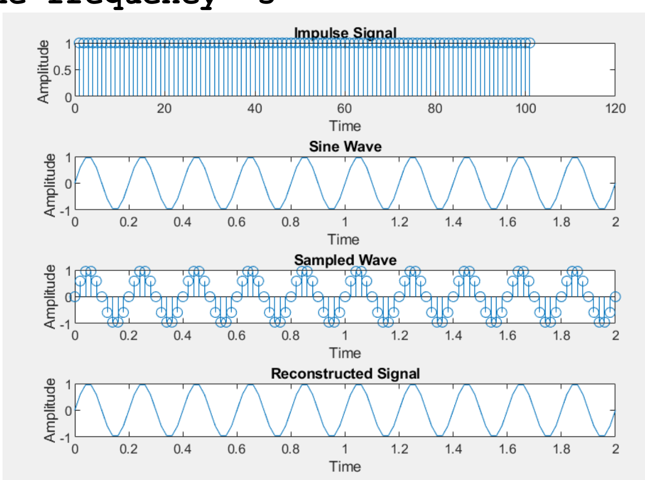
EC8561- COMMUNICATION SYSTEMS LAB

1. SAMPLING AND RECONSTRUCTION

```
clc;
close all;
clear all;
 f = input('Enter the frequency  = ');
t = 0:0.02:2; % for a total of 16 samples
x1 = square(20*pi*f*t); %generation of an impulse signal
x2 = sin(2*pi*f*t); %generation of sine wave
y = x1.*x2; %modulation step
subplot(4,1,1); %for impulse signal plot
stem(x1);
title('Impulse Signal');
xlabel('Time');
ylabel('Amplitude ');
subplot(4,1,2) %for sine wave plot
plot(t,x2);
title('Sine Wave');
xlabel('Time ');
ylabel('Amplitude ');
subplot(4,1,3) %for PAM wave plot
stem(t,y);
title('Sampled Wave');
xlabel('Time');
ylabel('Amplitude');
subplot(4,1,4)
plot(t,y);
title('Reconstructed Signal');
xlabel('Time');
ylabel('Amplitude');
```

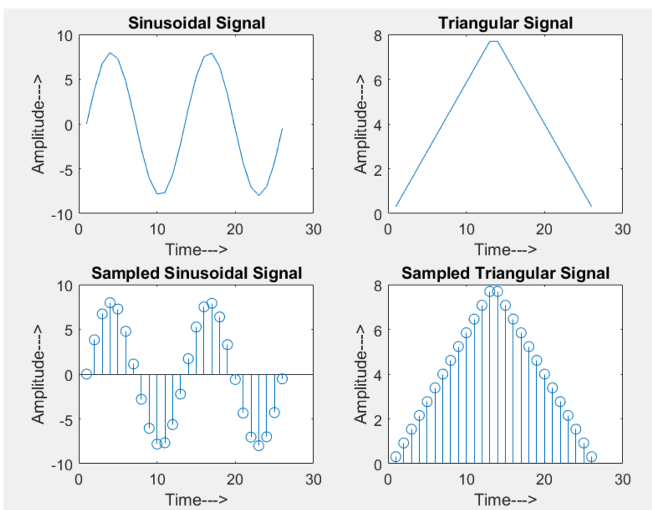
OUTPUT: Enter the frequency =5

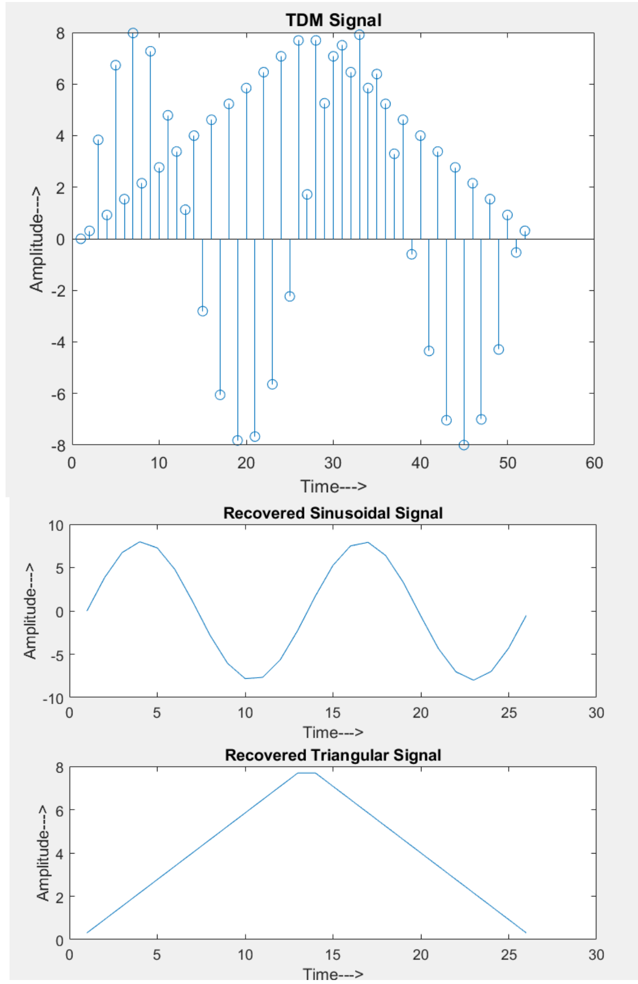


2. TIME DIVISION MULTIPLEXING AND DEMULTIPLEXING

```
clc;
close all;
clear all;
% Signal generation
x=0:.5:4*pi;                              % signal taken upto 4pi
sig1=8*sin(x);                            % generate 1st sinusoidal
signal
l=length(sig1);
sig2=8*triang(l);                        % Generate 2nd triangular
Sigal
% Display of Both Signal
subplot(2,2,1);
plot(sig1);
title('Sinusoidal Signal');
ylabel('Amplitude--->');
xlabel('Time--->');
subplot(2,2,2);
plot(sig2);
title('Triangular Signal');
ylabel('Amplitude--->');
xlabel('Time--->');
% Display of Both Sampled Signal
subplot(2,2,3);
stem(sig1);
title('Sampled Sinusoidal Signal');
ylabel('Amplitude--->');
xlabel('Time--->');
subplot(2,2,4);
stem(sig2);
title('Sampled Triangular Signal');
ylabel('Amplitude--->');
xlabel('Time--->');
l1=length(sig1);
l2=length(sig2);
 for i=1:l1
  sig(1,i)=sig1(i);                 % Making Both row vector to a
matrix
  sig(2,i)=sig2(i);
 end
% TDM of both quantize signal
tdmsig=reshape(sig,1,2*l1);
% Display of TDM Signal
figure
```

```
stem(tdmsig);
title('TDM Signal');
ylabel('Amplitude--->');
xlabel('Time--->');
% Demultiplexing of TDM Signal
 demux=reshape(tdmsig,2,l1);
 for i=1:l1
  sig3(i)=demux(1,i);        % Converting The matrix into row
vectors
  sig4(i)=demux(2,i);
 end

 % display of demultiplexed signal
 figure
 subplot(2,1,1)
 plot(sig3);
 title('Recovered Sinusoidal Signal');
 ylabel('Amplitude--->');
 xlabel('Time--->');
 subplot(2,1,2)
 plot(sig4);
 title('Recovered Triangular Signal');
 ylabel('Amplitude--->');
 xlabel('Time--->');
```

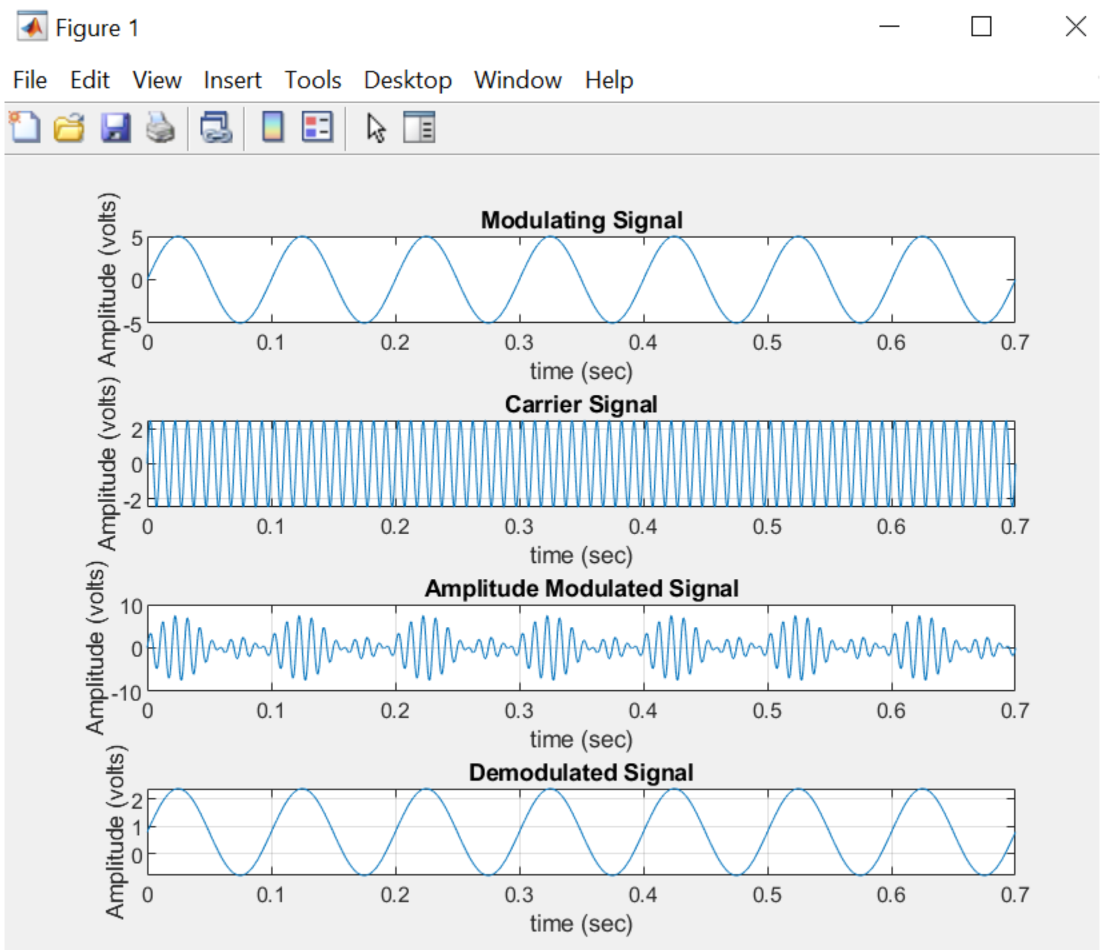



TDM Signal
Amplitude--->
Time--->
Recovered Sinusoidal Signal
Amplitude--->
Time--->
Recovered Triangular Signal
Amplitude--->
Time--->

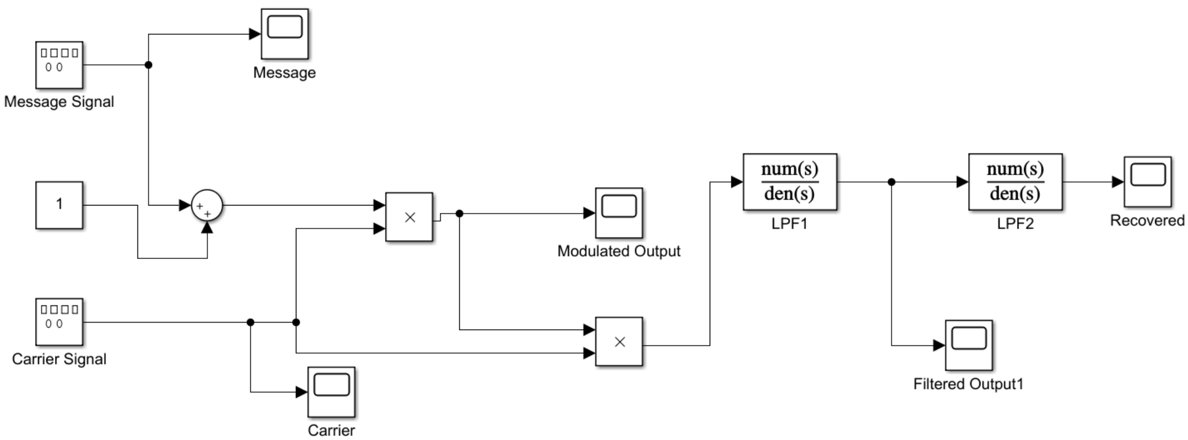
3. AM GENERATION AND DETECTION

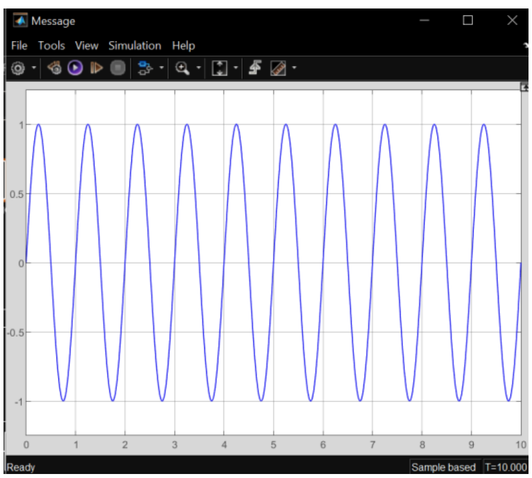
```
clc;
close all;
clear all;
%t=0:.001:.5;
m=1;
Am = 5; %Amp. of modulating signal
fm = 10; %frequency of modulating signal
Tm = 1/fm;
t = 0:Tm/999:6*Tm;
ym = Am*sin(2*pi*fm*t);
figure(1)
subplot(4,1,1)
plot(t,ym)
title('Modulating Signal')
xlabel('time (sec)');
ylabel('Amplitude (volts)');

%Carrier signal
Ac = Am/m;
fc = fm*10;
Tc = 1/fc;
yc = Ac*sin(2*pi*fc*t);
subplot(4,1,2)
plot(t,yc)
grid on;
title('Carrier Signal')
xlabel('time (sec)');
ylabel('Amplitude (volts)');

%AM Modulation
y = Ac + (1+m*sin(2*pi*fm*t)).*sin(2*pi*fc*t);
subplot(4,1,3)
plot(t,y)
title('Amplitude Modulated Signal')
xlabel('time (sec)');
ylabel('Amplitude (volts)');
grid on;
```

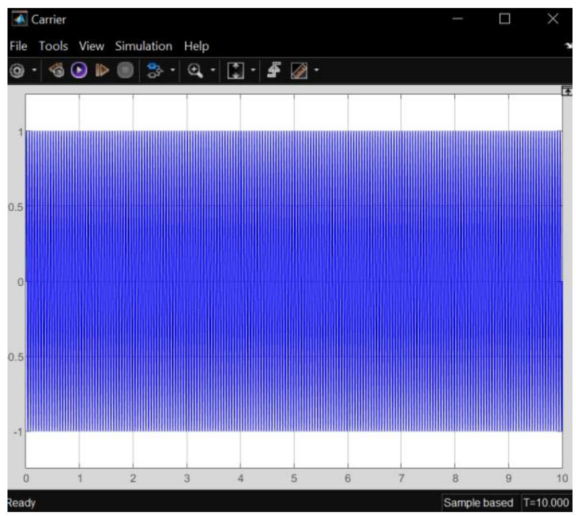
```
% AM Demodulation
s = (1/pi)*(Ac+ym);
subplot(4,1,4)
plot(t,s)
title('Demodulated Signal')
xlabel('time (sec)');
ylabel('Amplitude (volts)');
grid on;
```

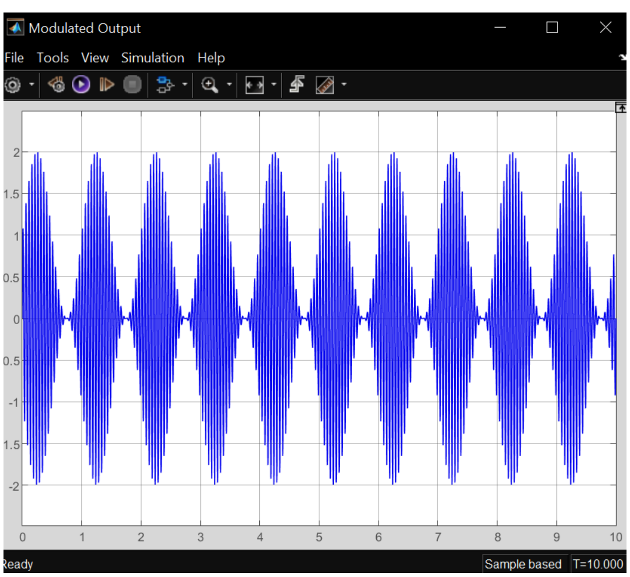


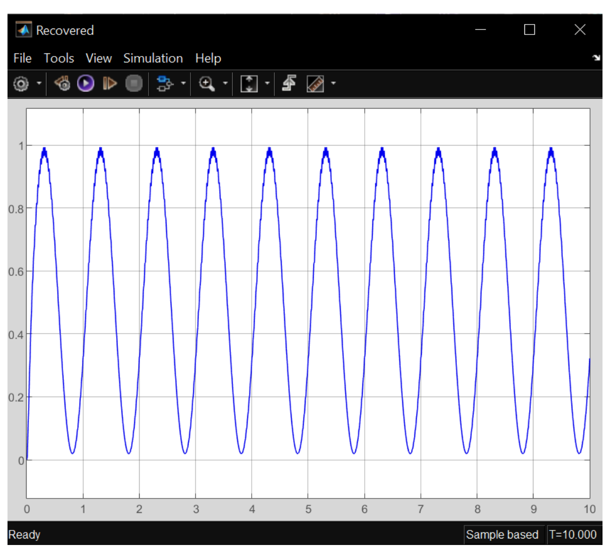
AMPLITUDE MODULATION USING SIMULINK











4.FM GENERATION AND DETECTION

```
clc;
clear all;
close all;
kf=0.75;
fc=1/10;
fm=1/100;
n=[1:fm/4:256];
x=sin(2*pi*fm*n);
y=cos(2*pi*fc*n);
z=cos(2*pi*fc*n+2*pi*kf*cos(2*pi*fm*n));
subplot(4,1,1)
plot(n,x);
title('Modulating Signal');
subplot(4,1,2)
plot(n,y);
title('Carrier Signal');
subplot(4,1,3)
plot(n,z);
title('Modulated Signal');
ts=1/(10*fc);
E=diff(z)/ts;
vout(1)=E(1);
t=(0:length(E)-1)*ts;
R=[10^5,10^4,10^3,10^2]*3.2;
c=10^-6;
for i=1:4
    for j=2:length(E)
        if E(j)>vout(j-1)
            vout(j)=E(j);
        else
            vout(j)=vout(j-1).*exp(-ts/(R(i)*c));
        end
    end
    subplot(4,1,4)
    plot(t,vout,t,E);
    title('AM and its Envelope');
end
```

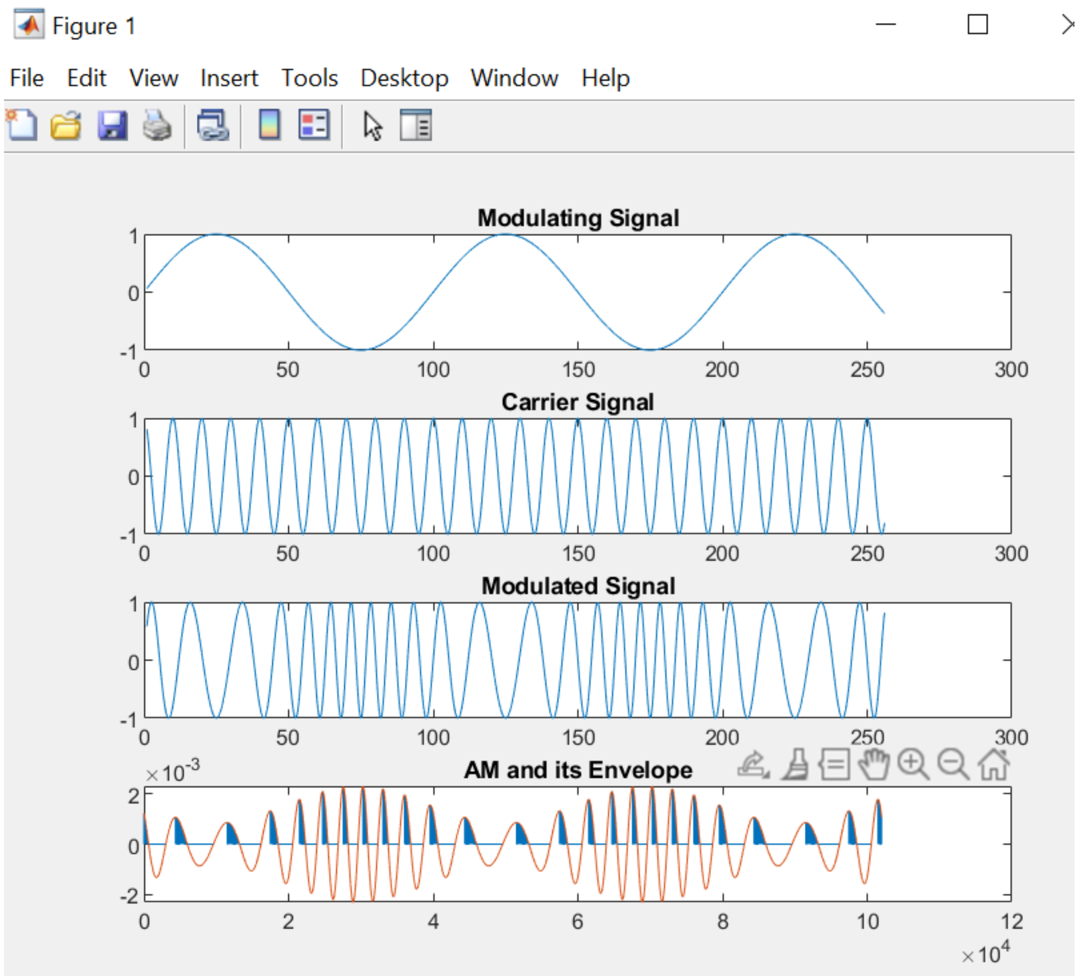
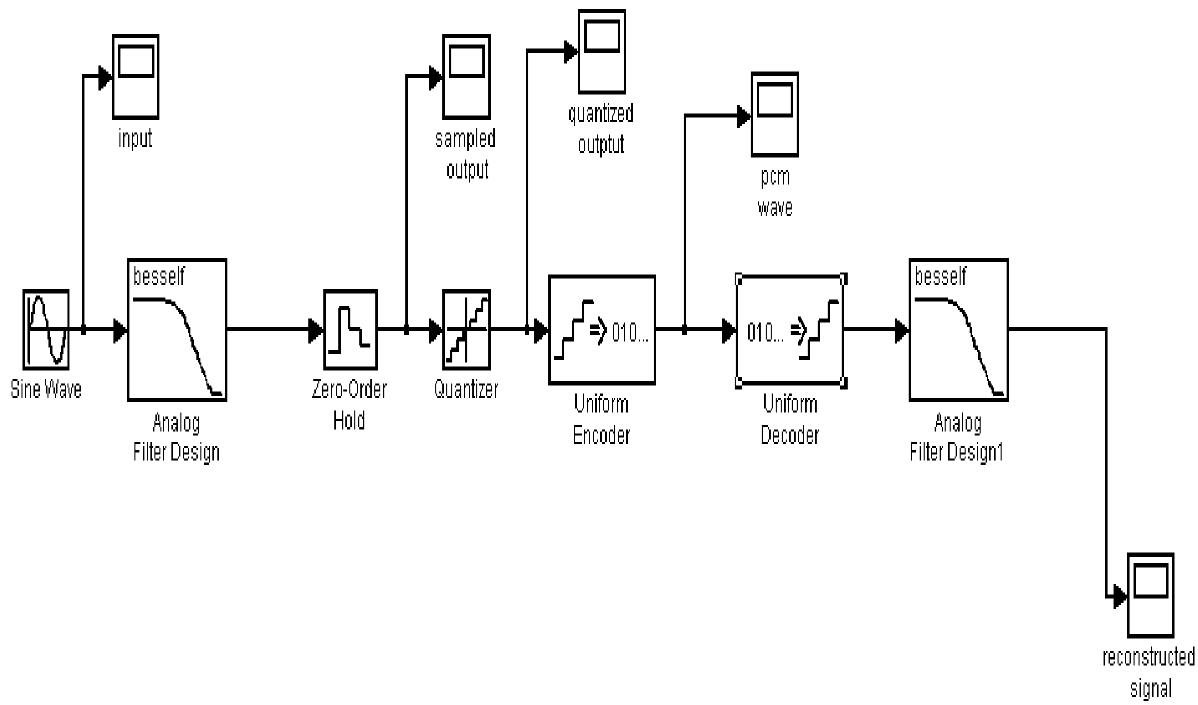
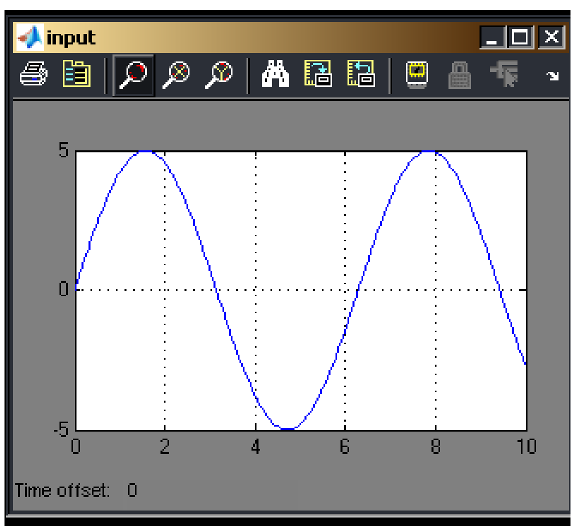
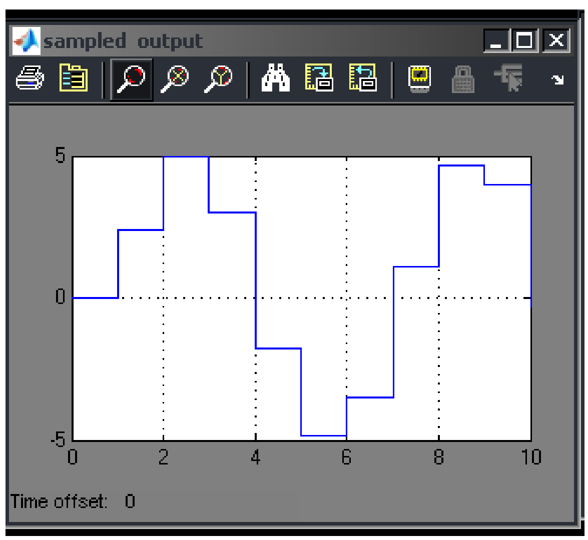


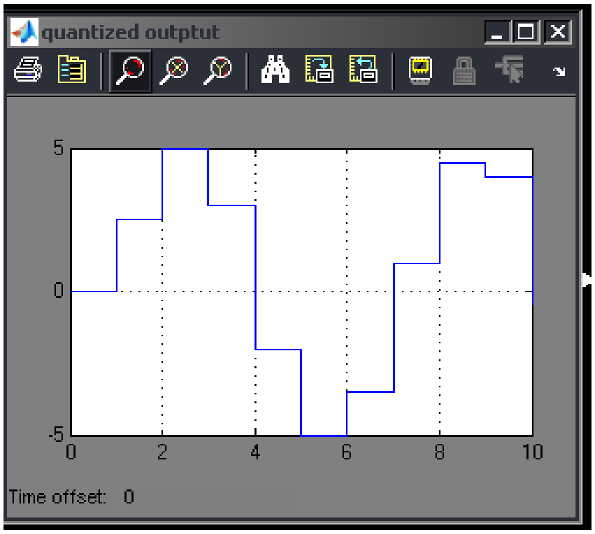
Figure 1
File Edit View Insert Tools Desktop Window Help
Modulating Signal
Carrier Signal
Modulated Signal
AM and its Envelope
×10^-3
×10^4

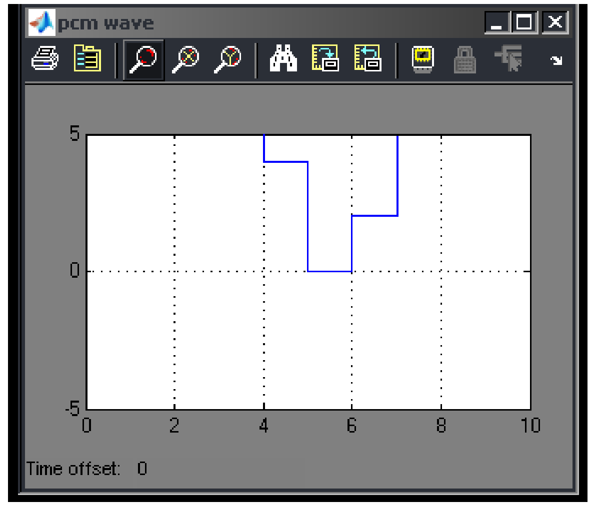
5. PULSE CODE MODULATION

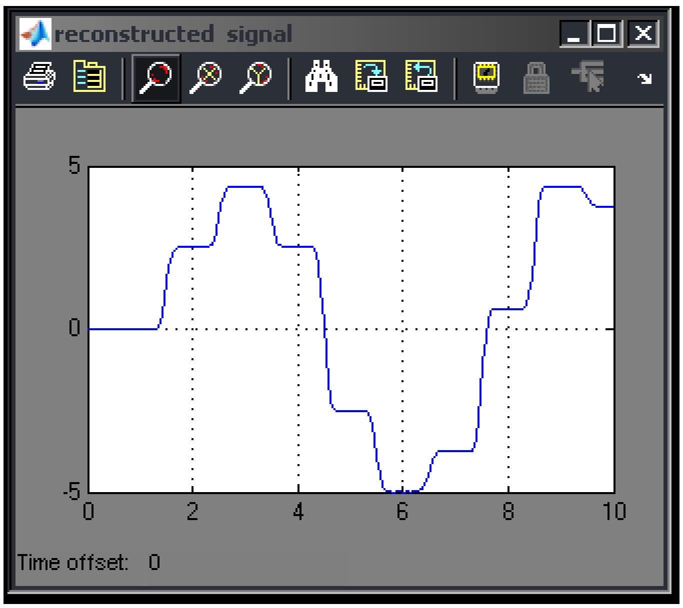




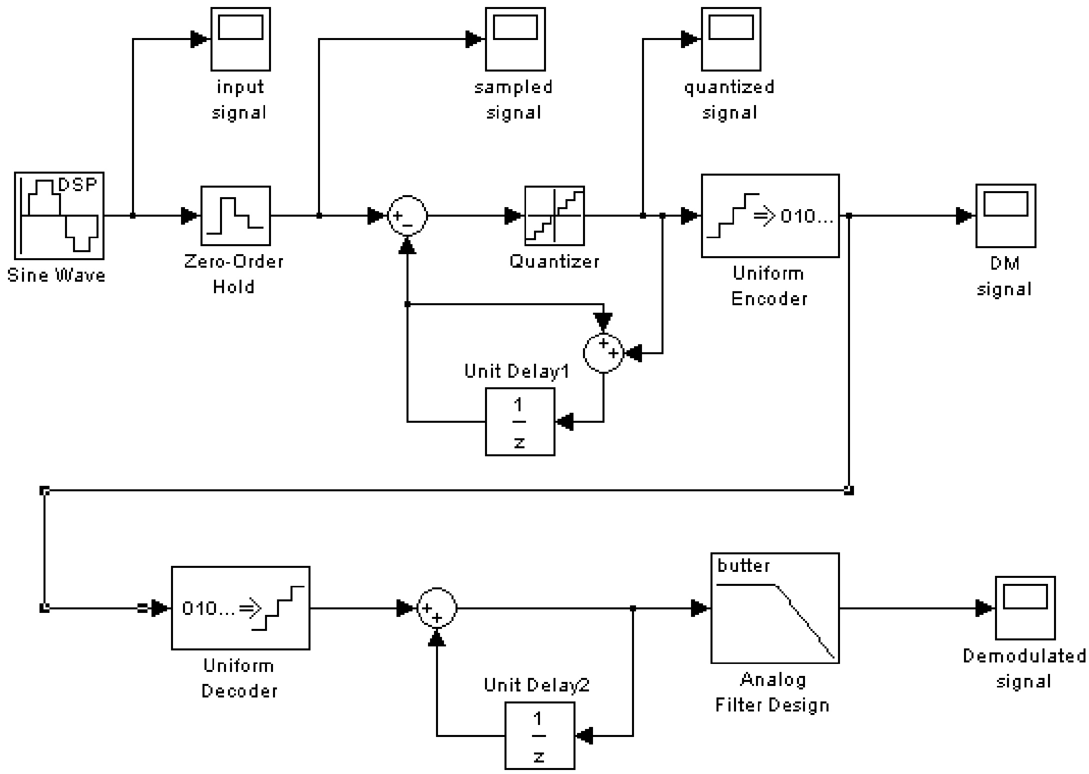


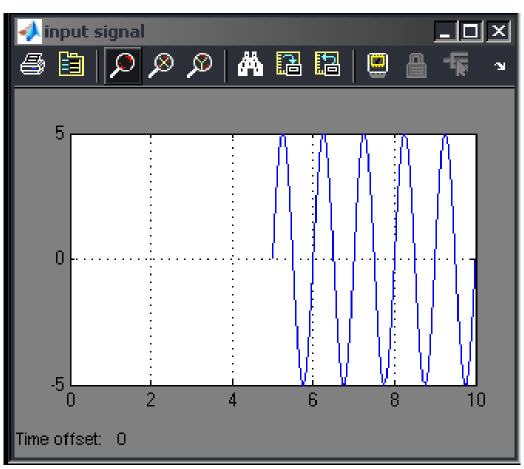

quantized outptut
5
0
-5
0
2
4
6
8
10
Time offset: 0

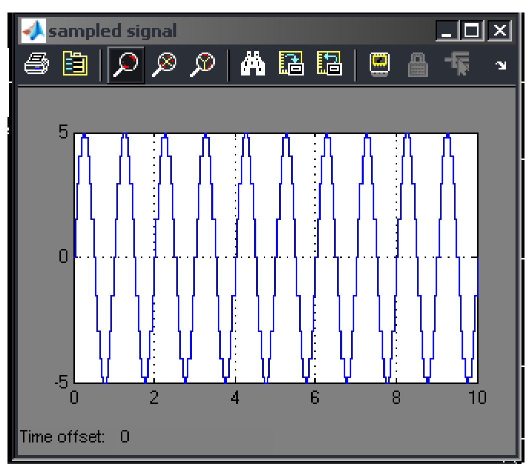

pcm wave
5
0
-5
0
2
4
6
8
10
Time offset: 0

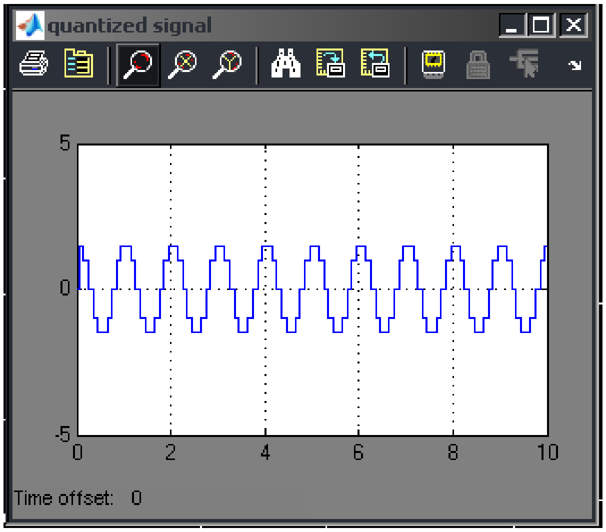

reconstructed signal
5
0
-5
0
2
4
6
8
10
Time offset: 0

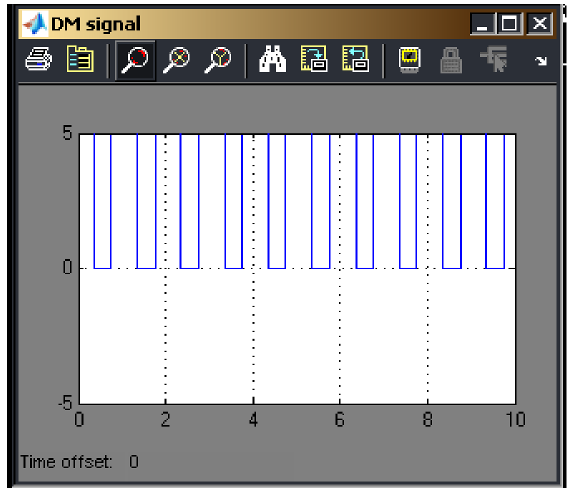
6. DELTA MODULATION

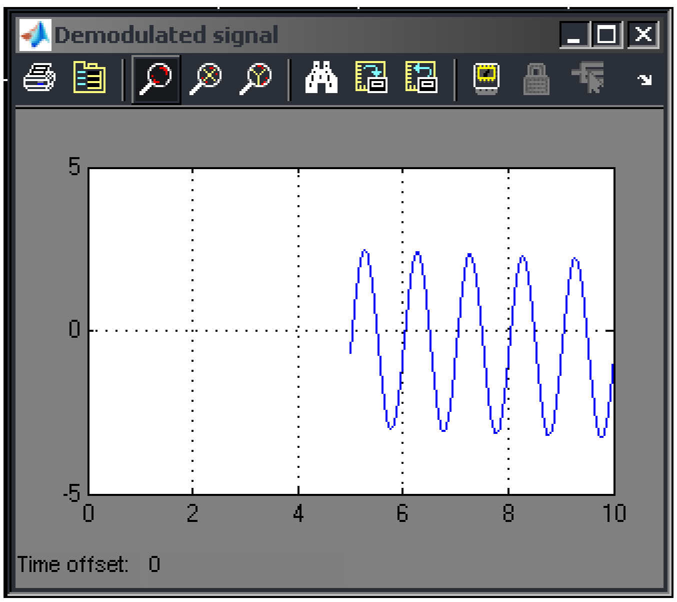







quantized signal
5
0
-5
0
2
4
6
8
10
Time offset: 0

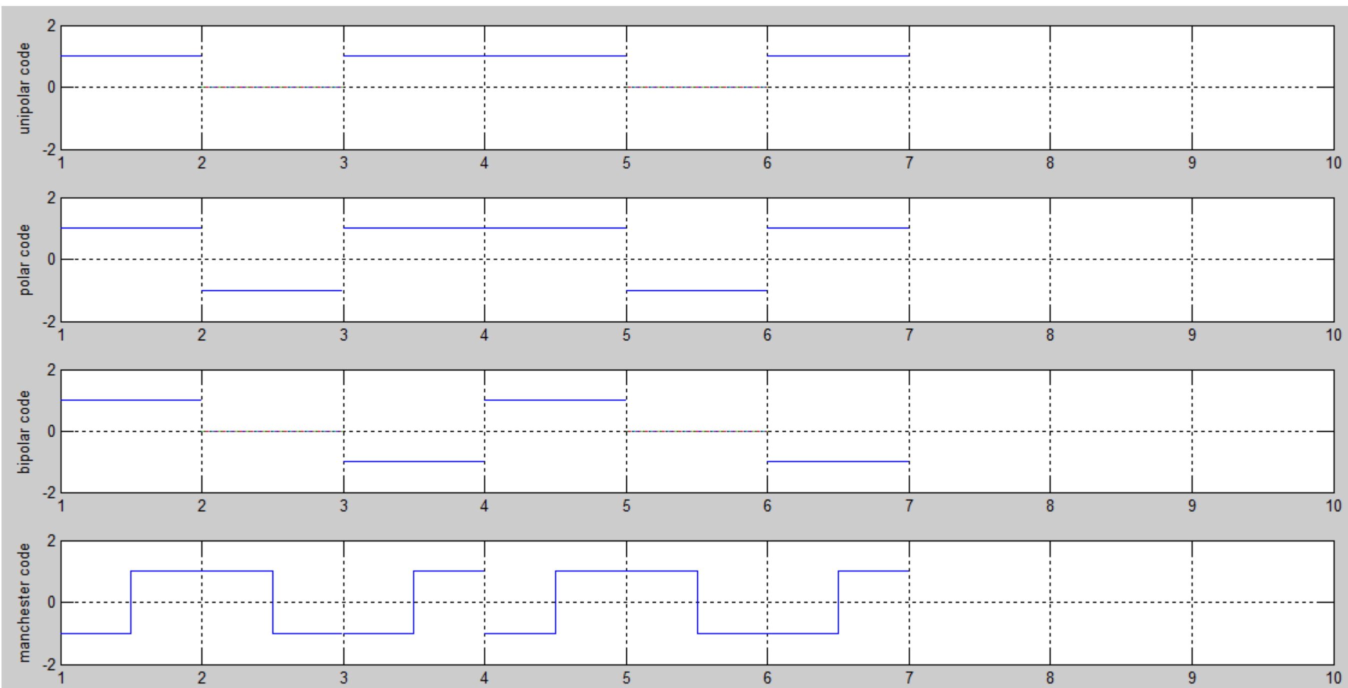

DM signal
5
0
-5
0
2
4
6
8
10
Time offset: 0


Demodulated signal
5
0
-5
0
2
4
6
8
10
Time offset: 0

7. LINE CODING

```
clc;
close all;
clear all;
N=6;
x=[1 0 1 1 0 1];
nx=size(x,2);
sign=1;
i=1;
while i<nx+1
    t = i:0.001:i+1-0.001;
    if x(i)==1
        unipolar_code=square(t*2*pi,100);
        polar_code=square(t*2*pi,100);
        bipolar_code=sign*square(t*2*pi,100);
        sign=sign*-1;
        manchester_code=-square(t*2*pi,50);
    else
        unipolar_code=0;
        polar_code=-square(t*2*pi,100);
        bipolar_code=0;
        manchester_code=square(t*2*pi,50);
    end
    subplot(4,1,1);
    plot(t,unipolar_code);
    ylabel('unipolar code');
    hold on;
    grid on;
    axis([1 10 -2 2]);

    subplot(4,1,2);
    plot(t,polar_code);
    ylabel('polar code');
    hold on;
    grid on;
    axis([1 10 -2 2]);

    subplot(4,1,3);
    plot(t,bipolar_code);
    ylabel('bipolar code');
    hold on;
```

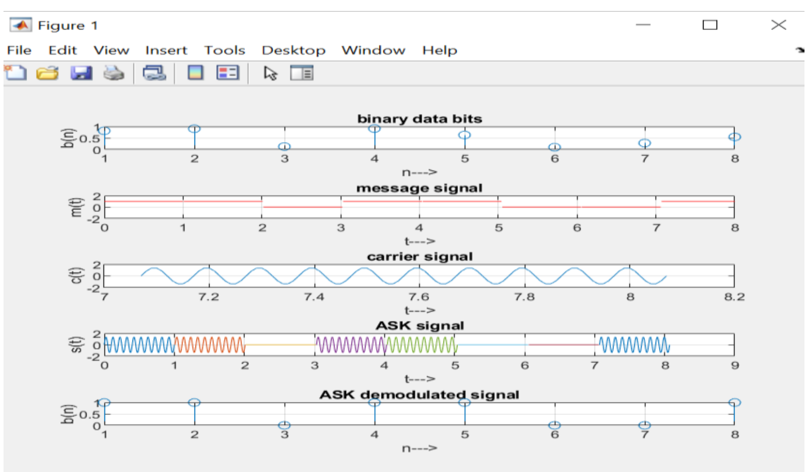
```
    grid on;
    axis([1 10 -2 2]);

    subplot(4,1,4);
    plot(t,manchester_code);
    ylabel('manchester code');
    hold on;
    grid on;
    axis([1 10 -2 2]);

    i=i+1;
end
```



8. AMPLTUDE SHIFT KEYING (ASK)

```
clc;
clear all;
close all;
%GENERATE CARRIER SIGNAL
Tb=1; fc=10;
t=0:Tb/100:1;
c=sqrt(2/Tb)*sin(2*pi*fc*t);
%generate message signal
N=8;
m=rand(1,N);
%Plot the input binary data
 subplot(5,1,1);stem(m);
 title('binary data bits');xlabel('n---
>');ylabel('b(n)');grid on
t1=0;t2=Tb;
for i=1:N
 t=[t1:.01:t2];
 if m(i)>0.5
 m(i)=1;
 m_s=ones(1,length(t));
 else
 m(i)=0;
 m_s=zeros(1,length(t));
 end
 message(i,:)=m_s;
 %plot the message (unipolar)
  subplot(5,1,2);axis([0 N -2
2]);plot(t,message(i,:),'r');
     title('message signal');xlabel('t---
>');ylabel('m(t)');grid on
    hold on
  %Plot the carrier signal
    subplot(5,1,3);plot(t,c);
    title('carrier signal');xlabel('t---
>');ylabel('c(t)');grid on

 %product of carrier and message
 ask_sig(i,:)=c.*m_s;
 t1=t1+(Tb+.01);
 t2=t2+(Tb+.01);
```

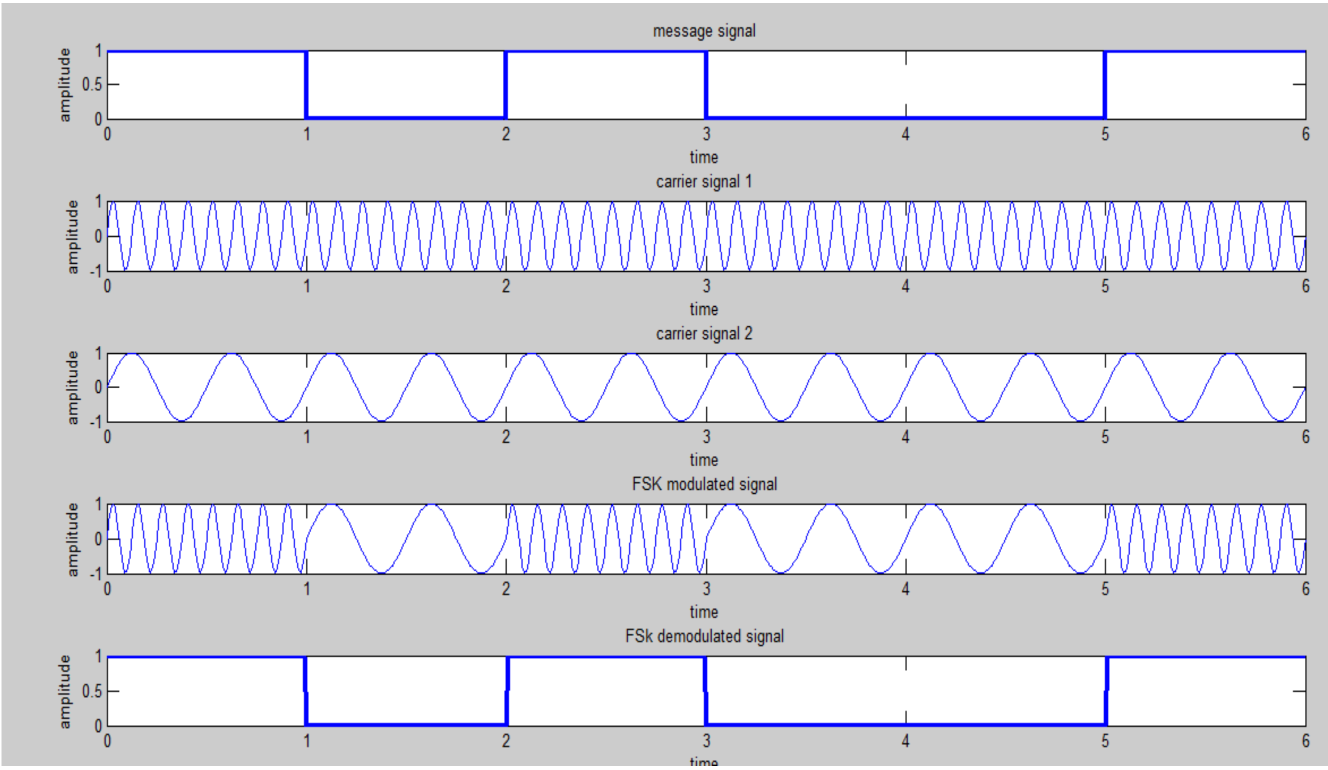
```
 %Plot the ASK signal
 subplot(5,1,4);plot(t,ask_sig(i,:));
 title('ASK signal');xlabel('t---
>');ylabel('s(t)');grid on
 hold on
 end
hold off
t1=0;t2=Tb
 for i=1:N
 t=[t1:Tb/100:t2];
 %correlator
 x=sum(c.*ask_sig(i,:));
 %decision device
 if x>0
 demod(i)=1;
 else
 demod(i)=0;
 end
 t1=t1+(Tb+.01);
 t2=t2+(Tb+.01);
 end
 %plot demodulated binary data bits
 subplot(5,1,5);stem(demod);
 title('ASK demodulated signal'); xlabel('n---
>');ylabel('b(n)');grid on
```



9. BFSK Modulation and Demodulation

```
clc;
clear all;
close all;
%carrier freq
f1=8;
f2=2;
a=1;
%6 bit are used
n=[1 0 1 1 0 0];
l=length(n);
if n(l)==1
    n(l+1)=1
else
    n(l+1)=0
end
L1=length(n);
t2=0:L1-1;
tn=0:l-1;
%plot message
subplot(5,1,1);
stairs(tn,n,'linewidth',3);
title('message signal');
xlabel('time');
ylabel('amplitude');
%plot carrier sig
t=0:0.01:6;
y1=a*sin(2*pi*f1*t);
y2=a*sin(2*pi*f2*t);
subplot(5,1,2);
plot(t,y1);
title('carrier signal 1');
xlabel('time');
ylabel('amplitude');
subplot(5,1,3);
```

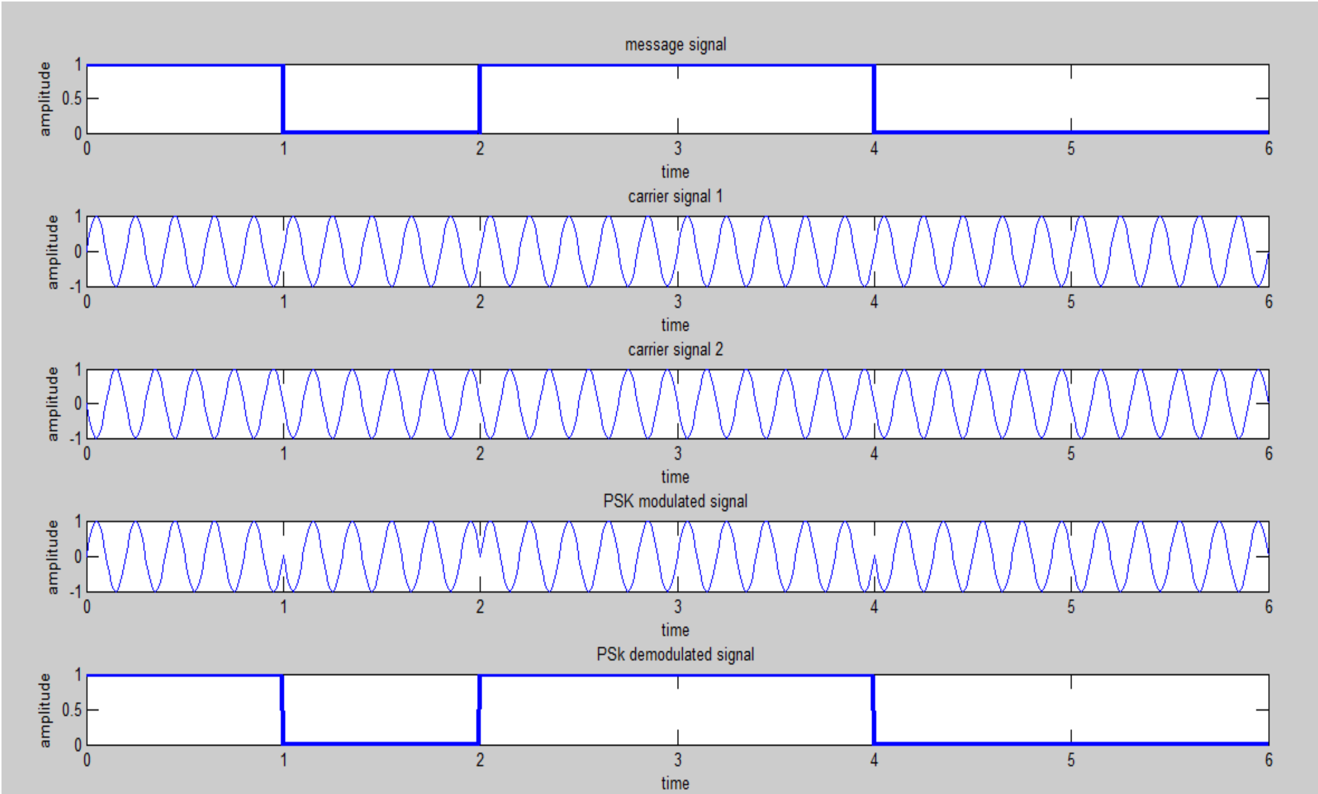
```
plot(t,y2);
title('carrier signal 2');
xlabel('time');
ylabel('amplitude');
%modulation process
for  i=1:6
    for j=(i-1)*100:i*100
        if (n(i)==1)
            s(j+1)=y1(j+1);
        else
            s(j+1)=y2(j+1);
        end
    end
end
%plot fsk signal
subplot(5,1,4);
plot(t,s);
title('FSK modulated signal');
xlabel('time');
ylabel('amplitude');
%demodulation process
for i=1:6
    for j=(i-1)*100:i*100
        if(s(j+1)==y1(j+1))
            s(j+1)=1;
        else
            s(j+1)=0;
        end
    end
end
%FSK demodulation
subplot(5,1,5);
plot(t,s,'linewidth',3);
title('FSK demodulated signal');
xlabel('time');
ylabel('amplitude');
```


message signal
amplitude
time
carrier signal 1
carrier signal 2
FSK modulated signal
FSk demodulated signal

10. BPSK Modulation and Demodulation

```
clc;
clear all;
close all;
%carrier freq
f=5;
a=1;
%6 bit are used
n=[1 0 1 1 0 0];
l=length(n);
if n(l)==1
    n(l+1)=1
else
    n(l+1)=0
end
L1=length(n);
t2=0:L1-1;
%plot message
subplot(5,1,1);
stairs(t2,n,'linewidth',3);
title('message signal');
xlabel('time');
ylabel('amplitude');
%plot carrier sig
t=0:0.01:6;
y1=a*sin(2*pi*f*t);
y2=-a*sin(2*pi*f*t);
subplot(5,1,2);
plot(t,y1);
title('carrier signal 1');
xlabel('time');
ylabel('amplitude');
subplot(5,1,3);
plot(t,y2);
title('carrier signal 2');
```

```
xlabel('time');
ylabel('amplitude');
%modulation process
for  i=1:6
for j=(i-1)*100:i*100
if (n(i)==1)
            s(j+1)=y1(j+1);
else
            s(j+1)=y2(j+1);
end
end
end
%plot psk signal
subplot(5,1,4);
plot(t,s);
title('PSK modulated signal');
xlabel('time');
ylabel('amplitude');
%demodulation process
for i=1:6
for j=(i-1)*100:i*100
if(s(j+1)==y1(j+1))
            x(j+1)=1;
else
            x(j+1)=0;
end
end
end
%plot demodulated sig
subplot(5,1,5);
plot(t,x,'linewidth',3);
title('PSk demodulated signal');
xlabel('time');
ylabel('amplitude');
```

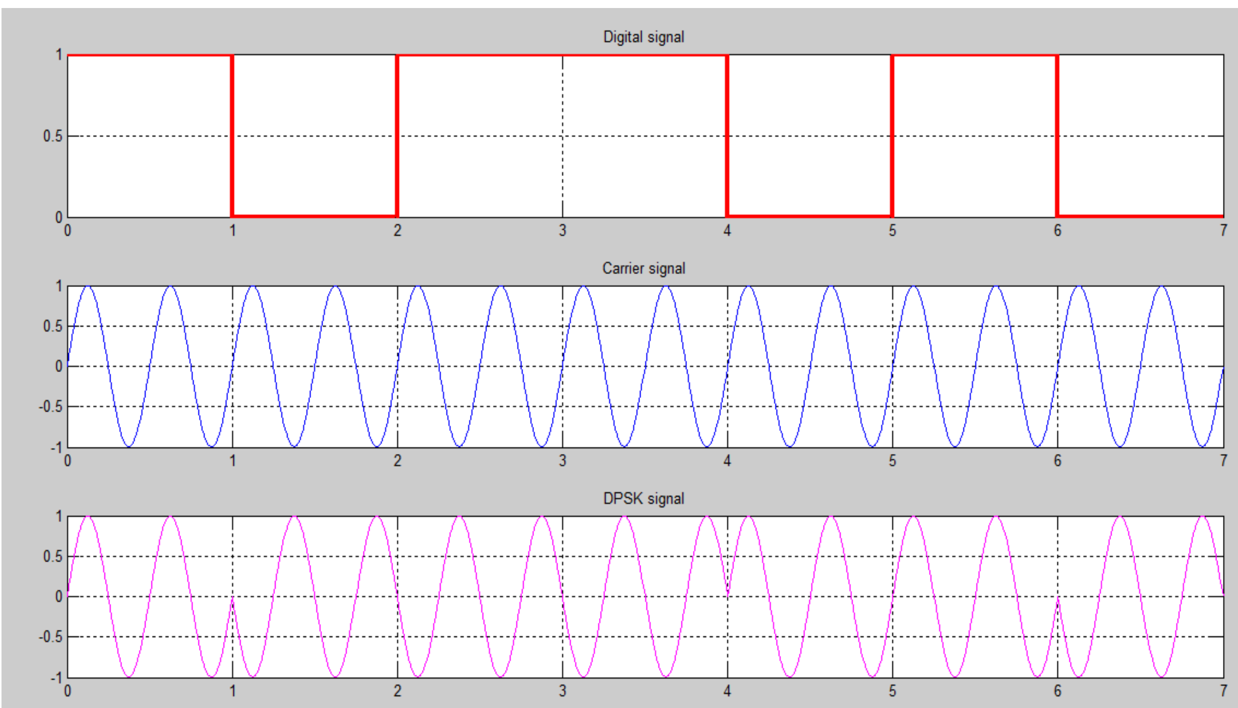


message signal
amplitude
time
carrier signal 1
carrier signal 2
PSK modulated signal
PSk demodulated signal

11. DPSK MODULATION

```
clc;
clear all;
close all;
t=0:.01:1;
fc = 2;
M = [1 0 1 1 0 1 0]
codedM=1;
for i=1:1:length(M)
    bit = not(xor(codedM(i),M(i)));
    codedM = [codedM bit];
end
codedM = codedM(2:length(codedM));

messageLength=length(M);
time=[];
digitalSignal=[];
dpskSignal=[];
carrierSignal=[];
for i=1:1:messageLength
    carrier = sin(2*pi*fc*t);
    carrierSignal = [carrierSignal carrier];
    if M(i) == 1
        bit = ones(1,length(t));
    else
        bit = zeros(1,length(t));
    end

    digitalSignal = [digitalSignal bit];

    if codedM(i) == 1
        DPSK = sin(2*pi*fc*t+0);
    else
        DPSK = sin(2*pi*fc*t+pi);
    end

    dpskSignal = [dpskSignal DPSK];
    time=[time t];
    t=t+1;
end
```

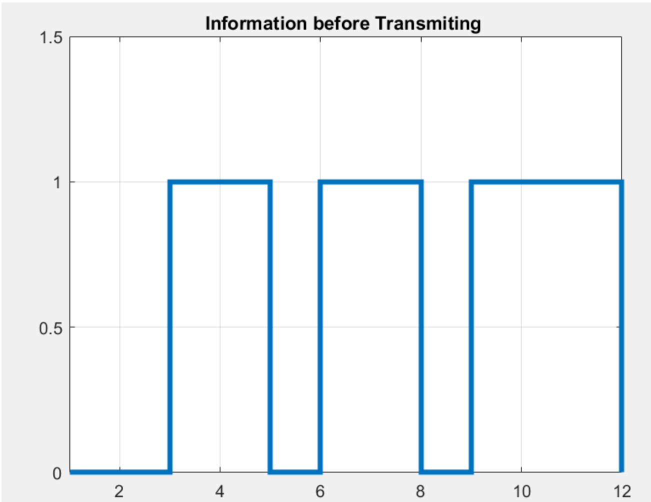
```
subplot(3,1,1);
plot(time,digitalSignal,'r','linewidth',3);
grid on;
title('Digital signal');

subplot(3,1,2);
plot(time,carrierSignal);
grid on;
title('Carrier signal');

subplot(3,1,3);
plot(time,dpskSignal,'m');
grid on;
title('DPSK signal');
```

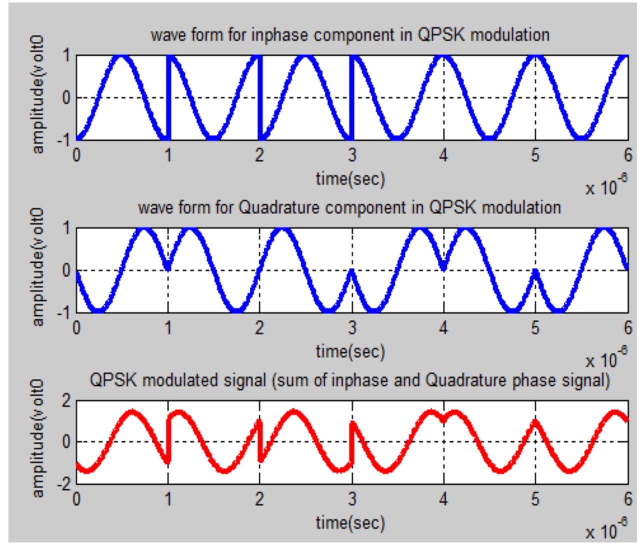


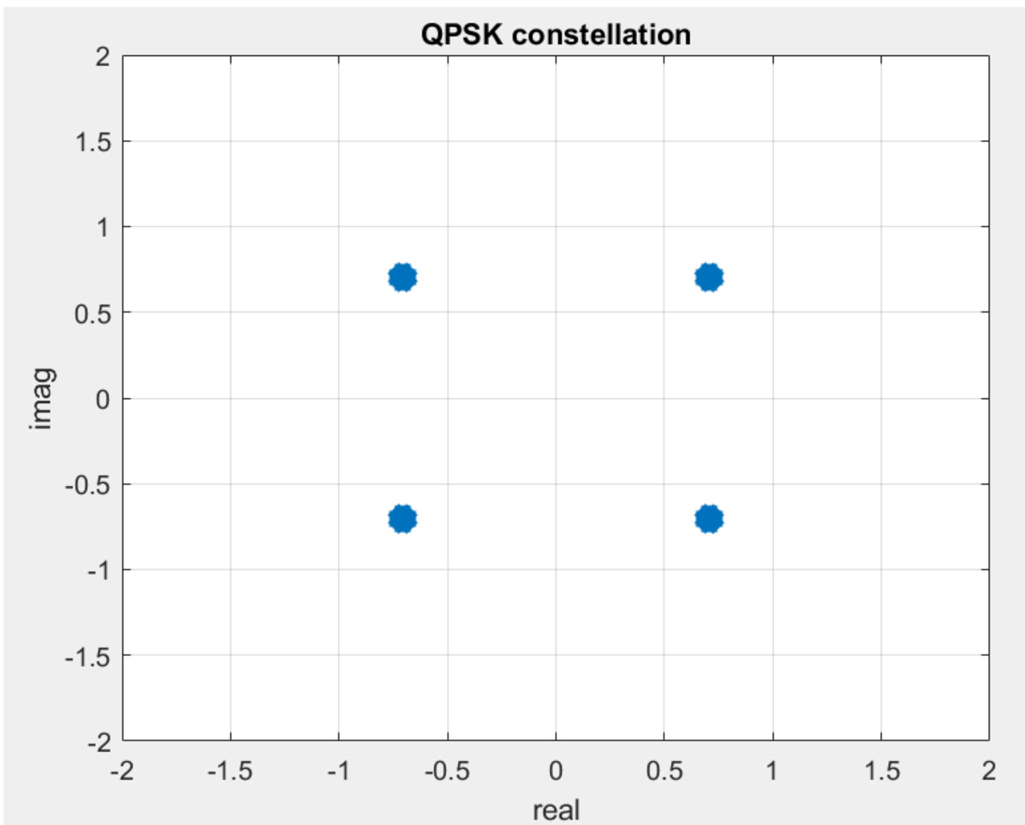
12. QPSK MODULATION AND CONSTELLATION

```
clc;
clear all;
close all;
data=[0 0 1 1 0 1 1 0 1 1 1 0]; % information
figure(1)
stairs(data, 'linewidth',3), grid on;
title('  Information before Transmiting ');
axis([ 1 12 0 1.5]);
data_NZR=2*data-1; % Data Represented at NZR form for
QPSK modulation
s_p_data=reshape(data_NZR,2,length(data)/2);  % S/P
convertion of data
br=10.^6; %Let us transmission bit rate  1000000
f=br; % minimum carrier frequency
T=1/br; % bit duration
t=T/99:T/99:T; % Time vector for one bit information
y=[];
y_in=[];
y_qd=[];
d=zeros(1,length(data)/2);
for i=1:length(data)/2
    p=data(2*i);
    imp=data(2*i - 1);
    y1=s_p_data(1,i)*cos(2*pi*f*t); % inphase component
    y2=s_p_data(2,i)*sin(2*pi*f*t) ;% Quadrature
component
    y_in=[y_in y1]; % inphase signal vector
    y_qd=[y_qd y2]; %quadrature signal vector
    y=[y y1+y2]; % modulated signal vector
    if (imp == 0) && (p == 0)
       d(i)=exp(j*pi/4);%45 degrees
    end
    if (imp == 1)&&(p == 0)
        d(i)=exp(j*3*pi/4);%135 degrees
    end
    if (imp == 1)&&(p == 1)
        d(i)=exp(j*5*pi/4);%225 degrees
    end
    if (imp == 0)&&(p == 1)
        d(i)=exp(j*7*pi/4);%315 degrees
```

```
    end
end
Tx_sig=y; % transmitting signal after modulation
qpsk=d;
tt=T/99:T/99:(T*length(data))/2;
figure(2)
subplot(3,1,1);
plot(tt,y_in,'linewidth',3), grid on;
title(' wave form for inphase component in QPSK
modulation ');
xlabel('time(sec)');
ylabel(' amplitude(volt)');
subplot(3,1,2);
plot(tt,y_qd,'linewidth',3), grid on;
title(' wave form for Quadrature component in QPSK
modulation ');
xlabel('time(sec)');
ylabel(' amplitude(volt)');
subplot(3,1,3);
plot(tt,Tx_sig,'r','linewidth',3), grid on;
title('QPSK modulated signal (sum of inphase and
Quadrature phase signal)');
xlabel('time(sec)');
ylabel(' amplitude(volt)');
figure(3);
plot(d,'*','linewidth',10);%plot constellation without
noise
axis([-2 2 -2 2]);
grid on;
xlabel('real'); ylabel('imag');
title('QPSK constellation');
```


Information before Transmiting

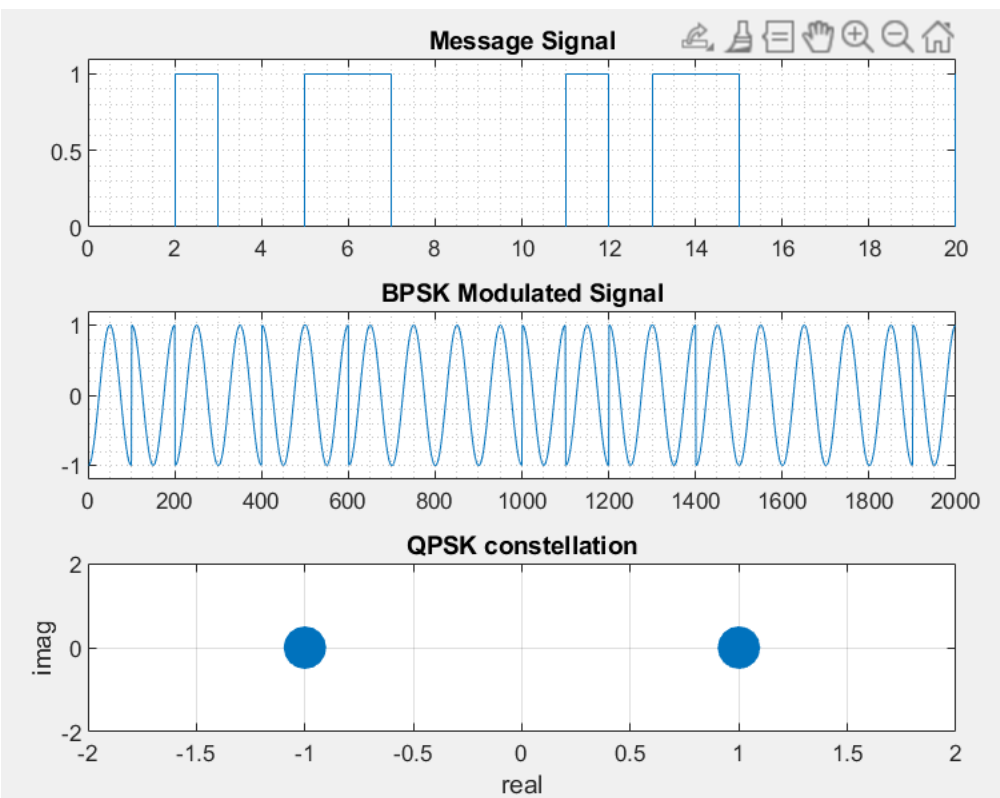

wave form for inphase component in QPSK modulation
amplitude(volt0
time(sec)
x 10^-6
wave form for Quadrature component in QPSK modulation
amplitude(volt0
time(sec)
x 10^-6
QPSK modulated signal (sum of inphase and Quadrature phase signal)
amplitude(volt0
time(sec)
x 10^-6


QPSK constellation
imag
real

13 (A). CONSTELLATION OF BPSK

```
clc;
clear all;
close all;
%****************************Message
Signal******************************
N=20; %Number of 1 and 0
data = randi([0 1],N,1); %Getting 1 and 0 randomly
%data representing at NZR
data_NZR = 2*data-1; % representing 1 as 1 and 0 as -1
s_p_data = reshape(data_NZR,1,length(data)); %Reshaping
as 1bits in array
bit_rate = 10.^3;
f = bit_rate; %minimum carrier frequency
Tb = 1/bit_rate ; %bit duration
t = 0:(Tb/99):Tb ; %Time vector for one bit information
%****************************BPSK
Modulation*****************************
x_mod = [];
Ac = 1;
xc = Ac*cos(2*pi*f*t);
for (l=1:length(data))
    xc = s_p_data(1,l)*Ac*cos(2*pi*f*t);
    x_mod = [x_mod xc]; %Modulated signal
end
M = 2; % MOD is BPSK therefor M = 2
x_mod_const = [];  % Constellation points of the
Modulated signal
for (n=1:length(data))
    if s_p_data(1,n) == -1  %point 0
        xC = exp(-i*((2*pi*0)/M))
    elseif s_p_data(1,n) == 1  %point 1
        xC = exp(-i*((2*pi*1)/M))
    end
    x_mod_const = [x_mod_const xC];
end
Tx = x_mod; %Transmitting Data
Tx_const = x_mod_const; %Constellation of Tx
figure(1)
subplot(3,1,1);
```

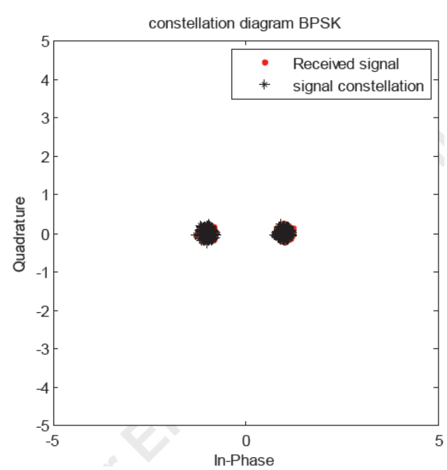
```
stairs(data_NZR); grid minor; xlim([0,N]);
ylim([0,1.1]);
title('Message Signal');
subplot(3,1,2);
plot(x_mod); grid minor; title('BPSK Modulated
Signal'); ylim([-1.2,1.2]);

%Constellation Diagram of the Rx
subplot(3,1,3);
plot(Tx_const,'o', 'linewidth',10);%plot constellation
without noise
axis([-2 2 -2 2]);
grid on;
xlabel('real'); ylabel('imag');
title('BPSK constellation');
```



13 (B). SIMULATION OF SIGNAL CONSTELLATIONS OF BPSK & QAM

```
clear all;
close all;
M=2;
k=log2(M);
n=3*1e5;
nsamp=8;
X=randint(n,1);
xsym = bi2de(reshape(X,k,length(X)/k).','left-msb');
Y_psk= modulate(modem.pskmod(M),xsym);
Ytx_psk = Y_psk;
EbNo=30;
SNR=EbNo+10*log10(k)-10*log10(nsamp);
Ynoisy_psk = awgn(Ytx_psk,SNR,'measured');
Yrx_psk = Ynoisy_psk;
h1=scatterplot(Yrx_psk(1:nsamp*5e3),nsamp,0,'r.');
hold on;
scatterplot(Yrx_psk(1:5e3),1,0,'k*',h1);
title('constellation diagram BPSK');
legend('Received signal' ,'signal constellation');
axis([-5 5 -5 5]);
hold off;
```

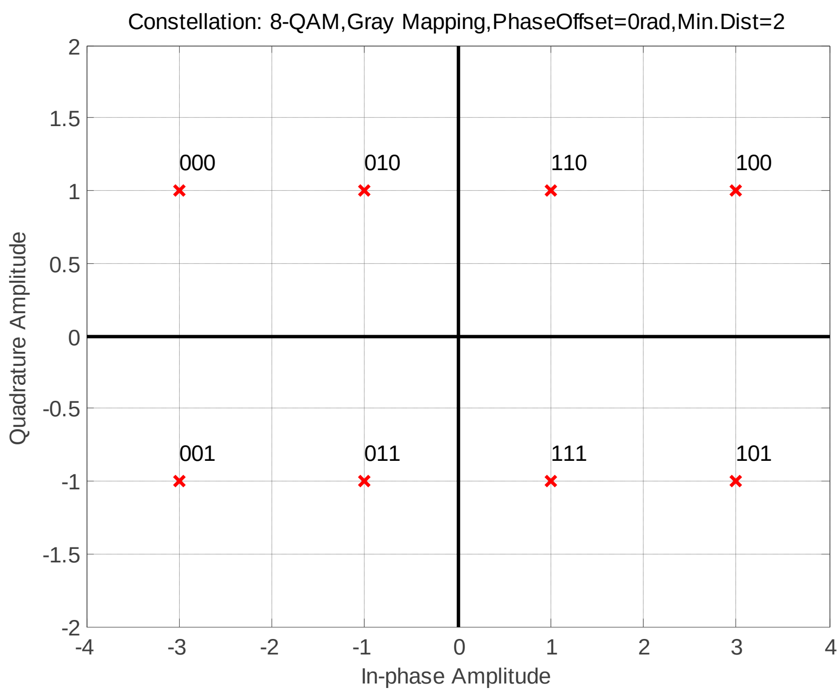
OUTPUT:



SIGNAL CONSTELLATION FOR 8-QAM

```
data=randi([0 1],96,1);
hModulator=comm.RectangularQAMModulator(8,'BitInput',tr
ue);
hModulator.PhaseOffset=0;
modData=step(hModulator,data);
constellation(hModulator)
```

SIMULATED OUTPUT:



14. SIMULATION OF LINEAR BLOCK AND CYCLIC ERROR CONTROL CODING SCHEMES

LINEAR BLOCK CODE

```
clc;
clear all;
n=7;          %Length of the output code word
k=4;    %Length of the input message bits

if(k<n)
    m=input('Enter the message:');
    p=[1 1 0;0 1 1;1 1 1;1 0 1];  %Parity matrix
    g=[eye(k),p];
%Generator matrix
    disp('Generator matrix:');
    disp (g);

    % encode
    c=rem(m*g,2);  %Create the code by multiplying
the message and generator polynomial
    disp('coded message at transmission side:');
    disp (c);

    % noise
    e=[0 0 1 0 0 0 0];% Assume error pattern
    r=xor(c,e);             %Introduce a random one
bit error in the message
    disp('Received code at received side:');
    disp(r);

    %Parity matrix
    h=[eye(n-k),[p]'];      %Creates a parity check
matrix
    disp('Parity matrix:');
    disp(h);
```

```
    %Syndrome
    disp('Syndrome');
    ht=h';                          % Transpose of the 
parity check matrix
    s=mod(r*ht,2);     % Calculates the syndrome 
value from the received code and parity matrix
    disp(s);

    %Find the error bit from the transpose of the 
parity check matrix
    for j=1:n
        t=n-k;
        for i=1:n-k
            if(s(1,i)==ht(j,i))
            t=t-1;
            end
        end
        if(t==0)
        break;
        end
    end

    disp('Position of errorbit is');
    disp(j);

    %Error pattern
    r(j)=~r(j); %Correct the error
    disp('Corrected code:');
    disp(r);

    for i=1:k
      dm(i)=r(i); %The first k bits of the 
corrected code word is the message
    end
    disp('Decoded message:');
    disp(dm);

else
    disp('k should be less than n');
end
```

SAMPLE OUTPUT

Enter the message:[1 0 1 1]

Generator matrix:

```
1 0 0 0 1 1 0
0 1 0 0 0 1 1
0 0 1 0 1 1 1
0 0 0 1 1 0 1
```

coded message at transmission side:

```
1 0 1 1 1 0 0
```

Random one bit error

```
0 1 0 0 0 0 0
```

Received code at received side:

```
1 1 1 1 1 0 0
```

Parity matrix:

```
1 0 0 1 0 1 1
0 1 0 1 1 1 0
0 0 1 0 1 1 1
```

Syndrome

```
0 1 0
```

Position of errorbit is

```
2
```

Corrected code:

```
1 0 1 1 1 0 0
```

Decoded message:

```
1 0 1 1
```

CYCLIC BLOCK CODE

```
clc;
clear all;

%%%%code can take values (7,4),(7,3),(8,3),(8,4),....
codeLength=input('Enter the length of the code :');
%Length of the output code word
messageLength=input('Enter the length of the
message:'); %Length of the input message bits
if (messageLength<codeLength)
    Message = input('Enter the message bits:');
    disp(Message);

    disp ('Cyclic polynomial');

cyclicPolynomial=cyclpoly(codeLength,messageLength,'min
'); %Creates a polynomials for cyclic codes
    disp(cyclicPolynomial);

    disp('Encoded word');
    %Encodes the message bits using cyclic code

code=encode(Message,codeLength,messageLength,'cyclic/fm
t',cyclicPolynomial);
    disp(code);

    disp('Error pattern generation');
    error=randerr(1,codeLength);
%Introduces a random one bit error
    disp(error);

    disp('Received vector');
    receivedCode = xor(code,error);   %Xor the error
with the code word
    disp(receivedCode);

    disp('Decode message bits');
```

```
   %Decodes the received code word and recovers the
original message

msg=decode(receivedCode,codeLength,messageLength,'cycli
c');
    disp(msg);
else
    disp('k value should be less than n');
end
```

SAMPLE OUTPUT

Enter the length of the code :7

Enter the length of the message:4

Enter the message bits:[1 0 1 0]

```
   1   0   1   0
```

Cyclic polynomial

```
   1   0   1   1
```

Encoded word

```
   0   1   1   1   0   1   0
```

Error pattern generation

```
   0   0   0   0   1   0   0
```

Received vector

```
   0   1   1   1   1   1   0
```

Decode message bits

```
   1   0   1   0
```

15. CONVOLUTION CODING SCHEME

```
Clc;
clear all;
close all;
m=[1 0 1 1];
p=2  %Number of flipflops
z=zeros(1,p);
mm=horzcat(m,z); %Additional zeros added with message
sequence
x=[]; % flipflop states
c=[]; %code vector
for i = 1:1:length(mm)
    d1(i+1) = mm(i);
    d2(i+1) = d1(i);
    x=[x; d1(i) d2(i)]; % states of shift register
    u(i) = xor(x(i,1),x(i,2));
    c1(i) = xor(u(i), mm(i));
    c2(i)=xor(mm(i), x(i,2));
    c= [c c1(i) c2(i)]
end
 disp (‘ States of the shift register:’)
x
disp(‘Code Vector’)
c
```

OUTPUT

States of the shift register:

x =

0	0
1	0
0	1
1	0
1	1
0	1

Code Vector

c =

1 1 1 0 0 0 0 1 0 1 1 1

16. COMMUNICATION LINK SIMULATION

```
clc;
clear all;
close all;
t=0:.01:1;
fc = 2;
M = input('Enter the message bits: ');
messageLength=length(M);
time=[];
digitalSignal=[];
pskSignal=[];
carrierSignal=[];
for i=1:1:messageLength

    carrier = sin(2*pi*fc*t);
    carrierSignal = [carrierSignal carrier];

    if M(i) == 1
        bit = ones(1,length(t));
    else
        bit = zeros(1,length(t));
    end
    digitalSignal = [digitalSignal bit];
    if M(i) == 1
        PSK = sin(2*pi*fc*t+0);
    else
        PSK = sin(2*pi*fc*t+pi);
    end
    pskSignal = [pskSignal PSK];
    time=[time t];
    t=t+1;
end

 % Add Rayleigh fading to the transmitted signal
ray=sqrt(randn(1,length(t)*messageLength).^2+randn(1,le
ngth(t)*messageLength).^2);
pskSignalRay= pskSignal*mod(rand(1,1),1)+ray;
demodMsgSignal=[];
```

```
for i=1:1:messageLength
    rx(i)=sum(pskSignalRay(length(t)*(i-
1)+1:length(t)*i).*carrier);
    if rx(i)< 0
        demodMsgSignal=[demodMsgSignal
zeros(1,length(t))];
    else
        demodMsgSignal=[demodMsgSignal
ones(1,length(t))];
    end
end


subplot(3,1,1);
plot(time,digitalSignal,'m','linewidth',3);
grid on;
axis([0 messageLength -0.5 1.5]);
title('Digital signal');

subplot(3,1,2);
plot(time,carrierSignal);
grid on;
title('Carrier signal');

subplot(3,1,3);
plot(time,pskSignal);
grid on;
title('PSK modulated signal');
figure,

subplot(2,1,1);
plot(time,pskSignalRay,'r');
grid on;
title('Received signal');

subplot(2,1,2);
plot(time,demodMsgSignal,'m','linewidth',3);
grid on;
axis([0 messageLength -0.5 1.5]);
title('Error removed and demodulated');
```

Output

Enter the message bits: [1 0 1 0 0 1]