LAB MANUAL

ON

DIGITAL COMMUNICATION

EC-502

EXPERIMENT NO: 1

AIM: Fourier analysis plays an important role in communication theory. The main objectives of this experiment are:

1) To gain a good understanding and practice with Fourier series and Fourier Transform techniques, and their applications in communication theory.

2) Learn how to implement Fourier analysis techniques using MATLAB.

APPRATUS REQUIRED: Spectrum Analyzer, Function Generator

THEORY:

Fourier Series and Fourier Transform:

1) In MATLAB, go to the command window and type **Fourier_series_demo.m.** This will bring up a Graphical User Interface (GUI) that can be used to test and demonstrate many concepts and properties of the Fourier series expansion.

2) Now, consider a periodic signal $x(t)$. Compute and plot the discrete magnitude and phase spectra of this signal given by $x(t) = e-t / 2$ where $\in [0,\pi]$. For this, you need to use the Fast Fourier Transform (FFT) function in MATLAB (refer to the notes below for more details). For the expansion of the signal x(t), the number of harmonics *o N* to be used should be 32, the period *o T* is π , and the step size is *s o o t = T / N* . The output should be in two figure windows. The first window should contain $x(t)$ while the second window should contain both the magnitude and phase spectra versus a vector of harmonics indices (for example, n). You also need to include labels and titles in all plots. What can you observe from these plots?

3) In the MATLAB command window, type **Fourier_trans_demo.m** to launch a GUI that will demonstrate and review the basic properties of the Fourier transform. The basic function used is a rectangular unit pulse.

• First, introduce a certain time delay in the function, and notice what happens to the amplitude spectra. Explain why?

• Next, introduce different scaling factors and comment on what you are observing.

• Now, introduce a frequency shift, which means that the unit pulse is multiplied by a given sine or cosine signal with some frequency (later, we will see this is known as Amplitude Modulation). Referring to the basic properties of the FT, explain what you are observing in the plots.

OBSERVATION TABLE:

S.NO	FUNCTION IN TIME DOMAIN	FREQUENCY DOMAIN		
		FRQUENCY	AMPLITUDE	PHASE
1.	SINE AND COSINE WAVE			
2.	SQUARE WAVE			
3.	TRIANGULAR WAVE			
4.	SINGULARITY FUNCTION			

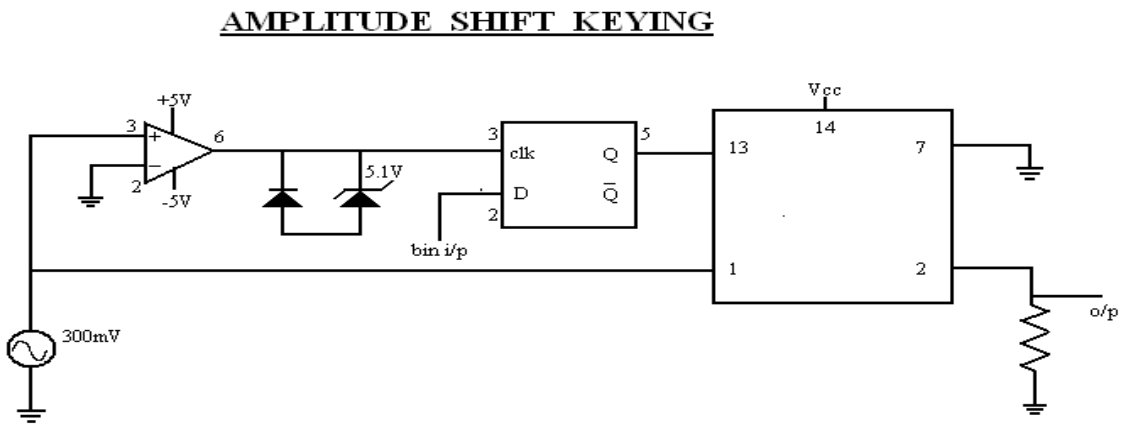
RESULT: Conversion of signals from time domain to frequency domain is done.

EXPERIMENT NO: 2

AIM: To set up a circuit to generate Binary Amplitude Shift keying and to plot the output waveforms.

APPRATUS REQUIRED: IC CD4016, IC 7474, Resistors, Zener diode 5.1, Breadboard, multimeter, CRO, Power supply, Signal generator.

CIRCUIT DIAGRAM:



THEORY: BASK is a type of modulation process in which the amplitude of the sinusoidal carrier is varying according to the incoming binary data. In BASK, sinusoidal carrier is transmitted for logic '1' and no carrier is transmitted for logic '0'.

Synchronized BASK modulator circuit consists of 3 sections, namely

a) Zero Crossing Detector

b) Sample & Hold circuit

c) Switch

The input sinusoidal signal is applied to the Zero Crossing Detector (+ve). The polar NRZ o/p from ZCD is converted into unipolar NRZ by using a zener diode. The DFF acts as S/H circuit. By applying the unipolar NRZ signal to the clock input. The frequency of the binary data signal is low as compared to the clock, satisfying the sampling theorem. The sample & Hold circuit acts as the synchronizer. The synchronized output from DFF is then applied to the switch. The analog

signal is applied to the input pin of IC 4016. If the control signal is logic high, then sinusoidal is transmitted at the o/p and if the control signal is low, the o/p goes to zero.

PROCEDURE:

- The Zero Crossing detector is assembled and checks the TTL compatibility.
- The Binary data is fed to the D input of the flip-flop & TTL compatible signal designed from the Zero Crossing Detector is given to the clock input of the DFF.
- **Q** & **Q'** outputs of the DFF are given to the control input of the IC 4016/ 4066. The amplitude of the carrier signal is fixed at 200mV.
- The carrier & 1800 phase shifted signals are given to the terminals of the analog switch and PSK output taken out of the other terminal terminated in a resistance.

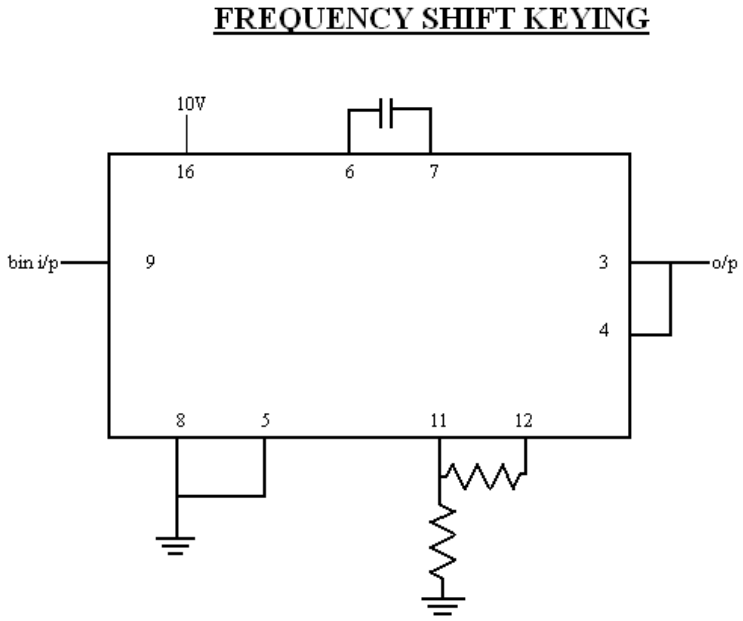
RESULT: The circuit is setup and the waveform is observed.

EXPERIMENT NO: 3

AIM: To design and set up a binary frequency shift keying generator circuit using PLL

APPRATUS REQUIRED: IC CD4046, capacitor-10μf, Resistor-100kohms,Breadboard ,multimeter ,CRO, Power supply ,Signal generator.

CIRCUIT DIAGRAM:



THEORY: BFSK is a linear method used for data modulation .In a BFSK system, symbols 1 and 0 are distinguished from each other by transmitting two different frequencies that differ by affixed amount. Binary 0 is transmitted by a shot duration of one frequency, f1 and binary 1 is transmitted by a short duration of another frequency, f2.The information about the transmitted data residues in the carrier frequency.BFSK signal can be considered as a sum of signals having two frequencies. In BFSK frequency of the carrier is shifted between two discrete values, one representing binary zero and other representing binary one. The carrier output amplitude is constant.BFSK signal can be considered to comprised of ASK signal with carrier frequency f1 and f2. BFSK is generated using voltage controlled oscillator (VCO) using PLL IC 4046.It generates frequency f1 at logic zero input, and frequency f2 at logic one input.

When the input is logic zero, f1=1/R2(c+32pF)

When the input is logic one, f2= [1/R1(c+32pF)] +f1

PROCEDURE:

- The circuit is assembled on the bread board. Initially no binary input data is given.

- Pin 9 is shorted to pin 16 to check the IC and free running frequency. Now TTL data input is given from signal generator. The BFSK waveform is observed and plotted.

RESULT: The circuit is setup and the waveform is observed as,

For 0, frequency =

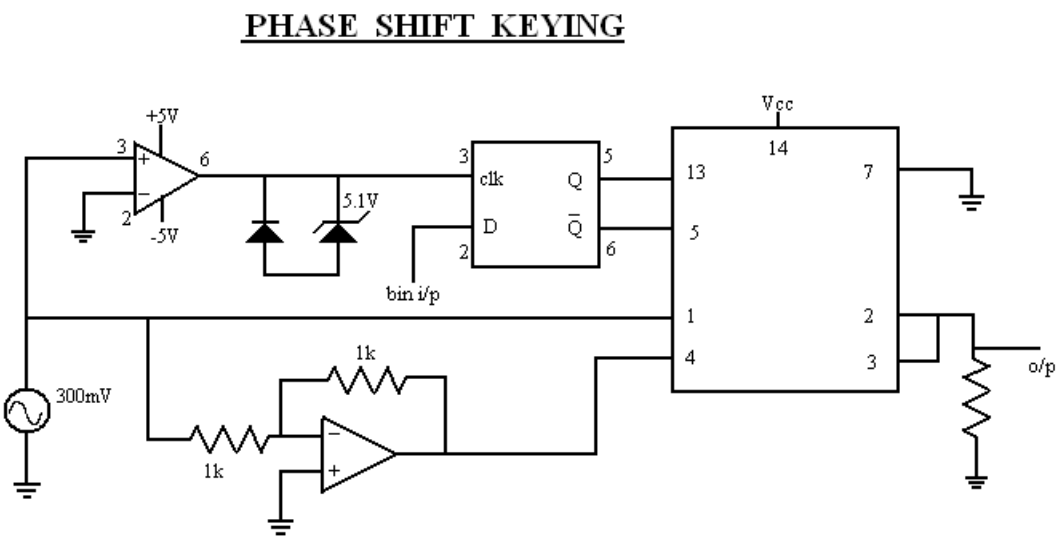
For 1, frequency =

EXPERMENT NO.4

AIM: To design and set up a Binary Phase Shift keying generator and to study its performance.

APPRATUS REQUIRED: IC CD4016, IC 7474, Resistors, Zener diode 5.1, Breadboard, multimeter, CRO, Power supply, Signal generator.

CIRCUIT DIAGRAM:



THEORY: BPSK is a digital modulation scheme in which the phase of the carrier is modulated to represent the binary values. In BPSK, Binary phase is changed between 0 & 180_0 by the bipolar digital signal. Binary states '1' & '0' are represented by by the –ve & +ve polarities of the digital signal.

Synchronized PSK modulation consists of:

d) Zero Crossing Detectors

e) Sample & Hold circuit

f) Inverter

g) Switch

The input sinusoidal signal is applied to the Zero Crossing Detector (+ve). The polar NRZ o/p from ZCD is converted into unipolar NRZ by using a zener diode. The DFF acts as S/H circuit. By applying the unipolar NRZ signal to the clock input. The frequency of the binary data signal is low as compared to the clock, satisfying the sampling theorem. The sample & Hold circuit acts

as the synchronizer. The synchronized output from DFF is then applied to the first control pin of the switch. The **Q'** output from the DFF is applied to the second control pin of the switch. The outputs are tied together. Then, a sinusoidal signal is transmitted for logic '1' and inverted sinusoidal signal is transmitted for logic '0'. PSK signals can be considered on 2 ASK signal with phases 0 and 1800.

PROCEDURE:

- In the BPSK, The Zero Crossing detector is assembled and its output is connected to the zener diode as shown in the figure, and checked for TTL compatibility.
- The Binary data is fed to the D input of the flip-flop & TTL compatible signal designed from the Zero Crossing Detector is given to the clock input of the DFF.
- **Q** & **Q'** outputs of the DFF are given to the control input of the IC 4016/ 4066. The amplitude of the carrier signal is fixed at 200mV. The carrier & 1800 phase shifted signals are given to the terminals of the analog switch and PSK output taken out of the other terminal terminated in a resistance.

RESULT: The circuit is setup and the waveform is observed.

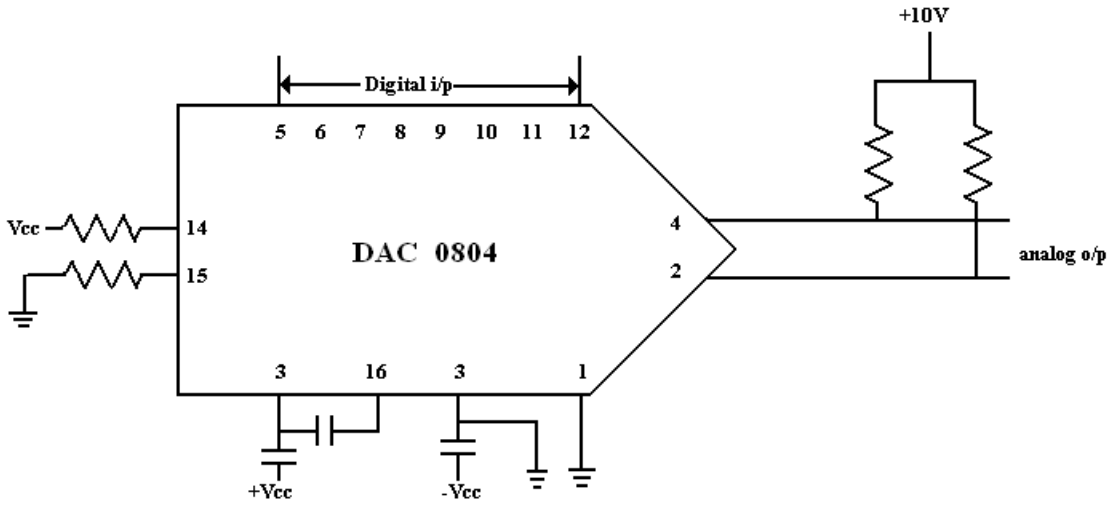
EXPERIMENT NO: 5

AIM: To familiarize the digital to analog converter DAC 0800.

APPRATUS REQUIRED: DAC 0800, resistors, capacitors, signal generator, dc supply, bread board and CRO.

CIRCUIT DIAGRAM:

DIGITAL TO ANALOG CONVERTOR



THEORY: A DAC accepts an n bit i/p word in binary and produce an analog signal proportional to it. DAC 0800 serves as monolithic 8_bit high speed current. O/p DAC is having typical settling time of 100ns.The analog signal at its o/p will be current signal. In order to convert it to a voltage signal a current to voltage converter is used.

PROCEDURE:

- The circuit is wired on the IC trainer kit. Digital i/p is applied at pin 5 to 12 of DAC 0800.
- The analog o/p are obtained at pin 4 and pin 2.
- Any of these is grounded and the other is taken as the i/p of current to voltage converter.
- The analog o/p is taken from the o/p of Opamp.

RESULT: The circuit was set up and the resolution is found to be 0.9v/bit.

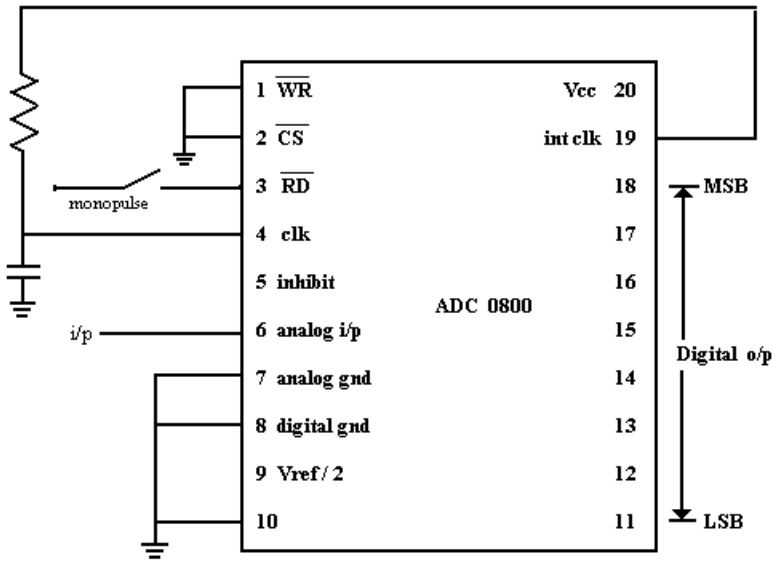
EXPERIMENT NO: 6

AIM: To convert analog values to corresponding digital values.

APPRATUS REQUIRED: IC 0804, Capacitor, Resistor breadboard, CRO, DC source multimeter.

CIRCUIT DIAGRAM:

ANALOG TO DIGITAL CONVERTOR



THEORY: ADC 0804 is a20 pin CMOS IC that perform analog to digital conversion using successive approximation methods with the conversion delay less than 100ms. It converts analog input values to 8 bit digital output values with a resolution of 19 mv. It has internal clock generator that produces a frequency of f =1/1.1 RC. If desired an external clock frequency can be used using clk IN (pin no. 4).

PROCEDURE:

- Circuit is wired on the breadboard as shown in the figure.
- Analog input is applied to Vin+ and Vin- is grounded.
- The cs pin is made low.
- The wr pin is made low to start with conversion.
- Rd is made low to enable the digital o/p buffer.
- The digital o/p pins will then have logic levels representing the last ADC.

RESULT: The circuit was set up and the resolution is found to be 0.9v/bit.

EXPERIMENT NO: 7

AIM: to plot amplitude modulation waveform using MATLAB.

SOFTWARE REQUIRED: MATLAB 10.0 Version.

CODING:

```
clc;
clear all;
close all;
t=0:.001:1;
set(0,'defaultlinelinewidth',2);
A=5;%Amplitude of signal
fm=input('Message frequency=');%Accepting input value
fc=input('Carrier frequency=');%Accepting input value (f2>f1)
mi=input('Modulation Index=');%Modulation Index
Sm=A*(sin(2*pi*fm*t));%Message Signal
subplot(3,1,1);%Plotting frame divided in to 3 rows and this fig appear at
1st
plot(t,Sm);
xlabel('Time');
ylabel('Amplitude');
title('Message Signal');
grid on;

Sc=A*(sin(2*pi*fc*t));%Carrier Signal
subplot(3,1,2);
plot(t,Sc);
xlabel('Time');
ylabel('Amplitude');
title('Carrier Signal');
grid on;

Sfm=(A+mi*Sm).*(sin(2*pi*fc*t));%AM Signal, Amplitude of Carrier changes to
(A+Message)
subplot(3,1,3);
plot(t,Sfm);
xlabel('Time');
ylabel('Amplitude');
title('AM Signal');
grid on;
```

RESULT: Plotting has done with undermodulation,100 % modulation and over modulated wave.

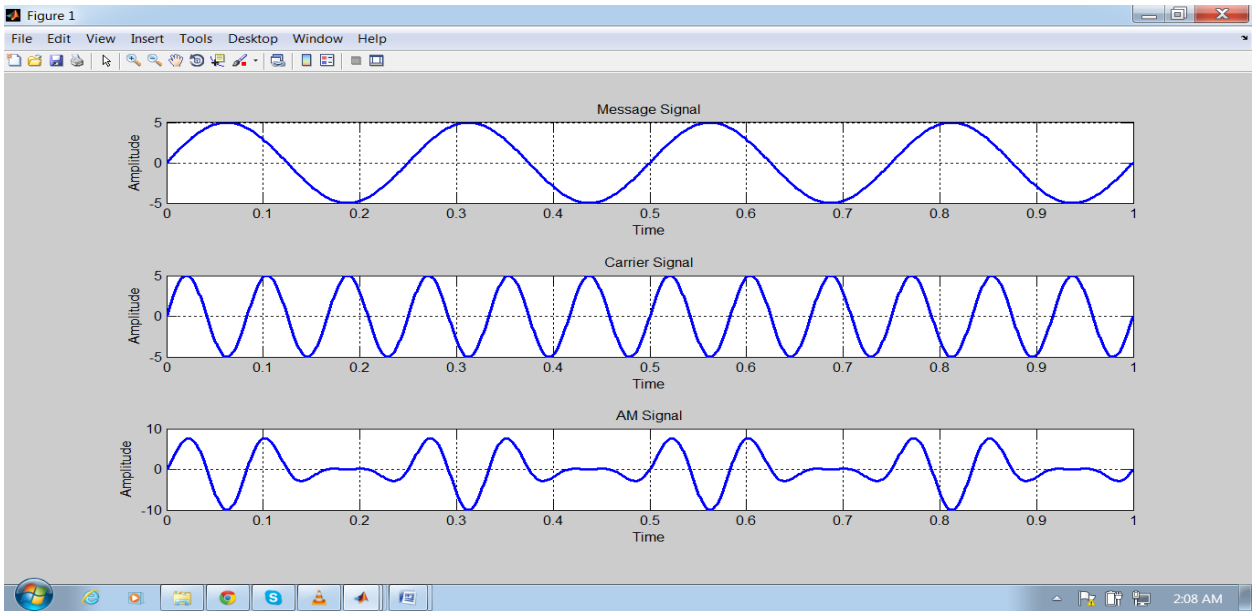


Fig 1:100% modulated wave

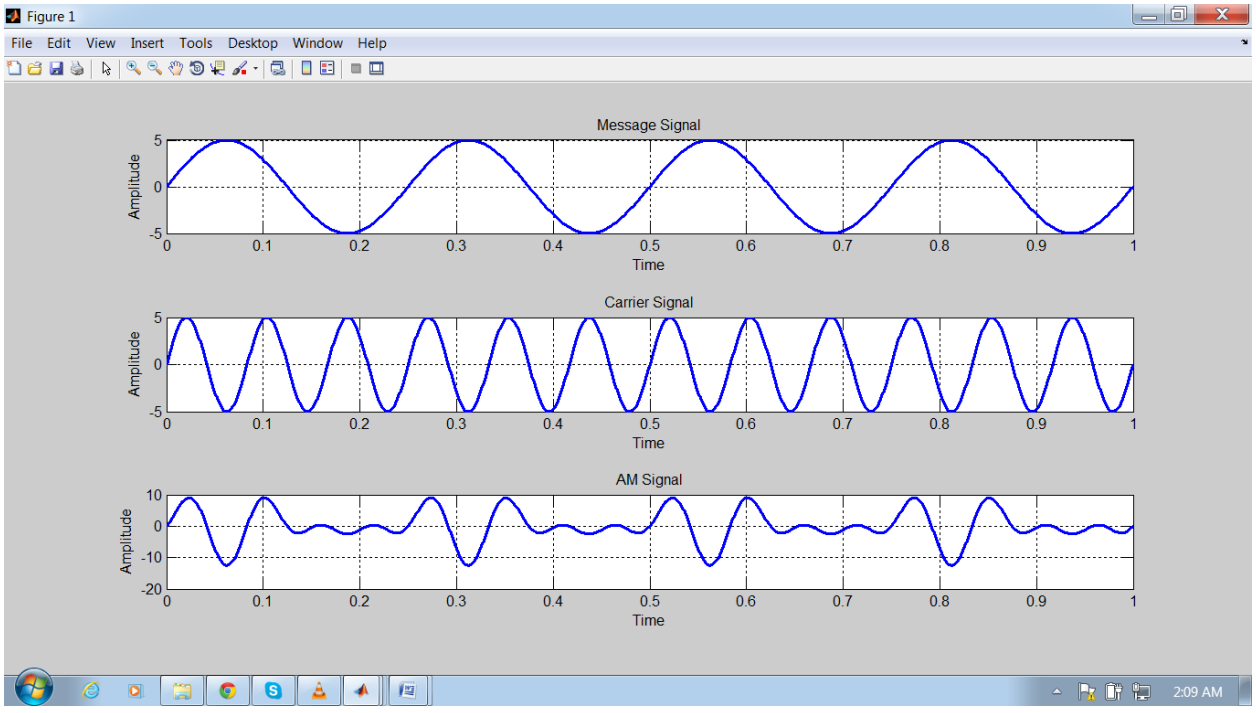


Fig 2: Overmodulated wave

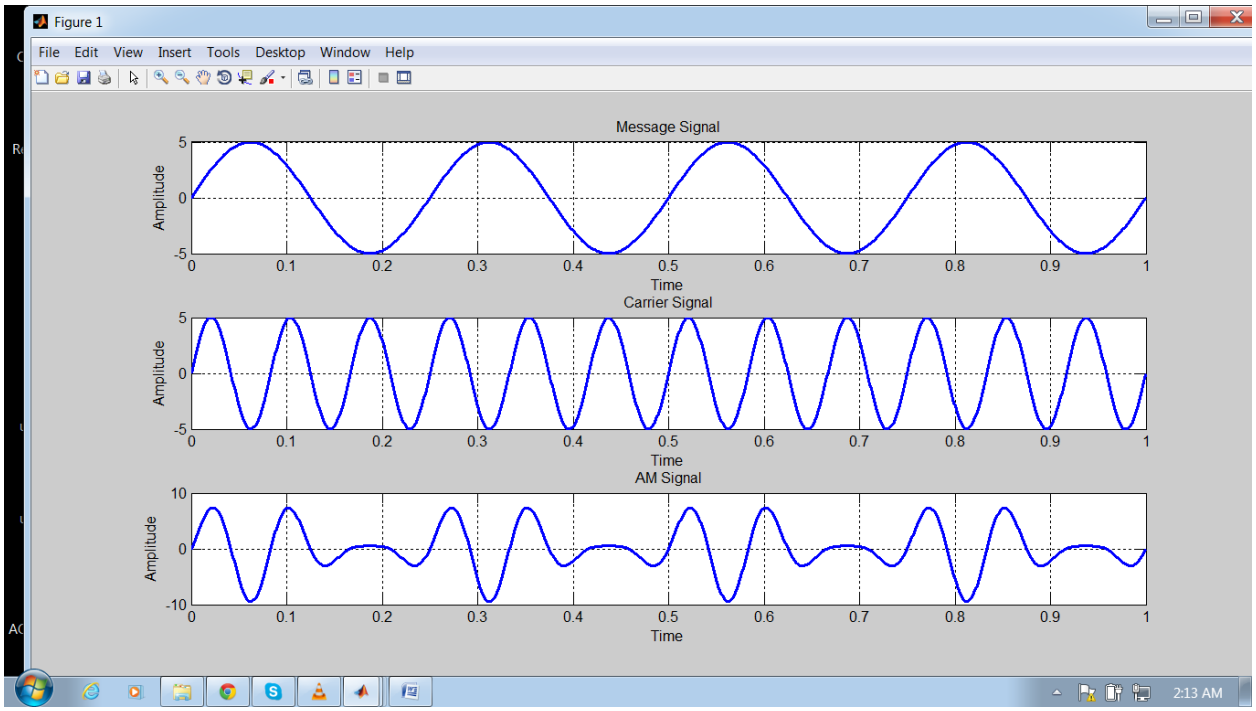


Fig:3 Under modulated wave

EXPERIMENT NO.8

AIM: To plot frequency shift keying waveform using MATLAB.

SOFTWARE: MATLAB 10.0 version.

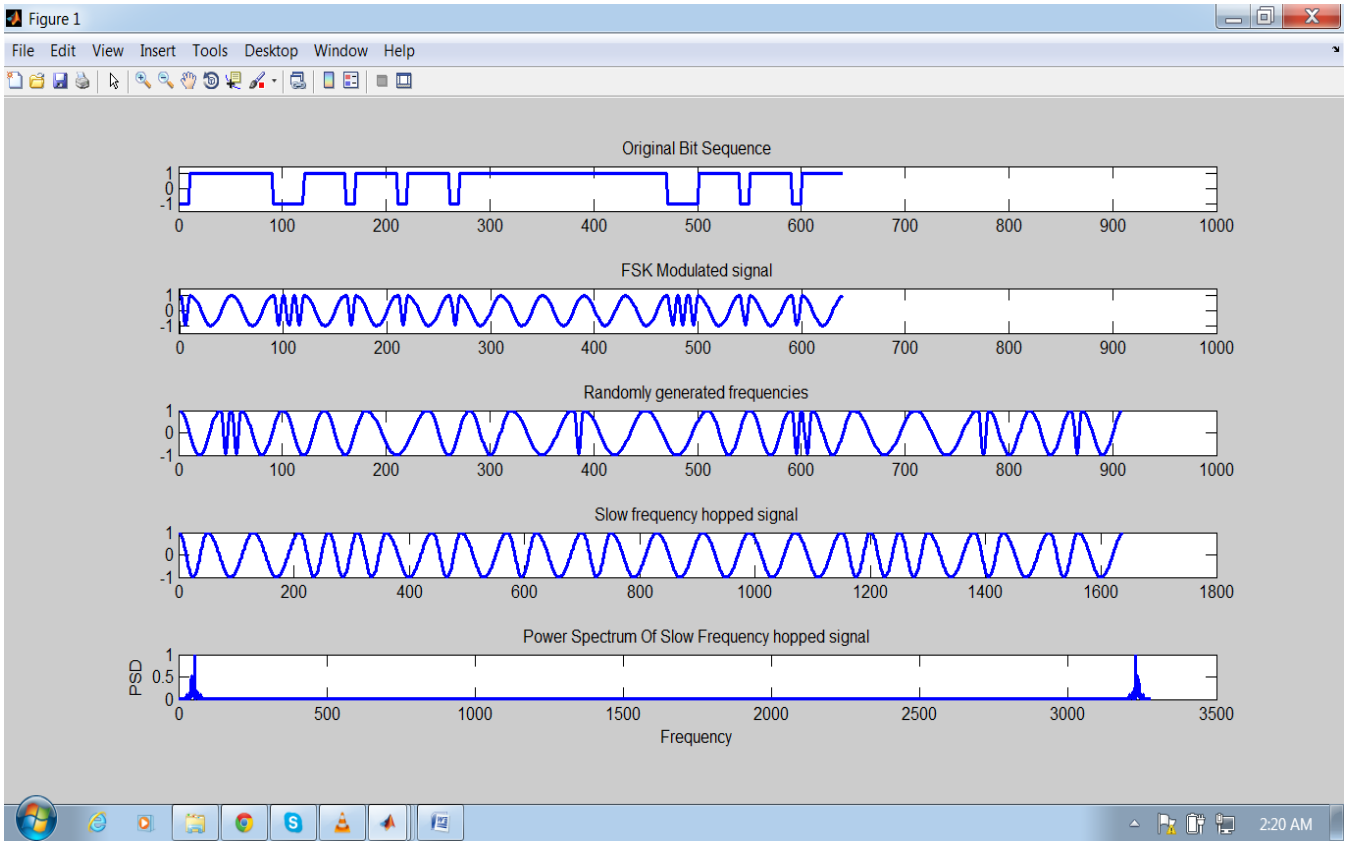
CODING:

```
%
%

% Slow Frequency Hop Spread Spectrum
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
clear
% Generating the data bit pattern
s=round(rand(1,25));

% FSK modulation of carrier by data signal
t1=0:2*pi/9:2*pi;
t2=0:2*pi/39:2*pi;
ca1=cos(t1);
ca2=cos(t2);
signal=[];
carrier=[];
for k=1:25
    if s(1,k)==0
        sig=-ones(1,10);    % 10 minus ones for bit 0
        wavef = ca1;
    else
        sig=ones(1,40);     % 40 ones for bit 1
        wavef = ca2;
    end
    signal=[signal sig];
    carrier=[carrier wavef];
end
subplot(5,1,1)
plot(signal);
axis([0 1000 -1.5 1.5]);
title('Original Bit Sequence');
subplot(5,1,2);
plot(carrier);
axis([-1 1000 -1.5 1.5]);
title('FSK Modulated signal');

% Preparation of 3 new carrier frequencies
ta1=[0:2*pi/9:2*pi];
ta2=[0:2*pi/39:2*pi];
ta3=[0:2*pi/59:2*pi];
cb1=cos(ta1);
cb2=cos(ta2);
cb3=cos(ta3);

spread_signal=[];
for n=1:25
c = randint(1,1,[1 3]);
        switch(c)
        case(1)
```

```
            spread_signal=[spread_signal cb1];
        case(2)
            spread_signal=[spread_signal cb2];
        case(3)
            spread_signal=[spread_signal cb3];
        end
 end
subplot(5,1,3)
plot(spread_signal);
title('Randomly generated frequencies');

% Generation of sum frequency
t1=0:2*pi/19:2*pi;
t2=0:2*pi/49:2*pi;
t3=0:2*pi/69:2*pi;
t4=0:2*pi/79:2*pi;
t5=0:2*pi/99:2*pi;
c1=cos(t1);
c2=cos(t2);
c3=cos(t3);
c4=cos(t4);
c5=cos(t5);

%Generation of transmitted sequence
txsig=[];
for i=1:25
    if (s(1,i)==0)& (c==1)
        txs=c1;
    else if (s(1,i)==0) & (c==2)
        txs=c2;
        else if (s(1,i)==0)& (c==3)
             txs=c3;
                else if (s(1,i)==1) & (c==1)
                     txs=c2;
                    else if (s(1,i)==1)&(c==2)
                         txs=c4;
                        else if (s(1,i)==1)&(c==3)
                             txs=c5;

                            end

                        end

                    end

            end

        end

    end
txsig=[txsig txs];
end
subplot(5,1,4)
plot(txsig)
title('Slow frequency hopped signal');
```

```
% Power Spectrum Of Slow Frequency hopped signal
power_spectrum=abs(fft(xcorr(txsig)));
subplot(5,1,5)
plot(power_spectrum/max(power_spectrum))
xlabel('Frequency')
ylabel('PSD')
title('Power Spectrum Of Slow Frequency hopped signal');
```

RESULT: Waveform has plotted.



EXPERIMENT NO.9

AIM: To plot Amplitude shift keying waveform using MATLAB.

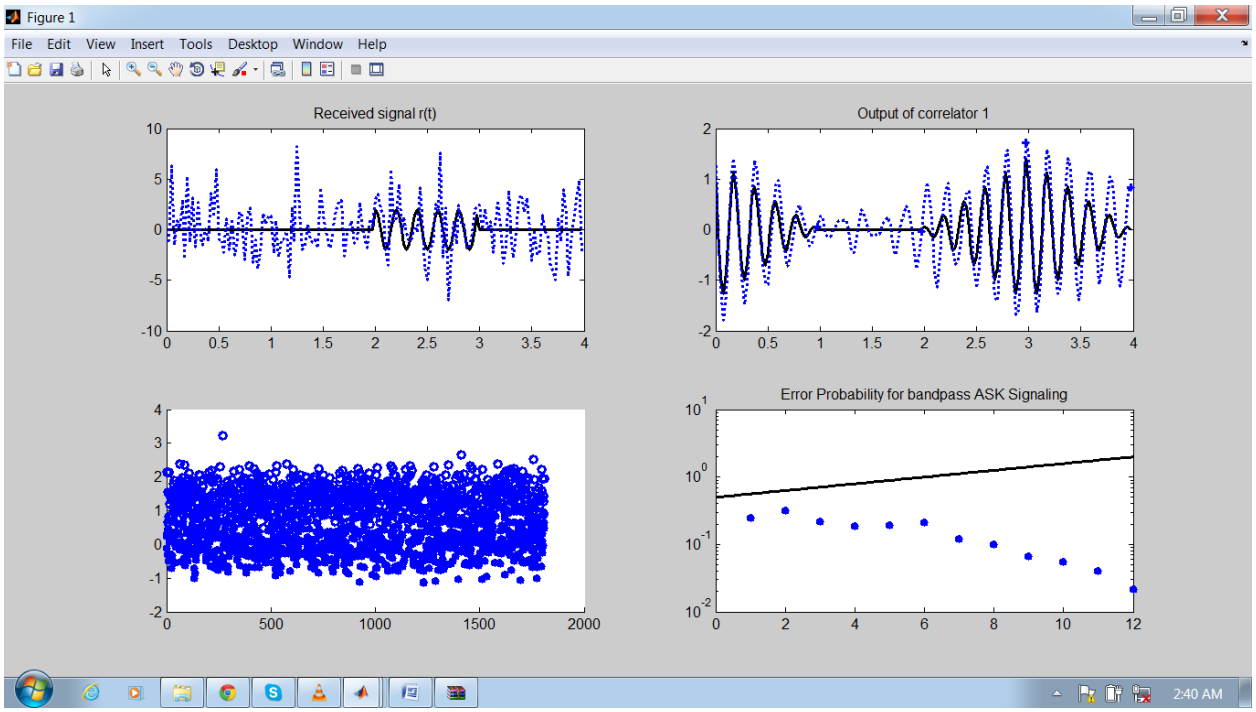
SOFTWARE: MATLAB 10.0 version.

CODING:

```
%sim_ASK_bandpass_coherent.m
% simulates a bandpass BASK signaling in Fig.7.2.1(coherent)
%Copyleft: Won Y. Yang, wyyang53@hanmail.net, CAU for academic use only
clear, clf
b=1; M=2^b;     % # of bits per symbol and modulation order
Tb=1; Ts=b*Tb; % Bit/Symbol time
Ns=40;          % # of sample times in Ts
T=Ts/Ns; LB=4*Ns; LBN1=LB-Ns+1; % Sample time and Buffer size
Es=2; A=sqrt(Es); % Energy of signal waveform
% Bandpass ASK Waveform by Eq.(7.1.1)
wc=10*pi/Ts; wcT=wc*T; t=[0:Ns-1]*T; tt=[0:LB-1]*T;
ss=[0; 1]; su(1,:)=sqrt(2/Ts)*cos(wc*t); suT=su*T; % Eq.(7.1.2)
sw=[A*su(1,:); zeros(1,Ns)];
SNRdBs=[1:12]; MaxIter=10000; % Range of SNRbdB and # of iterations
for iter=1:length(SNRdBs) % Iteration for SNRdBs
   SNRbdB= SNRdBs(iter);  SNR=10^(SNRbdB/10);
   N0=2*(Es/b)/SNR;  sigma2=N0/2;  sgmsT=sqrt(sigma2/T);
   st=zeros(1,LB); y=zeros(1,LB); r=zeros(1,LB); yr=zeros(1,LB);
   nobe= 0; % Number of bit errors to be accumulated
   thrshld= sqrt(Es)/2; %+sigma)/2;
   for k=1:MaxIter
      i= ceil(rand*M); s=ss(i); % Signal index, Data bits to transmit
      for n=1:Ns  % Operation per symbol time
         st=[st(2:LB) sw(i,n)]; % Uncontaminated received signal
         yn= suT*st(LBN1:LB)'; % Correlator output with no noise
         y=[y(:,2:LB) yn]; % Correlator outputs with no noise
         wct=wcT*(n-1);
         bp_noise= randn*cos(wct)-randn*sin(wct);
         r=[r(2:LB) sw(i,n)+sgmsT*bp_noise]; % Received signal
         yrn= suT*r(LBN1:LB)'; % Correlator output with noise
         yr=[yr(:,2:LB) yrn]; % Correlator output - DTR input
      end
      %Detector(DTR)
      if iter==10
        subplot(223), hold on
        if s==0, plot(k,yrn(1),'o');  else plot(k,yrn(1),'*'); end
      end
      d= (yrn(1)<thrshld); % Detected data bits
      nobe = nobe+(s~=d); % Number of bit errors in a detected symbol
      if nobe>100; break; end
   end
   pobe(iter)= nobe/(k*b);
end
SNRdBt=0:0.1:12;   SNRt=10.^(SNRdBt/10);
poe_on_theory= (sqrt(SNRt/4));  %Eq.(7.1.14)
subplot(224)
semilogy(SNRdBt, poe_on_theory, 'k-', SNRdBs, pobe, 'b*')
title('Error Probability for bandpass ASK Signaling')
subplot(221), plot(tt,st,'k-', tt,r,'b:')
```

```
title('Received signal r(t)')
subplot(222), plot(tt,y(1,:),'k-', tt,yr(1,:),'b:')
hold on, plot([1:4]*Tb-T,yr([1:4]*Ns),'+')
title('Output of correlator 1')
```

RESULT: waveform has plotted.



EXPERIMENT NO.10

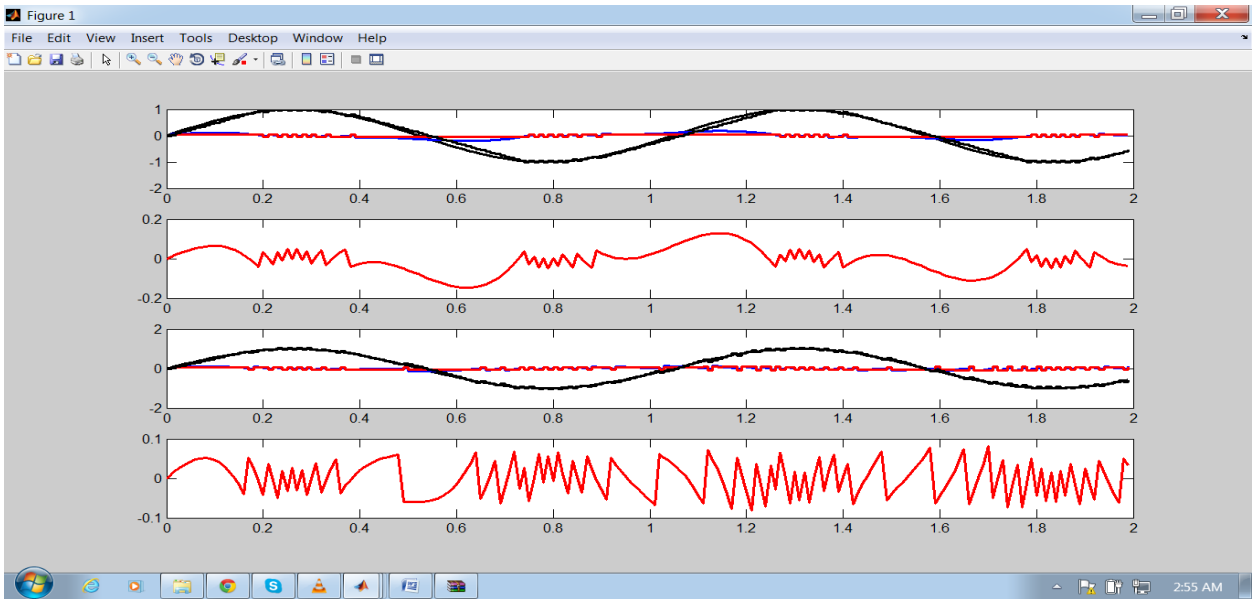
AIM: To plot Delta modulation waveform using MATLAB.

SOFTWARE: MATLAB 10.0 version.

CODING:

```
%do_dm.m
% simulates the DM (Delta Modulation) system in Fig.4.9
clear, clf
T=0.01; nmax=200; t=[0:nmax-1]*T;
b0=-0.1; bN=0.1; N=2;
for itr=1:2
   delta=(bN-b0)/N; % Width of the quantization interval
   xqhn=0; yqn=0; % Initialize the (memory) buffers.
   for n=1:nmax
      x(n)= sin(6*(n-1)*T); % +.2*(rand-0.5);
      xqh(n)=xqhn; % Predictor(delay) output
      d(n)=x(n)-xqh(n); % Comparator input
      dq(n)= sign(d(n))*delta/2; % Comparator
      %with adaptive adjustment of quantization interval width or not
      if itr==2
        delta=delta*50^(dq(n)*dqn); % Eq.(4.4.1)
      end
      dqn=dq(n); xqhn=xqh(n)+dqn;  % Store/Delay
      yq(n)=yqn+dq(n);             % Receiver(DM decoder)
      yqn=yq(n);                   % Store/Delay
   end
   subplot(4,1,2*itr-1)
   plot(t,x,'k-'), hold on
   stairs(t,d,'b-'), stairs(t,dq,'r-'), stairs(t,yq,'k-')
   subplot(4,1,2*itr)
   plot(t,x-yq,'r-')
   sum_of_squared_error=(x-yq)*(x-yq)'
end
```

RESULT: waveform has plotted,



EXPERIMENT NO.11

AIM: To plot frequency modulation waveform using MATLAB.

SOFTWARE: MATLAB 10.0 version.

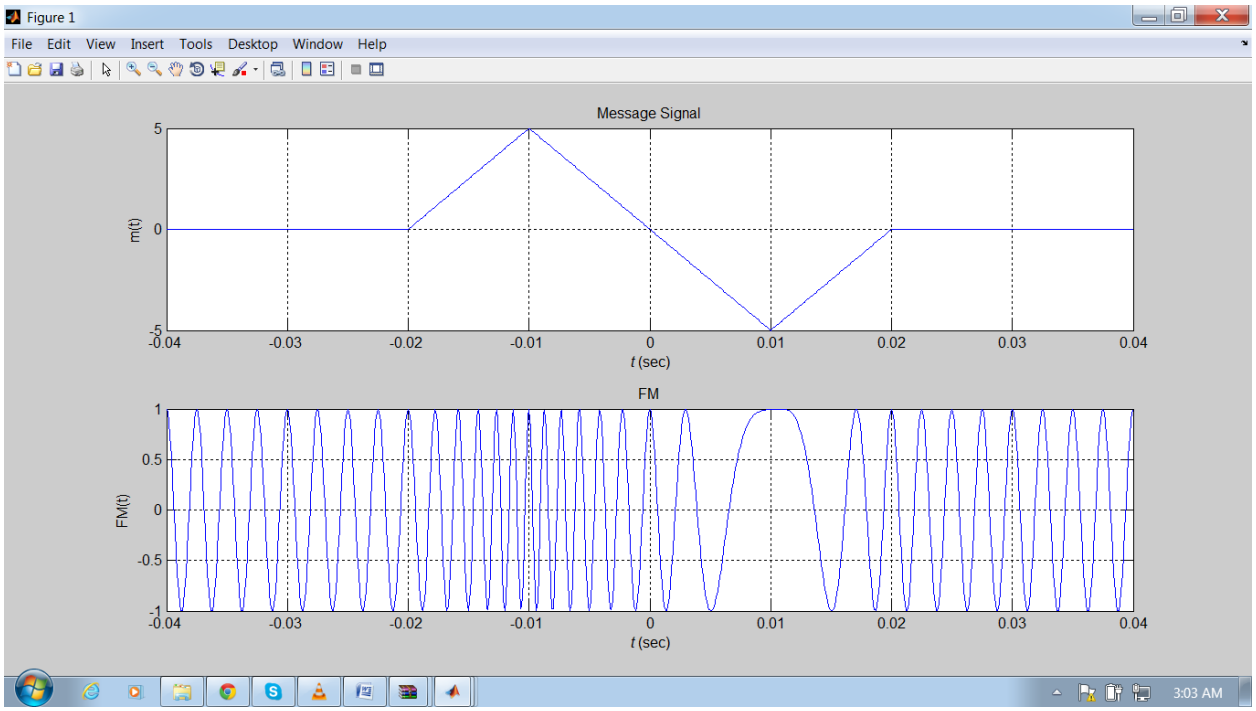
CODING:

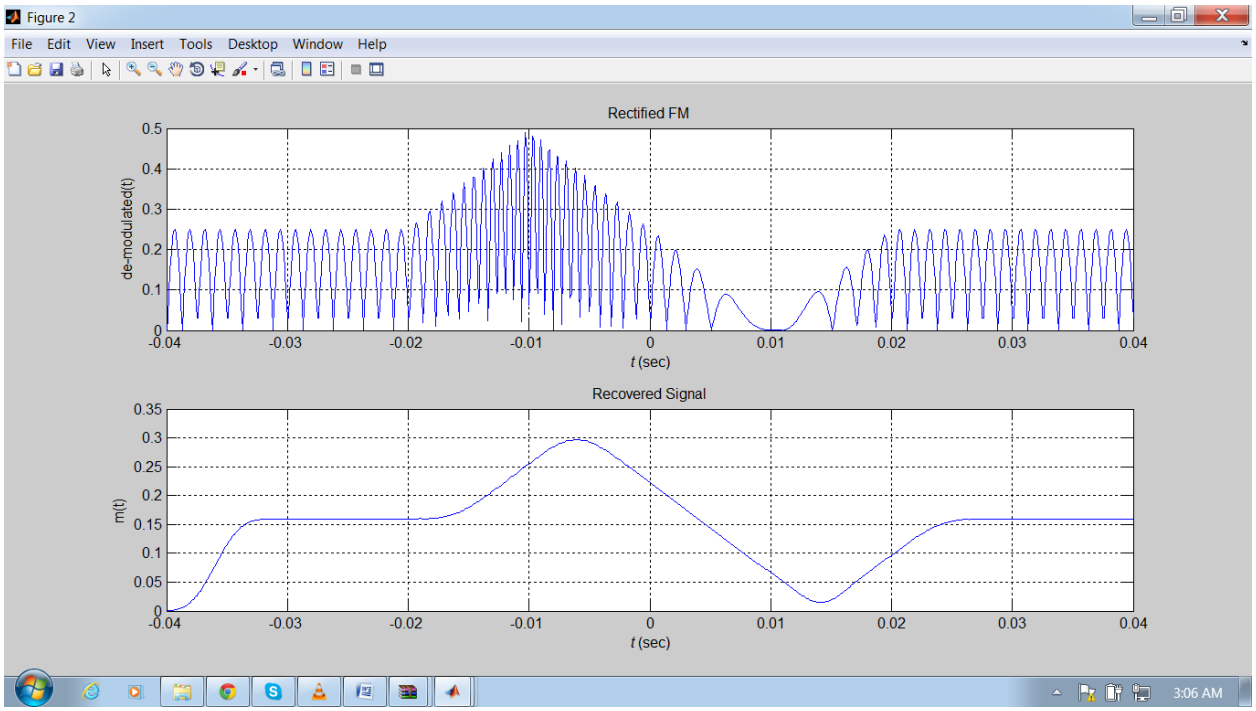
```
% ========================
% MSG FUNCTION TRIANGULAR
% ========================
 t1 = -0.02:1.e-4:0;
t2 = 0:1.e-4:0.02;
 Ta = 0.01;
m1 = 1 - abs((t1+Ta)/Ta);
m1 = [zeros([1 200]),m1,zeros([1 400])];
m2 = 1 - abs((t2-Ta)/Ta);
m2 = [zeros([1 400]),m2,zeros([1 200])];
m = m1 - m2;
m = 5*m;
t = -0.04:1.e-4:0.04;                                   %Time for plotting
fc = 400;                                               % Frquency of carrier
wave
% =============
% FM MODULATION
% =============
kf = 160*pi;
m_int = kf*1.e-4*cumsum(m);
fm = cos(2*fc*pi*t + m_int);
% ================
% FM DE-MODULATION
% ================
dem = diff(fm);
dem = [0,dem];
rect_dem = abs(dem);
% Filtering out High Frequencies
a = fir1(80,100*1.e-4);
b = 1;
rec = filter(a,b,rect_dem);
fl = length(t);
fl = 2^ceil(log2(fl));
f = (-fl/2:fl/2-1)/(fl*1.e-4);
mF = fftshift(fft(m,fl));                               % Frequency Responce
of Message Signal
fmF =  fftshift(fft(fm,fl));                            % Frequency Responce
of FM
rect_demF = fftshift(fft(rect_dem,fl));                 % Frequency Responce
of Rectified FM
recF = fftshift(fft(rec,fl));                           % Frequency Responce
of Recovered Message Signal
% ======================
% PLOTING IN TIME DOMAIN
% ======================
figure(1);
subplot(2,1,1);
plot(t,m);
title('Message Signal');
```

```
xlabel('{\it t} (sec)');
ylabel('m(t)');
grid;
subplot(2,1,2);
plot(t,fm);
title('FM');
xlabel('{\it t} (sec)');
ylabel('FM(t)');
grid;
figure(2);
subplot(2,1,1);
plot(t,rect_dem);
title('Rectified FM');
xlabel('{\it t} (sec)');
ylabel('de-modulated(t)')
grid;
subplot(2,1,2);
plot(t,rec);
title('Recovered Signal');
xlabel('{\it t} (sec)');
ylabel('m(t)');
grid;
% ================================
% Ploting Freq Responce of Signals
% ================================
figure(3);
subplot(2,2,1);
plot(f,abs(mF));
title('Freq Responce of Message Signal');
xlabel('f(Hz)');
ylabel('M(f)');
grid;
axis([-600 600 0 200]);


subplot(2,2,2);
plot(f,abs(fmF));
title('Freq Responce of FM');
grid;
xlabel('f(Hz)');
ylabel('C(f)');
axis([-600 600 0 200]);


subplot(2,2,3);
plot(f,abs(rect_demF));
title('Freq Responce of Rectified FM');
xlabel('f(Hz)');
ylabel('Rectified FM(f)');
grid;
axis([-600 600 0 150]);

subplot(2,2,4);
plot(f,abs(recF));
title('Freq Responce of Recoverd Signal');
xlabel('f(Hz)');
```

```
ylabel('M(f)');
grid;
axis([-600 600 0 150]);
```

RESULT: waveform has plotted.





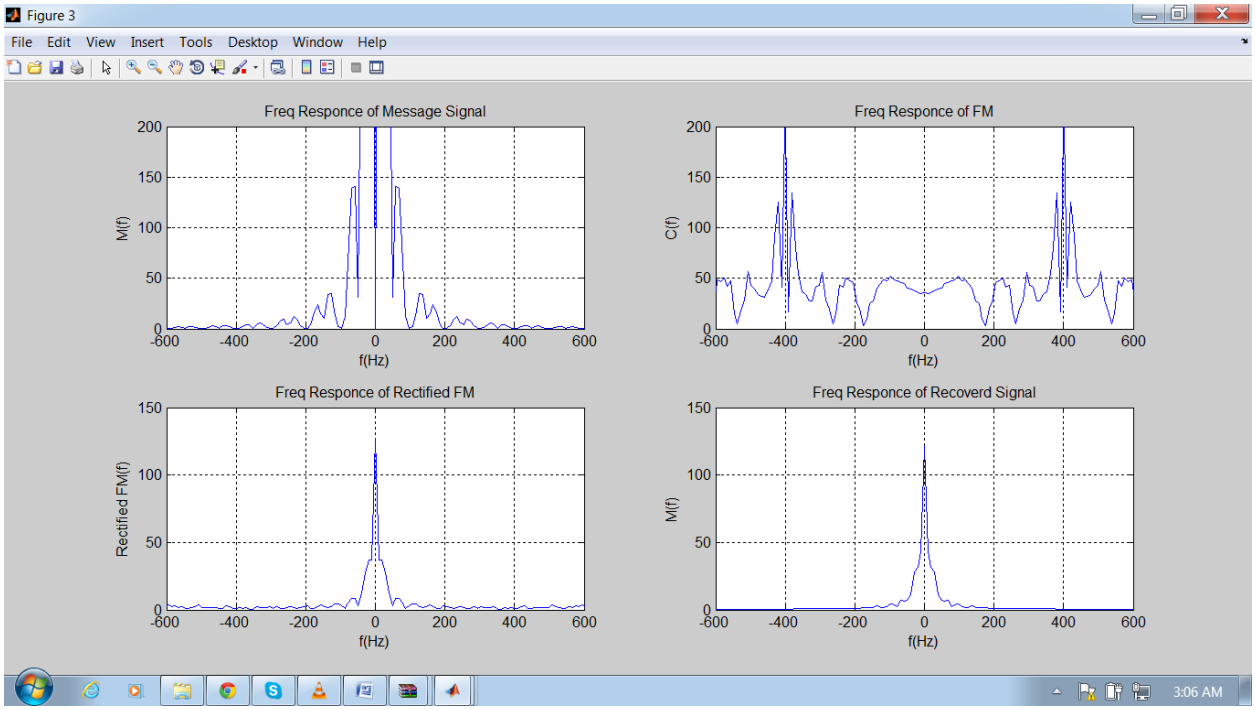


Figure 3
File Edit View Insert Tools Desktop Window Help
Freq Responce of Message Signal
M(f)
f(Hz)
Freq Responce of FM
C(f)
f(Hz)
Freq Responce of Rectified FM
Rectified FM(f)
f(Hz)
Freq Responce of Recoverd Signal
M(f)
f(Hz)
3:06 AM