GAUTAM BUDDHA UNIVERSITY

SCHOOL OF INFORMATION AND TECHNOLOGY

Digital Communication Lab (EC-373)

Programme: B.tech ECE
SESSION: 2018-2022
3RD YEAR-Vth semester

Submitted by: **Shekhar Tyagi**
18/BEC/046

Submitted to: **Dr. Ana Kumar**

EXPERIMENT 1

Aim : Learning the MATLAB/OCTAVE language in context of analog communication and perform basic array operations.

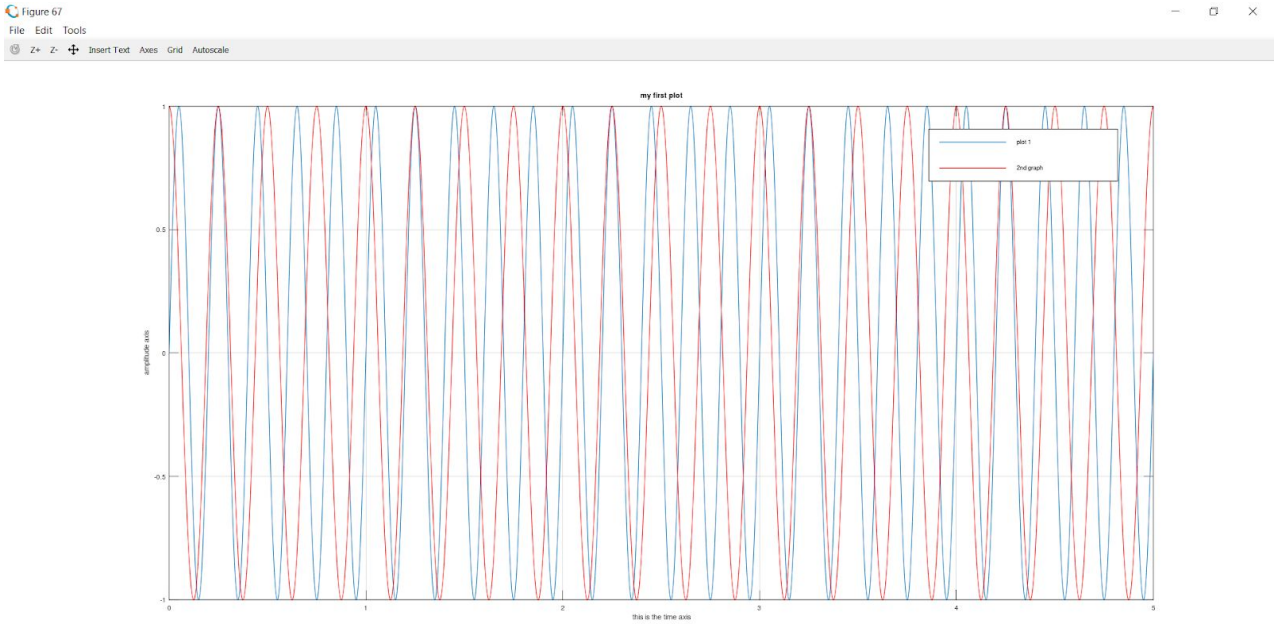
Theory : OCTAVE has powerful mathematics oriented syntax with built in 2D-3D plotting and visual tools. It can be used for various purposes like digital and analog communication. It helps in solving linear and non-linear problems numerically.

Software Used : GNU OCTAVE

Program :

```
clear all

close all

clc


t=0:.001:5;

z=sin(5*pi*2*t);

figure(67)

plot(t,z)

hold on

plot(t,cos(2*pi*4*t),'r')

xlabel('this is the time axis')

ylabel('amplitude axis')

title('my first plot')

grid on

legend('plot 1','2nd graph')
```

Observation :



Result : The above graph shows plots and graphs and the basics of how to use analog communication in OCTAVE.

Precautions : Make sure the code is written properly

The output is taken appropriately.

EXPERIMENT 2

Aim : WAPT simulate amplitude modulated signal

Theory : Amplitude modulation (**AM**) is a modulation technique used in electronic communication, most commonly for transmitting messages with a radio carrier wave. In amplitude modulation, the amplitude (signal strength) of the carrier wave is varied in proportion to that of the message signal, such as an audio signal.

To **produce** an **amplitude modulated wave** we need to superimpose this **signal** with another **signal** of higher frequency. This **signal** is the 'Carrier **Signal**' and differs from one carrier to another.

Software Used : GNU OCTAVE

Program : **(a) Program for magnitude spectrum generation :**

```
close all

clear all

clc


Fs = 500;           % Sampling frequency

T = 1/Fs;           % Sampling period

L = 50001;          % Length of signal

t = (0:L-1)*T;      % Time vector


S = sin(2*pi*5*t);

figure


subplot(2,1,1)

hold on
```

```
grid on
plot(t(1:150),S(1:150),'linewidth',2)
title('Signal','Fontsize',15,'color',[255 0 0]/255)
xlabel('t (Seconds) ')
ylabel('X(t)')

f = Fs*(-floor(L/2):floor(L/2))/L;

Y = fftshift(fft(S));
P= abs(Y/L);

subplot(2,1,2)
hold on
grid on
plot(f,P,'linewidth',2)
title('Magnitude Spectrum','Fontsize',15,'color',[0 0 255]/255)
xlabel('f (Hz)')
ylabel('|Signal|')
```

(b) Program for magnitude generation of message signal, carrier wave and modulating wave:

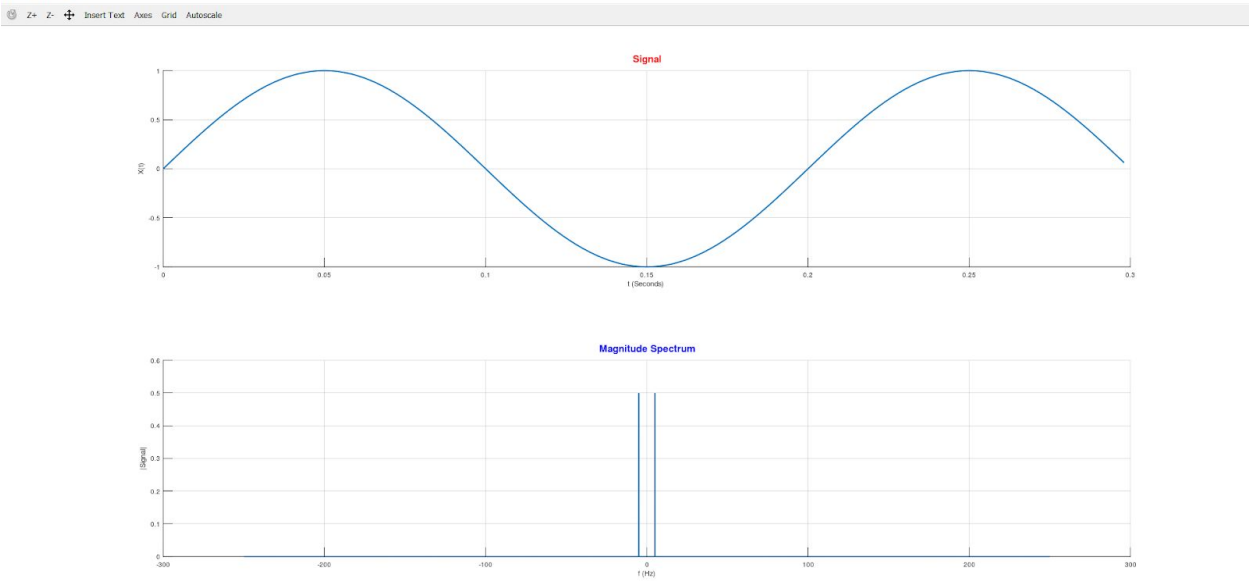
```
close all
clear all
clc

Fs = 400;          % Sampling frequency
```

```
T = 1/Fs;            % Sampling period
L = 401;             % Length of signal
t = (0:L-1)*T;       % Time vector

My_msg = sin(2*pi*10*t);
My_carrier = sin(2*pi*50*t);

figure

subplot(3,2,1)
hold on
grid on
plot(t(1:100),My_msg(1:100),'linewidth',2)
title('Message Signal','Fontsize',15,'color',[0 0 255]/255)
xlabel('t (Seconds) ')
ylabel('X(t)')

subplot(3,2,3)
hold on
grid on
plot(t(1:100),My_carrier(1:100),'linewidth',2)
title('Carrier Signal','Fontsize',15,'color',[255 0 0]/255)
xlabel('t (Seconds) ')
ylabel('X(t)')
```

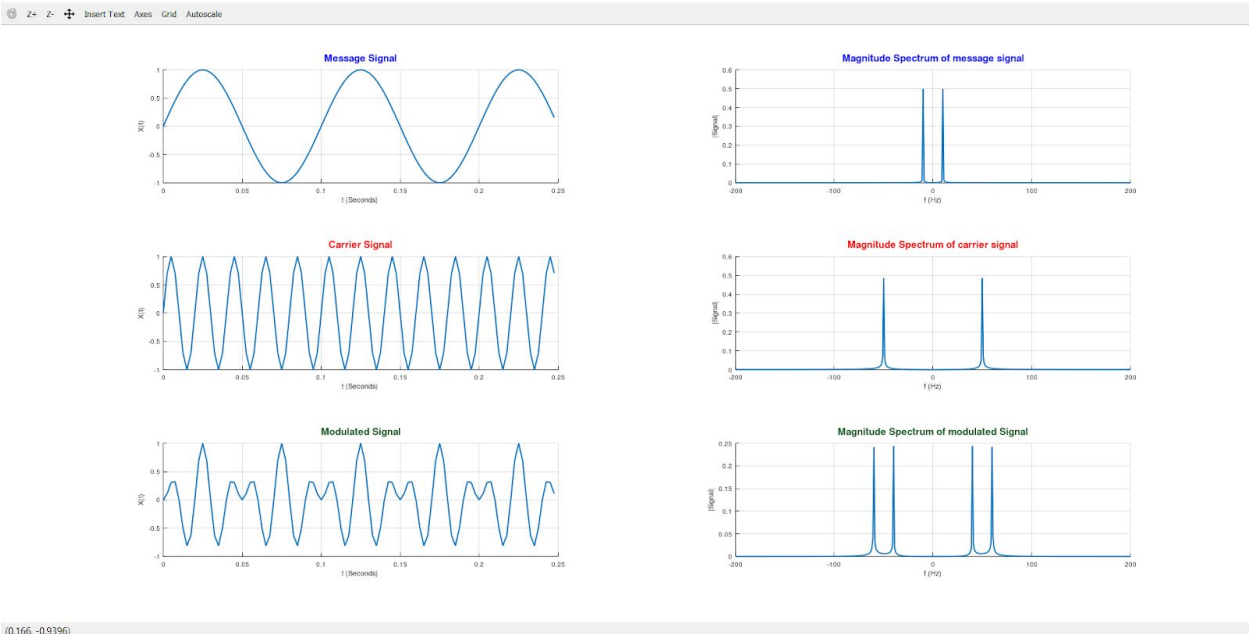
```
f = Fs*(-floor(L/2):floor(L/2))/L;

fft_msg = fftshift(fft(My_msg));
mag_msg= abs(fft_msg/L);

fft_carrier = fftshift(fft(My_carrier));
mag_carrier= abs(fft_carrier/L);

subplot(3,2,2)
hold on
grid on
plot(f,mag_msg,'linewidth',2)
title('Magnitude Spectrum of message signal','Fontsize',15,'color',[0 0 255]/255)
xlabel('f (Hz)')
ylabel('|Signal|')

subplot(3,2,4)
hold on
grid on
plot(f,mag_carrier,'linewidth',2)
title('Magnitude Spectrum of carrier signal','Fontsize',15,'color',[255 0 0]/255)
xlabel('f (Hz)')
ylabel('|Signal|')

modu_signal=My_msg.*My_carrier;
```

```
subplot(3,2,5)
hold on
grid on
plot(t(1:100),modu_signal(1:100),'linewidth',2)
title('Modulated Signal','Fontsize',15,'color',[20 90 40]/255)
xlabel('t (Seconds) ')
ylabel('X(t)')

fft_modu_signal = fftshift(fft(modu_signal));
mag_modu_signal= abs(fft_modu_signal/L);

subplot(3,2,6)
hold on
grid on
plot(f,mag_modu_signal,'linewidth',2)
title('Magnitude Spectrum of modulated Signal','Fontsize',15,'color',[20 90
40]/255)
xlabel('f (Hz)')
ylabel('|Signal|')
```

Observartions : (a) For magnitude spectrum generation



(b) For magnitude generation of message signal, carrier wave and modulating wave:



Result : We get the magnitude spectrum for message, carrier and modulated signal.

Precautions : Make sure the code is written properly

The output is taken appropriately.

EXPERIMENT – 3

Aim : WAPT simulate frequency modulated signal using OCTAVE.

Theory : Frequency modulation is the encoding of information in a carrier wave by varying the instantaneous frequency of the wave. The technology is used in telecommunications, radio broadcasting, signal processing, and computing.

To generate a **frequency modulated** signal, the **frequency** of the radio carrier is changed in line with the amplitude of the incoming audio signal. When the audio signal is **modulated** onto the radio **frequency** carrier, the new radio **frequency** signal moves up and down in **frequency**.

Software Used : GNU OCTAVE

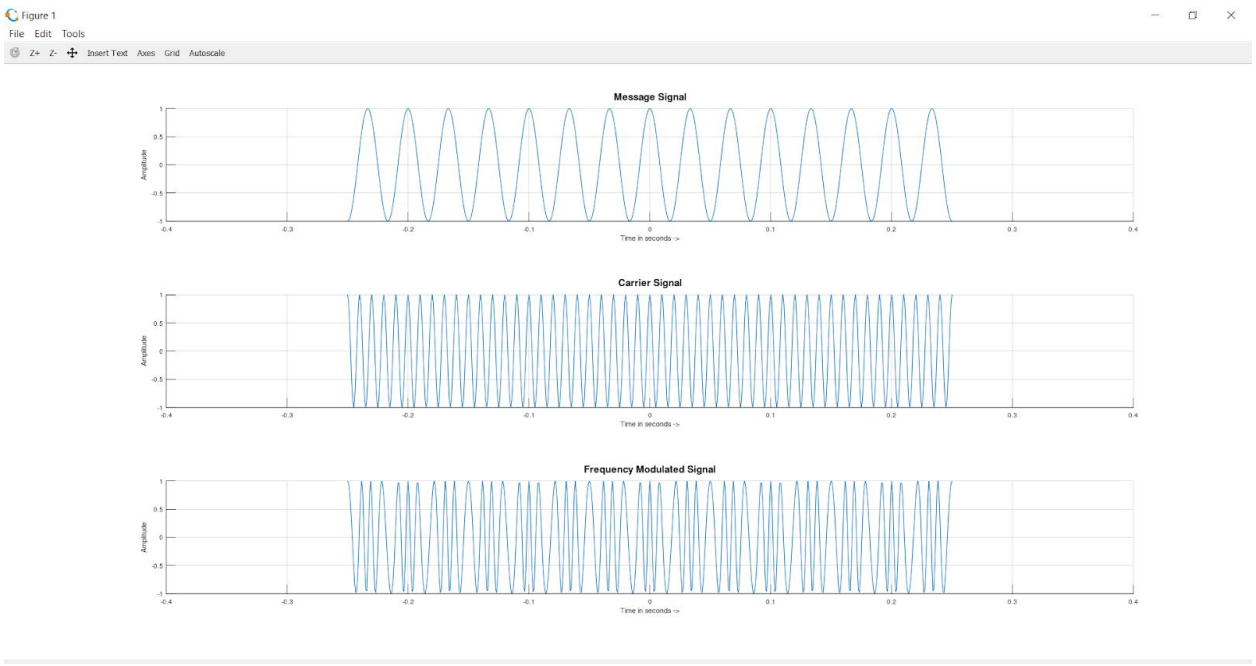
Program :

```
% Frequency Modulation

close all

clear all

clc


Fs=1200;

T=1/Fs;

L=600;

t=(-floor(L/2):floor(L/2))*T;


msg_sig=cos(2*pi*30*t);

car_sig=cos(2*pi*100*t);

kf=250;

mod_sig=cos(2*pi*100*t+kf*T*cumtrapz(msg_sig));
```

```
%% JUSTIFICATION FOR MULTIPLICATION OF 'T' in above expression
%Z = cumtrapz(Y) computes an approximation of the cumulative
%    integral of Y via the trapezoidal method (with unit spacing).
%    To compute the integral for spacing different from one,
%    multiply Z by the spacing increment.
   %%

figure

subplot(3,1,1)
hold on
grid on
plot(t,msg_sig)
xlabel('Time in seconds ->')
ylabel('Amplitude')
title('Message Signal','Fontsize',15);

subplot(3,1,2)
hold on
grid on
plot(t,car_sig)
xlabel('Time in seconds ->')
ylabel('Amplitude')
title('Carrier Signal','Fontsize',15);
```

```
subplot(3,1,3)
hold on
grid on
plot(t,mod_sig)
xlabel('Time in seconds ->')
ylabel('Amplitude')
title('Frequency Modulated Signal','Fontsize',15);
```

Observation :



Result : The frequency modulated signal has been generated .

Precautions : Make sure the code is written properly

The output is taken appropriately.

EXPERIMENT – 4

Aim : WAPT simulate phase modulated signal.

Theory : Phase modulation is a modulation pattern for conditioning communication signals for transmission. It encodes a message signal as variations in the instantaneous phase of a carrier wave. Phase modulation is one of the two principal forms of angle modulation, together with frequency modulation.

The phase of a carrier signal is modulated to follow the changing signal level (amplitude) of the message signal. The peak amplitude and the frequency of the carrier signal are maintained constant, but as the amplitude of the message signal changes, the phase of the carrier changes correspondingly.

Phase modulation is widely used for transmitting radio waves and is an integral part of many digital transmission coding schemes that underlie a wide range of technologies like Wi-Fi, GSM and satellite television.

Software Used : GNU OCTAVE

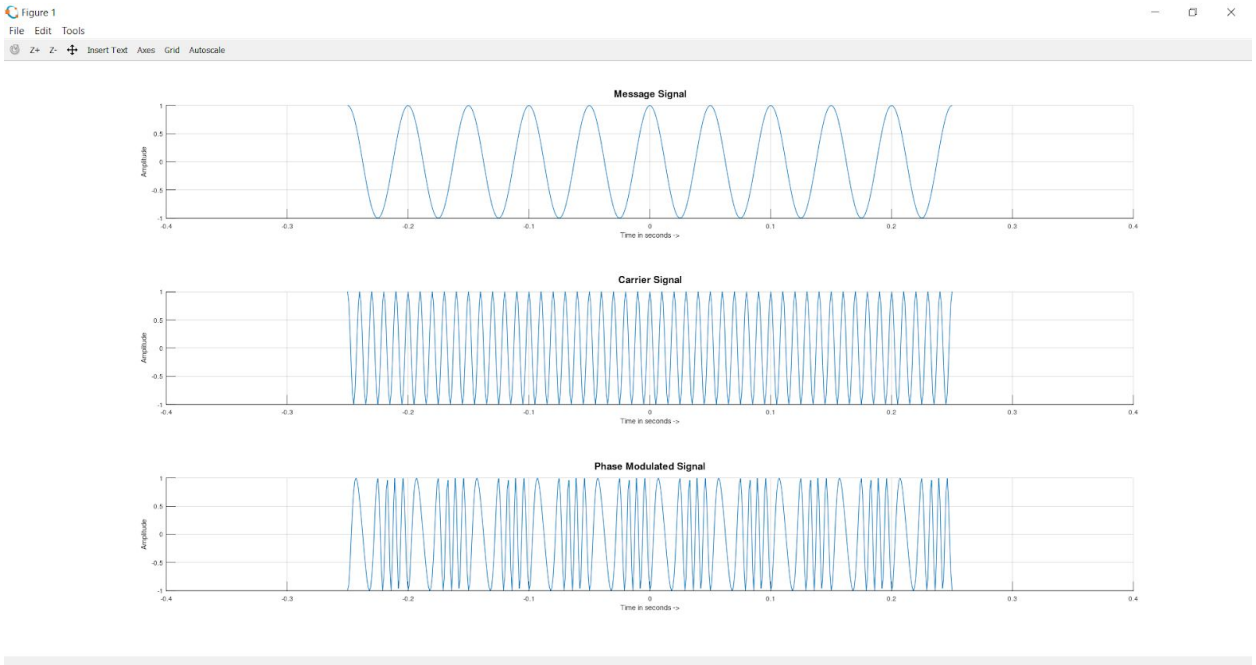
Program :

```
% Phase Modulation

close all

clear all

clc


Fs=1200;

T=1/Fs;

L=600;

t=(-floor(L/2):floor(L/2))*T;


msg_sig=cos(2*pi*30*t);
```

```
car_sig=cos(2*pi*100*t);

kp=3;
mod_sig=cos(2*pi*100*t+kp*(msg_sig));

figure

subplot(3,1,1)
hold on
grid on
plot(t,msg_sig)
xlabel('Time in seconds ->')
ylabel('Amplitude')
title('Message Signal','Fontsize',15);

subplot(3,1,2)
hold on
grid on
plot(t,car_sig)
xlabel('Time in seconds ->')
ylabel('Amplitude')
title('Carrier Signal','Fontsize',15);

subplot(3,1,3)
hold on
grid on
```

```
plot(t,mod_sig)
xlabel('Time in seconds ->')
ylabel('Amplitude')
title('Phase Modulated Signal','Fontsize',15);
```

Observation :



Result : We receive a phase modulated signal from message signal and carrier signal.

Precautions : Make sure the code is written properly

The output is taken appropriately.

EXPERIMENT – 5

Aim : WAPT simulate sampling theorem using OCTAVE

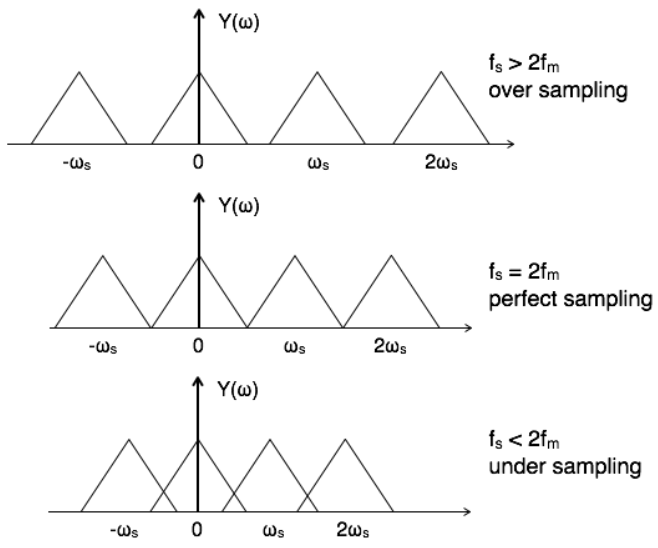
Theory : The sampling theorem specifies the minimum-sampling rate at which a continuous-time signal needs to be uniformly sampled so that the original signal can be completely recovered or reconstructed by these samples alone. This is usually referred to as Shannon's sampling theorem in the literature.

Sampling theorem:

If a continuous time signal contains no frequency components higher than *W* hz, then it can be completely determined by uniform samples taken at a rate f_s samples per second where

$f_s \geq 2W$

Sampling is shown in the given figure below :



Software Used : GNU OCTAVE

Program :

(a) Case 1 : When fs>2fm

```
fm=5 ; %maximum freq of signal

Fs=10*fm ; %sampling frequency
```

```
T=1/Fs;
L=1001; %lenght of signal
t=(0:L-1)*T %time axis
y =sin(2*pi*fm*t);  %sampled signal

figure;
subplot(2,1,1); %figure will contain 2 rows 1 columns
plot(t,y);

%Fs>=2fm
%3 cases: fs<2fm, fs=2fm, and fs>2fm
```

(b) Case 2 : When fs=2fm

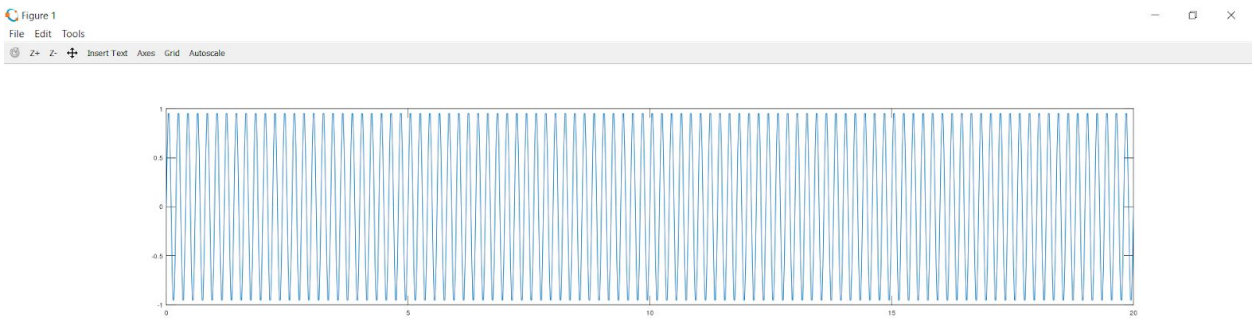
```
fm=25 ; %maximum freq of signal
Fs=1*fm ; %sampling frequency
T=1/Fs;
L=1001; %lenght of signal
t=(0:L-1)*T %time axis
y =sin(2*pi*fm*t);  %sampled signal

figure;
subplot(2,1,1); %figure will contain 2 rows 1 columns
plot(t,y);

%Fs>=2fm
%3 cases: fs<2fm, fs=2fm, and fs>2fm
```

(c) Case 3 : When fs<2fm

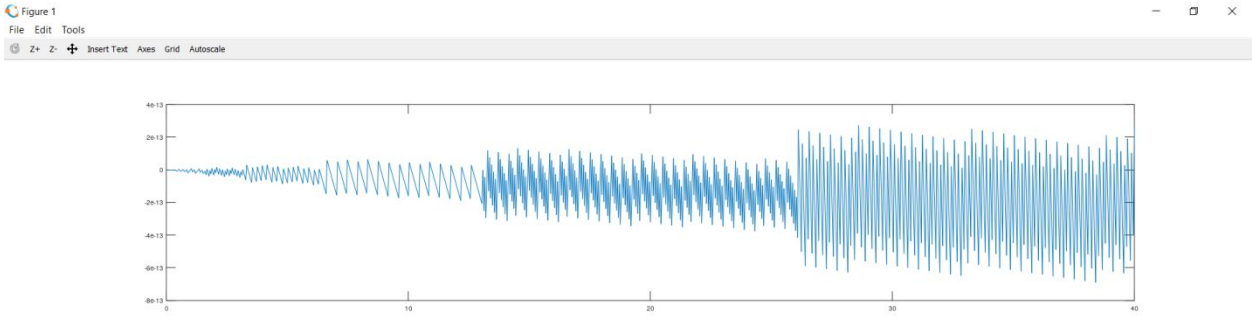
```
fm=25 ; %maximum freq of signal
Fs=2*fm ; %sampling frequency
T=1/Fs;
L=1001; %lenght of signal
t=(0:L-1)*T %time axis
y =sin(2*pi*fm*t);  %sampled signal

figure;
subplot(2,1,1); %figure will contain 2 rows 1 columns
plot(t,y);

%Fs>=2fm
%3 cases: fs<2fm, fs=2fm, and fs>2fm
```

<u>Observations :</u>

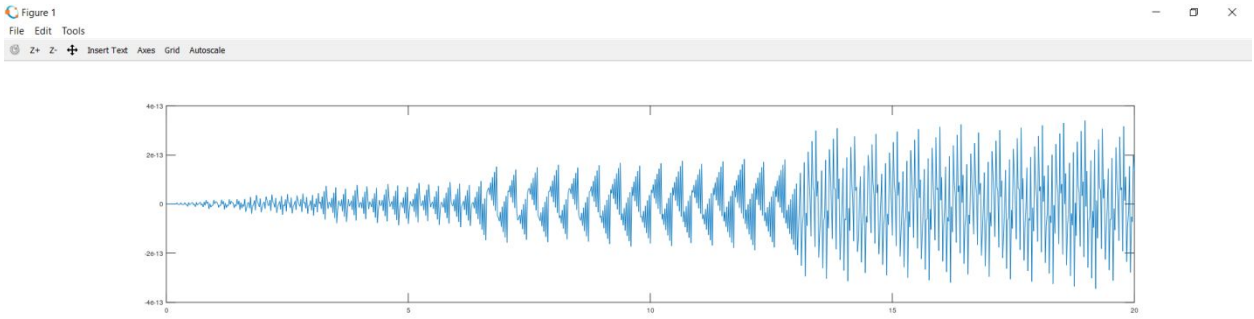
(a) Case 1 : When fs>2fm



(b) Case 2 : When fs=2fm



(c) Case 3 : When fs<2fm



Result : We verified all three cases of the sampling theorem

Precautions : Make sure the code is written properly

The output is taken appropriately.

EXPERIMENT – 6

Aim : WAP for demodulation of FM Wave using OCTAVE

Theory : FM demodulation is also called FM detection and sometimes the phrase "FM discrimination" is used, although this term tends to be used with older circuits and technology.

FM demodulation is a key process in the reception of a frequency modulated signal. Once the signal has been received, filtered and amplified, it is necessary to recover the original modulation from the carrier. It is this process that is called demodulation or detection.

FM demodulator circuits are found in any receiver that uses FM: broadcast receivers, two way radios like walkie talkies and handheld radios that use FM, and any receiver where frequency modulation is used.

In order to be able to demodulate FM it is necessary for the radio receiver to convert the frequency variations into voltage variations - it is a frequency to voltage converter. When the carrier frequency deviates to the lower end of the frequency range over which it deviates a lower voltage may be produced, then as it deviates higher in frequency, a higher voltage is produced.

One of the chief requirements for the FM demodulator is that it should have a response that is as linear as possible over the required bandwidth. In other words a shift of a given frequency produces the same output change wherever it may be found on the curve. If the response is not linear, then distortion will be introduced.

Software Used : GNU OCTAVE

Program :

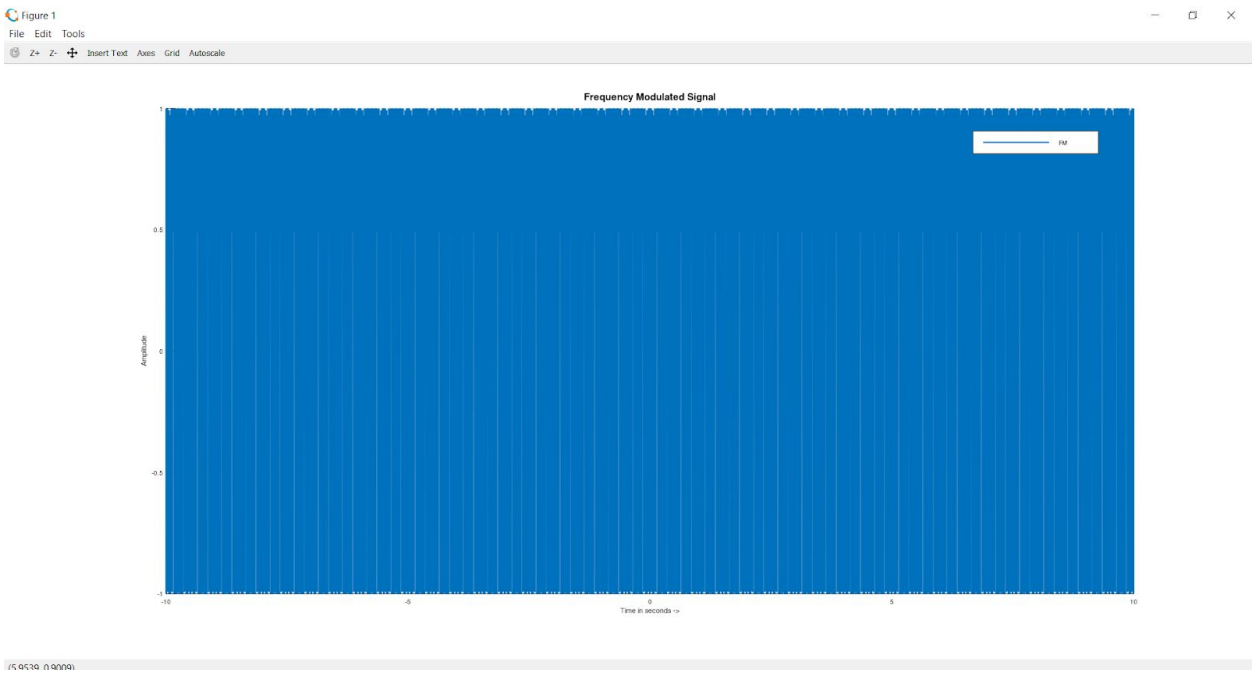
```
close all

clear all

clc

%%
```

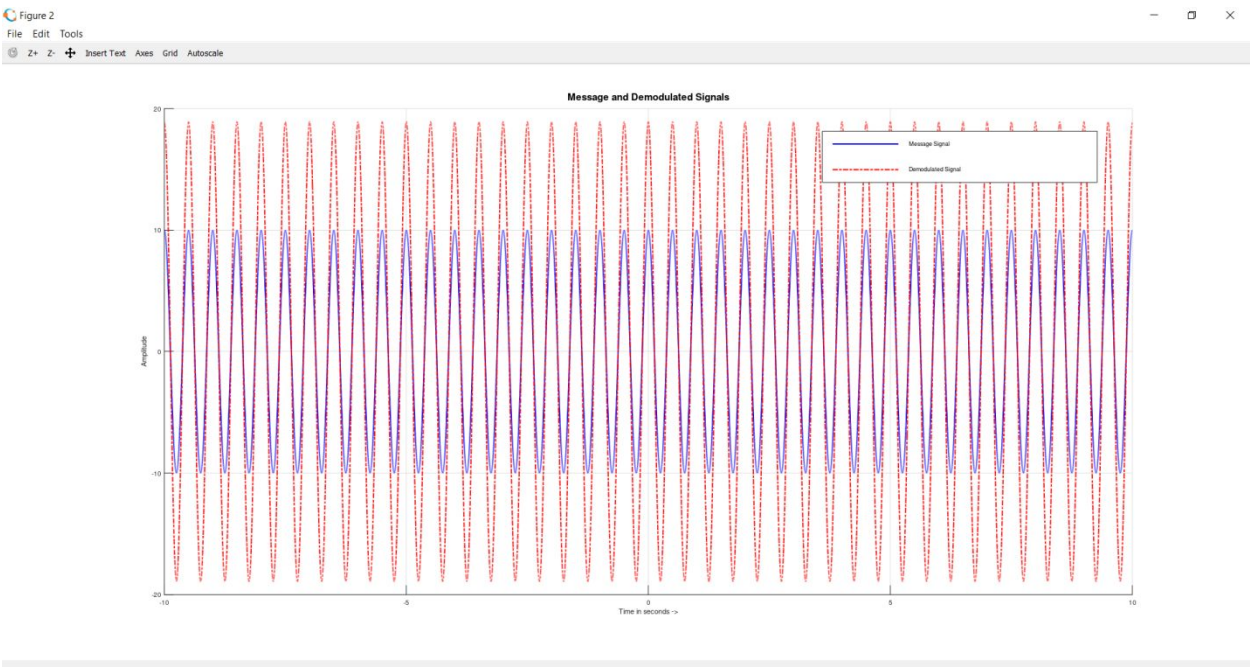
```
%% (2) Using Sampling Theorem for signal construction
Fs=2000;
T=1/Fs;
L=40000; % Keep this high in this code for good results
t=(-floor(L/2):floor(L/2))*T;
f=(-floor(L/2):floor(L/2))*Fs/L;
%%

%% (3) Defining Message Signal with amplitude = Am
kf=250;
Am=10;
fm=2;
msg_sig=Am*cos(2*pi*fm*t);
%%

%% (4) Defining Carrier Signal
A=1;
fc=100;
car_sig=A*cos(2*pi*fc*t);
%%

%% (5) Performing FM using actual equation
FM_sig=A*cos(2*pi*fc*t+kf*T*cumtrapz(msg_sig));
%%
```

```
%% (6) Taking derivative
derivative_FM_sig=diff(FM_sig);
derivative_FM_sig=[0 derivative_FM_sig];
%%

%% (7) Envelope Detection
derivative_FM_sig=derivative_FM_sig.^2;
derivative_fft=fftshift(fft(derivative_FM_sig));

f_lp=1.5*fm;
filter1=0*f;
filter1((f<f_lp) &  (f>-f_lp))=1;
filtered_signal_freq_domain=filter1.*derivative_fft;
demodulated_sig=ifft(ifftshift(filtered_signal_freq_domain));
demodulated_sig=demodulated_sig-mean(demodulated_sig);
%%

%% (8) Plotting Logic
figure
hold on
grid on
plot(t,FM_sig,'linewidth',2)
xlabel('Time in seconds ->')
ylabel('Amplitude')
title('Frequency Modulated Signal','Fontsize',15);
```

```
legend('FM')

figure
hold on
grid on
plot(t,msg_sig,'b','linewidth',2)
plot(t,60*demodulated_sig,'r-.','linewidth',2)
xlabel('Time in seconds ->')
ylabel('Amplitude')
title('Message and Demodulated Signals','Fontsize',15);
legend('Message Signal','Demodulated Signal')
```

Observations :





Result : The given FM Wave is demodulated.

Precautions : Make sure the code is written properly

The output is taken appropriately.

EXPERIMENT – 7

Aim : WAPT compare magnitude spectrum and phase spectrum FM vs NBFM

Theory : Modulating a carrier signal produces sidebands, which are a complex mixture of frequencies either side of the carrier frequency. This occurs for both AM and FM and all other kinds of modulation as well. AM's sidebands are fairly simple, being the sum and difference frequencies between the carrier and the modulating signal. As such they're easy to predict - if your maximum modulating frequency is X kHz, then the maximum width of the two sidebands are X, so the overall bandwidth is 2X.

FM also produces sidebands, but they are much more complex mathematically than AM sidebands. They are related not only to the modulating frequency but to the amount of deviation. Large deviation produces much more complex sidebands that occupy much more bandwidth.

NBFM produces sidebands that are very similar to AM sidebands - they are slightly more complex, but have a similar characteristic in that the total bandwidth is 2X, where X is the maximum modulating frequency. By contrast WBFM produces much more complex sidebands that occupy quite a bit more than 2X. The extra bandwidth allows more information to be carried in a much more robust, interference rejecting way, so WBFM is great for high-fidelity signals. NBFM allows many more channels to be fitted into a given band, and is adequate for speech.

WBFM is what is used for broadcast FM. NBFM is used for 2-way short range radio devices.

Software Used : GNU OCTAVE

Program :

```
close all

clear all

clc

%%
```

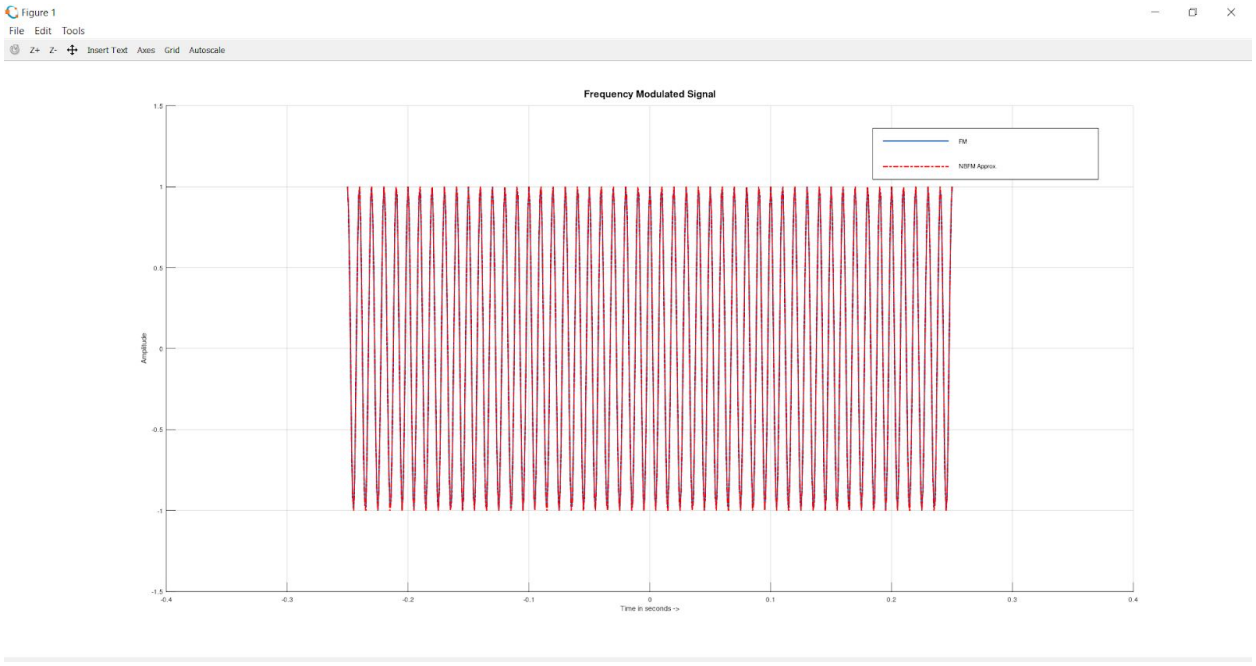
```
%% (2) Using Sampling Theorem for signal construction
Fs=1200;
T=1/Fs;
L=600;
t=(-floor(L/2):floor(L/2))*T;
f=(-floor(L/2):floor(L/2))*Fs/L;
%%

%% (3) Defining Message Signal with amplitude = Am
kf=250;
Am=.1;
fm=20;
msg_sig=Am*cos(2*pi*fm*t);
%%

%% (4) Defining Carrier Signal
A=1;
fc=100;
car_sig=A*cos(2*pi*fc*t);
%%

%% (5) Performing FM & NBFM modulations
FM_sig=A*cos(2*pi*fc*t+kf*T*cumtrapz(msg_sig));
NBFM_Approx_sig=A*[cos(2*pi*fc*t)-kf*T*cumtrapz(msg_sig).*sin(2*pi*fc*t)];
```

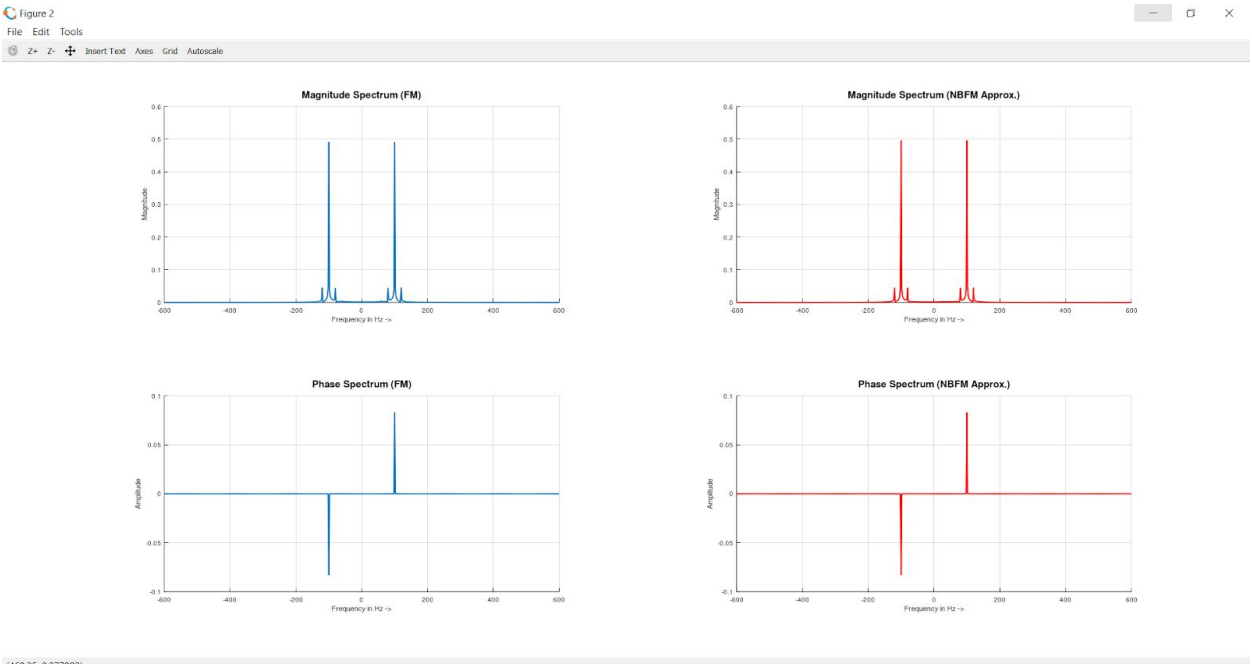
```
%%

%% (6) Bringing both signals in Frequency Domain
mod_fft=fftshift(fft(FM_sig));
NBFM_fft=fftshift(fft(NBFM_Approx_sig));
%%

%% (7) Plotting Logic
figure
hold on
grid on
plot(t,FM_sig,'linewidth',2)
plot(t,NBFM_Approx_sig,'r-.','linewidth',2)
xlabel('Time in seconds ->')
ylabel('Amplitude')
title('Frequency Modulated Signal','Fontsize',15);
legend('FM','NBFM Approx.')

figure
subplot(2,2,1)
hold on
grid on
plot(f,abs(mod_fft)/L,'linewidth',2)
xlabel('Frequency in Hz ->')
ylabel('Magnitude')
```

```
title('Magnitude Spectrum (FM)','Fontsize',15);

subplot(2,2,2)
hold on
grid on
plot(f,abs(NBFM_fft)/L,'r','linewidth',2)
xlabel('Frequency in Hz ->')
ylabel('Magnitude')
title('Magnitude Spectrum (NBFM Approx.)','Fontsize',15);

mod_fft=fftshift(fft(FM_sig));
NBFM_fft=fftshift(fft(NBFM_Approx_sig));
mod_fft(abs(mod_fft)<.7*max(abs(mod_fft)))=0;
NBFM_fft(abs(NBFM_fft)<.7*max(abs(NBFM_fft)))=0;

subplot(2,2,3)
hold on
grid on
plot(f,angle(mod_fft)/pi,'linewidth',2)
xlabel('Frequency in Hz ->')
ylabel('Amplitude')
title('Phase Spectrum (FM)','Fontsize',15);

subplot(2,2,4)
hold on
```

```
grid on
plot(f,angle(NBFM_fft)/pi,'r','linewidth',2)
xlabel('Frequency in Hz ->')
ylabel('Amplitude')
title('Phase Spectrum (NBFM Approx.)','Fontsize',15);
%%
```

Observation :





Result : Comparison of FM vs NBFM is seen through a graph

Precautions : Make sure the code is written properly

The output is taken appropriately.

EXPERIMENT – 8

Aim : WAPT compare magnitude spectrum and phase spectrum of FM, NBFM and AM

Theory : FM also produces sidebands, but they are much more complex mathematically than AM sidebands. They are related not only to the modulating frequency but to the amount of deviation. Large deviation produces much more complex sidebands that occupy much more bandwidth.

NBFM produces sidebands that are very similar to AM sidebands - they are slightly more complex, but have a similar characteristic in that the total bandwidth is 2X, where X is the maximum modulating frequency. By contrast WBFM produces much more complex sidebands that occupy quite a bit more than 2X. The extra bandwidth allows more information to be carried in a much more robust, interference rejecting way, so WBFM is great for high-fidelity signals. NBFM allows many more channels to be fitted into a given band, and is adequate for speech.

AM works by modulating (varying) the amplitude of the signal or carrier transmitted according to the information being sent, while the frequency remains constant. This differs from FM technology in which information (sound) is encoded by varying the frequency of the wave and the amplitude is kept constant.

Software Used : GNU OCTAVE

Program :

```
Fs=2000;

T=1/Fs;

L=500;

t=(-floor(L/2):floor(L/2))*T;

f=(-floor(L/2):floor(L/2))*Fs/L;

%%
```

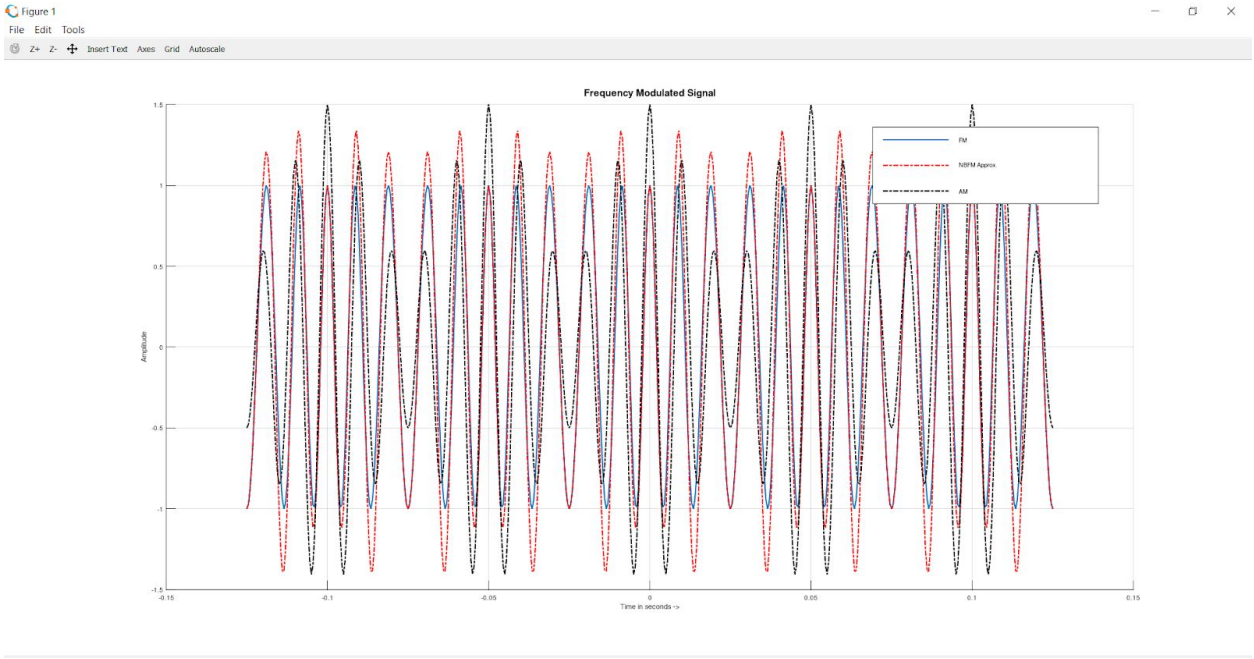
```
%% (3) Defining Message Signal with amplitude = Am
kf=250;
Am=.5;
fm=20;
msg_sig=Am*cos(2*pi*fm*t);
%%


%% (4) Defining Carrier Signal
A=1;
fc=100;
car_sig=A*cos(2*pi*fc*t);
%%

%% (5) Performing FM, NBFM and AM modulations
FM_sig=A*cos(2*pi*fc*t+kf*T*cumtrapz(msg_sig));
NBFM_Approx_sig=A*[cos(2*pi*fc*t)-kf*T*cumtrapz(msg_sig).*sin(2*pi*fc*t)];
AM_sig=A*cos(2*pi*fc*t)+msg_sig.*cos(2*pi*fc*t);
%%


%% (6) Bringing all signals in Frequency Domain
mod_fft=fftshift(fft(FM_sig));
NBFM_fft=fftshift(fft(NBFM_Approx_sig));
AM_fft=fftshift(fft(AM_sig));
%%
```

```
%% (7) Plotting Logic
figure
hold on
grid on
plot(t,FM_sig,'linewidth',2)
plot(t,NBFM_Approx_sig,'r-.','linewidth',2)
plot(t,AM_sig,'k-.','linewidth',2)
xlabel('Time in seconds ->')
ylabel('Amplitude')
title('Frequency Modulated Signal','Fontsize',15);
legend('FM','NBFM Approx.','AM')

figure
subplot(3,2,1)
hold on
grid on
plot(f,abs(mod_fft)/L,'linewidth',2)
xlabel('Frequency in Hz ->')
ylabel('Magnitude')
title('Magnitude Spectrum (FM)','Fontsize',15);

subplot(3,2,3)
hold on
grid on
```

```
plot(f,abs(NBFM_fft)/L,'r','linewidth',2)
xlabel('Frequency in Hz ->')
ylabel('Magnitude')
title('Magnitude Spectrum (NBFM Approx.)','Fontsize',15);

subplot(3,2,5)
hold on
grid on
plot(f,abs(AM_fft)/L,'k','linewidth',2)
xlabel('Frequency in Hz ->')
ylabel('Magnitude')
title('Magnitude Spectrum (AM)','Fontsize',15);


mod_fft=fftshift(fft(FM_sig));
NBFM_fft=fftshift(fft(NBFM_Approx_sig));
AM_fft=fftshift(fft(AM_sig));
thresh1=.7;
mod_fft(abs(mod_fft)<thresh1*max(abs(mod_fft)))=0;
NBFM_fft(abs(NBFM_fft)<thresh1*max(abs(NBFM_fft)))=0;
AM_fft(abs(AM_fft)<thresh1*max(abs(AM_fft)))=0;

subplot(3,2,2)
hold on
grid on
plot(f,angle(mod_fft)/pi,'linewidth',2)
```

```
xlabel('Frequency in Hz ->')
ylabel('Phase in Pi')
title('Phase Spectrum (FM)','Fontsize',15);

subplot(3,2,4)
hold on
grid on
plot(f,angle(NBFM_fft)/pi,'r','linewidth',2)
xlabel('Frequency in Hz ->')
ylabel('Phase in Pi')
title('Phase Spectrum (NBFM Approx.)','Fontsize',15);

subplot(3,2,6)
hold on
grid on
plot(f,angle(AM_fft)/pi,'k','linewidth',2)
xlabel('Frequency in Hz ->')
ylabel('Phase in Pi')
title('Phase Spectrum (AM)','Fontsize',15);%%
```

Observations :



Result : We can compare between FM,NBFM and AM using magnitude and phase spectrum plots

Precautions : Make sure the code is written properly

The output is taken appropriately.

EXPERIMENT – 9

Aim : WAPT generate noise in a baseband system

Theory : Random noise signals provide secure communications because they cannot, in general, be detected using conventional receivers and are jam-resistant. We describe the theoretical underpinnings of a novel spread spectrum technique that can be used for covert communications using transmissions over orthogonal polarization channels. The noise key and the noise-like modulated signal are transmitted over orthogonal polarizations to mimic unpolarized noise. Since the transmitted signal is featureless and appears unpolarized and noise-like, linearly polarized receivers are unable to identify, detect, or otherwise extract useful information from the signal. The wide bandwidth of the transmitting signal provides significant immunity from interference. Dispersive effects caused by the atmosphere and other factors are significantly reduced since both polarization channels operate over the same frequency band. The received signals are mixed together to accomplish demodulation.

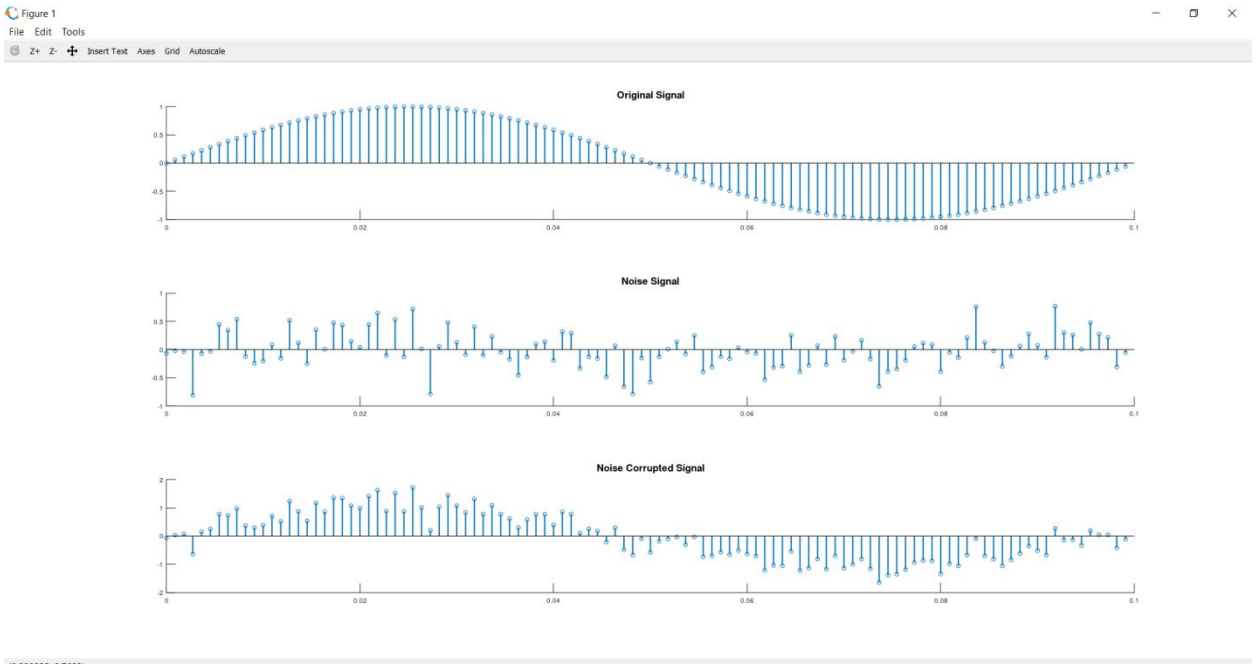
Software Used : GNU OCTAVE

Program :

```
close all

clear all

clc

pkg load communications

Fs=1100;

T=1/Fs;

L=110;

t=(0:L-1)*T;


sig1=sin(2*pi*10*t);
```

```
figure
subplot(3,1,1)
stem(t,sig1,'linewidth',2)
title('Original Signal','fontsize',15);

sig2=0*sig1+1;
sig2=awgn(sig2,10,'measured')
subplot(3,1,2)
stem(t,sig2-1,'linewidth',2)
title('Noise Signal','fontsize',15);

subplot(3,1,3)
stem(t,sig1+sig2-1,'linewidth',2)
title('Noise Corrupted Signal','fontsize',15);
```

Observations :



Result : Noise is generated in the baseband signal

Precautions : Make sure the code is written properly

The output is taken appropriately.

EXPERIMENT- 10

Aim : WAP to study Capture effect in FM enhancing Interference in Angle modulated signal

Theory : In a radio receiver, the capture effect, or FM capture effect, is a phenomenon associated with FM reception in which only the stronger of two signals at, or near, the same frequency or channel will be demodulated.

Capture effect is **caused by** the fact that responding tags may be scattered in different distances from the reader; the signals from a tag may overwhelm that from other responding tags that can be **captured** by the reader.

In order to study capture effect in FM, focus on both the carrier signals. 1st one is a Cosine wave and second one is a square wave.

Capture effect is based on the magnitude of the carrier wave. First take A1=100, A2=1. See effect.

Then take A1=1, A2=100. Then see the effect.

Software Used : GNU OCTAVE

Program : (a) When A1=100 and A2=1 :

```
% Signals and CAPTURE EFFECT

close all

clear all

clc

%%


%% (2) Using Sampling Theorem for signal construction

Fs=3000;

T=1/Fs;
```

```
L=50000; % Keep this high in this code for good results
t=(-floor(L/2):floor(L/2))*T;
f=(-floor(L/2):floor(L/2))*Fs/L;
%%

%% (3) Defining TWO DIFFERENT Message Signals
% for two nearby carriers
kf=250;
Am=1;
fm=2;
msg_sig1=Am*cos(2*pi*fm*t);

msg_sig2=Am*cos(2*pi*(fm/4)*t);
msg_sig2(msg_sig2>0)=1;
msg_sig2(msg_sig2<0)=-1;

%%
%% (4) Defining TWO NEARBY CARRIER Signals
A1=100;
fc1=100;
car_sig1=A1*cos(2*pi*fc1*t);

A2=1;
fc2=fc1+10;
car_sig2=A2*cos(2*pi*fc2*t);
```

```
%%

%% (5) Performing FM using actual equation
FM_sig1=A1*cos(2*pi*fc1*t+kf*T*cumtrapz(msg_sig1));
FM_sig2=A2*cos(2*pi*fc2*t+kf*T*cumtrapz(msg_sig2));
FM_sig=FM_sig1+FM_sig2;
%%


%% (6) Taking derivative
derivative_FM_sig=diff(FM_sig);
derivative_FM_sig=[0 derivative_FM_sig];
%%

%% (7) Envelope Detection
derivative_FM_sig=derivative_FM_sig.^2;
derivative_fft=fftshift(fft(derivative_FM_sig));

f_lp=10*fm;
filter1=0*f;
filter1((f<f_lp) &  (f>-f_lp))=1;
filtered_signal_freq_domain=filter1.*derivative_fft;
demodulated_sig=ifft(ifftshift(filtered_signal_freq_domain));
demodulated_sig=demodulated_sig-mean(demodulated_sig);
%%
```

```
%% (8) Plotting Logic
figure
hold on
grid on
plot(t,msg_sig1,'b','linewidth',1)
plot(t,msg_sig2,'k','linewidth',2)
plot(t,.001*demodulated_sig,'r','linewidth',2)
xlabel('Time in seconds ->')
ylabel('Amplitude')
title('DEMO OF CAPTURE EFFECT IN FM','Fontsize',15);
legend('Message Signal 1','Message Signal 2','Demodulated Signal')
```

(b) When A1=1 and A2=100 :

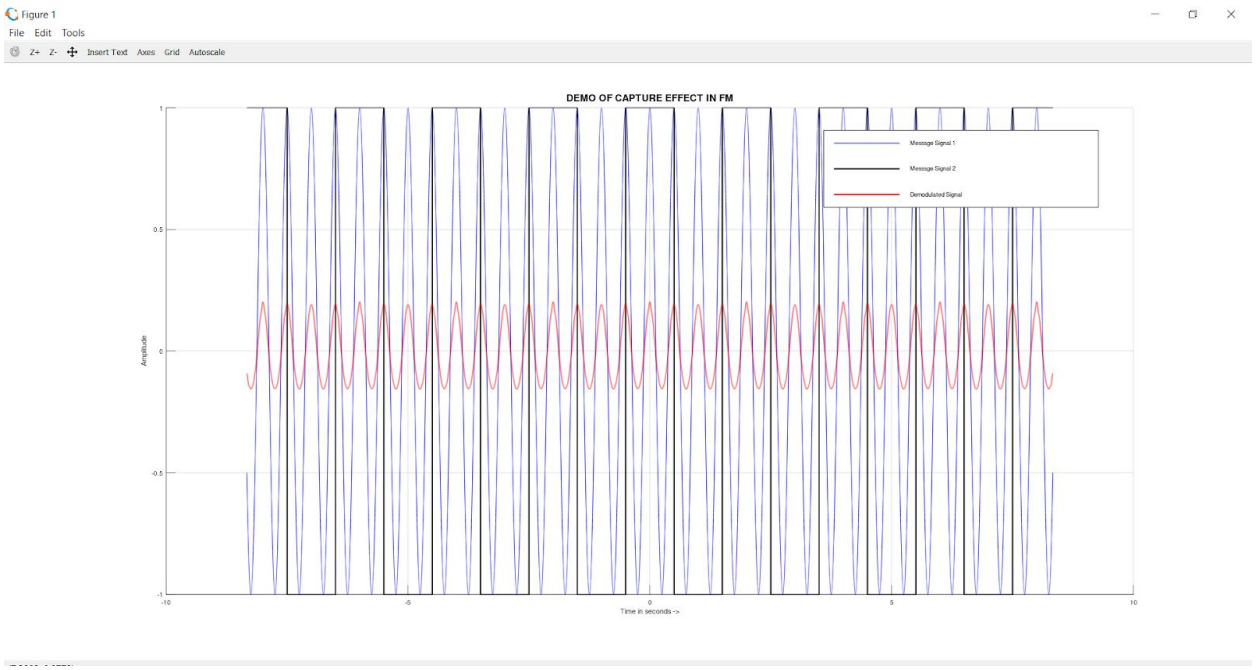
```
% Signals and CAPTURE EFFECT
close all
clear all
clc
%%

%% (2) Using Sampling Theorem for signal construction
Fs=3000;
T=1/Fs;
L=50000; % Keep this high in this code for good results
t=(-floor(L/2):floor(L/2))*T;
```

```
f=(-floor(L/2):floor(L/2))*Fs/L;
%%

%% (3) Defining TWO DIFFERENT Message Signals
% for two nearby carriers
kf=250;
Am=1;
fm=2;
msg_sig1=Am*cos(2*pi*fm*t);

msg_sig2=Am*cos(2*pi*(fm/4)*t);
msg_sig2(msg_sig2>0)=1;
msg_sig2(msg_sig2<0)=-1;

%%
%% (4) Defining TWO NEARBY CARRIER Signals
A1=1;
fc1=100;
car_sig1=A1*cos(2*pi*fc1*t);

A2=100;
fc2=fc1+10;
car_sig2=A2*cos(2*pi*fc2*t);
%%

%% (5) Performing FM using actual equation
```

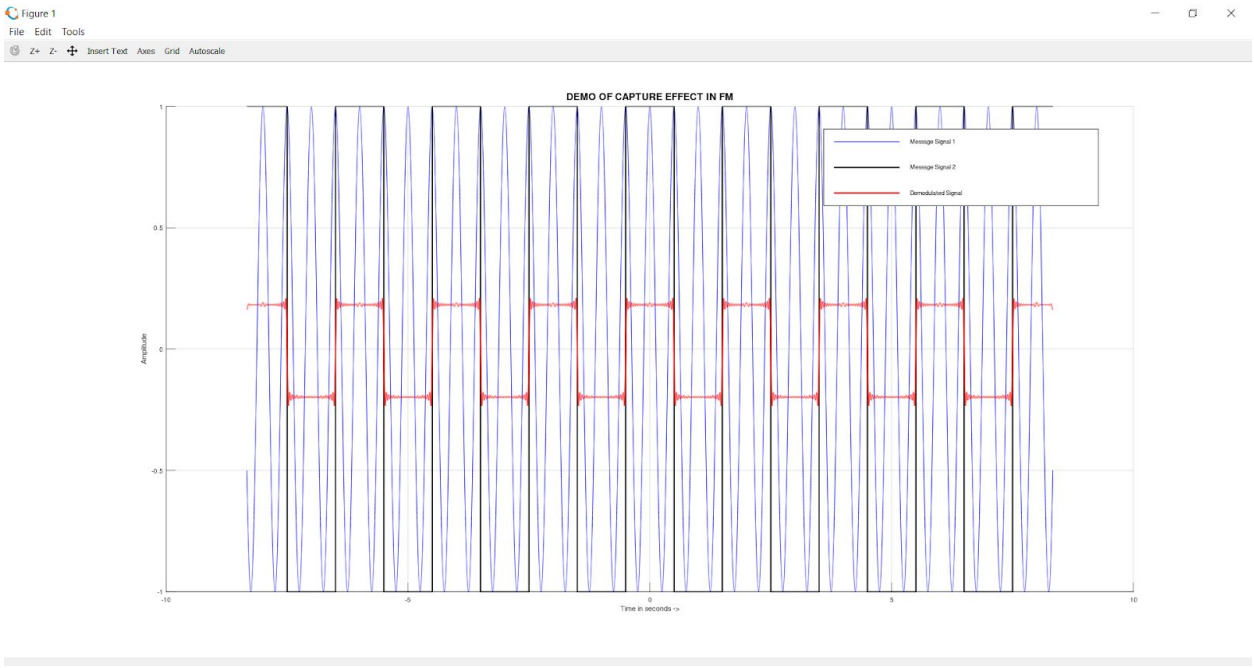
```
FM_sig1=A1*cos(2*pi*fc1*t+kf*T*cumtrapz(msg_sig1));
FM_sig2=A2*cos(2*pi*fc2*t+kf*T*cumtrapz(msg_sig2));
FM_sig=FM_sig1+FM_sig2;
%%

%% (6) Taking derivative
derivative_FM_sig=diff(FM_sig);
derivative_FM_sig=[0 derivative_FM_sig];
%%

%% (7) Envelope Detection
derivative_FM_sig=derivative_FM_sig.^2;
derivative_fft=fftshift(fft(derivative_FM_sig));

f_lp=10*fm;
filter1=0*f;
filter1((f<f_lp) &  (f>-f_lp))=1;
filtered_signal_freq_domain=filter1.*derivative_fft;
demodulated_sig=ifft(ifftshift(filtered_signal_freq_domain));
demodulated_sig=demodulated_sig-mean(demodulated_sig);
%%

%% (8) Plotting Logic
figure
hold on
```

```
grid on
plot(t,msg_sig1,'b','linewidth',1)
plot(t,msg_sig2,'k','linewidth',2)
plot(t,.001*demodulated_sig,'r','linewidth',2)
xlabel('Time in seconds ->')
ylabel('Amplitude')
title('DEMO OF CAPTURE EFFECT IN FM','Fontsize',15);
legend('Message Signal 1','Message Signal 2','Demodulated Signal')
```

<u>Observations :</u> (a) When A1=100 and A2=1 :



(b) When A1=1 and A2=100 :



Result : The capture effect is shown in the above graphs

Precautions : Make sure the code is written properly

The output is taken appropriately.