Ac+m(t)) *cos(2*wc*t)

After using a low pass filter with the cutoff frequency slightly higher than message bandwidth and Dc blocking we recover the transmitted signal.

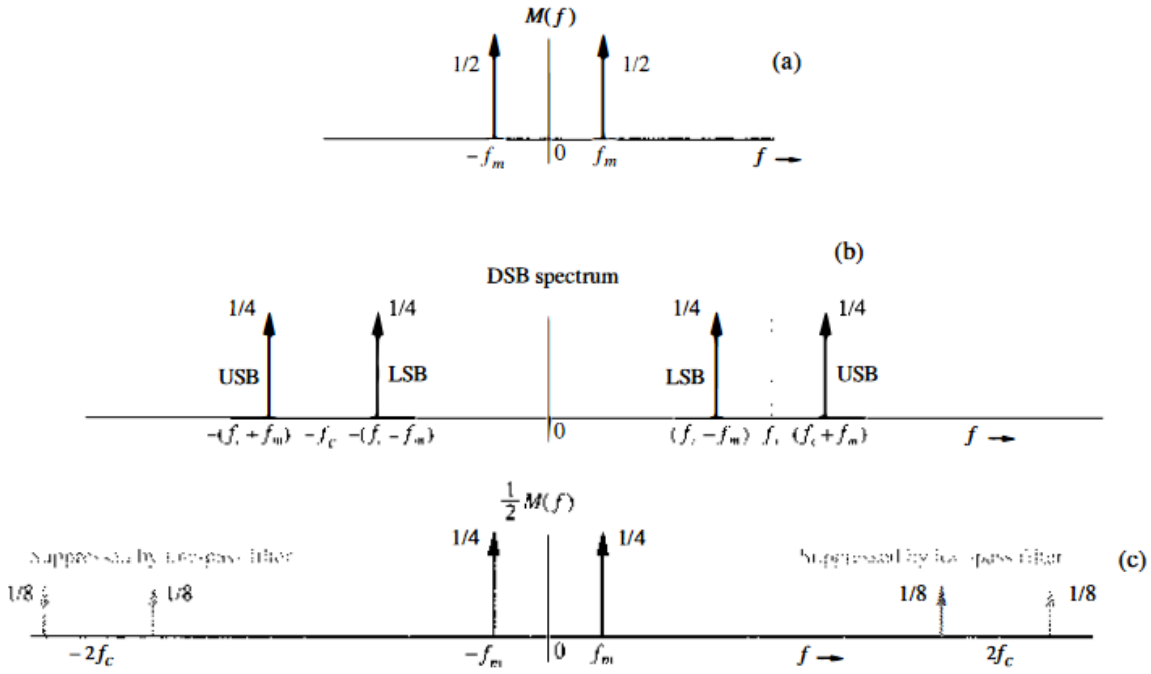


Fig : DSB modulation and demodulation in the frequency domain

Block Diagram of the DSB transmission system :

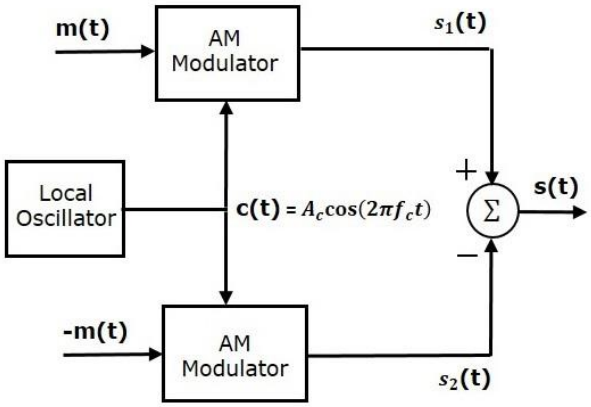


Fig: DSB-SC Modulation block diagram

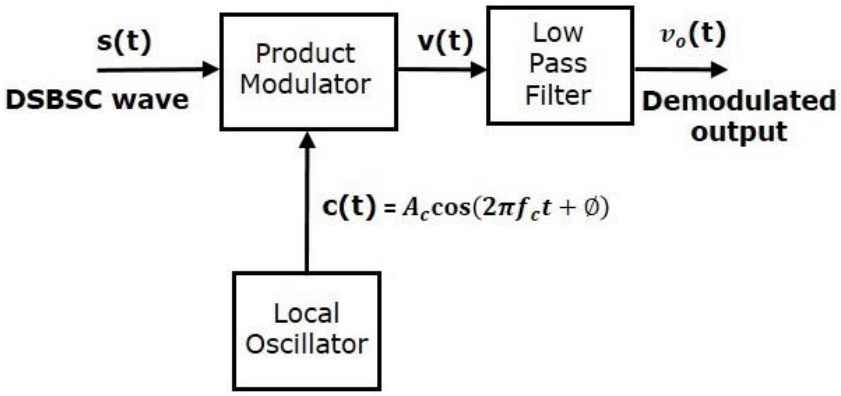


Fig: DSB-SC demodulation block diagram

2 . Single Sideband Modulation(SSB)

As I have mentioned before, DSB modulation and demodulation techniques are very simple and less expensive but it also wastes a lot of bandwidth and power to transmit message signals that can be sent more efficiently in other ways. SSB is such a technique where we filter out the LSBs or USBs from both sides of the frequency axis and only one sideband remains on both sides of the spectrum. As it's demonstrated in figure 1, the final reconstructed signal comes from both sides of the spectrum and the modulated and reconstructed signal have the magnitude because of two spectrum being merged there. So if we filter out one of the spectrums, everything works as it is just the magnitude gets halved which is easily managed by amplifying the signal. By doing this simple technique we can use the same amount of bandwidth as the message signal to send the message signal more efficiently than DSB.

Modulation and Demodulation principle:

SSB modulation is done using Hilbert Transform. Hilbert transform is a ideal phase shifting method where the phase of every frequency component is shifted by **-π**/2 rad.

H(f) = -j*sgn(f), is defined as the hilbert transformer

Xh(f) = H(f)*X(f)

M+(f) = M(f)u(f) = M(f)*0.5*[1+sgn(f)] = 0.5*[M(f) + j*Mh(f)]
M-(f) = M(f)u(-f) = M(f)*0.5*[1-sgn(f)] = 0.5*[M(f) - j*Mh(f)]

Here, Mh(f) = H(f) * X(f)

Φusb(f) = M+(f-fc) + M-(f-fc)

So, **Φ**usb(t) = m(t)*cos(wc*t) - mh(t)*sin(wc*t)
Similarly, **Φ**lsb(t) = m(t)*cos(wc*t) + mh(t)*sin(wc*t)

Φssb(t) = m(t)*cos(wc*t) -/+ mh(t)*sin(wc*t)

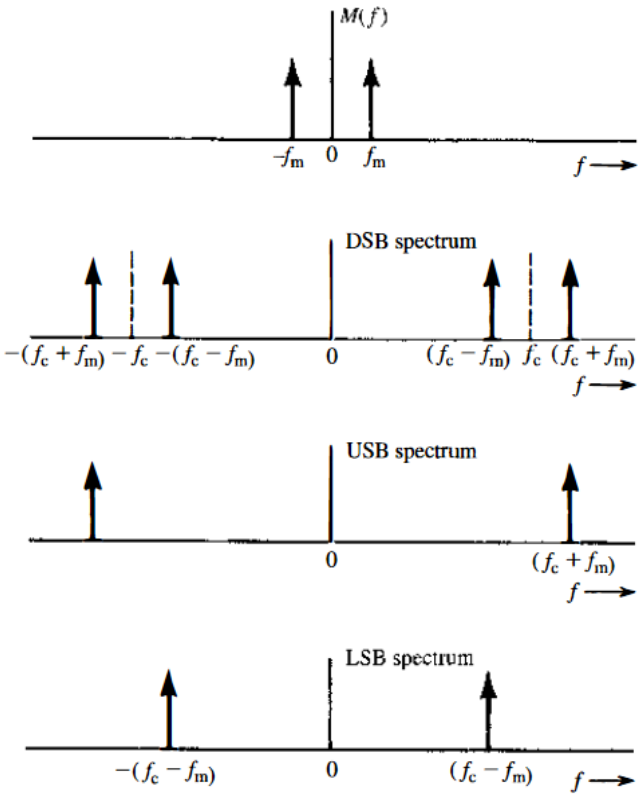


Fig: Extracting the single sideband from double sideband

Thus SSB modulated signals can be generated. In practical system these are implemented by different types of modulation techniques like -

1. Phase-shift method
2. Selective filtering method
3. Weaver method

In our project, we used a selective filtering method for extracting SSB modulated signals.

For demodulation Coherent detection method is also applicable here.

p(t) = ***Φ***ssb(t) * 2* cos(wc*t) = m(t) + [m(t)*cos(2*wc*t) -/+ mh(t)*sin(2*wc*t)]

After using a low pass filter after this with a cutoff frequency the same as that of the bandwidth of the message signal we recover it.

Block Diagram of the SSB transmission system :

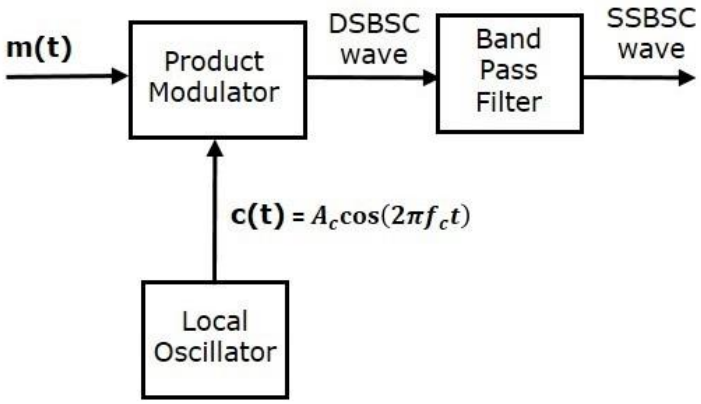


Fig : SSB-SC modulation block diagram

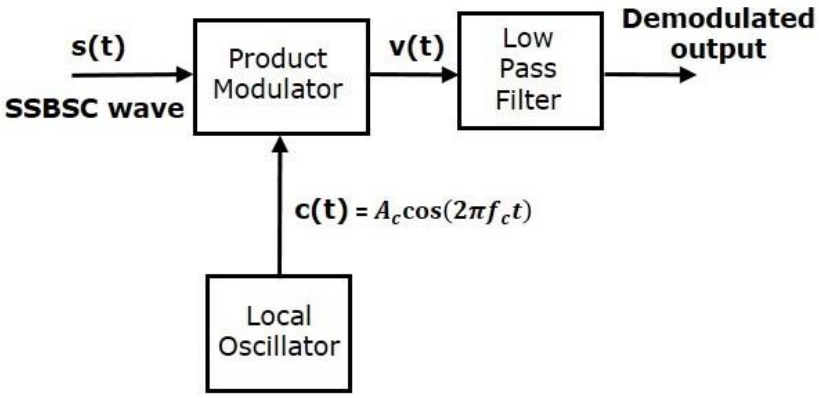


Fig: SSB-SC Demodulation block diagram

3 . Vestigial Sideband Modulation(VSB)

We have chosen SSB as the alternative of DSB as the same transmission systems stand solid with that even after using only half of the bandwidth. But in reality, exactly cutting one of the sidebands needs a very sharp filter which is not easy to design. Usually filters have a slop in the borders rather than a sharp straight line. Because of that some parts of the filtered out portion(usually upto 25%) of the other side band remains in most of the cases and it's easier to design filter for that. VSB inherits advantages of both DSB and SSB but it avoids their disadvantages at a small cost.

Modulation and Demodulation principle:

$\boldsymbol{\Phi}$vsb(f) = [M(f-fc) +M(f+fc)] * Hi(f)

Hi(f) is the transfer function of the vestigial shaping filter which is a bandpass filter to filter out the LSB of the modulated signal

For demodulation coherent detection is used.

P(f) = [$\boldsymbol{\Phi}$vsb(f-fc) + $\boldsymbol{\Phi}$vsb(f+fc)]*H0(f)

Where H0(f) = 1/[Hi(f_fc) + Hi(f-fc)]; | f | < B is the transfer function of the low pass filter used for synchronous detection.

Block Diagram of the VSB transmission system :

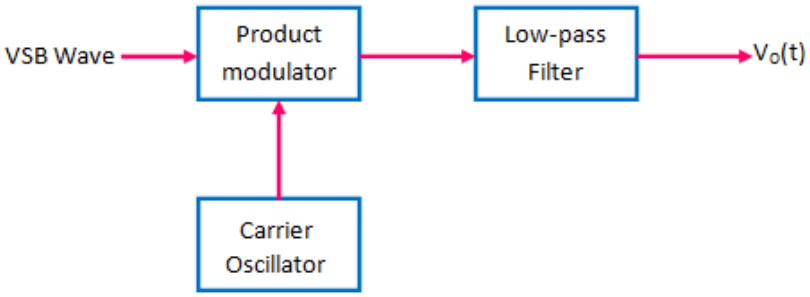


Fig: VSB-SC modulation block diagram

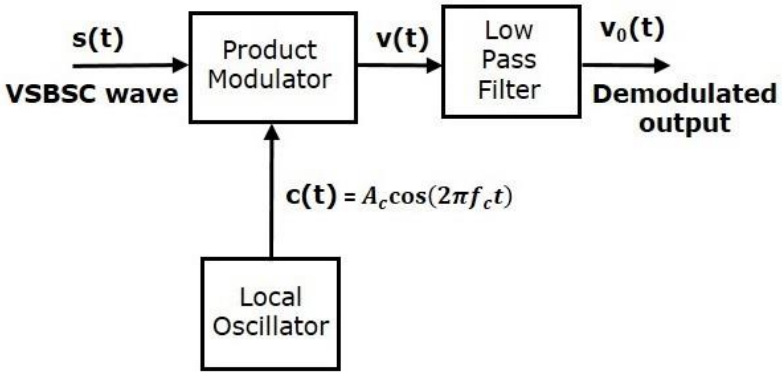


Fig: VSB Demodulation block diagram

4 . Quadrature Amplitude Modulation (QAM)

As SSBs are difficult to generate along with VSB, another attractive alternative is QAM modulation technique. In this modulation system two baseband signals, each with bandwidth B, are sent over the same bandwidth with 2B bandwidth in total like DSB but the upside is we need only B bandwidth per signal. As we are simultaneously sending 2 signals they can interfere between themselves. For this, we choose the carrier frequencies to be the same with 90 degrees phase shift, which helps us to send 2 signals simultaneously without worrying about their interference. As this method is one

type of multiplexing technique ,it's also known as the Quadrature Multiplexing technique.
If it can be implemented accurately then there is no problem but if the synchronisation is lost(carriers have a phase difference other than 90 degrees) -

a. Loss of powers
b. Interference

Then the message signals may not be recovered anymore ans we would get interfered version of the signals in both channels.

Modulation and Demodulation principle:

Here we need a set of message signals, one carrier and a 90 degree phase shifter.

$\boldsymbol{\Phi}$qam(t) = m1(t)*cos(wc*t) + m2(t)*sin(wc*t)
For demodulation we use synchronous detection.

In phase(I) channel:
x1(t) = $\boldsymbol{\Phi}$qam(t) * 2* cos(wc*t) = m1(t) + m1(t)*cos(2*wc*t) + m2(t)*sin(2*wc*t)

Quadrature(Q) Channel:
x2(t) = $\boldsymbol{\Phi}$qam(t) * 2* sin(wc*t) = m2(t) - m2(t)*cos(2*wc*t) + m1(t)*sin(2*wc*t)

Block Diagram of the QAM transmission system :

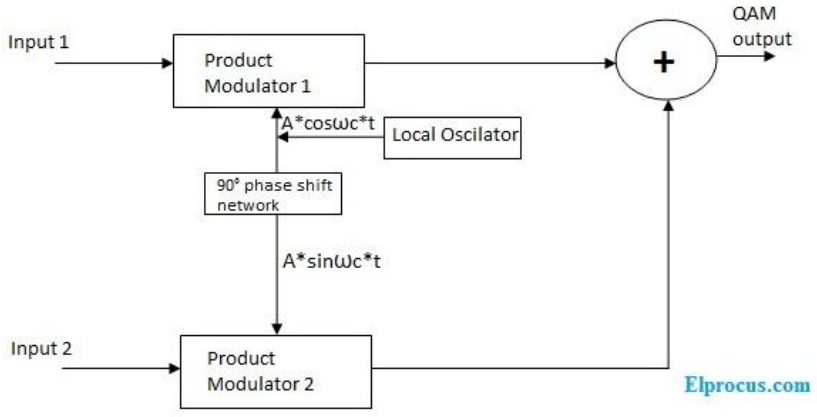


Fig: Block Diagram of QAM modulation block diagram

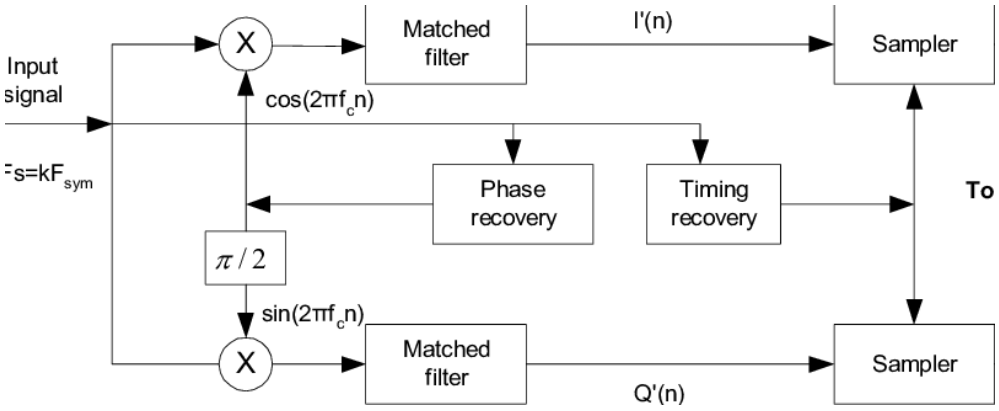


Fig: Block Diagram of QAM Demodulation block diagram

Along with these modulation techniques, there are multiplexing techniques that allow us to send multiple signals at the same time which also enables transmission of the message signal more effectively. Some of the multiplexing techniques are -

1. Time Division Multiplexing (TDM)
2. Frequency Division Multiplexing (FDM)
3. Code Division Multiplexing(CDM)
4. Orthogonal Frequency DIvision Multiplexing (OFDM)
5. Wavelength Division Multiplexing (WDM)

Among these techniques, we used Frequency Division Multiplexing technique to transmit multiple voice signals using AM modulation and demodulation techniques. Descriptions about the FDM system is added below:

5 . Frequency Division Multiplexing (FDM)

In the FDM system available bandwidth of the common channel is divided into bands. Signals are separated in frequency domain in those bands. Thus if we transmit the signals will be in separate frequency regions from other signals. But here we need guard bands between signal bands to avoid interference. And it also requires filtering to minimize adjacent channel interference which makes it costly. But overall we can send signals successfully using this system. A block diagram illustrating the system is shown below:
(block diagram inserted here)

Project Implementation

We have used matlab and simulink as our medium for simulating a real world AM modulation and demodulation system. In both of these mediums we implemented four types of modulation system and at last used a FDM system to illustrate how we can use multiplexing using these modulation techniques. For making it more real world like we have used AWGN block in simulink (awgn function in matlab) as the noise added by channel in our system, For data we have used simple signals and voice as our input and performed simulation on them. Detailed descriptions are added in the following sections.

Signal Characteristics

At first, we used simple signals like-

1. A Sine wave signal
2. A triple sinc function (sum of three sinc functions)
3. An aperiodic triangular function

We used these signals as input to use a variety of signals to test our validity. As sine function is simple and ideal for testing if our system is acting properly. Then we used a little complex signal like triplesinc and at last an aperiodic signal which is a little more complex than that. Thus at first we confirmed that our system works for simple signals.
Then we moved to voice signals as our input as it was our primary goal to demonstrate a real world example. Here we have used 3 types of signals-

1. Our voice input(voice signal without music like sound)
2. Music (only sound no voice)
3. Songs (voice + music , a little more complex)

The reasoning behind it is that our voice signal has range 20 Hz to 20000 Hz and sometimes over 50000 hz, theoretically but we can see from the spectrum plotted from these signals that after a certain frequency the significant frequencies are no longer existent. As a result even if we filter out those frequencies we can still hear almost perfectly the original signal. This is why telephone companies use 300-3400 Hz range for voice over communications. We played with the dataset a little and observed that if we filtered out voice signals at 2500-2800 Hz bandwidth we got almost perfect signal in no noise condition.

The second option musical sounds generally contains combinations of harmonically related sinusoids which creates such nice tones and songs contain both voice and music so we can get a taste of the combination of these things.

Model parameters

We used awgn block to mix noise to our signal to model real world signals. Here we used values generally used in transmission systems from online sources. In the following table we illustrated the SNR values and their acceptability in the transmission system.

SNR range(dB)	Acceptability
10 - 15	Minimum to establish connection
15 - 25	Minimally acceptable level to establish poor connectivity
25 - 40	Good
41 or higher	Excellent

We used SNR 10 to show that our model can handle bare minimum acceptable SNR in producing good results.

Other important values are added in the following table:

Characteristics	Values
Sampling frequency, Fs	44100 Hz
Channel	1
length(time)	3s
Used Bandwidth	3000 Hz
SNR	20

Matlab Implementation

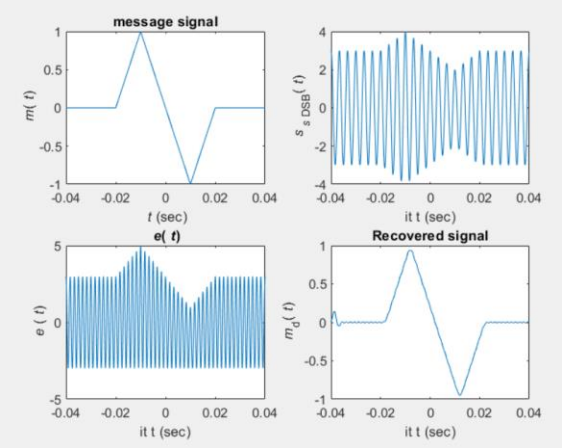
In matlab we implemented the following techniques and generated their outputs:-

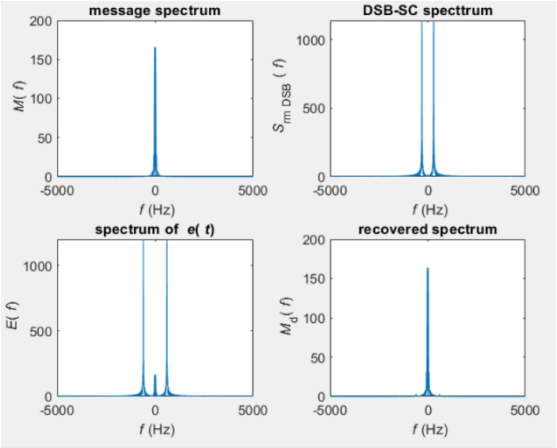
1. DSB-WC and DSB-SC
2. SSB-WC and SSB-SC
3. VSB-SC
4. QAM
5. FDM using SSB modulation.

Here we mainly used the 3 simple inputs discussed before and voice signal. The codes used are added in the appendix. The input and signals in different stages are illustrated input-wise below for clear depiction of the process:

1. Aperiodic Triangular wave

A. DSB-WC



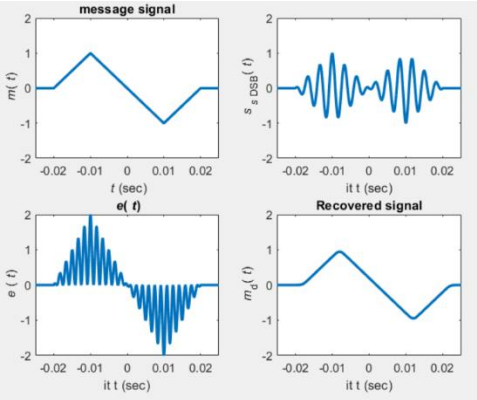


Time domain

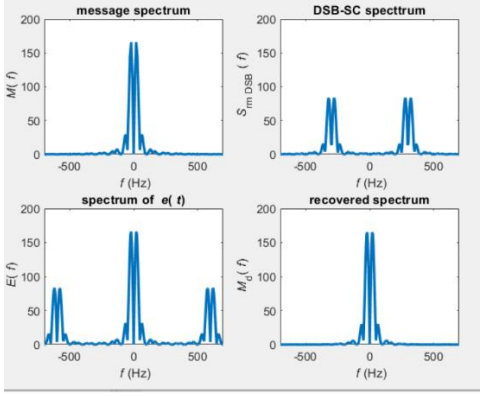
Freq domain

B. DSB-SC

Time Domain

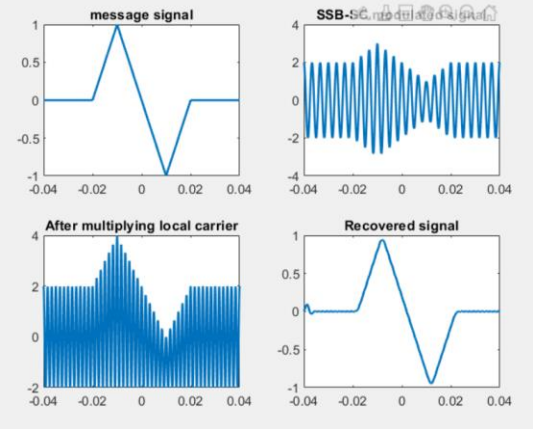


Frequency Domain

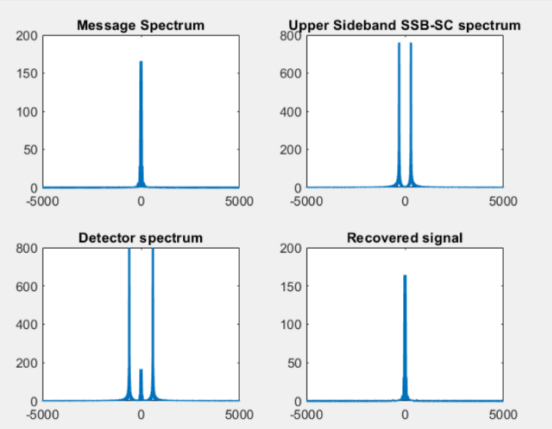


C. SSB-WC

Time domain

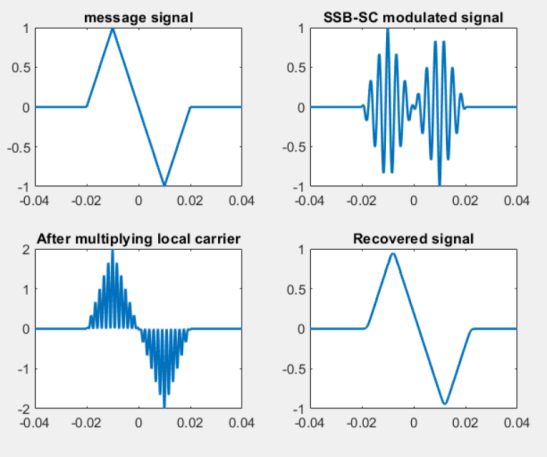


Frequency Domain

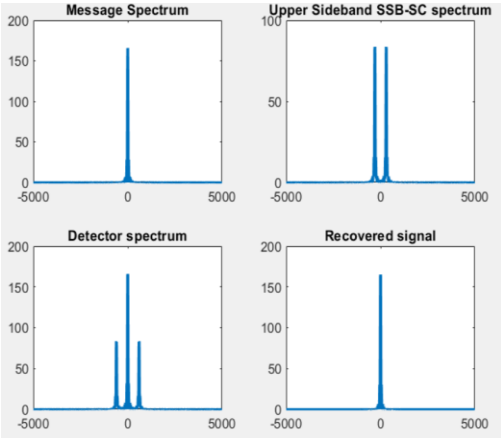


D. SSB-SC

Time domain



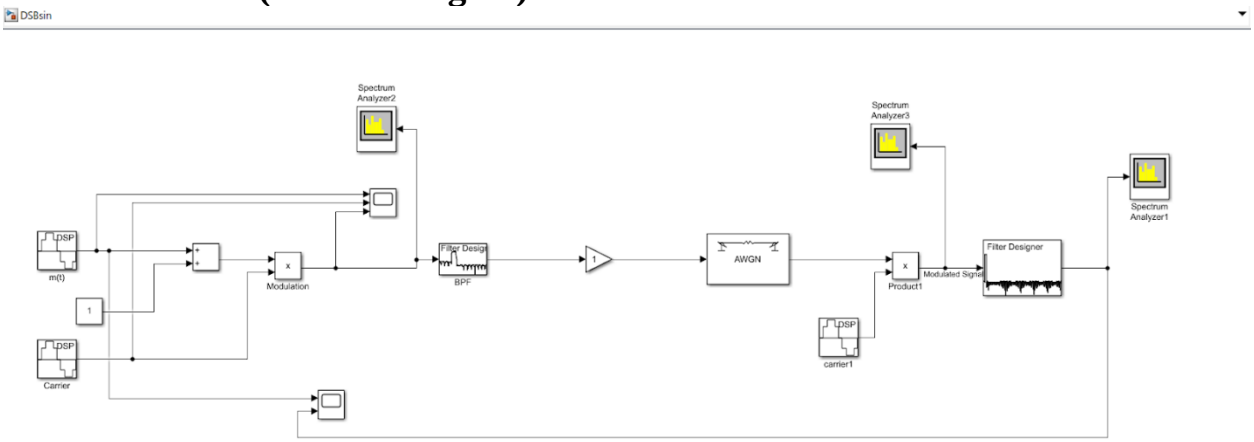
Frequency Domain



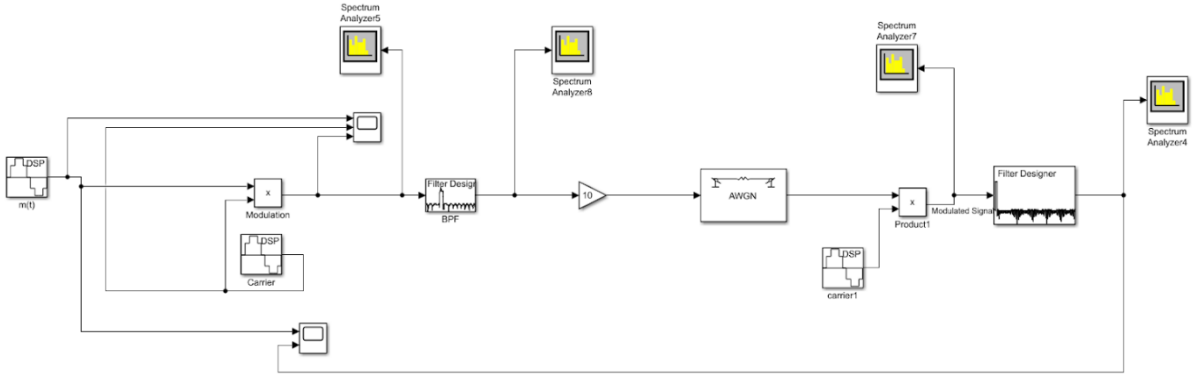
Simulink Implementation

Simulink block diagrams:

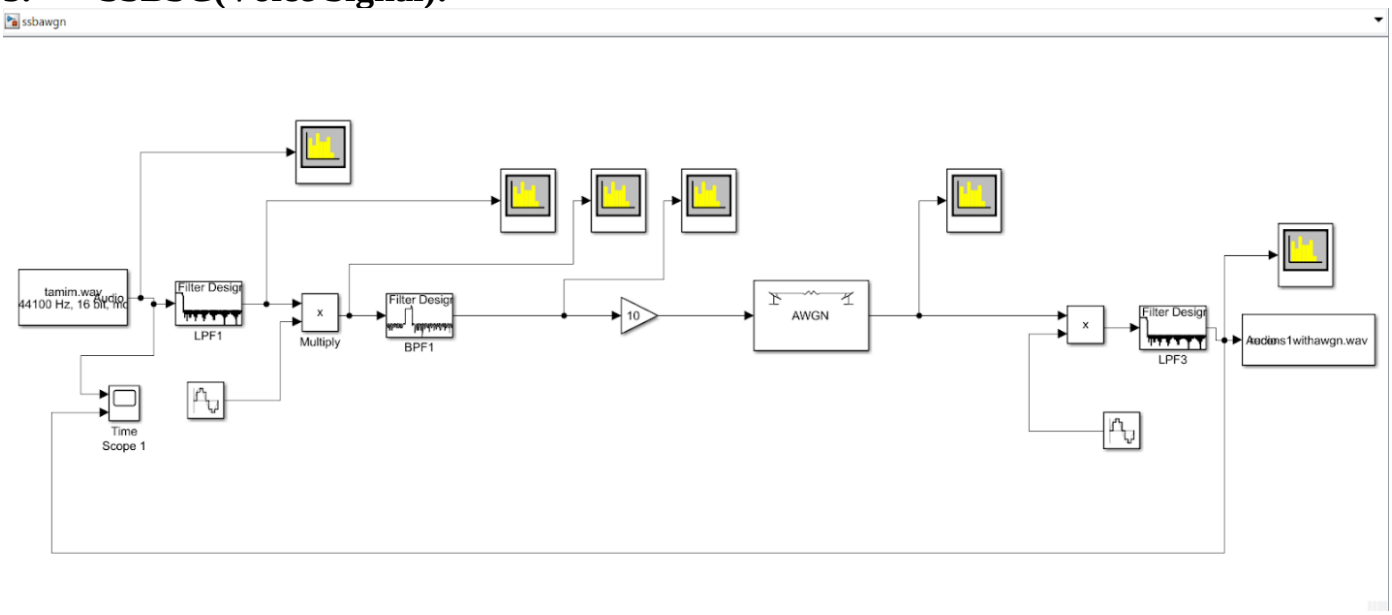
1. **DSB-WC (Sinusoid Signal):**



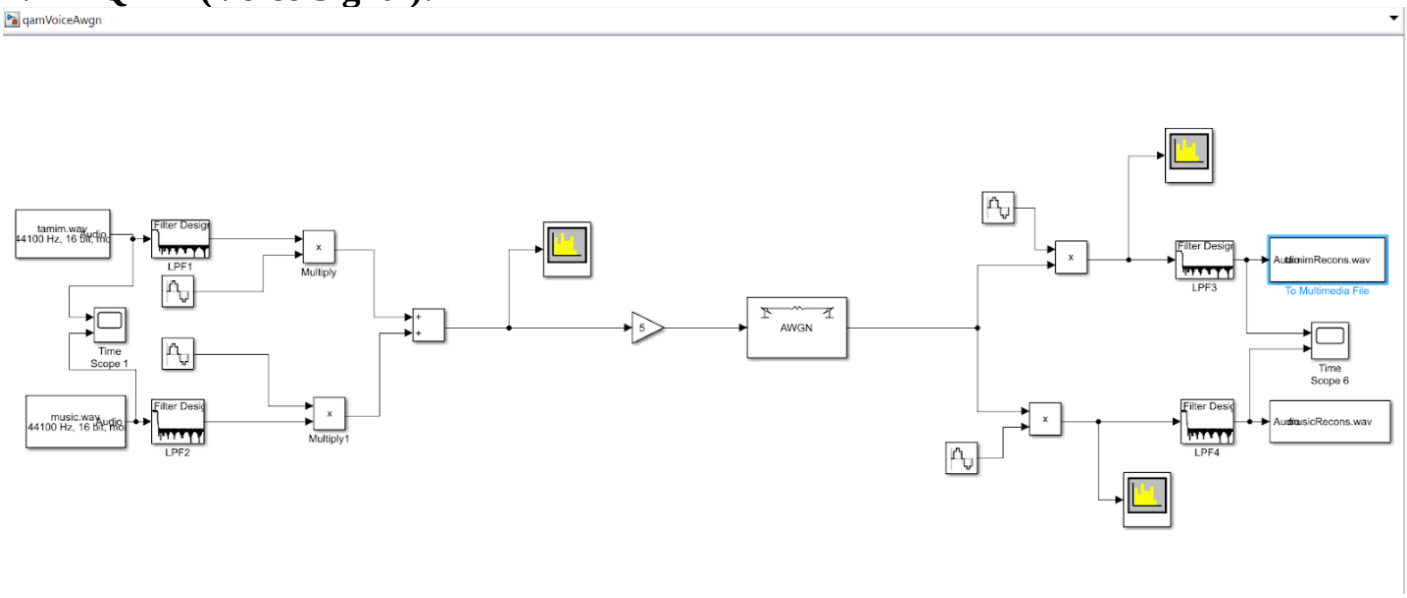
2. VSBSC (Sinusoid Signal):



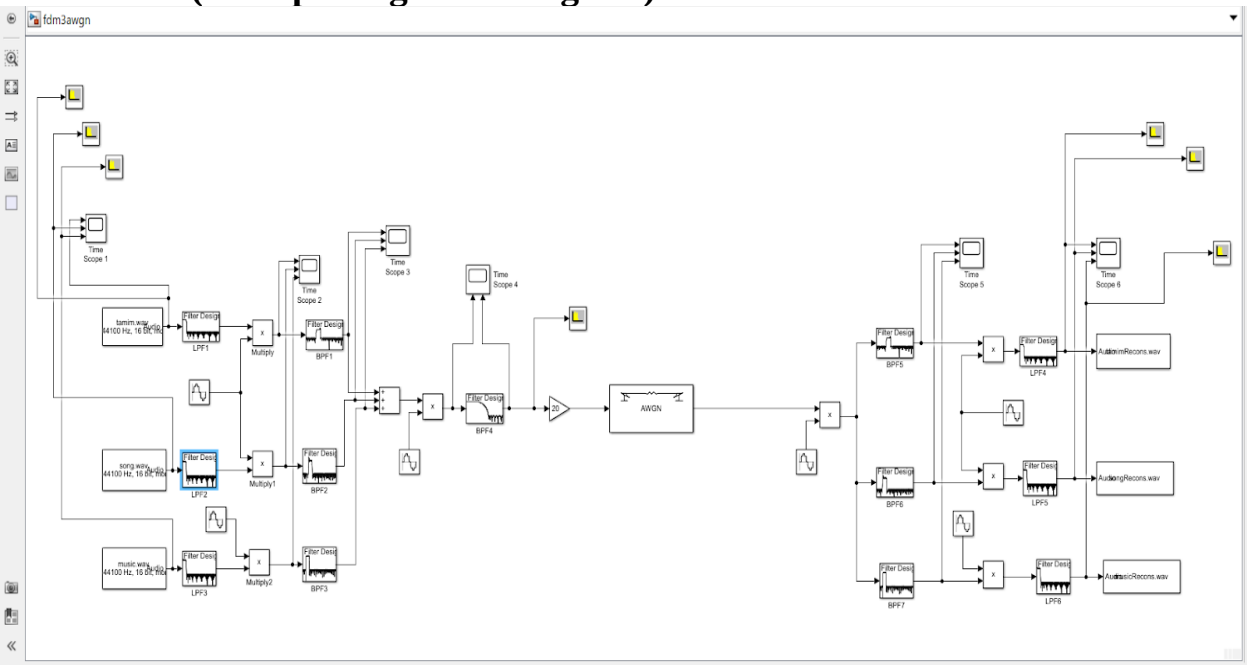
3. SSBSC(Voice Signal):



4. QAM (Voice Signal):



5. FDM (Multiplexing 3 voice signals):



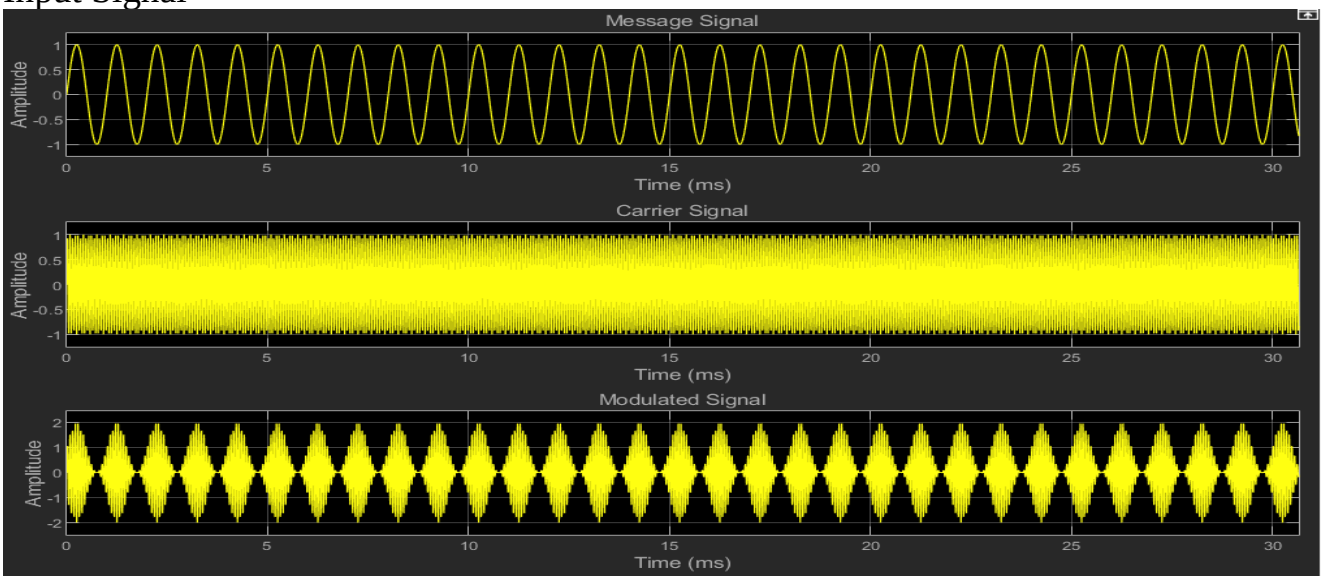
Simulink input and outputs:

We are presenting output for the same input signal from different simulink block together so that we can compare.

1. Sine wave (1000 Hz)

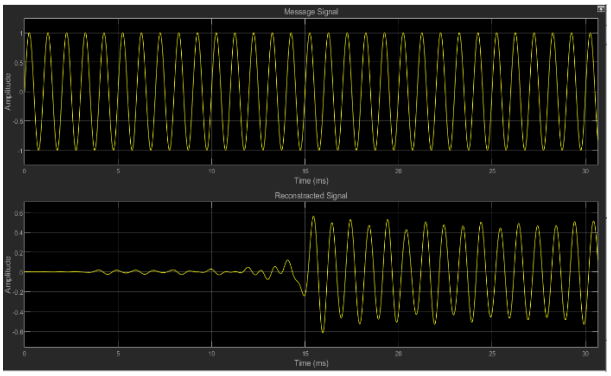
Time Domain:

Input Signal

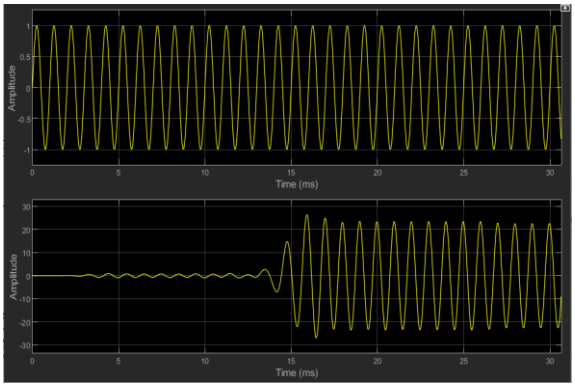


Reconstructed signals

DSB



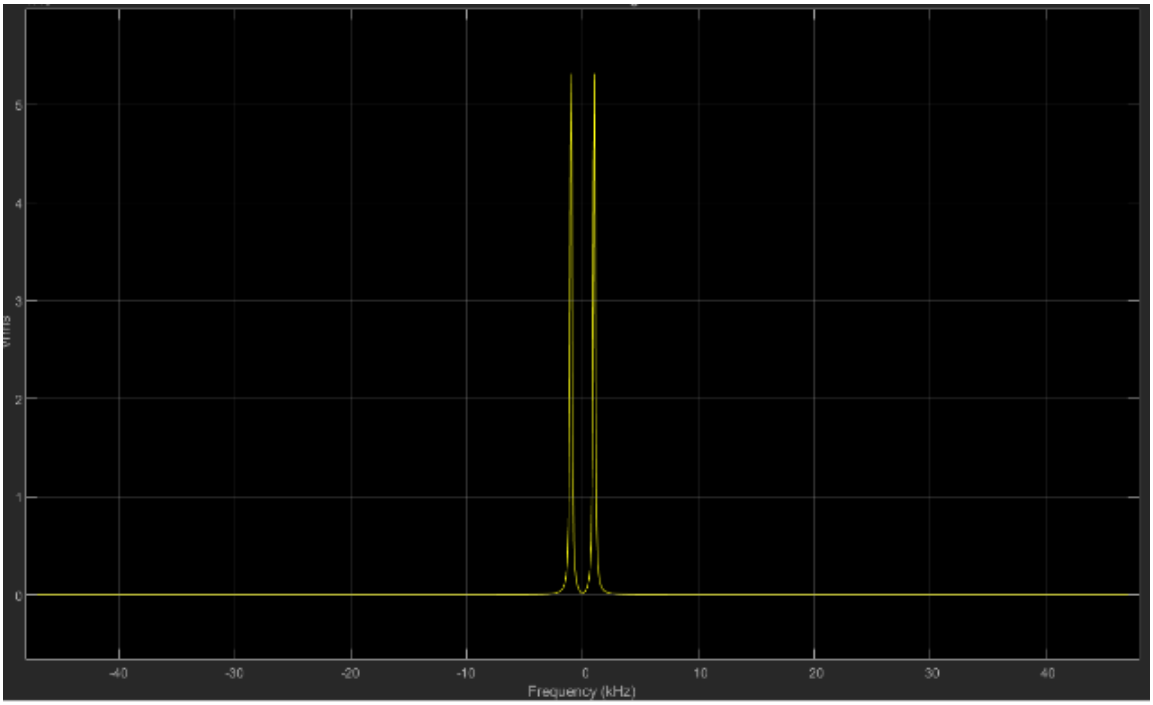
VSB



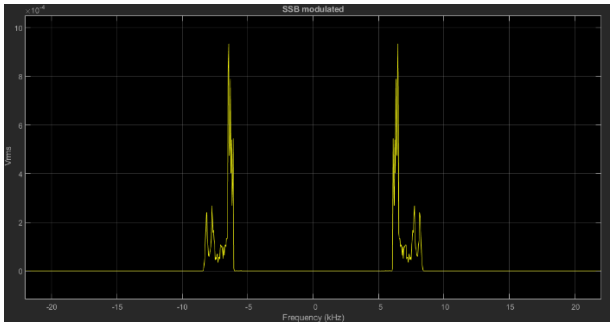
Frequency Domain:

Modulated signal

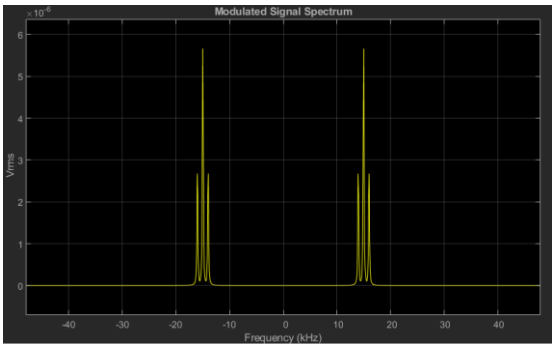
Input signal



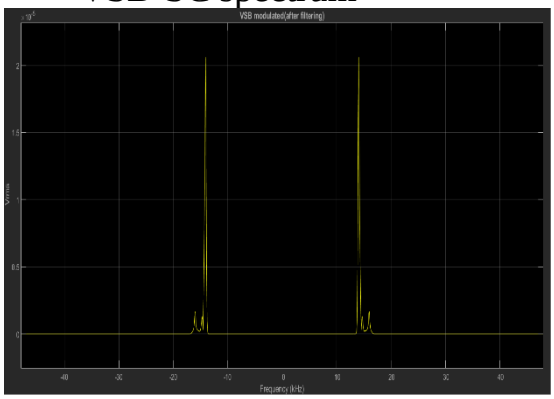
SSB-SC spectrum



DSB-SC spectrum

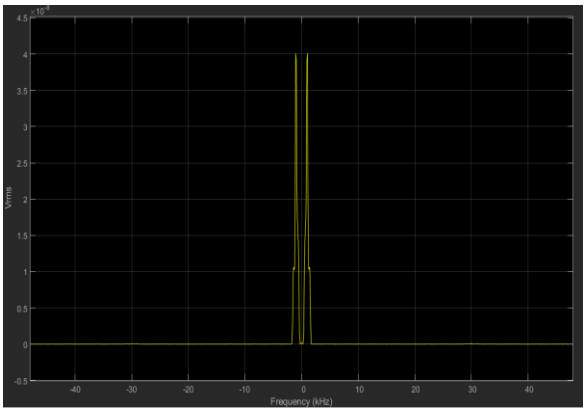


VSB-SC spectrum

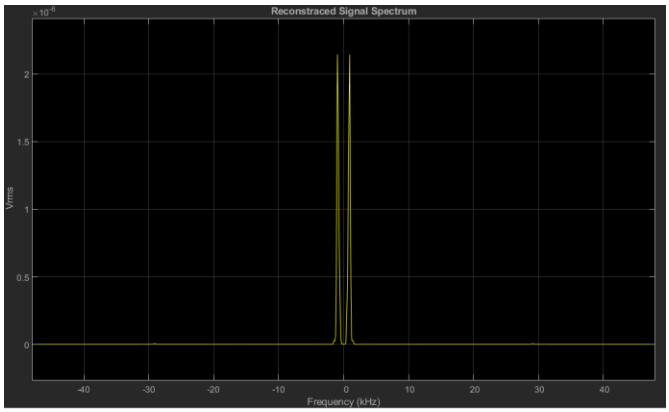


Reconstructed spectrum

DSB

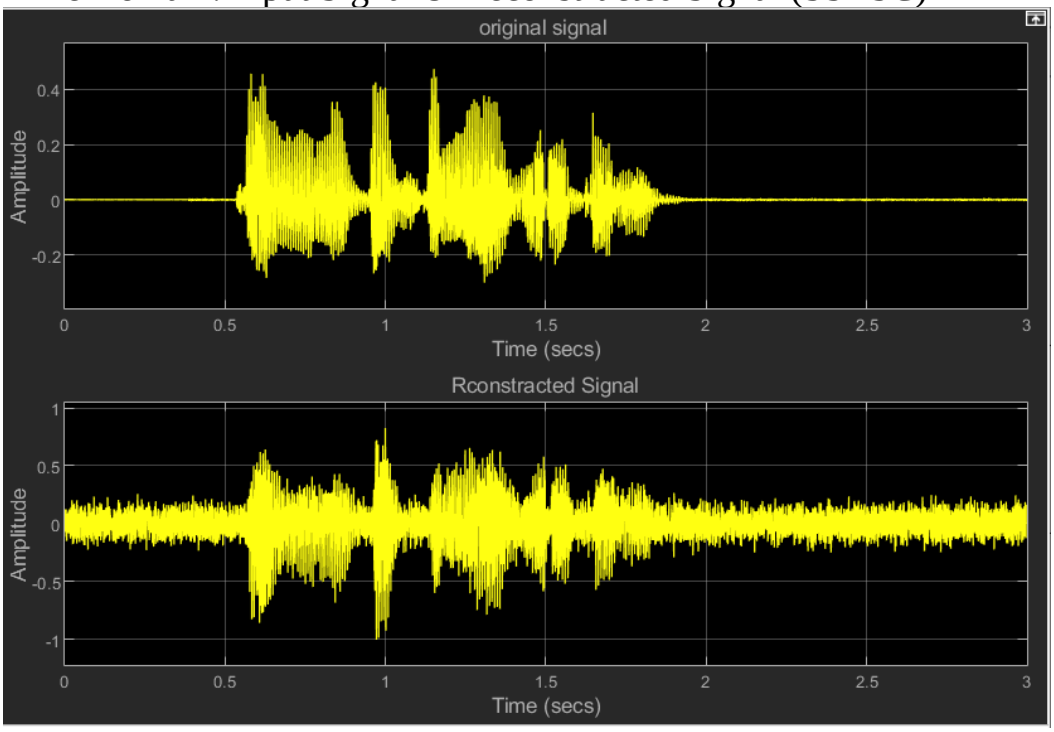


VSB

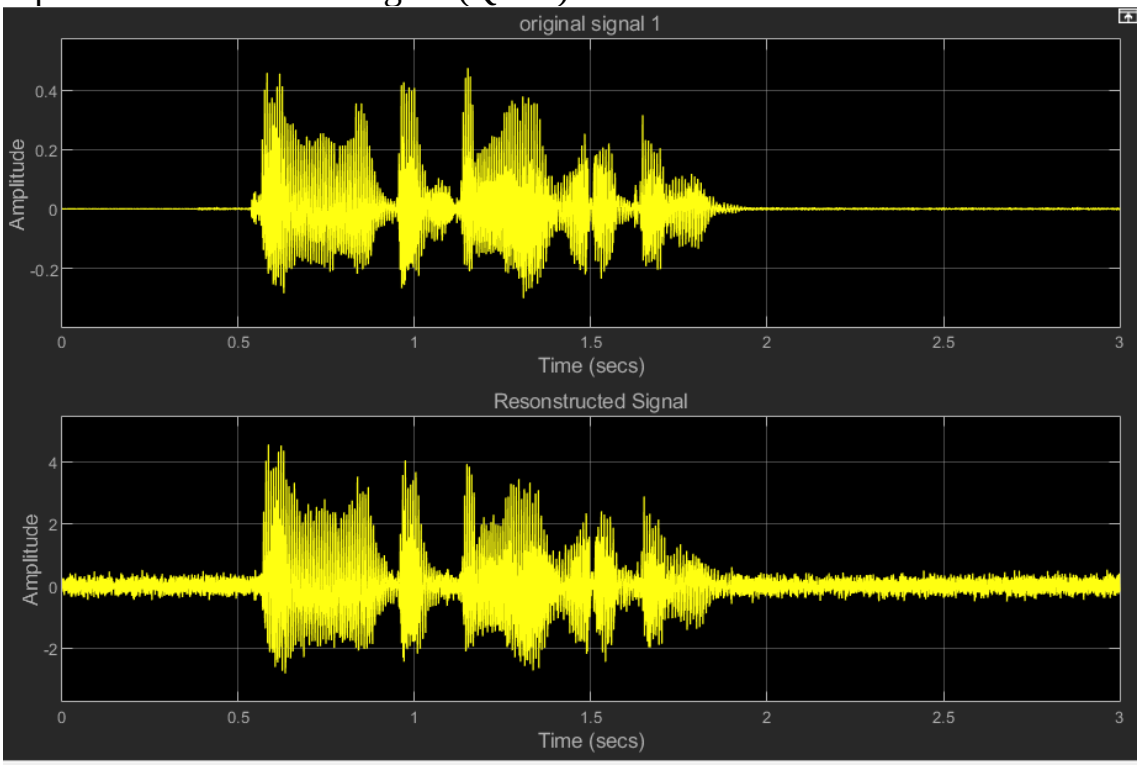


2. **Wave (Voice):**

Time Domain: Input Signal & Reconstructed Signal (SSBSC)

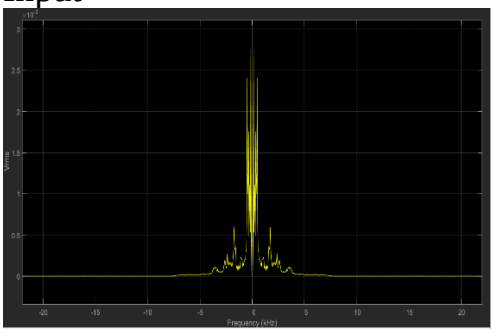


Input & Reconstructed Signal (QAM):

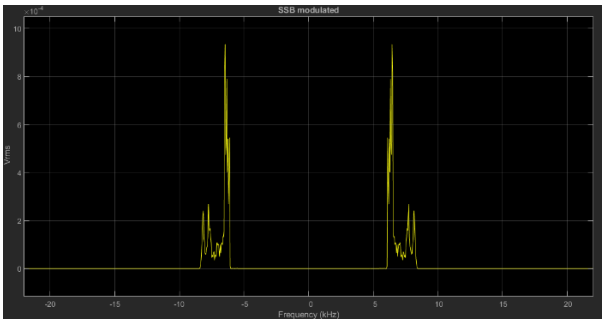


Frequency domain:
Modulated Spectrum

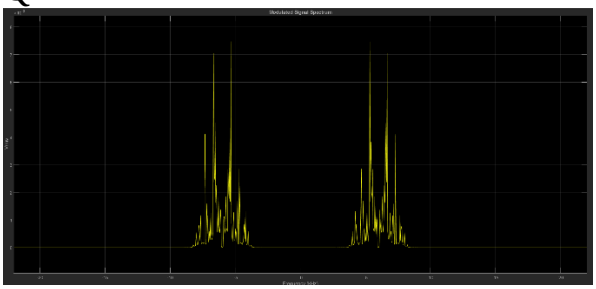
Input



SSBSC

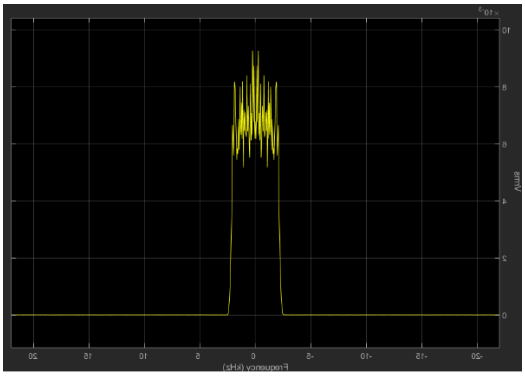


QAM

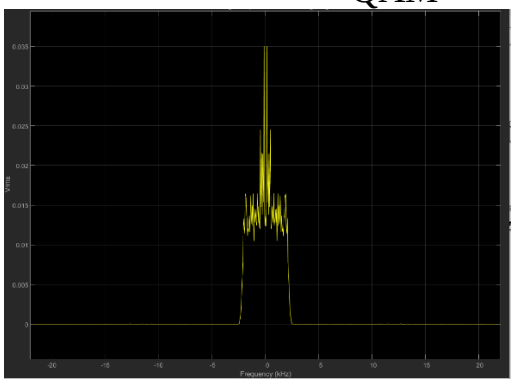


Reconstructed Spectrum

SSBSC

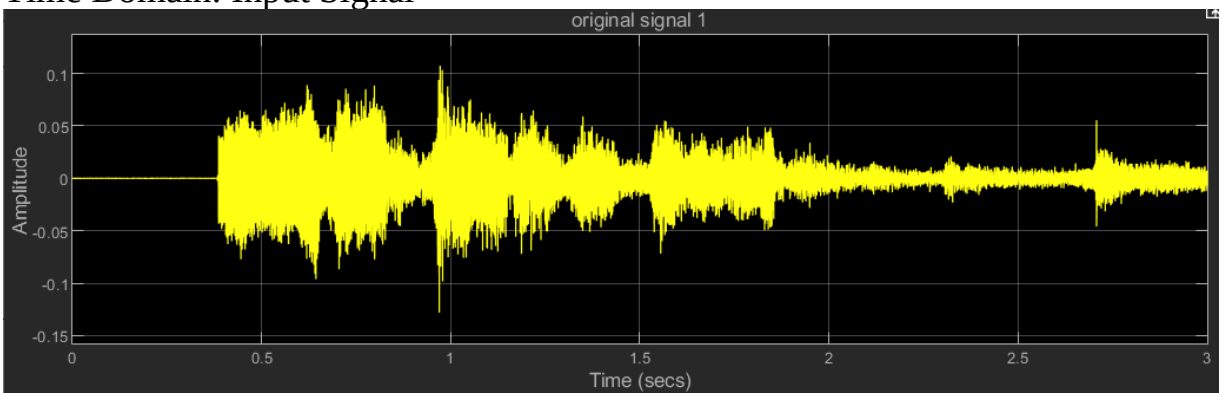


QAM



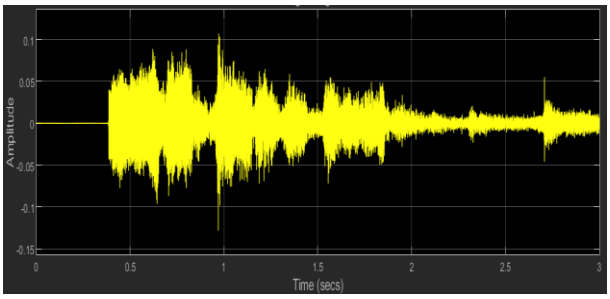
Song:

Time Domain: Input Signal

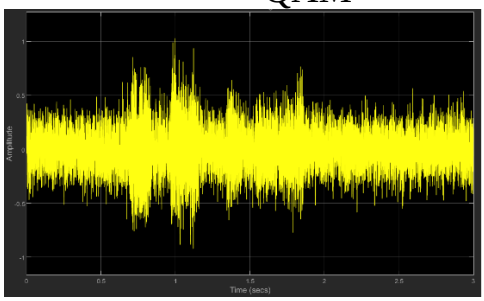


Reconstructed Signal

SSBSC



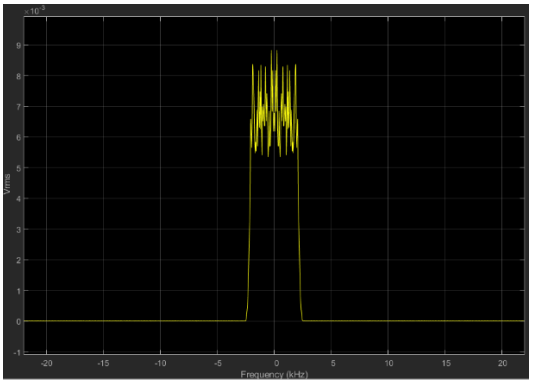
QAM



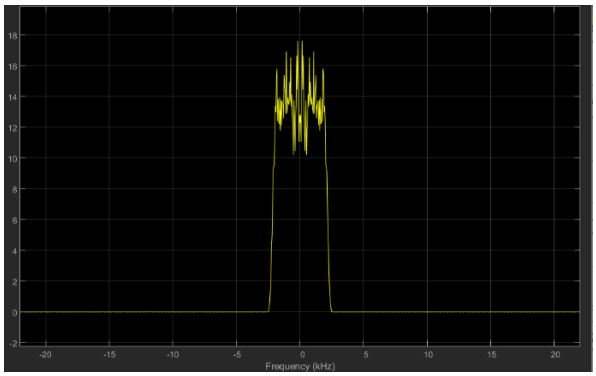
Frequency Domain:

Reconstructed

SSBSC

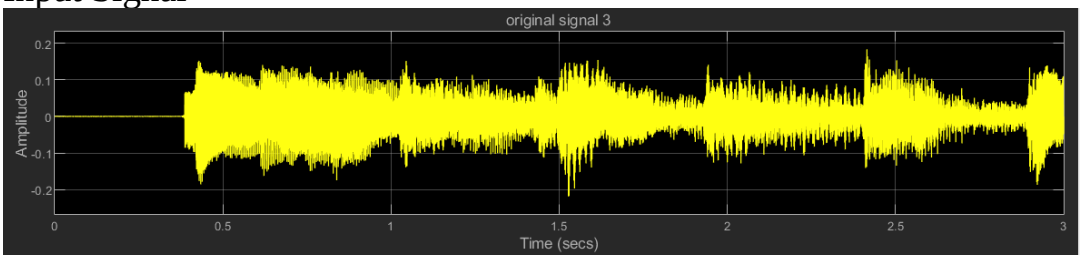


QAM

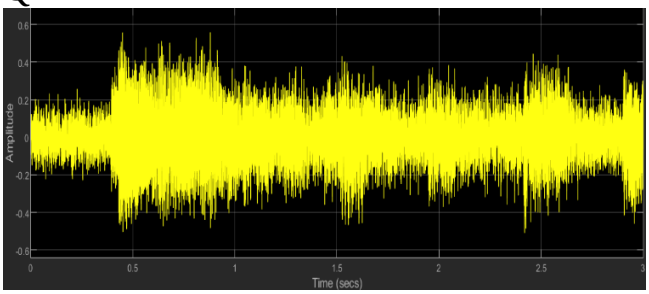


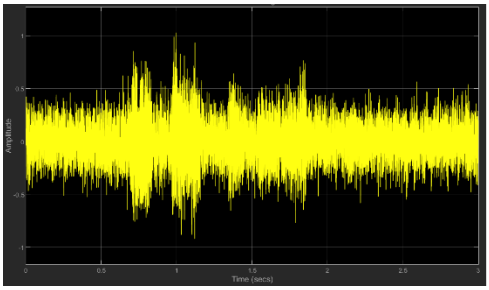
Music:

Time domain
Input Signal



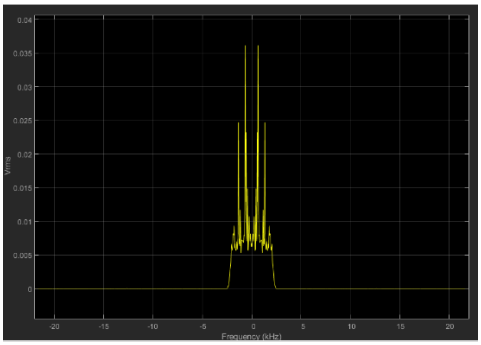
Reconstructed Signal:
SSBSC
QAM



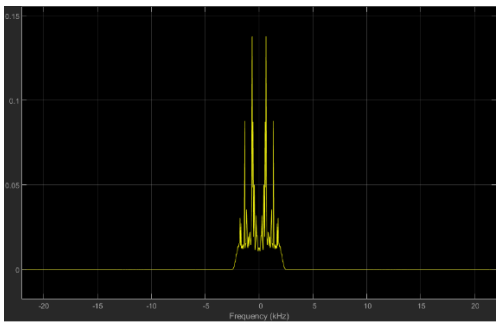


Frequency domain:

Reconstructed:
SSBSC



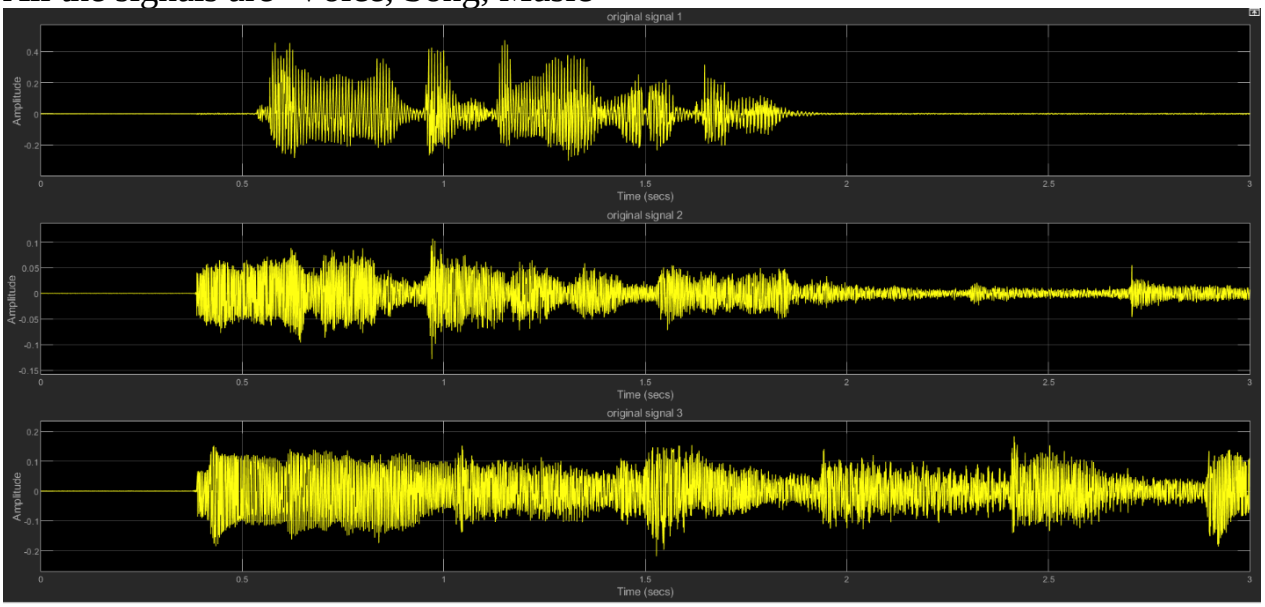
QAM



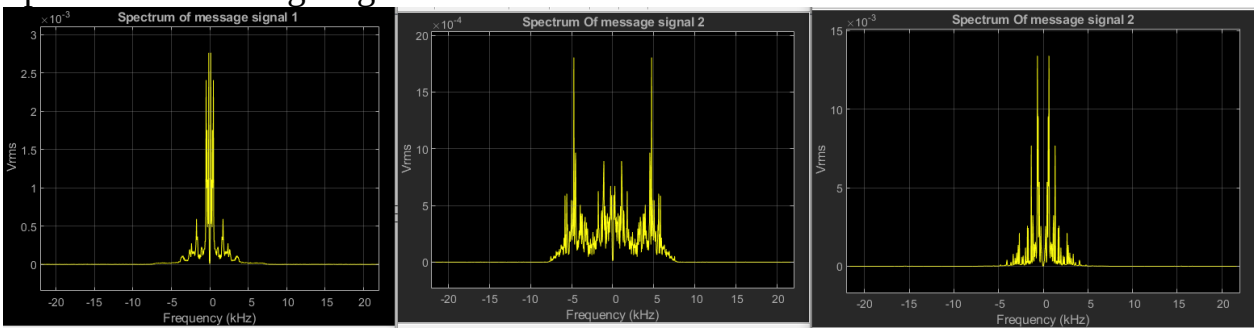
FDM:

Input 3 signals (Voice, Song, Music)
In FDM we use SSBSC modulation techniques
We use two carriers in order to modulate in desired frequency range

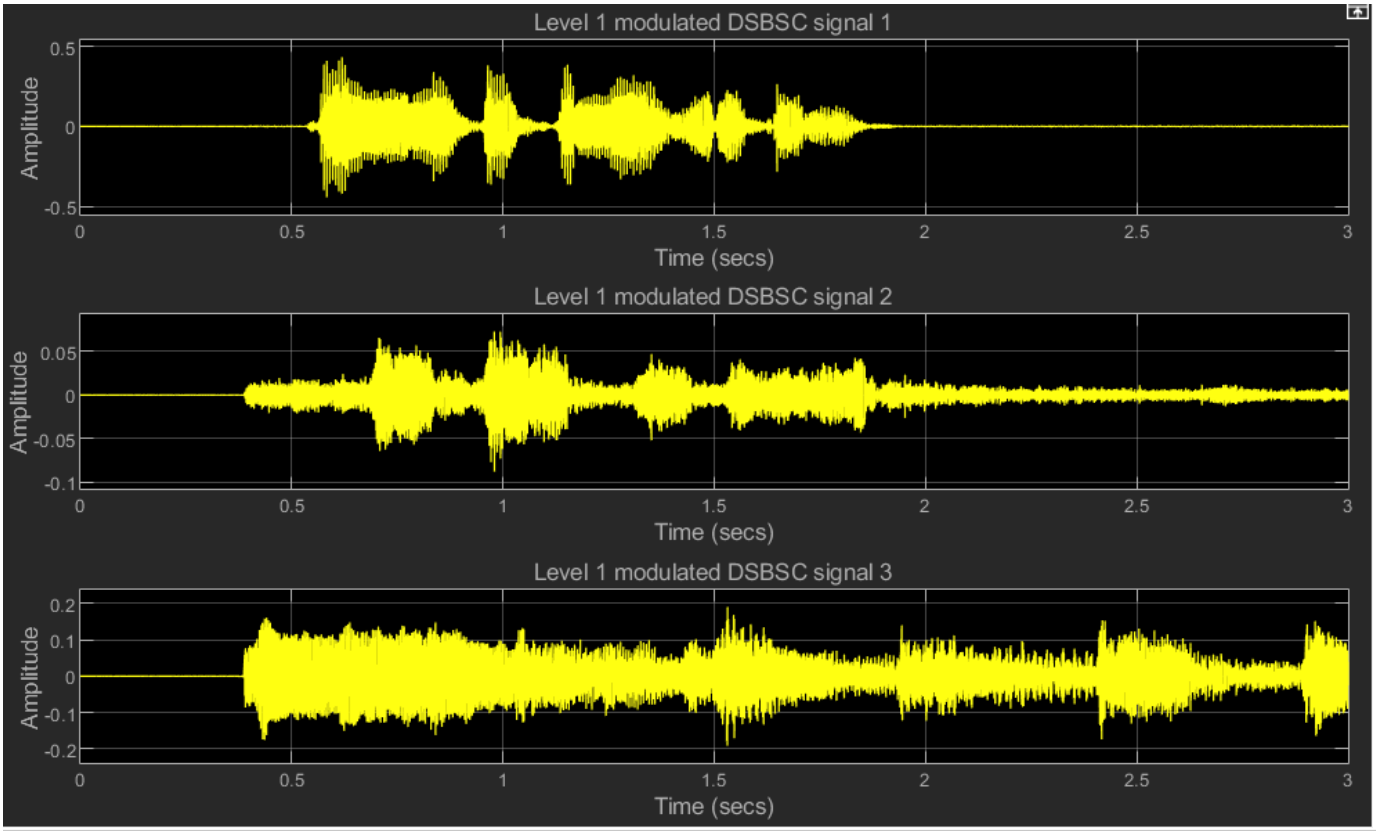
All the signals are- Voice, Song, Music

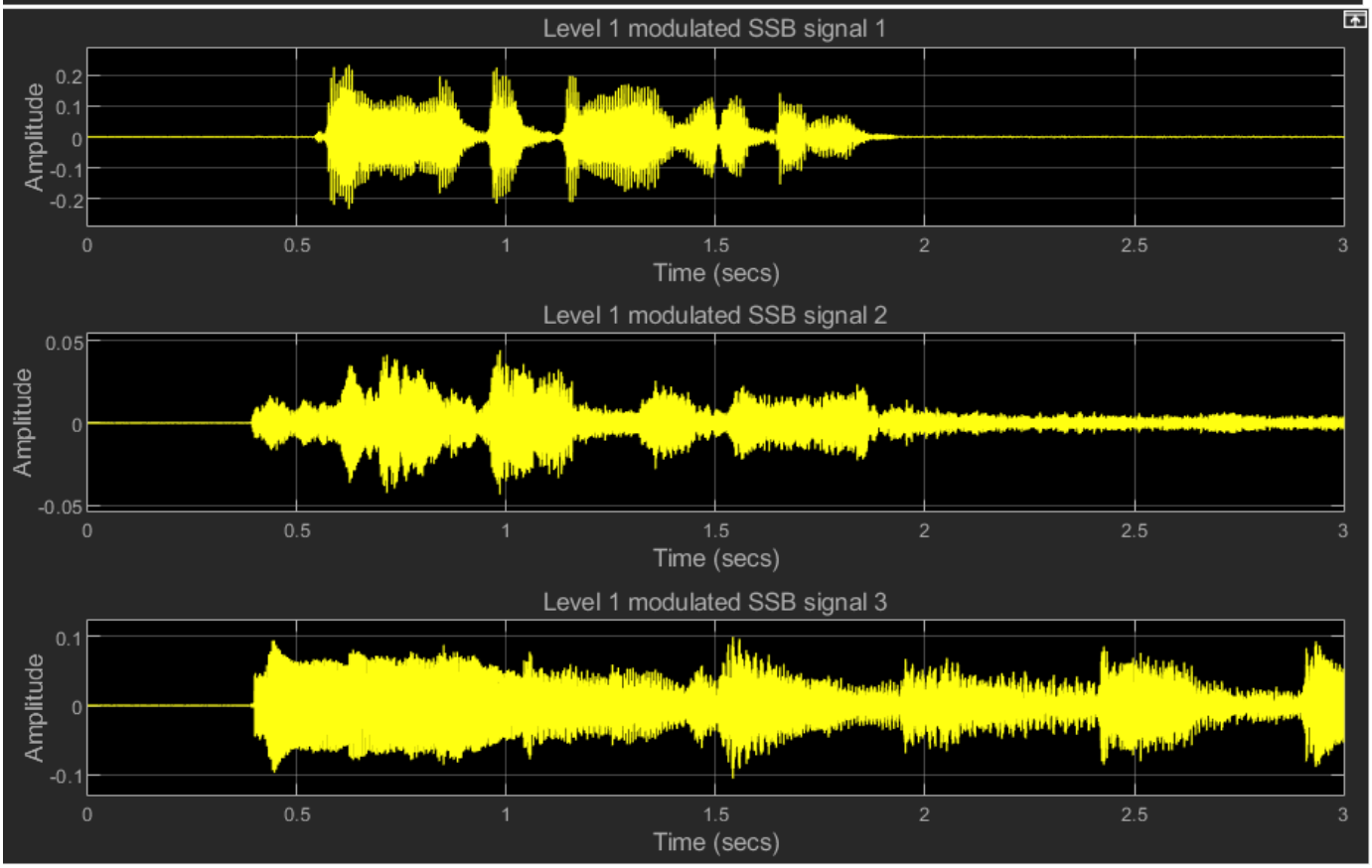


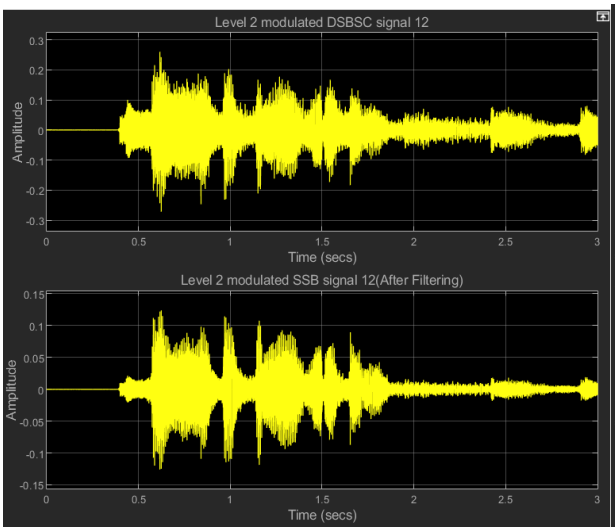
Spectrum of message signals

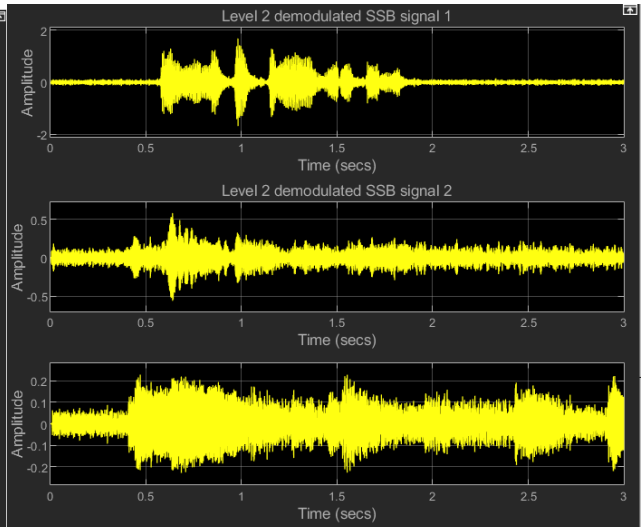


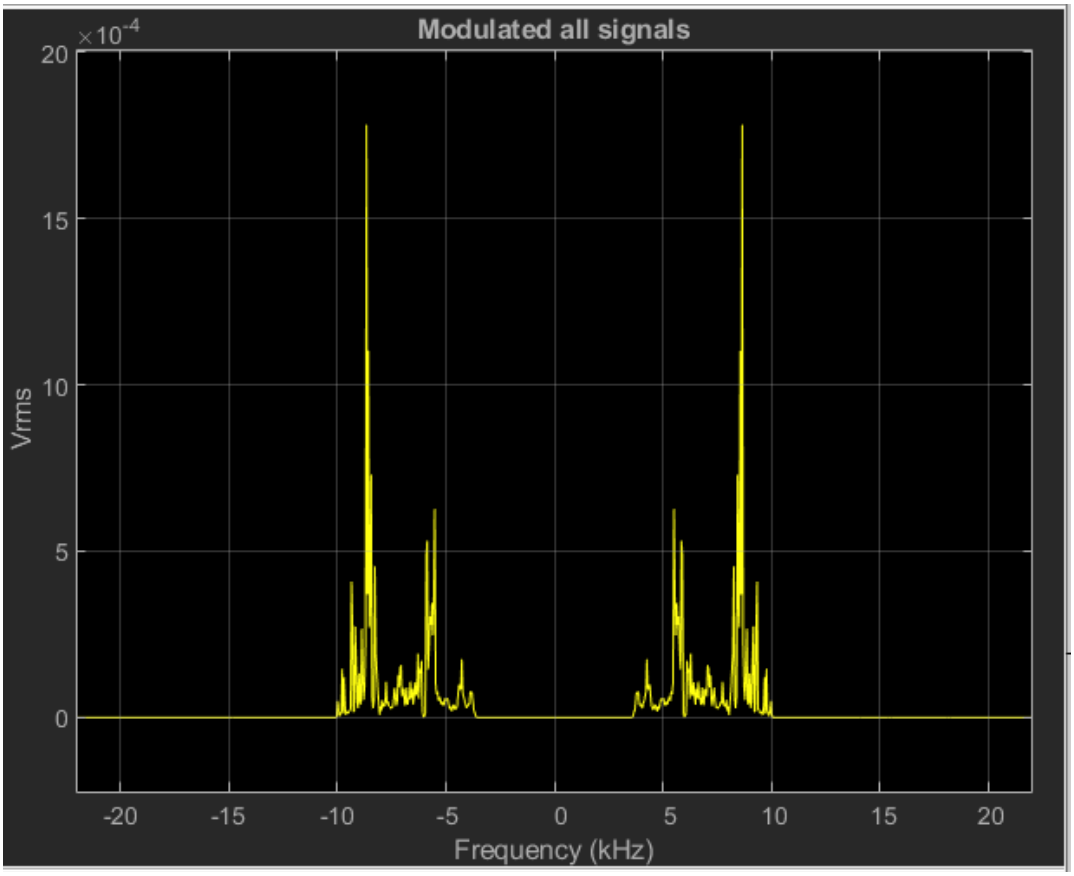
Creating DSB and then modulate again to get SSB, then it passes through an Bandpass Filter


Level 1 modulated DSBSC signal 1
Amplitude
Time (secs)
Level 1 modulated DSBSC signal 2
Amplitude
Time (secs)
Level 1 modulated DSBSC signal 3
Amplitude
Time (secs)

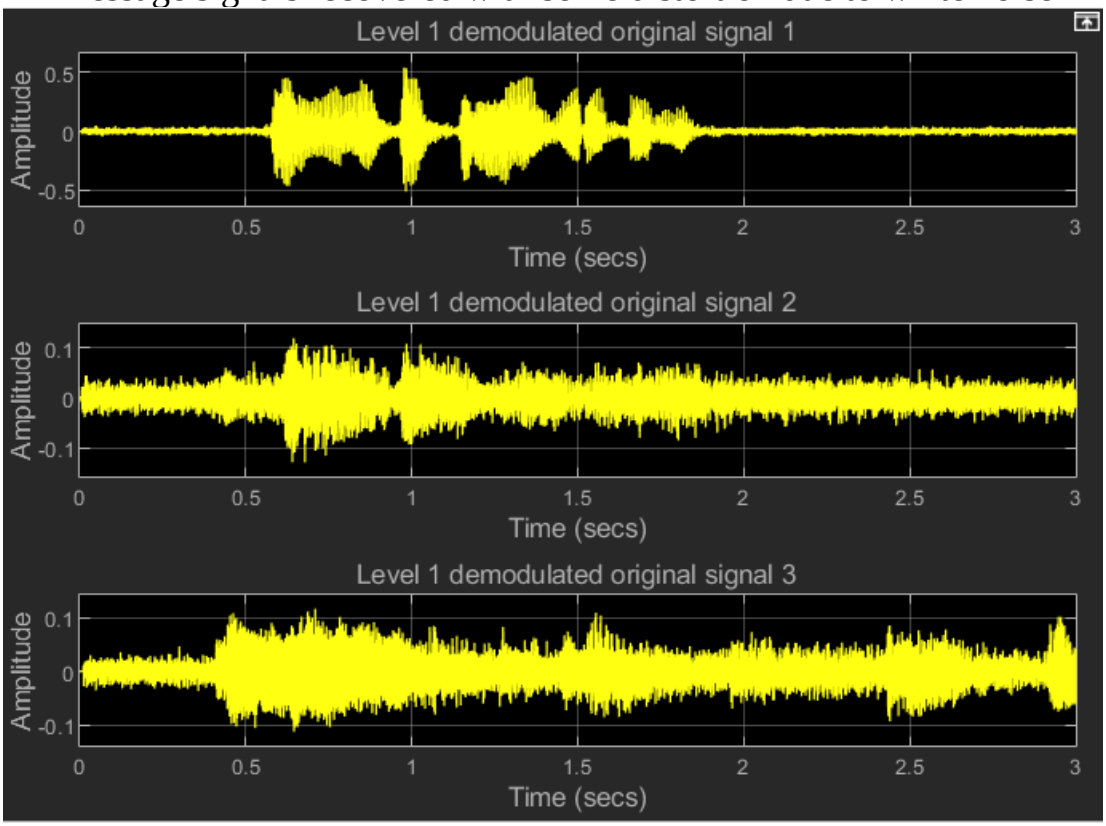

Level 1 modulated SSB signal 1
Amplitude
Time (secs)
Level 1 modulated SSB signal 2
Amplitude
Time (secs)
Level 1 modulated SSB signal 3
Amplitude
Time (secs)


Level 2 modulated DSBSC signal 12
Amplitude
Time (secs)
Level 2 modulated SSB signal 12(After Filtering)
Amplitude
Time (secs)


Level 2 demodulated SSB signal 1
Amplitude
Time (secs)
Level 2 demodulated SSB signal 2
Amplitude
Time (secs)
Amplitude
Time (secs)


Modulated all signals
×10^-4
Vrms
Frequency (kHz)

Output:
All message signals recovered with some distortion due to white noise



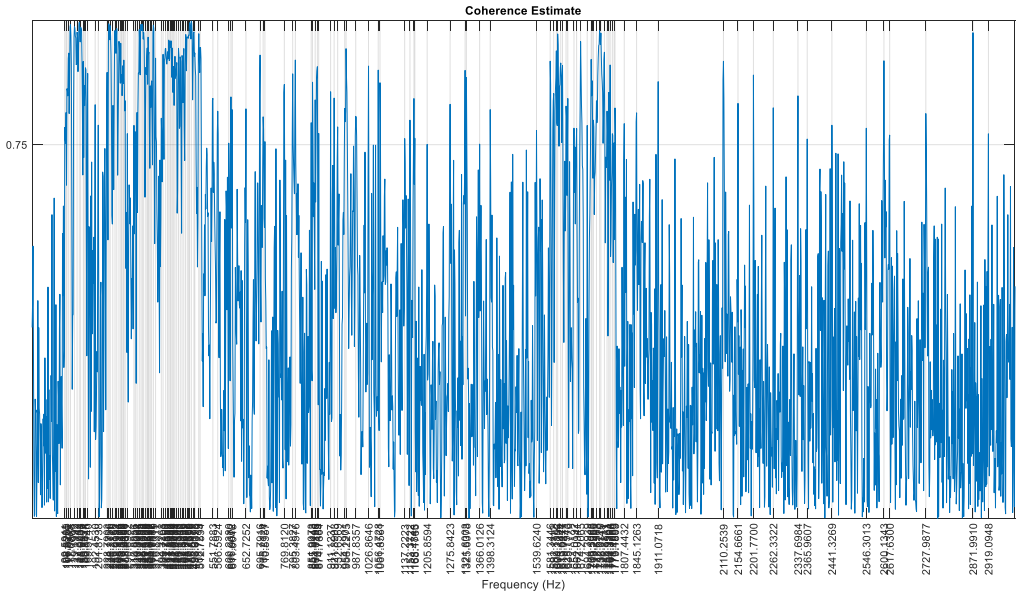
Result Analysis

From the images added in previous sections we noticed there is some difference in the frequency distribution of 3 types of audio data we used and that was the reason for using them, But all of them were reconstructed properly even if we filtered out the frequency values out of a certain range.

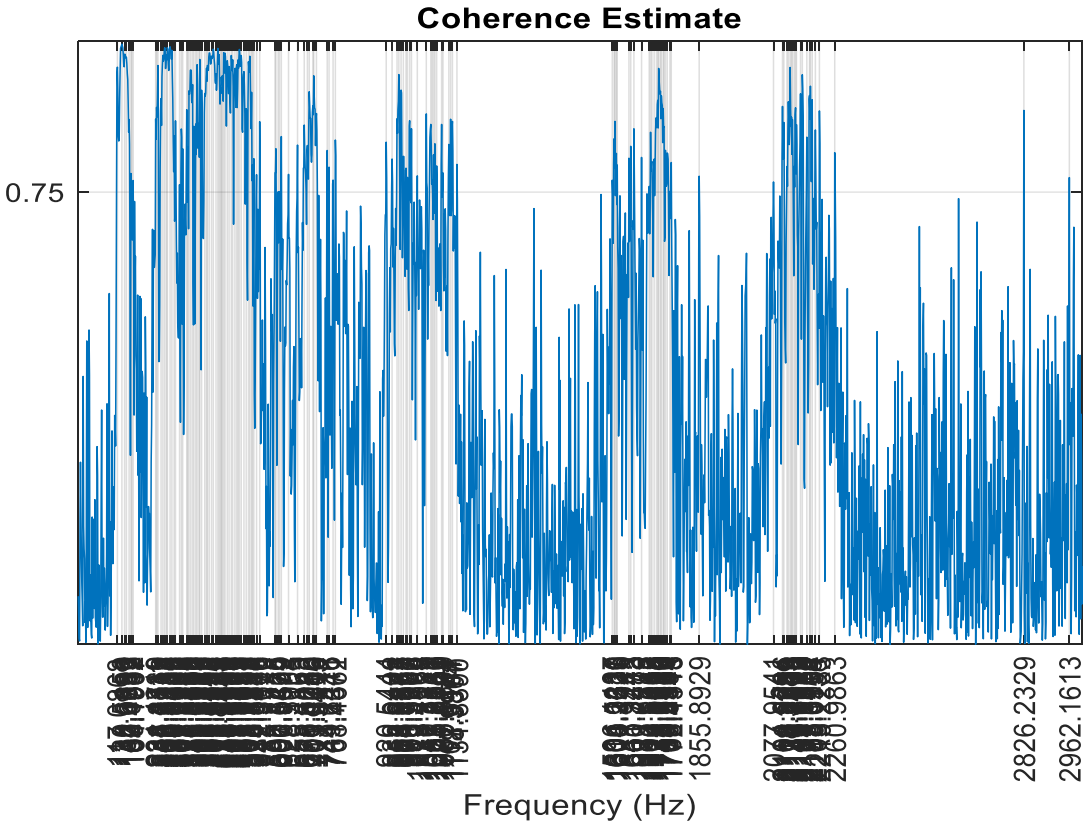
For further analysis, we observed similarities between the reconstructed and original signal and noticed their correlation coefficient.

Similarities analysis:

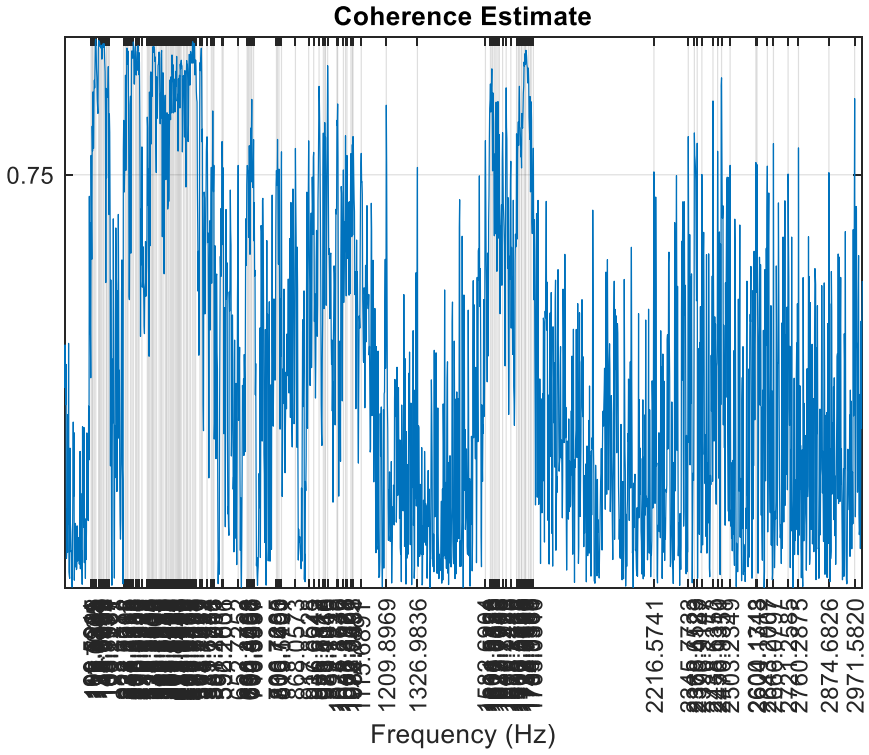
1. **DSB**



2. **SSB**

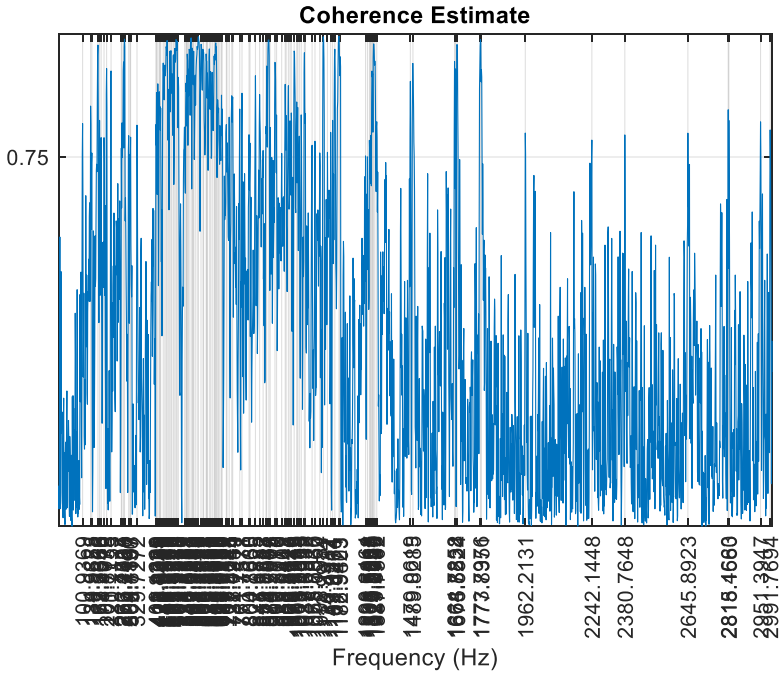


3. QAM

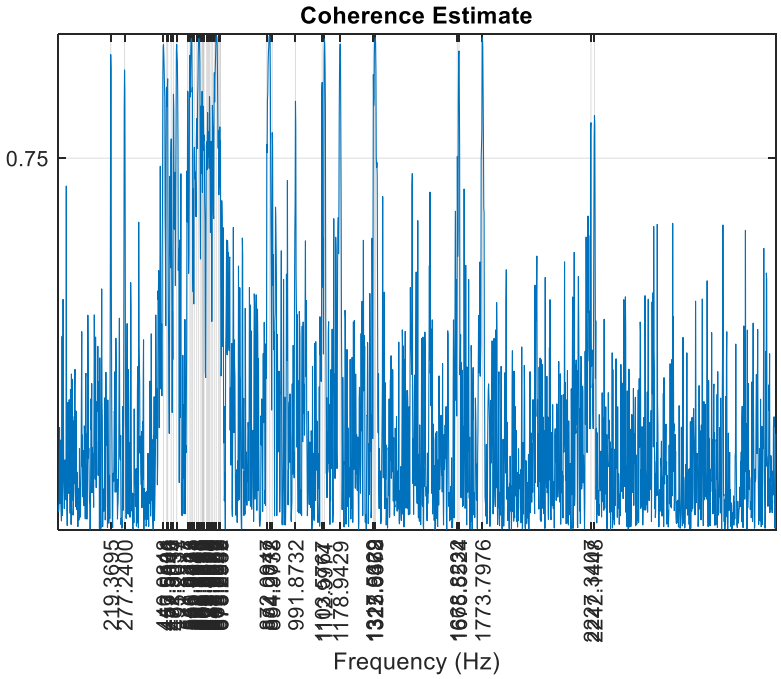


Music

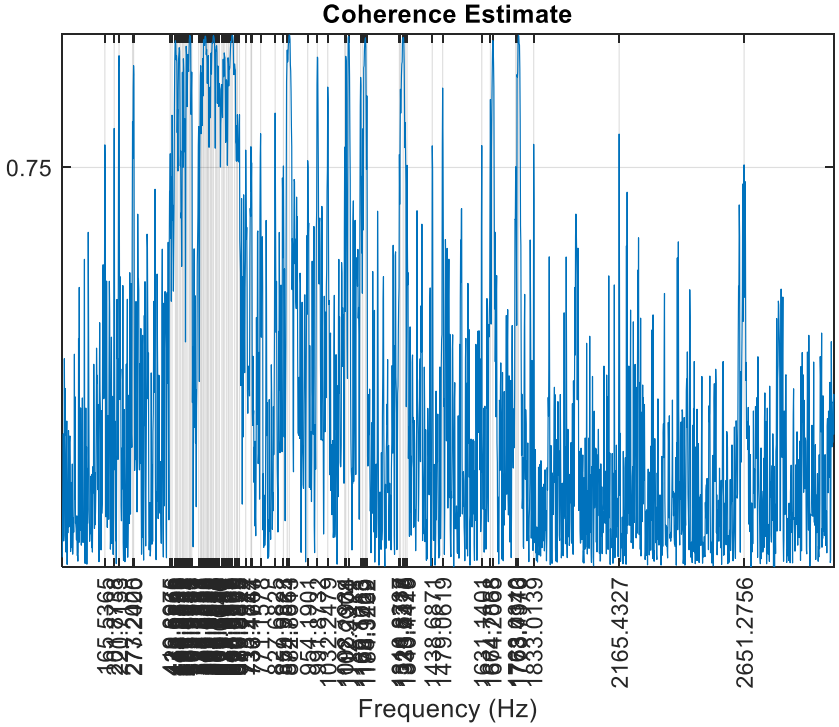
1. DSB



2. SSB

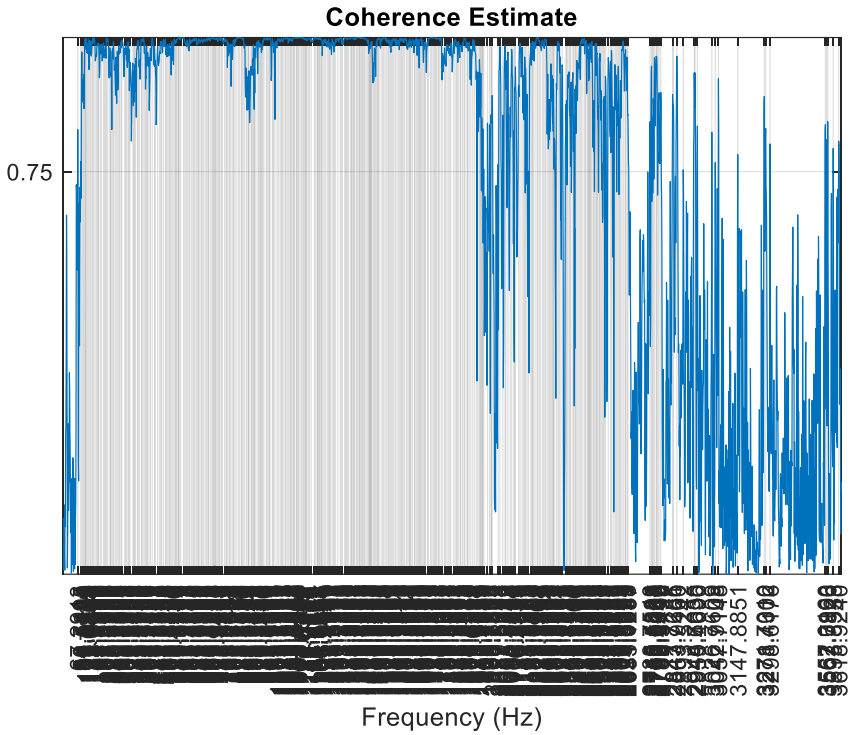

Coherence Estimate
0.75
991.8732
1178.9429
1773.7976
Frequency (Hz)

3. QAM

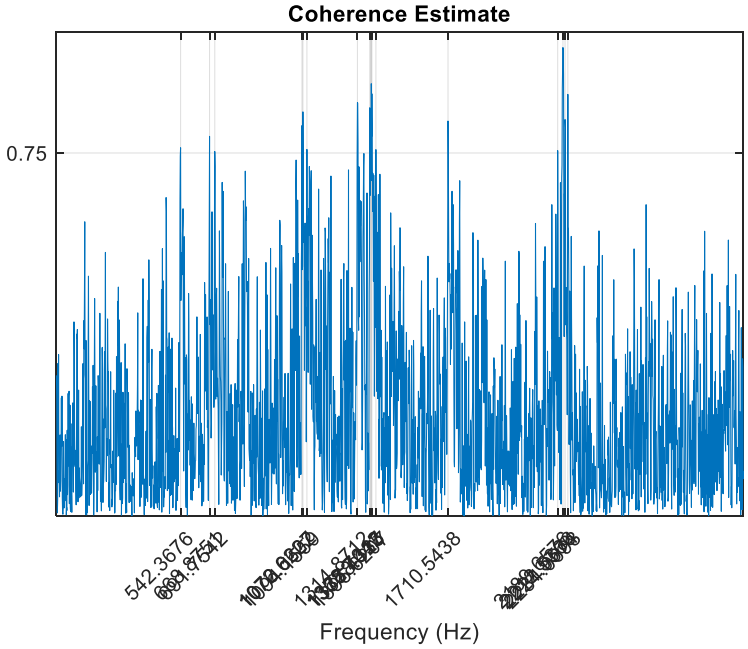

Coherence Estimate
0.75
1479.6871
2165.4327
2651.2756
Frequency (Hz)

Song

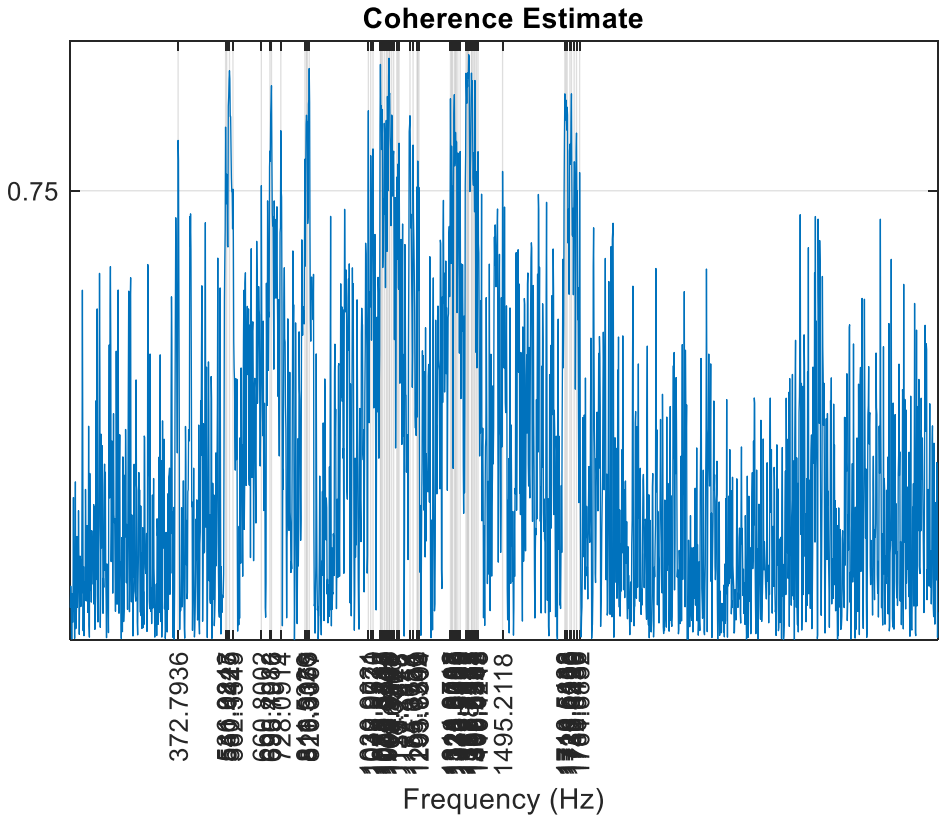
1. DSB



2. SSB



3. QAM



Comparison among correlation coefficient values:

correlation coefficient represents how strong is the relation between two variables. It ranges from -1 to 1. If the value is closer to 1 then the two variables are very related and if closer to 0 there is little relation in them, for -1 there seems to be a negative relation between them. As we have discarded the outlier frequencies in all of the cases, we expect to have lower values, but still it'd be a good comparison among the performance of the modulation technique used.

Signal used	DSB	SSB	QAM
Voice	0.5739	0.3104	-0.4125
Music	-0.2944	-0.5320	-0.3649
Song	-0.3657	0.0491	-0.0698

Challenges Faced

Before adding Noise the main problem was finding out the best possible values for filters used in our system, but they were solved easily. But then we noticed slight distortion in the frequency spectrum even before adding noise. We identified these to be because of the delay happening. As we know, every system has delay, and in the delayed part we notice some frequencies other than message signals, which distort the reconstructed signal in time domain. We couldn't got rid of them completely.

Removing noise was another challenge, for this we used an amplifier (10x) before transmission. It helped us get a good enough recovered signal but the noise was still there. Because of the noise there can be noticed discrepancy in the result obtained in the previous section.

We also faced size related problems in matlab. If we took larger values for Fs and took 10s or more data, the software crashed, and array size was out of its capability. So, we restricted ourselves within 10s and Bandwidth 3000 Hz for message signal and Fs = 16000. Thus, we needed to lose some of the data related frequencies of the outlier region before even the modulation process started.

From the table included above we can see for some processes we got a negative correlation coefficient, the reason for this being the signal being too noisy. We used 10x gain here, by increasing it more we can get good values for this.

Conclusion

In this project, we implemented different types of amplitude modulation and demodulation systems and analyzed their performance for different kinds of signals. We applied noise as channel loss and tried to solve them. While facing some unexpected situation we tried to come up with logical reasoning and tried to solve as much as we could.

References

1. https://www.mathworks.com/help/signal/ug/measuring-signal-similarities.html (for performance analysis)
2. https://uk.mathworks.com/help/signal/ug/compare-the-frequency-content-of-two-signals.html (for performance analysis)
3. https://documentation.meraki.com/MR/WiFi_Basics_and_Best_Practices/Signal-to-Noise_Ratio_(SNR)_and_Wireless_Signal_Strength (for SNR value range)

Appendix (Matlab code used)

DSB-SC(TRIANGLE)

```
clc;
clear all;
close all;
ts=1.e-4
t=-0.04:ts:0.04;
Ta=0.01;
m_sig=triangl((t+0.01)/0.01)-triangl((t-0.01)/0.01);
Lm_sig=length(m_sig);
Lfft=length(t);
Lfft=2^ceil(log2(Lfft));
M_fre=fftshift(fft(m_sig,Lfft));
freqm=(-Lfft/2:Lfft/2-1)/(Lfft*ts);
B_m=150;
%h=fir1(40, [B_m*ts])
h=fir1(40,.015);
t=-0.04:ts:0.04;
Ta=0.01;
fc=300;
s_dsb=m_sig.*cos(2*pi*fc*t);
Lfft=length(t);
Lfft=2^ceil(log2(Lfft)+1);
S_dsb=fftshift(fft(s_dsb,Lfft));
freqs=(-Lfft/2:Lfft/2-1)/(Lfft*ts);
s_dem=s_dsb.*cos(2*pi*fc*t)*2;
S_dem=fftshift(fft(s_dem,Lfft));

s_rec=filter(h,1,s_dem);
S_rec=fftshift(fft(s_rec,Lfft));

Trange=[-0.025 0.025 -2 2];
figure(1)
subplot(221); td1=plot(t,m_sig);
axis(Trange); set(td1,'Linewidth',2);
xlabel('{\it t} (sec)');
ylabel('{\it m}({\it t})')
title('message signal');
subplot(222); td2=plot(t,s_dsb);
axis(Trange);
set(td2,'Linewidth',2);
xlabel('{it t} (sec)');
ylabel('{\it s}_{\it s}_{\rm DSB}({\it t})')
subplot(223); td3=plot(t,s_dem);
axis(Trange);
set(td3,'Linewidth',2);
xlabel('{it t} (sec)');
ylabel('{\it e} ({\it t})')
title('{\it e}({\it t})');
subplot(224); td4=plot(t,s_rec);
axis(Trange);
set(td4,'Linewidth',2);
xlabel('{it t} (sec)');
ylabel('{\it m}_d({\it t})')
title('Recovered signal');

Frange =[-700 700 0 200]
figure(2)
subplot(221);
fd1=plot(freqm,abs(M_fre));
axis(Frange); set(fd1,'Linewidth',2);
xlabel('{\it f} (Hz)');
ylabel('{\it M}({\it f})')
title('message spectrum')
subplot(222);fd2=plot(freqs,abs(S_dsb));
axis(Frange);set(fd2,'Linewidth',2);
xlabel('{\it f} (Hz)');
ylabel('{\it S}_{rm DSB} ({\it f})')
title('DSB-SC specttrum')
subplot(223);
fd3=plot(freqs,abs(S_dem));
axis(Frange); set(fd3,'Linewidth',2);
xlabel('{\it f} (Hz)');
ylabel('{\it E}({\it f})');
title('spectrum of {\it e}({\it t})')
subplot(224);
fd4=plot(freqs,abs(S_rec));
axis(Frange); set(fd4,'Linewidth',2);
xlabel('{\it f} (Hz)');

ylabel('{\it M}_d({\it f})');
title('recovered spectrum')


DSB-SC(VOICE)

clc;
clear all;
close all;
%% Input
[y, Fs] = audioread('tamimDSB.wav');
y = y';
y=y(1,:);
n = length(y);
t = 0:1/Fs:(n-1)*(1/Fs);
m_sig=y;
bw=3400;
ts=1/Fs;
Lm_sig=length(m_sig);
Lfft=length(t);
Lfft=2^ceil(log2(Lfft));
M_fre=fftshift(fft(m_sig,Lfft));
freqm=(-Lfft/2:Lfft/2-1)/(Lfft*ts);
h=fir1(40, [bw*ts]);
%% Modulation
fc=4000;
s_dsb=m_sig.*cos(2*pi*fc*t);
Lfft=length(t);
Lfft=2^ceil(log2(Lfft)+1);
S_dsb=fftshift(fft(s_dsb,Lfft));
%% Channel
s_ch=awgn(100*s_dsb,20);

%% SIgnal recieve
fr=[2*4300/Fs 2*7400/Fs];
s_rcv=bandpass(s_ch,fr);


%% Demodulation
freqs=(-Lfft/2:Lfft/2-1)/(Lfft*ts);
s_dem=s_rcv.*cos(2*pi*fc*t);
S_dem=fftshift(fft(s_dem,Lfft));
s_rec=filter(h,1,s_dem);
S_rec=fftshift(fft(s_rec,Lfft));

%% Plot
figure(1)
subplot(221); td1=plot(t,m_sig);
xlabel('{\it t} (sec)');
ylabel('{\it m}({\it t})')
title('message signal');

subplot(222); td2=plot(t,s_dsb);
```

```
xlabel('{it t} (sec)');
ylabel('{\it s}_{\it s}_{\rm DSB}({\it t})')

subplot(223); td3=plot(t,s_dem);
xlabel('{it t} (sec)');
ylabel('{\it e} ({\it t})')
title('{\it e}({\it t})');

subplot(224); td4=plot(t,s_rec);
xlabel('{it t} (sec)');
ylabel('{\it m}_d({\it t})')
title('Recovered signal');


figure(2)
subplot(221);
fd1=plot(freqm,abs(M_fre));
xlabel('{\it f} (Hz)');
ylabel('{\it M}({\it f})')
title('message spectrum')

subplot(222);fd2=plot(freqs,abs(S_dsb));
xlabel('{\it f} (Hz)');
ylabel('{\it S}_{rm DSB} ({\it f})')
title('DSB-SC specttrum')

subplot(223);
fd3=plot(freqs,abs(S_dem));
xlabel('{\it f} (Hz)');
ylabel('{\it E}({\it f})');
title('spectrum of {\it e}({\it t})')

subplot(224);
fd4=plot(freqs,abs(S_rec));
xlabel('{\it f} (Hz)');
ylabel('{\it M}_d({\it f})');
title('recovered spectrum')
%% Output
sound(s_rec,Fs)
```

DSB-WC(TRIANGLE)

```
clc;
clear all;
close all;
ts=1.e-4
t=-0.04:ts:0.04;
Ta=0.01;
m_sig=triangl((t+0.01)/0.01)-triangl((t-
0.01)/0.01);
Lm_sig=length(m_sig);
Lfft=length(t);
Lfft=2^ceil(log2(Lfft));
M_fre=fftshift(fft(m_sig,Lfft));
freqm=(-Lfft/2:Lfft/2-1)/(Lfft*ts);
B_m=150;
%h=fir1(40, [B_m*ts])
h=fir1(40,.015);
t=-0.04:ts:0.04;
Ta=0.01;
fc=300;
Ac=3;
carrier=Ac*cos(2*pi*fc*t);
s_dsb=carrier+m_sig.*cos(2*pi*fc*t);
Lfft=length(t);
Lfft=2^ceil(log2(Lfft)+1);
S_dsb=fftshift(fft(s_dsb,Lfft));
freqs=(-Lfft/2:Lfft/2-1)/(Lfft*ts);
s_dem=s_dsb.*cos(2*pi*fc*t)*2-
Ac*ones(1,length(s_dsb));
S_dem=fftshift(fft(s_dem,Lfft));

s_rec=filter(h,1,s_dem);
S_rec=fftshift(fft(s_rec,Lfft));

% Trange=[-0.025 0.025 -2 2];
figure(1)
subplot(221); td1=plot(t,m_sig);
% axis(Trange); set(td1,'Linewidth',2);
xlabel('{\it t} (sec)');
ylabel('{\it m}({\it t})')
title('message signal');
subplot(222); td2=plot(t,s_dsb);
% axis(Trange);
% set(td2,'Linewidth',2);
xlabel('{it t} (sec)');
ylabel('{\it s}_{\it s}_{\rm DSB}({\it t})')
subplot(223); td3=plot(t,s_dem);
% axis(Trange);
% set(td3,'Linewidth',2);
xlabel('{it t} (sec)');
ylabel('{\it e} ({\it t})')
title('{\it e}({\it t})');
subplot(224); td4=plot(t,s_rec);
% axis(Trange);
% set(td4,'Linewidth',2);
xlabel('{it t} (sec)');
ylabel('{\it m}_d({\it t})')
title('Recovered signal');

% Frange =[-700 700 0 200]
figure(2)
subplot(221);
fd1=plot(freqm,abs(M_fre));
% axis(Frange); set(fd1,'Linewidth',2);
xlabel('{\it f} (Hz)');
ylabel('{\it M}({\it f})')
title('message spectrum')
subplot(222);fd2=plot(freqs,abs(S_dsb));
% axis(Frange);set(fd2,'Linewidth',2);
xlabel('{\it f} (Hz)');
ylabel('{\it S}_{rm DSB} ({\it f})')
title('DSB-SC specttrum')
subplot(223);
fd3=plot(freqs,abs(S_dem));
% axis(Frange); set(fd3,'Linewidth',2);
xlabel('{\it f} (Hz)');
ylabel('{\it E}({\it f})');
title('spectrum of {\it e}({\it t})')
subplot(224);
fd4=plot(freqs,abs(S_rec));
% axis(Frange); set(fd4,'Linewidth',2);
xlabel('{\it f} (Hz)');

ylabel('{\it M}_d({\it f})');
title('recovered spectrum')
```

DSB-WC(VOICE)

```
clc;
clear all;
close all;
%% Input
[y, Fs] = audioread('tamimDSB.wav');
y = y';
n = length(y);
t = 0:1/Fs:(n-1)*(1/Fs);
ts=1/Fs;
m_sig=y;
Lm_sig=length(m_sig);
Lfft=length(t);
Lfft=2^ceil(log2(Lfft));
M_fre=fftshift(fft(m_sig,Lfft));
freqm=(-Lfft/2:Lfft/2-1)/(Lfft*ts);
bw=3400;
h=fir1(40, [bw*ts]);
%% Modulation
fc=4000;
Ac=1;
carrier=Ac*cos(2*pi*fc*t);
s_dsb=carrier+m_sig.*cos(2*pi*fc*t);
%% Channel
s_ch=awgn(100*s_dsb,20);

%% Reciever
fr=[2*4300/Fs 2*7400/Fs];
s_rcv=bandpass(s_ch,fr);


%% Demodulation
Lfft=length(t);
Lfft=2^ceil(log2(Lfft)+1);
S_dsb=fftshift(fft(s_dsb,Lfft));
freqs=(-Lfft/2:Lfft/2-1)/(Lfft*ts);
s_dem=s_rcv.*cos(2*pi*fc*t)-
Ac*ones(1,length(s_rcv));
S_dem=fftshift(fft(s_dem,Lfft));
s_rec=filter(h,1,s_dem);
S_rec=fftshift(fft(s_rec,Lfft));

%% Plot
figure(1)
subplot(221); td1=plot(t,m_sig);
xlabel('{\it t} (sec)');
ylabel('{\it m}({\it t})')
title('message signal');

subplot(222); td2=plot(t,s_dsb);
xlabel('{it t} (sec)');
ylabel('{\it s}_{\it s}_{\rm DSB}({\it t})')

subplot(223); td3=plot(t,s_dem);
xlabel('{it t} (sec)');
ylabel('{\it e} ({\it t})')
title('{\it e}({\it t})');

subplot(224); td4=plot(t,s_rec);
xlabel('{it t} (sec)');
ylabel('{\it m}_d({\it t})')
title('Recovered signal');

figure(2)
```

```
subplot(221);
fd1=plot(freqm,abs(M_fre));
xlabel('{\it f} (Hz)');
ylabel('{\it M}({\it f})')
title('message spectrum')

subplot(222);fd2=plot(freqs,abs(S_dsb));
xlabel('{\it f} (Hz)');
ylabel('{\it S}_{rm DSB} ({\it f})')
title('DSB-SC specttrum')

subplot(223);
fd3=plot(freqs,abs(S_dem));
xlabel('{\it f} (Hz)');
ylabel('{\it E}({\it f})');
title('spectrum of {\it e}({\it t})')

subplot(224);
fd4=plot(freqs,abs(S_rec));
xlabel('{\it f} (Hz)');
ylabel('{\it M}_d({\it f})');
title('recovered spectrum')
%% Output
sound(s_rec,Fs)


SSB-SC(TRIANGLE)

clear all;
clc;

ts=10^(-4);
t=-0.04:ts:0.04;
Ta=0.01;
m_sig=triangl((t+0.01)/0.01)-triangl((t-
0.01)/0.01);
Lm_sig=length(m_sig);
Lfft=length(t);
Lfft=2^ceil(log2(Lfft))
M_sig=fftshift(fft(m_sig,Lfft))

freqm=(-Lfft/2:Lfft/2-1)/(Lfft*ts)
B_m=150;
h=fir1(40,[B_m*ts])

%% SSB modulation
fc=300; % carrier frequency
s_dsb=(m_sig).*cos(2*pi*fc*t)
Lfft=length(t);
Lfft=2^ceil(log2(Lfft)+1)
S_dsb=fftshift(fft(s_dsb,Lfft));
L_lsb=floor(fc*ts*Lfft)
SSBfilt=ones(1,Lfft);
%% SSBfilt(Lfft/2-
L_lsb+1:Lfft/2+L_lsb)=zeros(1,2*L_lsb);
S_ssb=S_dsb.*SSBfilt;
freqs=(-Lfft/2:Lfft/2-1)/(Lfft*ts)
s_ssb=real(ifft(fftshift(S_ssb)));
s_ssb=s_ssb(1:Lm_sig);



%% Demodulation begins by using a
rectifier
s_dem=s_ssb.*cos(2*pi*fc*t)*2 %
S_dem=fftshift(fft(s_dem,Lfft));
% Using an ideal low pass filter with
bandwidth 150 Hz
s_rec=filter(h,1,s_dem)
S_rec=fftshift(fft(s_rec,Lfft));


% Trange=[-0.025 0.025 -1 1]
% Frange=[-700 700 0 200]
figure(1)
subplot(221);
plot(t,m_sig,'Linewidth',1.5)
% axis(Trange)
title('message signal')

subplot(222);
plot(t,s_ssb,'Linewidth',1.5)
% axis(Trange)
title('SSB-SC modulated signal')

subplot(223); plot(t,
s_dem,'Linewidth',1.5)
% axis(Trange)
title('After multiplying local carrier')

subplot(224);
plot(t,s_rec,'Linewidth',1.5)
% axis(Trange)
title('Recovered signal')


figure(2)
subplot(221);
plot(freqm,abs(M_sig),'Linewidth',1.5)
% axis(Frange)
title('Message Spectrum')

subplot(222);
plot(freqs,abs(S_ssb),'Linewidth',1.5)
% axis(Frange)
title('Upper Sideband SSB-SC spectrum')

subplot(223); plot(freqs,
abs(S_dem),'Linewidth',1.5)
% axis(Frange)
title('Detector spectrum')

subplot(224);
plot(freqs,abs(S_rec),'Linewidth',1.5)
% axis(Frange)
title('Recovered signal')


SSB-SC(VOICE)

clc;
clear all;
close all;
%% Input
[y, Fs] = audioread('tamimDSB.wav');
y = y';
n = length(y);
t = 0:1/Fs:(n-1)*(1/Fs);
ts=1/Fs;
m_sig=y;
bw=3400;
ts=1/Fs;
Lm_sig=length(m_sig);
Lfft=length(t);
Lfft=2^ceil(log2(Lfft)) ;
M_sig=fftshift(fft(m_sig,Lfft));
freqm=(-Lfft/2:Lfft/2-1)/(Lfft*ts);
h=fir1(40,[bw*ts]);

%% SSB modulation
fc=4000; % carrier frequency
s_dsb=(m_sig).*cos(2*pi*fc*t);
Lfft=length(t);
Lfft=2^ceil(log2(Lfft)+1);
S_dsb=fftshift(fft(s_dsb,Lfft));
L_lsb=floor(fc*ts*Lfft);
SSBfilt=ones(1,Lfft);
%% SSBfilt(Lfft/2-
L_lsb+1:Lfft/2+L_lsb)=zeros(1,2*L_lsb);
S_ssb=S_dsb.*SSBfilt;
freqs=(-Lfft/2:Lfft/2-1)/(Lfft*ts);
s_ssb=real(ifft(fftshift(S_ssb)));
s_ssb=s_ssb(1:Lm_sig);
%% Channel
s_noise=awgn(100*s_ssb,20);
fr=[2*4500/Fs 2*7500/Fs];
%% Reciever
s_rcv=bandpass(s_noise,fr);


%% Demodulation begins by using a
rectifier
s_dem=s_rcv.*cos(2*pi*fc*t)*2;
S_dem=fftshift(fft(s_dem,Lfft));
%% Using an ideal low pass filter with
bandwidth 150 Hz
s_rec=filter(h,1,s_dem);

S_rec=fftshift(fft(s_rec,Lfft));
%% Plot
figure(1)
subplot(221);
plot(t,m_sig,'Linewidth',1.5)
title('message signal')

subplot(222);
plot(t,s_ssb,'Linewidth',1.5)
title('SSB-SC modulated signal')

subplot(223); plot(t,
s_dem,'Linewidth',1.5)
title('After multiplying local carrier')

subplot(224);
plot(t,s_rec,'Linewidth',1.5)
title('Recovered signal')


figure(2)
subplot(221);
plot(freqm,abs(M_sig),'Linewidth',1.5)
title('Message Spectrum')
```

```
subplot(222);
plot(freqs,abs(S_ssb),'Linewidth',1.5)
title('Upper Sideband SSB-SC spectrum')

subplot(223); plot(freqs,
abs(S_dem),'Linewidth',1.5)
title('Detector spectrum')

subplot(224);
plot(freqs,abs(S_rec),'Linewidth',1.5)
title('Recovered signal')
%% Output
sound(s_rec,Fs)
```

SSB-WC(TRIANGLE)

```
clear all;
clc;

ts=10^(-4);
t=-0.04:ts:0.04;
Ta=0.01;
m_sig=triangl((t+0.01)/0.01)-triangl((t-
0.01)/0.01);
Lm_sig=length(m_sig);
Lfft=length(t);
Lfft=2^ceil(log2(Lfft))
M_sig=fftshift(fft(m_sig,Lfft))

freqm=(-Lfft/2:Lfft/2-1)/(Lfft*ts)
B_m=150;
h=fir1(40,[B_m*ts])

%% SSB modulation
fc=300; % carrier frequency
Ac=2;
carrier=Ac*cos(2*pi*fc*t);
s_dsb=carrier+(m_sig).*cos(2*pi*fc*t)
Lfft=length(t);
Lfft=2^ceil(log2(Lfft)+1)
S_dsb=fftshift(fft(s_dsb,Lfft));
L_lsb=floor(fc*ts*Lfft)
SSBfilt=ones(1,Lfft);
%% SSBfilt(Lfft/2-
L_lsb+1:Lfft/2+L_lsb)=zeros(1,2*L_lsb);
S_ssb=S_dsb.*SSBfilt;
freqs=(-Lfft/2:Lfft/2-1)/(Lfft*ts)
s_ssb=real(ifft(fftshift(S_ssb)));
s_ssb=s_ssb(1:Lm_sig);



%% Demodulation begins by using a
rectifier
s_dem=s_ssb.*cos(2*pi*fc*t)*2-
Ac*ones(1,length(s_ssb));
S_dem=fftshift(fft(s_dem,Lfft));
% Using an ideal low pass filter with
bandwidth 150 Hz
s_rec=filter(h,1,s_dem)
S_rec=fftshift(fft(s_rec,Lfft));
```

```
% Trange=[-0.025 0.025 -1 1]
% Frange=[-700 700 0 200]
figure(1)
subplot(221);
plot(t,m_sig,'Linewidth',1.5)
% axis(Trange)
title('message signal')

subplot(222);
plot(t,s_ssb,'Linewidth',1.5)
% axis(Trange)
title('SSB-SC modulated signal')

subplot(223); plot(t,
s_dem,'Linewidth',1.5)
% axis(Trange)
title('After multiplying local carrier')

subplot(224);
plot(t,s_rec,'Linewidth',1.5)
% axis(Trange)
title('Recovered signal')


figure(2)
subplot(221);
plot(freqm,abs(M_sig),'Linewidth',1.5)
% axis(Frange)
title('Message Spectrum')

subplot(222);
plot(freqs,abs(S_ssb),'Linewidth',1.5)
% axis(Frange)
title('Upper Sideband SSB-SC spectrum')

subplot(223); plot(freqs,
abs(S_dem),'Linewidth',1.5)
% axis(Frange)
title('Detector spectrum')

subplot(224);
plot(freqs,abs(S_rec),'Linewidth',1.5)
% axis(Frange)
title('Recovered signal')
```

SSB-WC(VOICE)

```
clc;
clear all;
close all;
[y, Fs] = audioread('tamimDSB.wav');
y = y';
n = length(y);
t = 0:1/Fs:(n-1)*(1/Fs);
ts=1/Fs;
m_sig=y;
Lm_sig=length(m_sig);
Lfft=length(t);
Lfft=2^ceil(log2(Lfft))
M_sig=fftshift(fft(m_sig,Lfft));
freqm=(-Lfft/2:Lfft/2-1)/(Lfft*ts);
bw=3400;
h=fir1(40,[bw*ts]);

%% SSB modulation
fc=4000; % carrier frequency
Ac=1;
carrier=Ac*cos(2*pi*fc*t);
s_dsb=carrier+(m_sig).*cos(2*pi*fc*t);
Lfft=length(t);
Lfft=2^ceil(log2(Lfft)+1);
S_dsb=fftshift(fft(s_dsb,Lfft));
L_lsb=floor(fc*ts*Lfft);
SSBfilt=ones(1,Lfft);
%% SSBfilt(Lfft/2-
L_lsb+1:Lfft/2+L_lsb)=zeros(1,2*L_lsb);
S_ssb=S_dsb.*SSBfilt;
freqs=(-Lfft/2:Lfft/2-1)/(Lfft*ts);
s_ssb=real(ifft(fftshift(S_ssb)));
s_ssb=s_ssb(1:Lm_sig);
%% Channel
s_noise=awgn(100*s_ssb,20);
fr=[2*4300/Fs 2*7400/Fs];
%% Reciever
s_rcv=bandpass(s_noise,fr);


%% Demodulation begins by using a
rectifier
s_dem=s_rcv.*cos(2*pi*fc*t)*2 -
Ac*ones(1,length(s_rcv));
S_dem=fftshift(fft(s_dem,Lfft));
% Using an ideal low pass filter with
bandwidth 150 Hz
s_rec=filter(h,1,s_dem);
S_rec=fftshift(fft(s_rec,Lfft));

%% Plot
figure(1)
subplot(221);
plot(t,m_sig,'Linewidth',1.5)
% axis(Trange)
title('message signal')

subplot(222);
plot(t,s_ssb,'Linewidth',1.5)
% axis(Trange)
title('SSB-SC modulated signal')

subplot(223); plot(t,
s_dem,'Linewidth',1.5)
% axis(Trange)
title('After multiplying local carrier')

subplot(224);
plot(t,s_rec,'Linewidth',1.5)
% axis(Trange)
title('Recovered signal')


figure(2)
subplot(221);
plot(freqm,abs(M_sig),'Linewidth',1.5)
% axis(Frange)
title('Message Spectrum')

subplot(222);
plot(freqs,abs(S_ssb),'Linewidth',1.5)
% axis(Frange)
title('Upper Sideband SSB-SC spectrum')
```

```
subplot(223); plot(freqs,
abs(S_dem),'Linewidth',1.5)
% axis(Frange)
title('Detector spectrum')

subplot(224);
plot(freqs,abs(S_rec),'Linewidth',1.5)
% axis(Frange)
title('Recovered signal')
sound(s_rec,Fs)
```

VSB(VOICE)

```
clc;
close all;
clear all;

%% generation of message signal and
carrier signal

[y_tr, Fs] = audioread('tamimDSB.wav');
nBits = 16;
t =
linspace(0,length(y_tr)/Fs,length(y_tr));
ts=1/Fs;
t =
linspace(0,length(y_tr)/Fs,length(y_tr));
bw = 3400;
fc = 4000;
m_sig=y_tr;
Lm_sig=length(m_sig);
Lfft=length(t);
Lfft=2^ceil(log2(Lfft));
M_sig=fftshift(fft(m_sig,Lfft));
freqm=(-Lfft/2:Lfft/2-1)/(Lfft*ts) ;
h=fir1(40,[bw*ts]);

%% VSB Modulation
s_dsb=(m_sig)'.*cos(2*pi*fc*t);
%Transposed the signal to match
dimension
Lfft=length(t);
Lfft=2^ceil(log2(Lfft)+1);
S_dsb=fftshift(fft(s_dsb,Lfft));
freqs=(-Lfft/2:Lfft/2-1)/(Lfft*ts);
percentageOfLsb = 0.25;
fv = percentageOfLsb*bw/2;
L_vsb=floor((fc-fv)*ts*Lfft);
hfv = floor((fc+fv)*ts*Lfft);

%setting up the vsb filter
VSBfilt=ones(1,Lfft);

p = Lfft/2+L_vsb:Lfft/2+hfv;
q = Lfft/2-hfv+1:Lfft/2-L_vsb+1;

x = linspace(fc-fv,fc+fv,length(p));
m = 0.5/(2*fv);
y = m.*x - m*(fc-fv);

VSBfilt(p)=y;
VSBfilt(q)=fliplr(y);

VSBfilt(Lfft/2-
L_vsb+1:Lfft/2+L_vsb)=zeros(1,2*L_vsb);

S_vsb=S_dsb.*VSBfilt;
% freqs=(-Lfft/2:Lfft/2-1)/(Lfft*ts);
s_vsb=real(ifft(fftshift(S_vsb)));
s_vsb=s_vsb(1:Lm_sig);

%% Channel
s_noise=awgn(100*s_vsb,20);
fr=[2*4300/Fs 2*7400/Fs];
%% Reciever
s_rcv=bandpass(s_noise,fr);


%% demodulation
%%% Demodulation begins by using a
rectifier
s_dem=s_rcv.*cos(2*pi*fc*t)*2;
S_dem=fftshift(fft(s_dem,Lfft));
% Using an ideal low pass filter with
bandwidth 150 Hz
s_rec=filter(h,1,s_dem);%ei filter ta niye
pech lagbe

%need to design only this filter
S_rec=fftshift(fft(s_rec,Lfft));

%% Plot
figure(2)
subplot(221);
plot(t,m_sig,'Linewidth',1.5)
title('message signal')

subplot(222);
plot(t,s_vsb,'Linewidth',1.5)
title('VSB-SC modulated signal')

subplot(223); plot(t,
s_dem,'Linewidth',1.5)
title('After multiplying local carrier')

subplot(224);
plot(t,s_rec,'Linewidth',1.5)
title('Recovered signal')

figure(3)
subplot(221);
plot(freqm,abs(M_sig),'Linewidth',1.5)
title('Message Spectrum')

subplot(222);
plot(freqs,abs(S_vsb),'Linewidth',1.5)
title('Upper Sideband VSB-SC spectrum')

subplot(223); plot(freqs,
abs(S_dem),'Linewidth',1.5)
title('Detector spectrum')

subplot(224);
plot(freqs,abs(S_rec),'Linewidth',1.5)
title('Recovered spectrum')
%% Output
sound(s_rec,Fs,nBits)
```

QAM(VOICE)

```
clc;
clear all;
close all;
[m_sig1 Fs]=
audioread('tamimDSB.wav');
[m_sig2 Fs]= audioread('tamim.wav');
nBits=16;
BW=3000;
fc=5000;
% [s_rec] =
qam_mod_demod(nBits,Fs,m_sig1,BW,f
c,m_sig2);
ts = 1/Fs;
t =
linspace(0,length(m_sig1)/Fs,length(m_
sig1));

Lm_sig = length(m_sig1);
Lfft = length(t);
Lfft = 2^ceil(log2(Lfft));
M1_fre = fftshift(fft(m_sig1,Lfft));
M2_fre = fftshift(fft(m_sig2,Lfft));
freqm = (-Lfft/2:Lfft/2 -1)/(Lfft*ts);

h = fir1(40,[BW*ts]);

%% QAM modulation

% QAM signal generated adding a
carrier to DSB-SC
s_qam =
m_sig1'.*cos(2*pi*fc*t)+m_sig2'.*sin(2*
pi*fc*t);
Lfft = length(t);  Lfft =
2^ceil(log2(Lfft)+1);
S_qam = fftshift(fft(s_qam,Lfft));
freqs = (-Lfft/2:Lfft/2 -1)/(Lfft*ts);

%% QAM demodulation

% demodulation begins by using a
rectifier
s_dem1 = s_qam.*cos(2*pi*fc*t)*2;
S_dem1 = fftshift(fft(s_dem1,Lfft));
% demodulate 2nd signal
s_dem2 = s_qam.*sin(2*pi*fc*t)*2;
S_dem2 = fftshift(fft(s_dem2,Lfft));

% using ideal LPF with BW
s_rec1 = filter(h,1,s_dem1);
S_rec1 = fftshift(fft(s_rec1,Lfft));
s_rec2 = filter(h,1,s_dem2);
S_rec2 = fftshift(fft(s_rec2,Lfft));
Trange = [-0.025 0.025 -2 2];
Trange2 = [-0.025 0.025 -2 4];

figure(1)
subplot(221);
plot(t,m_sig1,'Linewidth',1.5);  axis(Tran
ge);
xlabel('t,(sec)');  ylabel('m1(t)');  title('m
essage signal 1');
subplot(222);  plot(t,s_qam,'Linewidth',
1.5);  axis(Trange);
```

```
xlabel('t,(sec)'); ylabel('s1_DSB(t)'); title
('QAM modulated signal');
subplot(223); plot(t,s_dem1,'Linewidth'
,1.5); axis(Trange2);
xlabel('t,(sec)'); ylabel('e1(t)'); title('firs
t demodulator output');
subplot(224); plot(t,s_rec1,'Linewidth',
1.5); axis(Trange);
xlabel('t,(sec)'); ylabel('m_d1(t)'); title('
detected signal 1');

figure(2)
subplot(221); plot(t,m_sig2,'Linewidth',
1.5); axis(Trange);
xlabel('t(sec)'); ylabel('m2(t)'); title('me
ssage signal 2');
subplot(222); plot(t,s_qam,'Linewidth',
1.5); axis(Trange);
xlabel('t(sec)'); ylabel('s2_DSB(t)'); title(
'QAM modulated signal');
subplot(223); plot(t,s_dem2,'Linewidth'
,1.5); axis(Trange2);
xlabel('t(sec)'); ylabel('e2(t)'); title('sec
ond demodulator output');
subplot(224);
plot(t,s_rec2,'Linewidth',1.5); axis(Tran
ge);
xlabel('t(sec)'); ylabel('m_d2(t)'); title('
detected signal 2');

Frange = [-700 700 0 250];
figure(3)
subplot(221); plot(freqm,abs(M1_fre),'L
inewidth',1.5); axis(Frange);
xlabel('f(Hz)'); ylabel('M1(f)'); title('mes
sage 1 spectrum');
subplot(222); plot(freqs,abs(S_qam),'Li
newidth',1.5); axis(Frange);
xlabel('f(Hz)'); ylabel('S1_AM(f)'); title('
QAM spectrum magnitude');
subplot(223); plot(freqs,abs(S_dem1),'L
inewidth',1.5); axis(Frange);
xlabel('f(Hz)'); ylabel('E1(f)'); title('first
demodulator spectrum');
subplot(224); plot(freqs,abs(S_rec1),'Li
newidth',1.5); axis(Frange);
xlabel('f(Hz)'); ylabel('M_d1(f)'); title('r
ecovered spectrum 1');

figure(4)
subplot(221); plot(freqm,abs(M2_fre),'L
inewidth',1.5); axis(Frange);
xlabel('f(Hz)'); ylabel('M2(f)'); title('mes
sage 2 spectrum');
subplot(222); plot(freqs,abs(S_qam),'Li
newidth',1.5); axis(Frange);
xlabel('f(Hz)'); ylabel('S2_AM(f)'); title('
QAM spectrum magnitude');
subplot(223); plot(freqs,abs(S_dem2),'L
inewidth',1.5); axis(Frange);
xlabel('f(Hz)'); ylabel('E2(f)'); title('seco
nd demodulator spectrum');
subplot(224); plot(freqs,abs(S_rec2),'Li
newidth',1.5); axis(Frange);
xlabel('f(Hz)'); ylabel('M_d2(f)'); title('r
ecovered spectrum 2');
```

FDM(VOICE)

```
clc;
clear all;
close all;


[y1 Fs1]=audioread('tamim01.wav');
[y2 Fs2]=audioread('tamim02.wav');

fpass=2000;
ssbm1=ssbmod(y1, 3000, 8000);
ssbm2=ssbmod(y2, 3000, 8000);
ssbmodu=ssbm1+ssbm2;
t =
linspace(0,length(y1)/Fs1,length(y1));

ssbmod2=ssbmod(ssbmodu, 5000,8000
)
ssbdemod1=bandpass(ssbmod2,[6000
8000],17000);
ssbdemod2=bandpass(ssbmod2,[4000
5900],17000);

r1=ssbdemod1 .* sin(2*pi*3000*t);
r2=ssbdemod2 .* sin(2*pi*3000*t);
recons1= lowpass(r1, fpass, Fs1);
recons2= lowpass(r2, fpass, Fs1);
audiowrite(sprintf('reconstructed1.wav'
),recons1, Fs);
audiowrite(sprintf('reconstructed2.wav'
),recons2, Fs)
```