# PABNA UNIVERSITY OF SCIENCE AND TECHNOLOGY


***Department Of***
***Information and Communication Engineering***
***Faculty Of***
***Engineering and Technology***
***Course Code: ICE-3206***
***Course Title: Digital Communication Sessional***

# Lab Report

| Submitted By: | Submitted to: |
|---|---|
| **Joana Hossain Dip**<br>Roll : 180638<br>Registration no: 1065321<br>3rd Year 2nd Semester<br>Session: 2017-18<br>Department of Information and Communication Engineering<br>Pabna University of Science and Technology<br>Pabna-6600,Bangladesh | **Md. Imran Hossain**<br>Assistant Professor,<br>**Tarun debnath**<br>Lecturer<br>Department of Information and Communication Engineering<br>Pabna University of Science and Technology<br>Pabna-6600, Bangladesh<br><br>***Signature:*** |

Tarun Debnath
Lecturer
Dept. of Information and Communication Engineering
Pabna University of Science and Technology

# INDEX

## Experiment No: 01

**Experiment Name:** Write a MATLAB code to demonstrate Unipolar Non-Return to Zero signaling

## Theory:

A line code is the code used for data transmission of a digital signal over a transmission line. This process of code is chosen so as to avoid overlap and distortion of signal such as inter symbol interference.

## Properties of Line Coding:

↳ As the coding is done to make more bits transmission a signal, the bandwidth used is much reduced.

↳ For a given bandwidth the power is efficiently used.

↳ Error detection is done and the bipolar too has a correction capability.

↳ The probability of error is much reduced.

↳ Power density is much favorable.

↳ The timing content is adequate.

↳ Long strings of 1s and 0s is avoided to maintain transparency.

## Unipolar NRZ signaling:

In this line code, symbol 1 is represented by transmitting a pulse of amplitude A for the duration of the symbol ; and symbol 0 is represented by switching of the pulse as illustrated in figure-1.

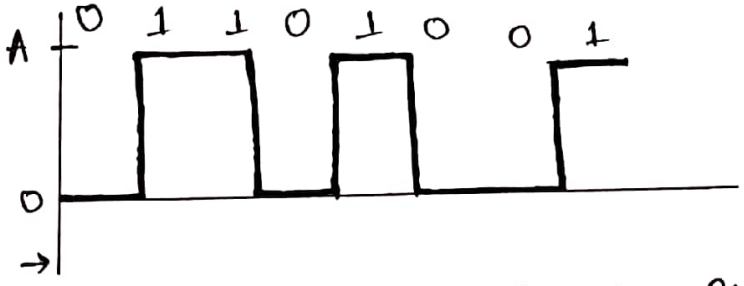



Figure-1 : Unipolar Non Return to Zero signaling.

## Advantages:

↳ It is simple.

↳ A lesser bandwidth is required.

## Disadvantages:

↳ No error correction done.

↳ Presence of low frequency components may cause the signal drop

↳ No clock is present.

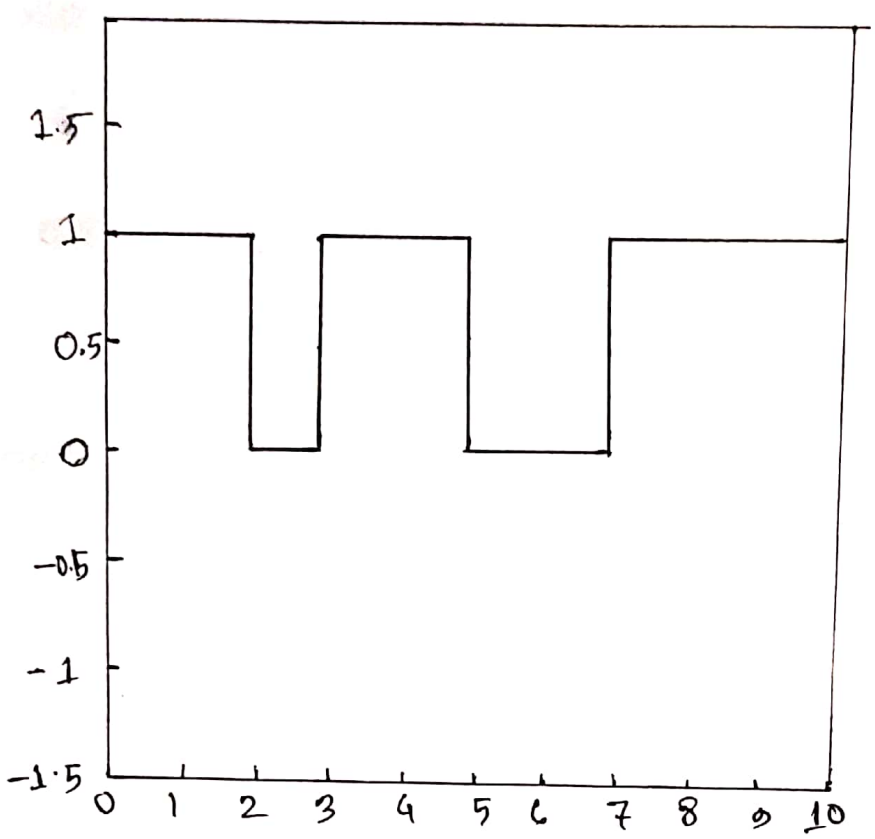
↳ Loss of synchronization is likely to occur (specially for long strings of 1s and 0s).

## Source code in MATLAB:

```
%unipolar non return to zero signal
clc;
clear all;
close all;
N=10;                          %number of bits
n=randi([0,1],1,N)             %random bit generation

%mapping function
for m=1:N
    if n(m)==1
        nn(m)=1;
    else
        nn(m)=0;
    end
end
nn

%signal shaping
i=1;
t=0:0.01:length(n);            %t=0,0.01,0.02,0.1,0.2,1,1.01,1.1...10,t=1000
%times duretion set up for a single binary bit
for j=1:length(t)              %j=1,2,3,4,5,6,7,8,9,10,....1000  %indexing
setup for time duration
    if t(j)<=i                 %binary input data index checkup condition
        y(j)=nn(i);            %assign value from the mapping function
    else
        y(j)=nn(i);
        i=i+1;                 %binary input data index increment
    end
end
plot(t,y,'linewidth',2);
axis([0,N,-1.5,1.5]);          %axis set_up
grid on;
title("unipolar non-return-to-zero(NRZ) signaling");
```

## Output:

Unipolar non-return-to-zero (NRZ) signalling



Experiment No: 02

Experiment Name: Write a MATLAB Program to represent the line code of Polar Non-Return to Zero (NRZ) signalling.

Theory:

In Polar Non-Return-to-Zero (NRZ) signalling symbol 1 and 0 are represented by transmitting pulses of amplitudes +A and -A respectively as illustrated in figure-2.

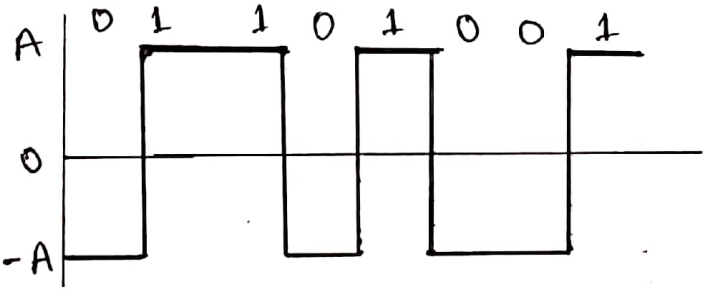



Figure-2. Polar Non-Return-to Zero Signalling

This line code is relatively very easy to generate but its disadvantage is that the power

Spectrum of the signal is large near zero frequency.

Advantages:

↳ It is simple

↳ No low-frequency components are present.

Disadvantage:

↳ No error correction.

↳ NO clock is present

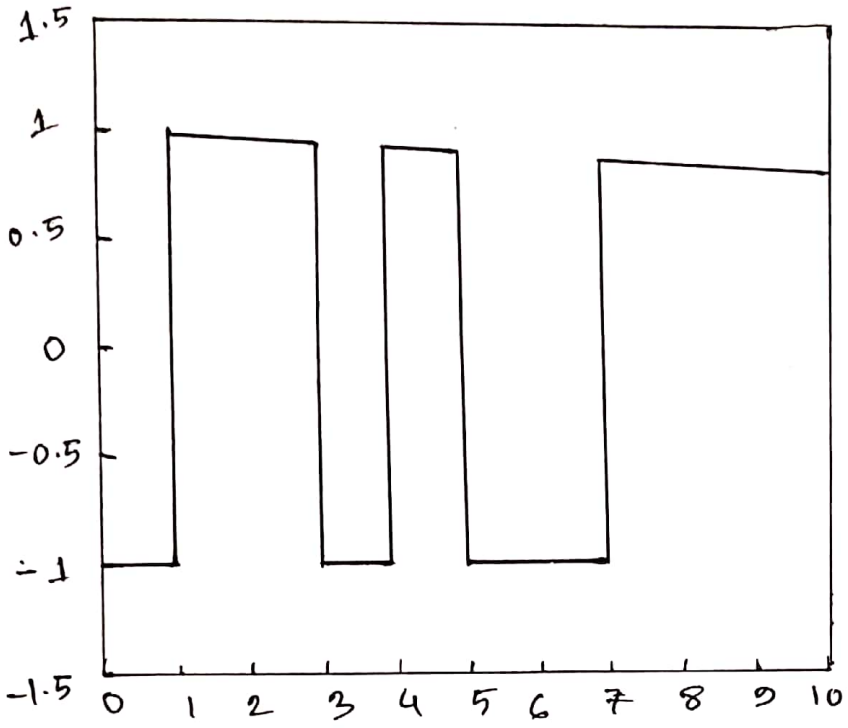
↳ The signal drop is caused at the places where the signal is non-zero to 0 Hz.

## Source code in MATLAB:

```
%polar non return to zero signal
clc;
clear all;
close all;
N=10;                          %number of bits
n=randi([0,1],1,N)             %random bit generation

%mapping function
for m=1:N
    if n(m)==1
        nn(m)=1;
    else
        nn(m)=-1;
    end
end
nn

%signal shaping
i=1;
t=0:0.01:length(n);            %t=0,0.01,0.02,0.1,0.2,1,1.01,1.1...10,t=1000
%times duretion set up for a single binary bit
for j=1:length(t)              %j=1,2,3,4,5,6,7,8,9,10,....1000  %indexing
setup for time duration
    if t(j)<=i                 %binary input data index checkup condition
        y(j)=nn(i);            %assign value from the mapping function
    else
        y(j)=nn(i);
        i=i+1;                 %binary input data index increment
    end
end
plot(t,y,'linewidth',2);
axis([0,N,-1.5,1.5]);          %axis set_up
grid on;
title("polar non-return-to-zero(NRZ) signaling");
```

Output :

Polar Non-return-to-Zero (NRZ) Singling.



Experiment No: 03

Experiment Name: Write a MATLAB Program to represent the Unipolar Return-to-Zero (RZ) signalling.

Theory:

In this type of signalling, symbol 1 is represented by a rectangular pulse of amplitude A and half-symbol width and symbol 0 is represented by transmitting no pulse as described in figure-3.

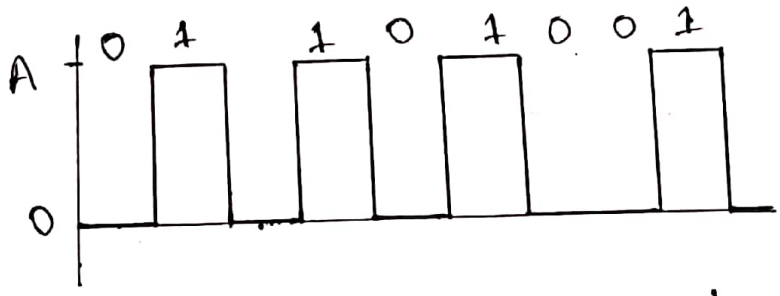



Figure-3: Unipolar Return to Zero(RZ) signalling.

An attractive features of this line code is the presence of delta function at $f=0, \pm 1/T_b$ is the power spectrum of the transmitted signal, which can be used for bit-timing recovery

at the receiver. So half of the bit duration remains high but it immediately return to zero and shows the absence of pulse during the remaining half of the bit duration.

Advantage:

↳ It is simple.

↳ The spectral line present at the symbol rate can be used as a clock.

Disadvantage:

↳ No error correction.

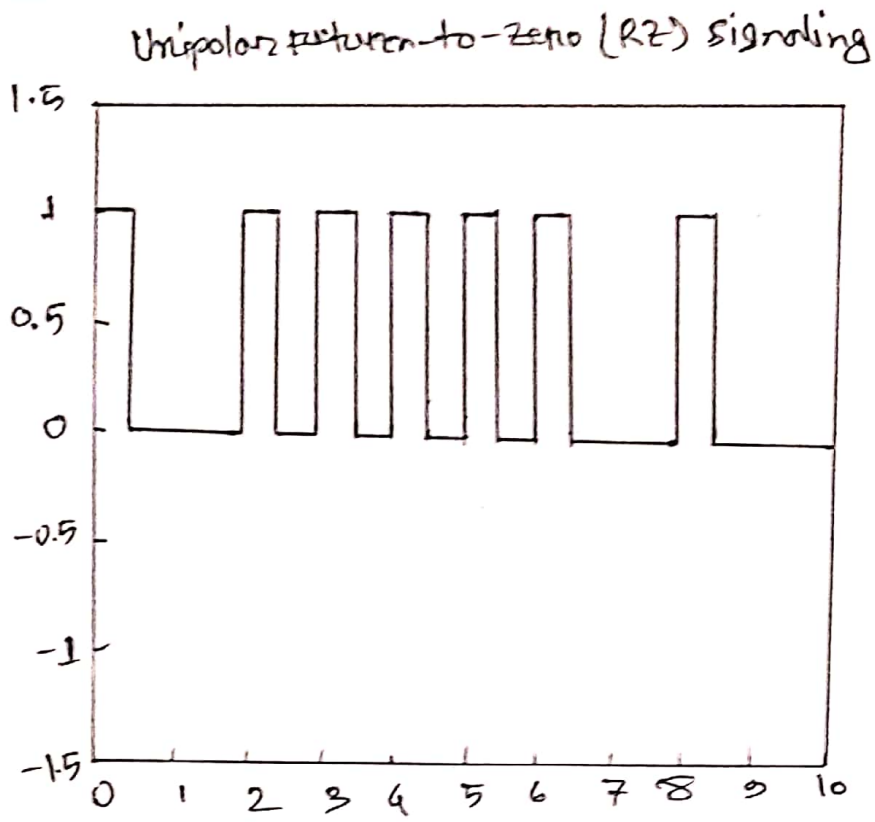
↳ Occupies twice the bandwidth as unipolar Non-return-to-zero (NRZ).

↳ The signal droop is caused at the places where signal is non-zero at 0Hz.

## Source code in MATLAB:

```
%unipolar return to zero signal
clc;
clear all;
close all;
N=10;                          %number of bits
n=randi([0,1],1,N)             %random bit generation

%RZ pulse shaping
i=1;
a=0;                           %initial value for first half cycle
b=0.5;                         %initial value for second half cycle
t=0:0.01:length(n);
for j=1:length(t)
    if t(j)>=a && t(j)<=b          %condition for the first half cycle
        y(j)=n(i);                 %assign first 50 values for
    elseif t(j)>b && t(j)<=i       %condition for second half cycle
            y(j)=0;                %set all values 0 for the second half cycle
    else
            i=i+1;                 %binary input data index increment
            a=a+1;                 %initial value for first half cycle
increment
            b=b+1;                 %initial value for second half cycle
increment
    end
end

plot(t,y,'linewidth',2);       %linewidth 2 for clear visualization
axis([0,N,-1.5,1.5]);          %axis set_up
grid on;
title("unipolar return-to-zero(RZ) signaling");
```

Output:



Experiment No: 04

Experiment Name: Write a MATLAB program to represent the Bipolar Return-to Zero signalling.

Theory:

Bipolar Return to zero: It uses three amplitude levels as describe below in figure-4.

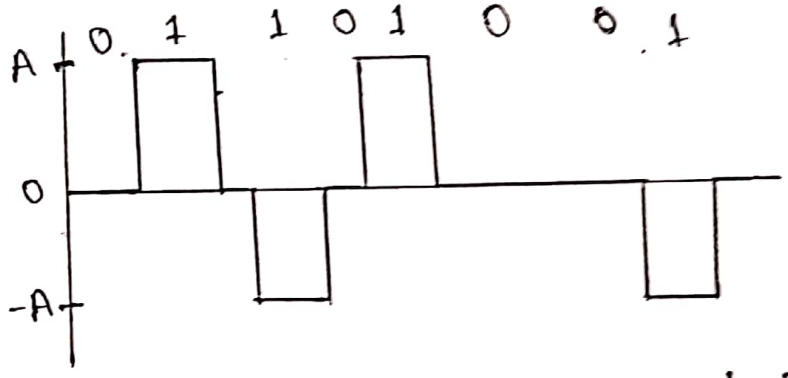



figure-4: Bipolar Return to zero signalling.

Specifically, positive and negative pulses of equal amplitude (i.e. +A and -A) are used alternatingly for symbol 1, with each pulse having a half symbol width; no pulses is always used for symbol 0. This line code is also called alternate mark inversion (AMI) signalling.

## Advantage:

↳ It is simple.

→ No low frequency components are present.

→ occupies low bandwidth than unipolar and bipolar non-Return-to-zero schemes.

↳ This technique is suitable for transmission over AC coupled lines, as signal drooping doesn't occur here.

↳ A single error detection capability is present in this.

## Disadvantage:

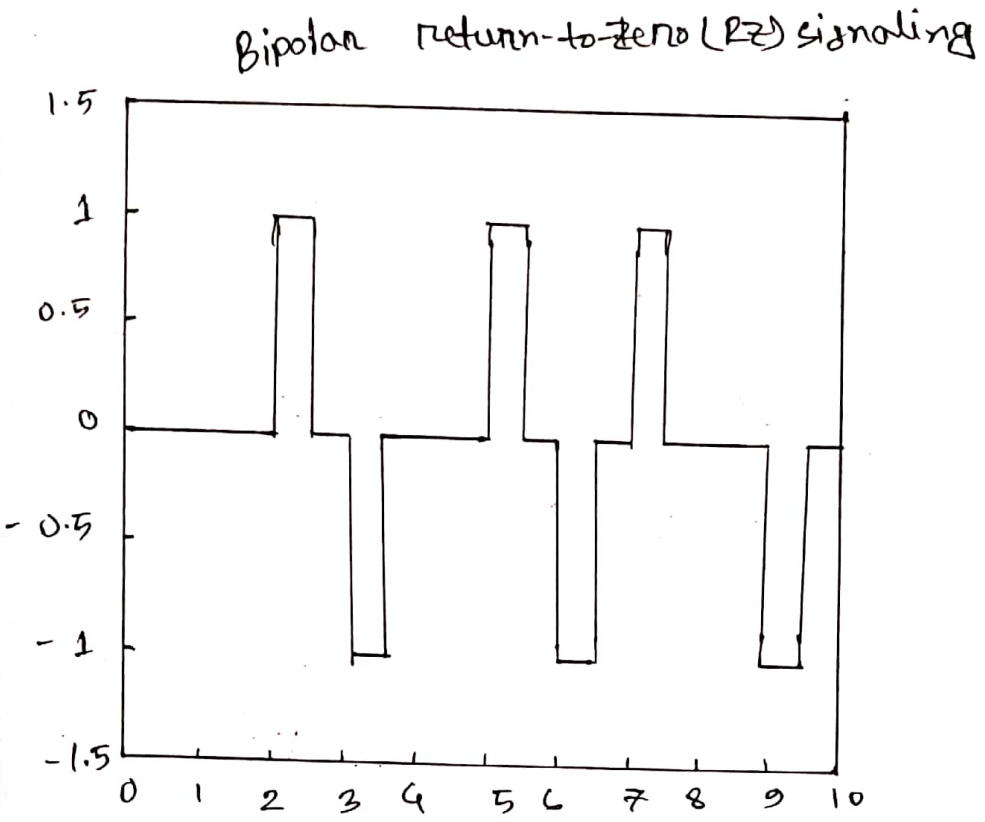
↳ No clock is present

↳ Long string of data causes loss of synchronization.

## Source code in MATLAB:

```
%Bipolar return to zero signaling
clc;
clear all;
close all;
N=10;            %number of bits
n=[0 0 1 1 0 1 1 1 0 1];
%n=randi([0,1],1,N);      %random bit generation
%Binary to polar conversion
f=1;
for m=1:N
    if n(m)==1
        if f==1
            nn(m)=1;
            f=-1;
        else
            nn(m)=-1;
            f=1;
        end
    else
        nn(m)=0;
    end
end
%bipolar rz pulse shaping
i=1;
a=0;             %initial value for first half cycle
b=0.5;           %initial value for second half cycle
t=0:0.01:length(n);
for j=1:length(t)
    if t(j)>=a && t(j)<=b        %condition for the first half cycle
        y(j)=nn(i);              %assign first 50 values for
    elseif t(j)>b && t(j)<=i     %condition for the second half cycle
        y(j)=0;                  %set all values 0 for the second half cycle
    else
        i=i+1;          %binary input data index increment
        a=a+1;          %initial value for the first half cycle increment
        b=b+1;           %initial value for second half cycle increment
    end
end
plot(t,y,'linewidth',2);         %linewidth 2 for clear visualization
axis([0,N,-1.5,1.5]);            %axis set-up
grid on;
title('Bipolar return-to-zero(BRZ) signalling');
```

)utput :



Experiment No : 05

Experiment Name: Write a MATLAB Program to represent the split-phase (Manchester code) signalling.

Theory :

In Manahastan coding or split-phase signaling is describe below in Figure-5.

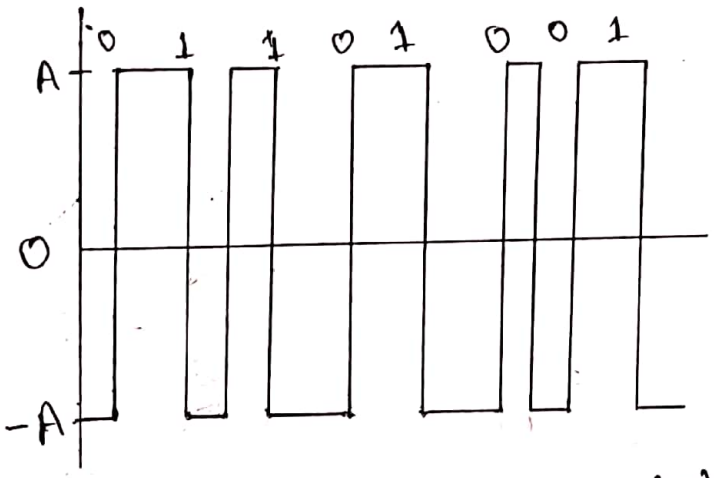



Figure-5: (Manchester coding) split-phase signaling.

Symbol 1 is represented by a positive pulse of amplitude A followed by a negative pulse of Amplitude -A, with both pulses being half-symbol wide.

For symbol 0, the polarities of these two pulses are reversed. The Manchester code suppresses the DC component and has relatively insignificant low-frequency components, regardless of the signal statistics.

Advantages:

↳ Manchester coding's main advantage is signal synchronization.

↳ It minimizes the error rate and optimizes reliability.

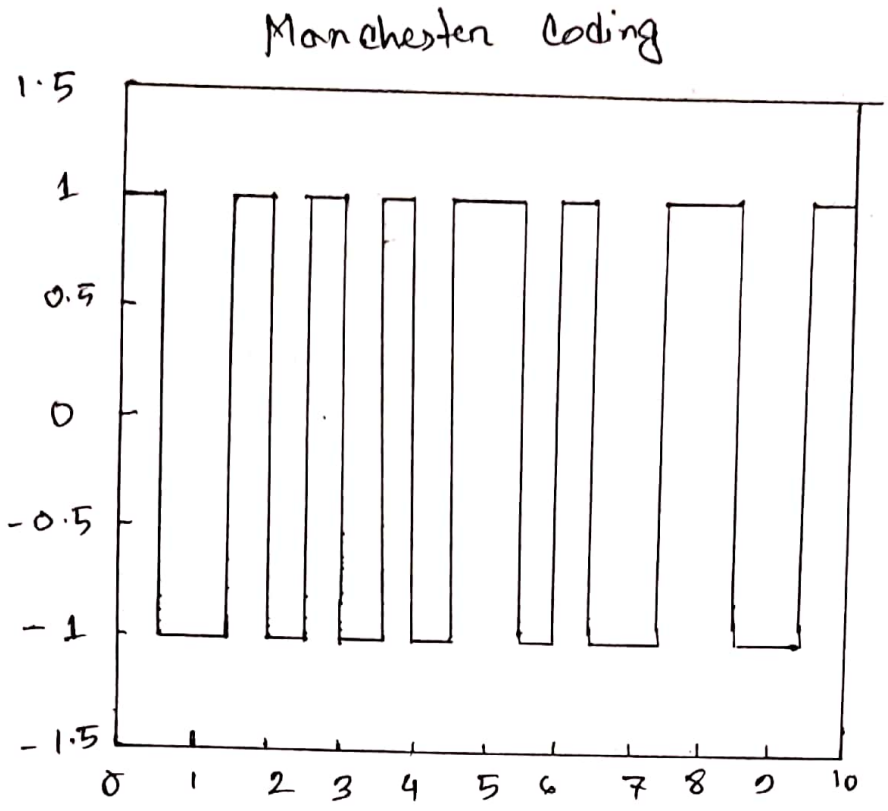
Disadvantage:

↳ The only drawback of Manchester coding scheme is its minimum bandwidth requirements.

## Source code in MATLAB:

```
%Split Phase-Manchester Coding
clc;
clear all;
close all;
N=10; %Number of bits
n=randi([0,1],1,N) %Random bit generation
%Binary to Manchester Conversion
nnn=[];
for m=1:N
if n(m)==1
 nn=[1 -1];
else
 nn=[-1 1];
end
 nnn=[nnn nn];
end
nnn
%Manchester Coding Pulse Shaping
i=1;
l=0.5;
t=0:0.01:length(n);
for j=1:length(t)
if t(j)<=l
 y(j)=nnn(i);
else
 y(j)=nnn(i);
 i=i+1;
 l=l+0.5;
end
end
plot(t,y,'lineWidth', 2);
axis([0,N,-1.5,1.5]); %Axis set-up
grid on;
title('Manchester Coding');
```

Output:



Experiment No: 06

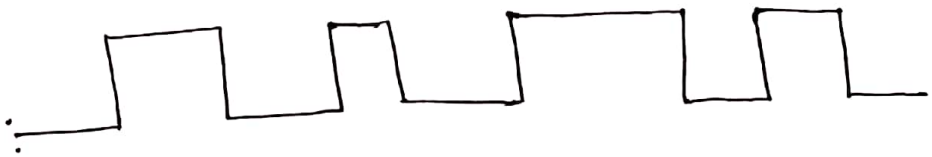
Experiment Name: Write a MATLAB code for Amplitude Shift Keying (ASK) Modulation and demodulation.

Theory:

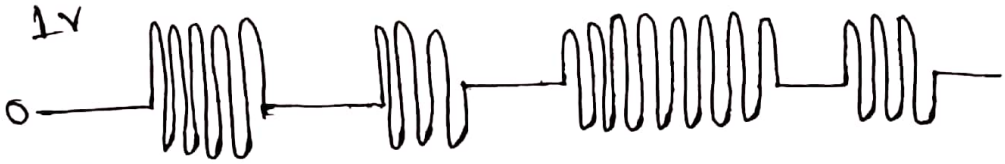
Amplitude Shift keying: (ASK) is a type of Amplitude Modulation which represent the binary data in the form of variations in the amplitude of a signal.

Any Modulated signal has a high frequency carrier

The following figure represents ASK modulated waveform along with its input.



(a) Input binary sequence.




(b) ASK Modulated output wave

Figure-6: ASK Modulation.

ASK Modulation: The ASK Modulator block diagram comprises of the carrier signal generation, the binary sequence from the message signal and the band-limited filter. Following is the block diagram of the ASK modulation.

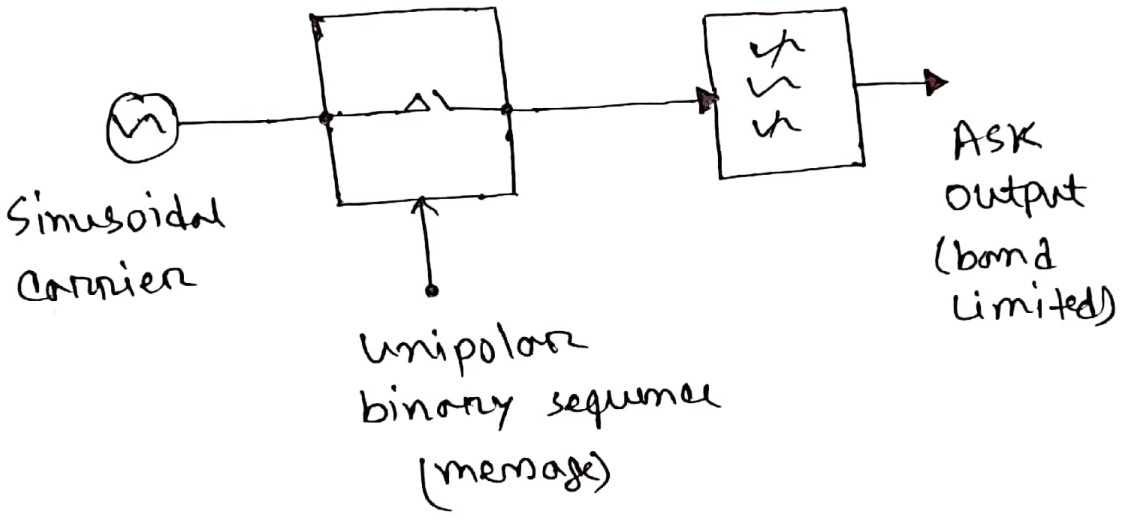



Figure-4: ASK Generation Method

The carrier generator, sends a continuous high-frequency carrier. The binary sequence from the message signal makes the unipolar input to be either High or Low. The high signal closes. The switch, allowing a carrier wave. Hence the output will be the carrier signal at high input. When there is low input, the switch

opens, allowing no voltage to appear. Here, the output will be low.

Demodulation :

Ask signal has a well defined envelope. Thus it is amenable to demodulation by an envelope detector. Some sort of decision making circuitry is necessary for detecting the message. The signal is recovered by using a correlator and decision making circuitry is used to recover the binary sequence.

Algorithm :

Ask Modulation :

Step-1 : Generate carrier signal.

Step-2 : Start FOR loop.

Step-3 : Generate binary data, message signal.

Step-4 : Generate Ask Modulated signal.

Step-5 : plot message signal and Ask modulated signal.

Step-6 : End FOR Loop.

Step-7 : Plot the binary data and carrier.

## ASK Demodulation:

Step-1 : Start For Loop;

Step-2 : Perform correlation of ASK signal with carrier to get decision. variable.

Step-3 : Make decision to get demodulated Binary data. If $x > 0$, choose '1' else choose '0'.

Step-4 : Plot the demodulated binary data.

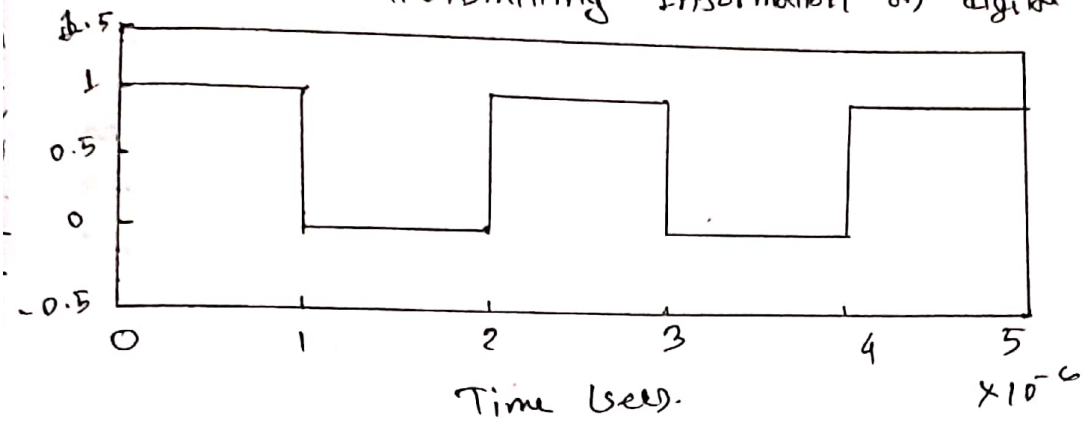
## Source code in MATLAB:

```
clc;
clear all;
close all;
x=[1 0 1 0 1 0 1 0 1];
bp=0.000001;
disp('Binary information at transmitter');
disp(x);

%representation of transmitting binary information as digital signal
bit=[];
for n=1:1:length(x)
    if x(n)==1;
        se=ones(1,100);
    else
        x(n)==0;
        se=zeros(1,100);
    end
    bit=[bit se];
end
t1=bp/100:bp/100:100*length(x)*(bp/100);
subplot(3,1,1);
plot(t1,bit,'linewidth',2.5);
grid on;
axis([0 bp*length(x) -0.5 1.5]);
ylabel('amplitude (volt)');
xlabel('Time(sec)');
title('Transmitting Information as Digital Signal');

%binary ASK Modulation
A1=10;
A2=5;
br=1/bp;
f=br*10;
t2=bp/99:bp/99:bp;
ss=length(t2);
m=[];
for(i=1:1:length(x))
    if x(i)==1;
        y=A1*cos(2*pi*f*t2);
    else
        y=A2*cos(2*pi*f*t2);
    end
    m=[m y];
end
t3=bp/99:bp/99:bp*length(x);
subplot(3,1,2);
plot(t3,m);
grid on;
xlabel('Time(sec)');
ylabel('Amplitude(volt)');
title('Waveform for binary ASK modulation corresponding binary
information');

%binary ASK Demodulation
mn=[];
for n=ss:ss:length(m);
    t=bp/99:bp/99:bp;
    y=cos(2*pi*f*t);
```

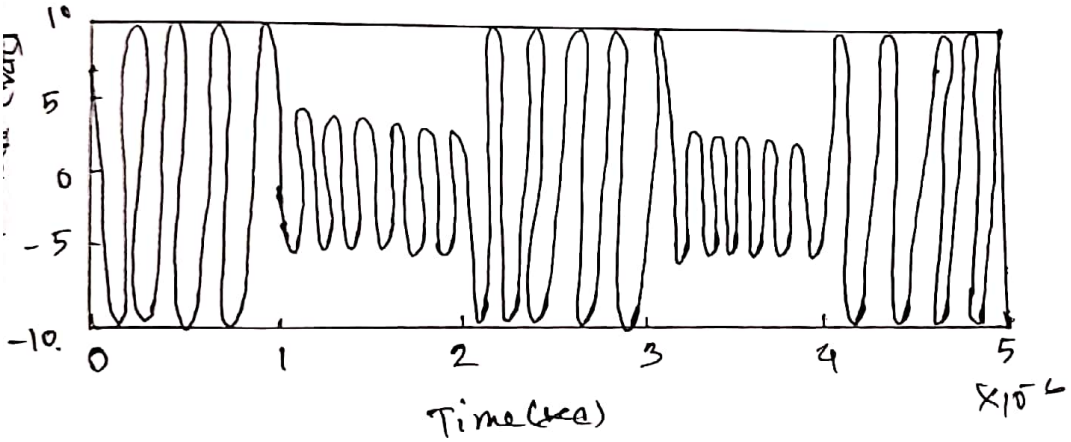
```
    mm=y.*m((n-(ss-1)):n);
    t4=bp/99:bp/99:bp;
    z=trapz(t4,mm);
    zz=round((2*z/bp));
    if(zz>7.5)
        a=1;
    else
        a=0;
    end
    mn=[mn a];
end
disp('binary information in receiver');
disp(mn);

%representation of binary data into digital signal
bit=[];
for n=1:length(mn)
    if mn(n)==1;
        se=ones(1,100);
    else
        se=zeros(1,100);
    end
    bit=[bit se];
end
t4=bp/100:bp/100:100*length(mn)*(bp/100);
subplot(3,1,3);
plot(t4,bit,'linewidth',2.5);
grid on;
axis([0 bp*length(mn) -0.5 1.5]);
xlabel('Time(sec)');
ylabel('Amplitude(volt)');
title('received information as digital signal');
```

Output:

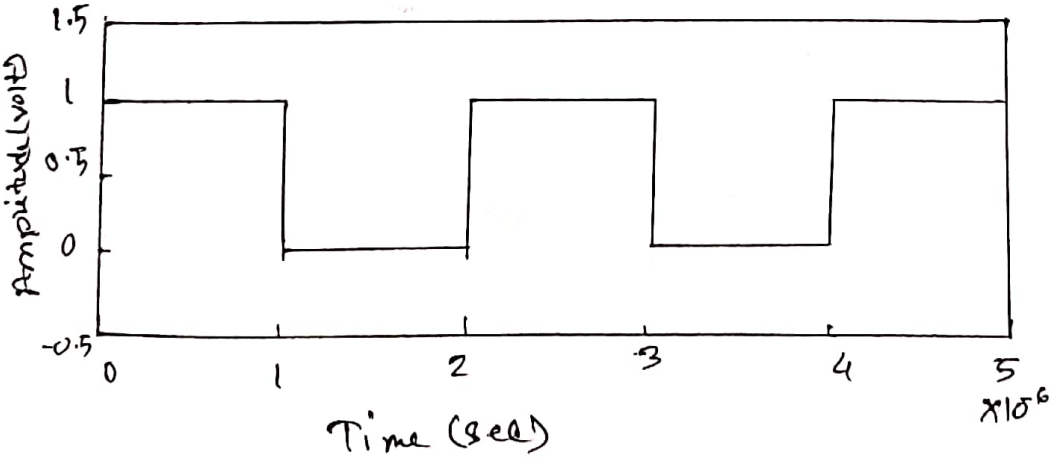
Transmitting Information on digital signal.




ASK Modulation.




ASK DeModulation



Experiment No : 07

Experiment Name : Write a MATLAB code to generate frequency shift keying (FSK) Modulation and Demodulation.

Theory :

Generation of FSK :

Frequency shift keying (FSK) is a frequency modulation scheme in which digital information is transmitted through discrete frequency changes of a carrier wave. The simplest FSK is binary FSK (BFSK). BFSK uses a pair of discrete frequencies to transmit binary (0s and 1s) information. With this scheme the '1' is called the mark frequency and the '0' is called the space frequency.

In binary FSK system, symbol 1 and 0 are distinguished from each other by transmitting one of the two sinusoidal waves that differ in frequency by a fixed amount.

$$S_i(t) = \sqrt{2}E/T_b \cos 2\pi f_i t \; ; \; 0 \le t \le T_b$$

Where $i = 0, 1, 2 \ldots$ and

$E_b$ = Transmitted Energy /bit

$T_b$ = bit interval

$n$ = constant.

BFSK Transmitter: The input binary sequence is represented in its ON-OFF form, with symbol 1 represents by constant amplitude of $\sqrt{E_b}$ with. and symbol 0 represented by zero volts.

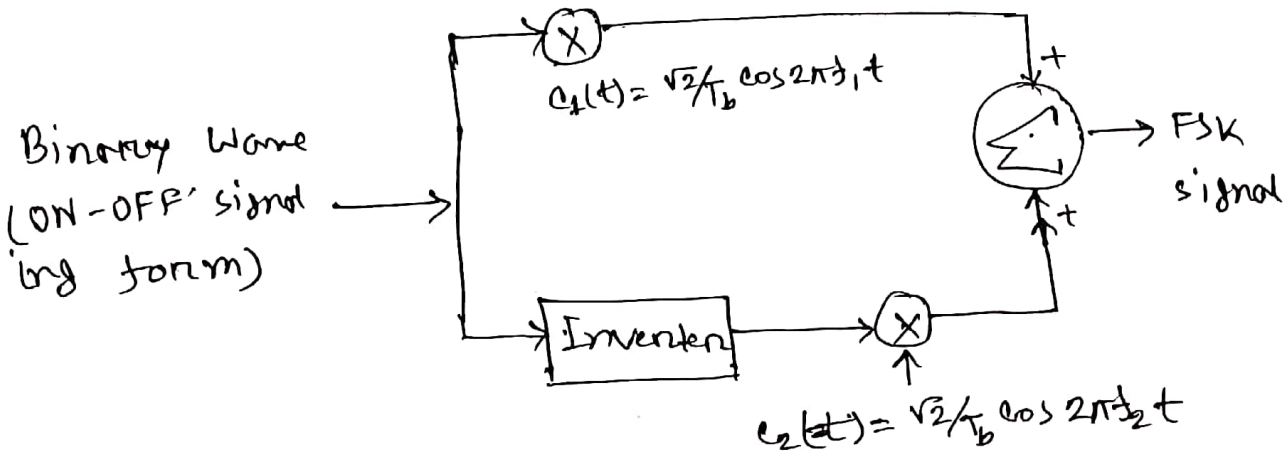



Figure - 8: Binary FSK transmitter.

By using inverter in the lower channel, we make sure that when symbol 1 is at the input. The two frequency $f_1$ and $f_2$ are chosen to be

Integer multiples of the bit rate $1/T_b$. By summing the upper and lower channel outputs we get BFSK signal.

**BFSK Receiver:** The receiver consists of two correlator with common inputs which are supplied with locally generated coherent reference signal $c_1(t)$ and $c_2(t)$.

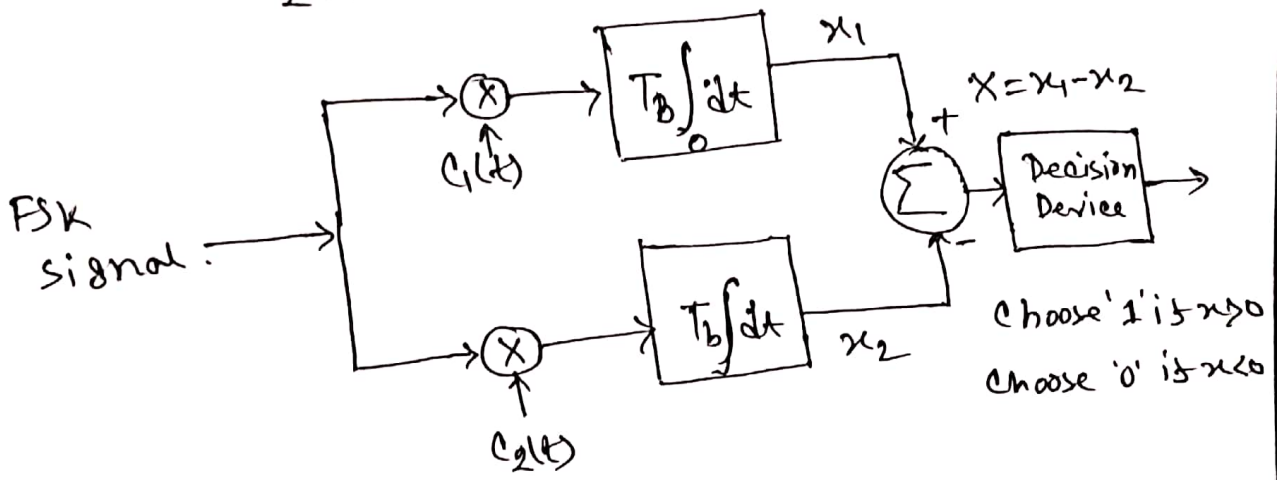



Figure-10: Binary FSK Receiver.

The correlator output are then subtracted one from the other, and the resulting difference x is compared with a threshold of zero volts. If $x > 0$, the receiver decides in favor of symbol 1 and if $x < 0$ the receiver decides in favor of symbol 0.

## Algorithm :

### FSK Modulation :

Step-1 :

Step-2 : Start FOR Loop.

Step-3 : Multiply carrier 1 with message signal and carrier 2 with inverted message signal.

Step-4 : Perform addition to get the FSK modulated signal.

Step-5 : plot message signal and FSK modulated signal.

Step-6 : End FOR loop.

Step-7 : plot the binary data and carriers.

### FSK Demodulation :

Step-1 : Start FOR loop

Step-2 : Perform correlation of FSK modulated signal with carrier 1 and carrier 2 to get two decision variables $x_1$ and $x_2$.

Step-3 : Make decision on $x = x_1 - x_2$ to get demodulated binary path. If $x > 0$ choose '1' else choose '0'.

Step-4 : End FOR loop.

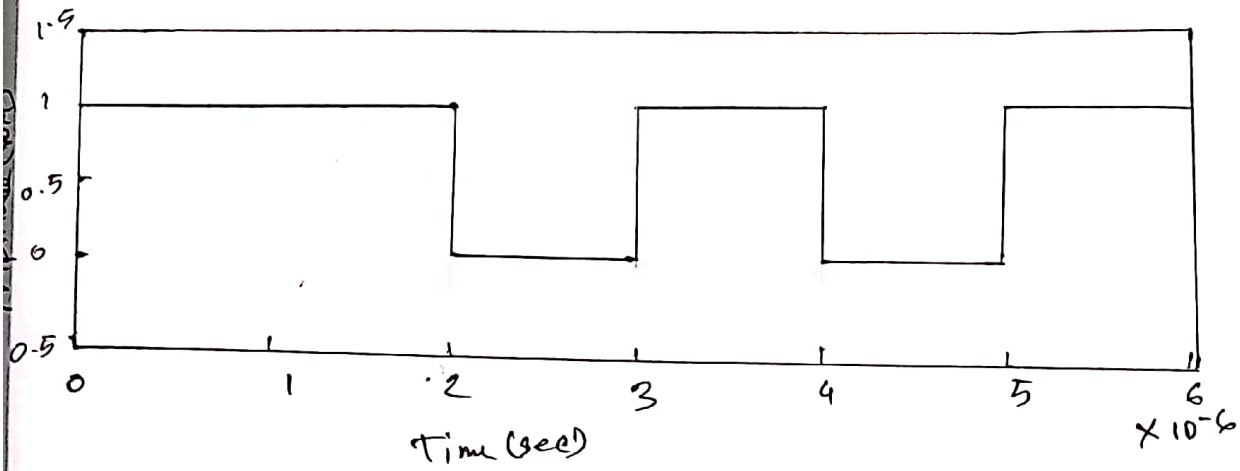
Step-5 : plot the demodulated binary data.

## Source code in MATLAB:

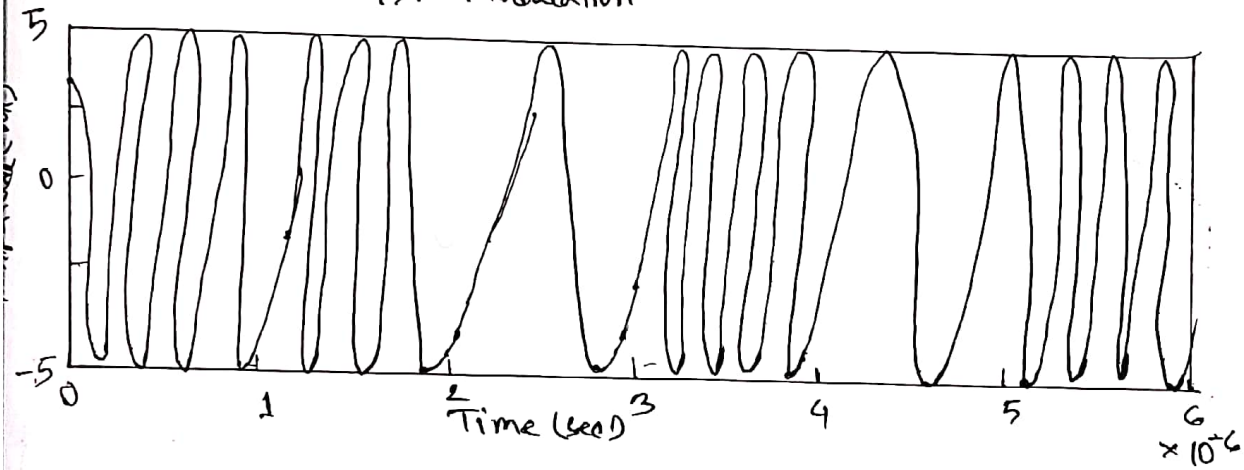
```
clc;
clear all;
close all;
x=[1 1 0 1 0 1];
bp=0.000001;
disp('Binary information at transmitter');
disp(x);

%representation of transmitting binary information as digital signal
bit=[];
for n=1:1:length(x)
    if x(n)==1;
        se=ones(1,100);
    else
        x(n)==0;
        se=zeros(1,100);
    end
    bit=[bit se];
end
t1=bp/100:bp/100:100*length(x)*(bp/100);
subplot(3,1,1);
plot(t1,bit,'linewidth',2.5);
grid on;
axis([0 bp*length(x) -0.5 1.5]);
ylabel('amplitude (volt)');
xlabel('Time(sec)');
title('Transmitting Information as Digital Signal');

%binary FSK Modulation
A=5;
br=1/bp;
f1=br*8;
f2=br*2;
t2=bp/99:bp/99:bp;
ss=length(t2);
m=[];
for(i=1:1:length(x))
    if (x(i)==1)
        y=A*cos(2*pi*f1*t2);
    else
        y=A*cos(2*pi*f2*t2);
    end
    m=[m y];
end
t3=bp/99:bp/99:bp*length(x);
subplot(3,1,2);
plot(t3,m);
grid on;
xlabel('Time(sec)');
ylabel('Amplitude(volt)');
title('Waveform for binary FSK modulation corresponding binary
information');

%binary FSK Demodulation
mn=[];
for n=ss:ss:length(m);
    t=bp/99:bp/99:bp;
    y1=cos(2*pi*f1*t);
```

Output:

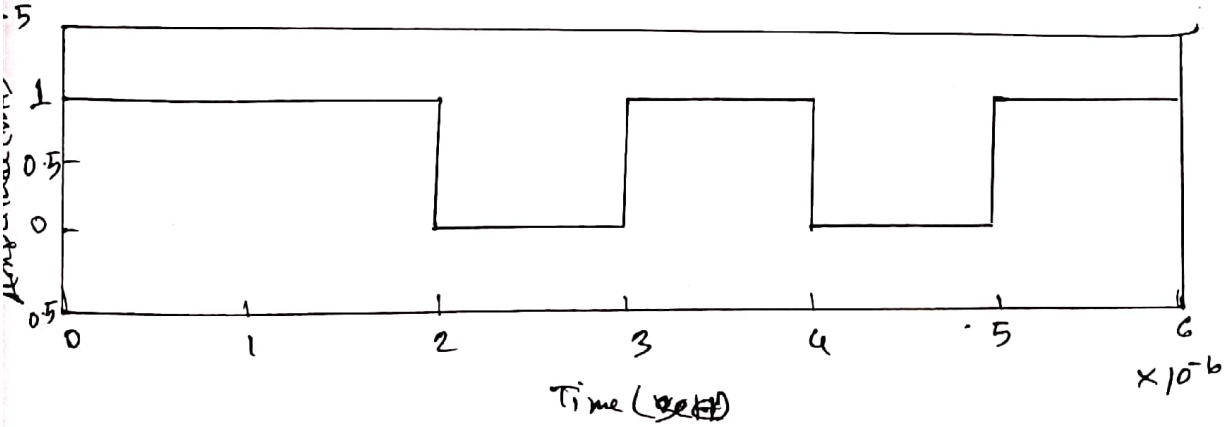
Transmitting information at digital signal.




FSK Modulation




FSK Demodulation.



## Experiment No: 08

**Experiment Name:** Write a MATLAB code to generate phase shift keying (PSK) Modulation and demodulation.

**Theory:**

**Generation of PSK signal:**

PSK is a digital modulation scheme that conveys data by changing, or modulating the phase of a reference signal (the carrier wave). PSK uses a finite number of phase, each assigni a unique pattern of binary digits. Usually, each phase encodes an equal number of bits. Each pattern of bits forms the symbol that is represented by the particular phase.

In BPSK system the pair of signal $S_1(t)$ and $S_2(t)$ is used to represent binary symbol 1 and 0 are defined by,

$$S_1(t) = \sqrt{2}E_b/T_b \cos 2\pi f_c t.$$

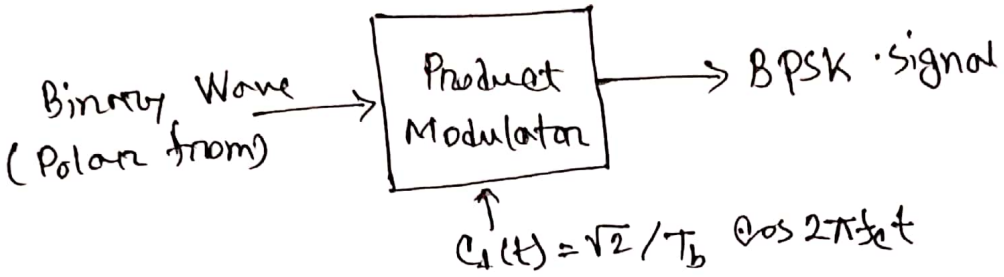
$$S_2(t) = \sqrt{2}E_b/T_b \cos(2\pi f_c t + \pi) \cdot \text{ where } 0 \le t < T_b$$

$E_b$ = Transmitted Energy for bit.

$$f_c = n/T_b$$

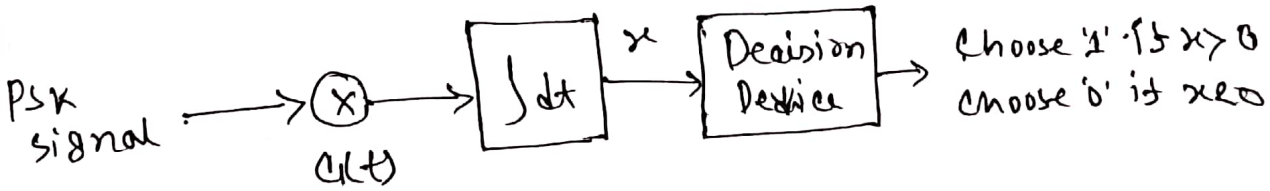
The pair of sinusoidal waves that differs only in a relative phase shift of 180° are called antipodal signals.

## BPSK Transmitter:




The input binary symbol are represented in polar form with symbols 1 and 0 represented by constant amplitude levels $\sqrt{E_b}$ and $-\sqrt{E_b}$. Thus binary wave is multiplied by a sinusoidal carrier in a product modulator. This result in a BPSK signal.

## BPSK Receiver:




The received BPSK signal is applied to a correlator which is also supplied with a locally generated reference signal $c_1(t)$. The correlator operation is compared with a threshold of zero volts. If $x > 0$ the receiver decids in favour of symbol '1'. If $x < 0$, it decides in favour of symbol 0.

## Algorithm:

### PSK Modulation:

Step-1 : Generate carrier signal.

Step-2 : Start FOR Loop.

Step-3 : Generate binary data, message signal in polar form.

Step-4 : Generate PSK modulated signal.

Step-5 : plot message signal and PSK Modulated signal.

Step-6 : End FOR loop.

Step-7 : plot the binary data and carrier.

### PSK Demodulation:

Step-1 : Start FOR loop.

Step-2 : Perform correlation of PSK signal with carrier to get decision variable.

Step-3 : Make decision to get demodulated binary data. If $x>0$ choose '1' else choose '0'.

Step-4 : End FOR loop.

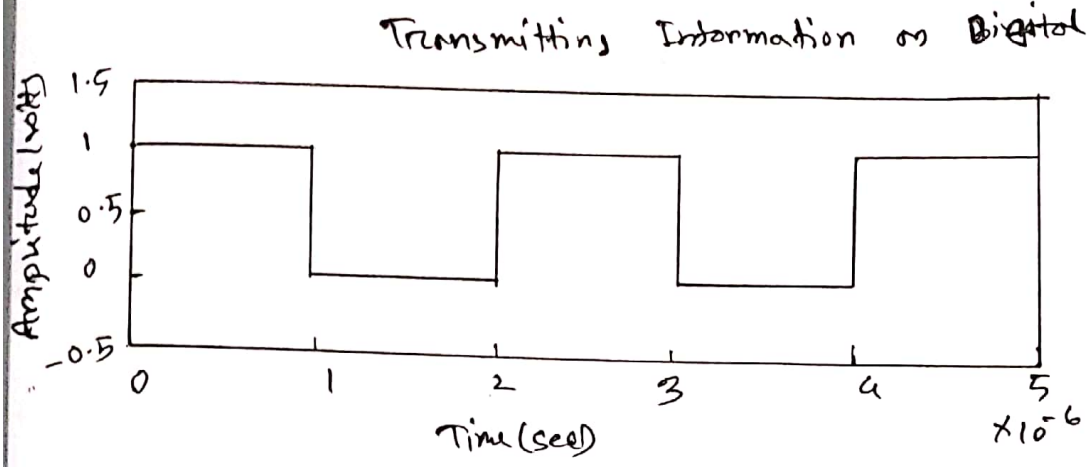
Step-5 : plot the demodulated binary data.

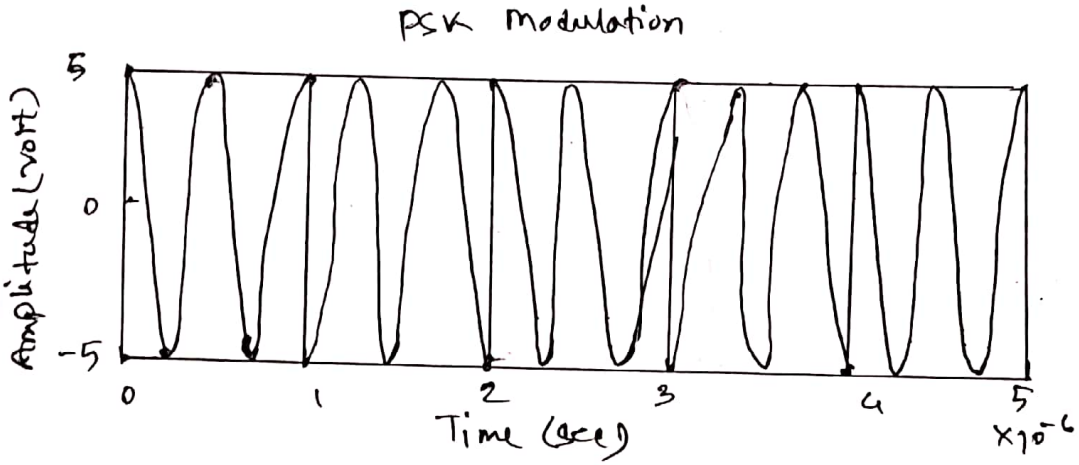
## Source code in MATLAB:

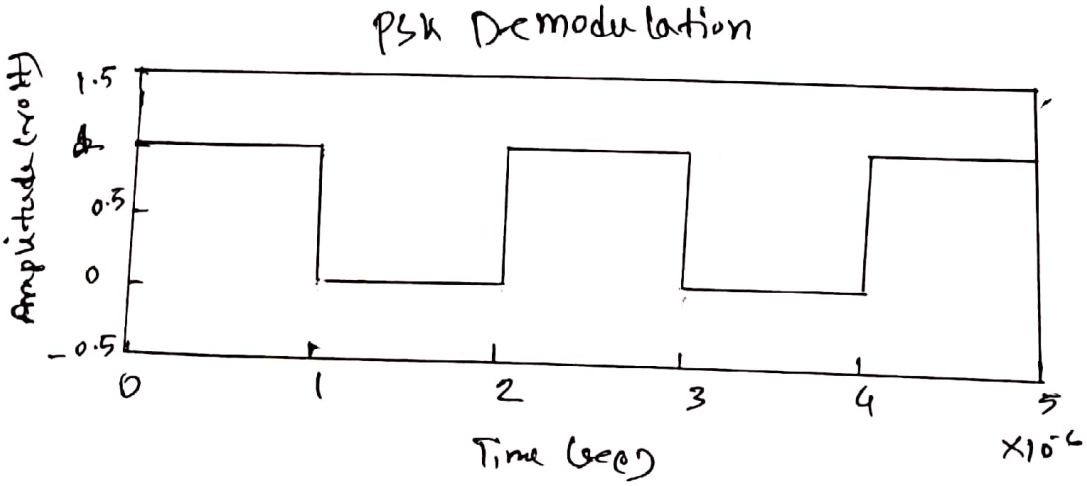
```
clc;
clear all;
close all;
x=[1 0 1 0 1 0 1 0 1];
bp=0.000001;
disp('Binary information at transmitter');
disp(x);

%representation of transmitting binary information as digital signal
bit=[];
for n=1:1:length(x)
    if x(n)==1;
        se=ones(1,100);
    else
        x(n)==0;
        se=zeros(1,100);
    end
    bit=[bit se];
end
t1=bp/100:bp/100:100*length(x)*(bp/100);
subplot(3,1,1);
plot(t1,bit,'linewidth',2.5);
grid on;
axis([0 bp*length(x) -0.5 1.5]);
ylabel('amplitude (volt)');
xlabel('Time(sec)');
title('Transmitting Information as Digital Signal');

%binary PSK Modulation
A=5;
br=1/bp;
f=br*2;
t2=bp/99:bp/99:bp;
ss=length(t2);
m=[];
for(i=1:1:length(x))
    if x(i)==1;
        y=A*cos(2*pi*f*t2);
    else
        y=A*cos(2*pi*f*t2+pi);      %Acos(2*pi*f*t+pi) means -Acos(2*pi*f*t)
    end
    m=[m y];
end
t3=bp/99:bp/99:bp*length(x);
subplot(3,1,2);
plot(t3,m);
grid on;
xlabel('Time(sec)');
ylabel('Amplitude(volt)');
title('Waveform for binary PSK modulation corresponding binary
information');

%binary PSK Demodulation
mn=[];
for n=ss:ss:length(m);
    t=bp/99:bp/99:bp;
    y=cos(2*pi*f*t);
    mm=y.*m((n-(ss-1)):n);
```

```
    t4=bp/99:bp/99:bp;
    z=trapz(t4,mm);
    zz=round((2*z/bp));
    if(zz>0)
        a=1;
    else
        a=0;
    end
    mn=[mn a];
end
disp('binary information in receiver after PSK Demodulation');
disp(mn);

%representation of binary data into digital signal
bit=[];
for n=1:length(mn)
    if mn(n)==1;
        se=ones(1,100);
    else
        se=zeros(1,100);
    end
    bit=[bit se];
end
t4=bp/100:bp/100:100*length(mn)*(bp/100);
subplot(3,1,3);
plot(t4,bit,'linewidth',2.5);
grid on;
axis([0 bp*length(mn) -0.5 1.5]);
xlabel('Time(sec)');
ylabel('Amplitude(volt)');
title('received information as digital signal');
```

Output:


Transmitting Information on Digital signal
Amplitude (volt)
1.5
1
0.5
0
-0.5
0
1
2
3
4
5
Time (sec)
X10^-6


PSK Modulation
Amplitude (volt)
5
0
-5
0
1
2
3
4
5
Time (sec)
X10^-6


PSK Demodulation
Amplitude (volt)
1.5
1
0.5
0
-0.5
0
1
2
3
4
5
Time (sec)
X10^-6

Experiment No : 09

Experiment Name : Write a MATLAB Program to generate Quadrature Phase Shift Keying (QPSK) signal.

Theory :

Generation of Quadrature Phase Shift Keying (QPSK):

QPSK is also known as quaternary PSK, quadriphase PSK, 4-PSK or 4-QAM. It is a phase modulation technique that transmits two bits in four modulation states.

phase of the carrier takes on one of four equally spaced values such as $\pi/4$, $3\pi/4$, $5\pi/4$ and $7\pi/4$.

$$s_i(t) = \begin{cases} \sqrt{2}\, E/T_b \cos\{2\pi f_c t + (2i-1)\pi/4\} & 0 \le t \le T_b \\ 0 & \text{otherwise} \end{cases}$$

Where $i = 1, 2, 3, 4 \ldots$

$E$ = Tx signal energy per symbol

$T_b$ = symbol duration.

Each of the possible value of phase corresponds to a pair of bits called dibits.

Thus the gray encoded set of dibits: 10, 00, 01, 11

$$s_i(t) = \begin{cases} \sqrt{2E/T_b}\cos[(2i-1)\pi/4]\cdot\cos(2\pi f_c t) - \sqrt{2E/T_b}\cdot \\ \qquad \sin[(2i-1)\pi/4]\sin(2\pi f_c t);\ 0 \le t \le T_b \\ 0, \quad \text{Otherwise} \end{cases}$$

There are two orthonormal basis functions.

$$\phi_1(t) = \sqrt{2/T_b}\cos 2\pi f_c t;\ 0 \le t \le T_b$$

$$\phi_2(t) = \sqrt{2/T_b}\sin 2\pi f_c t;\ 0 \le t \le T_b$$

There are four message points:

| Input dibits | Phase of QPSK signal | Co-ordinates of message signals | |
|---|---|---|---|
| | | $s_1$ | $s_2$ |
| 10 | $\pi/4$ | $+\sqrt{E/2}$ | $-\sqrt{E/2}$ |
| 00 | $3\pi/4$ | $-\sqrt{E/2}$ | $-\sqrt{E/2}$ |
| 01 | $5\pi/4$ | $-\sqrt{E/2}$ | $+\sqrt{E/2}$ |
| 11 | $7\pi/4$ | $+\sqrt{E/2}$ | $+\sqrt{E/2}$ |

## QPSK Transmitter:

The input binary sequence b(t) is represented in polar form with symbol 1 and 0 represented as $+\sqrt{E/2}$ and $-\sqrt{E/2}$.

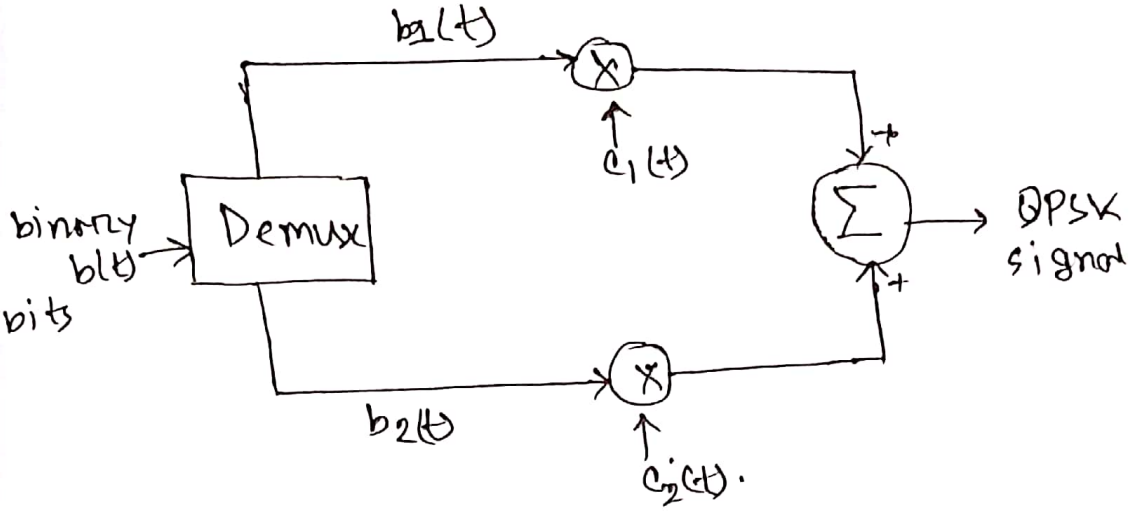



Figure- QPSK Transmitter

This binary wave is demultiplexed into two seperate binary waves consisting of odd and even numbered input bits denoted by $b_1(t)$ and $b_2(t)$. The $b_1(t)$ and $b_2(t)$ are used to modulate a pair of quadrature carrier. The result is two PSK waves. These two binary PSK waves are added to produce the desired QPSK signal.

## QPSK Receiver:

It consists of a pair of correlators with common input and supplied with locally generated signal $c_1(t)$ and $c_2(t)$.

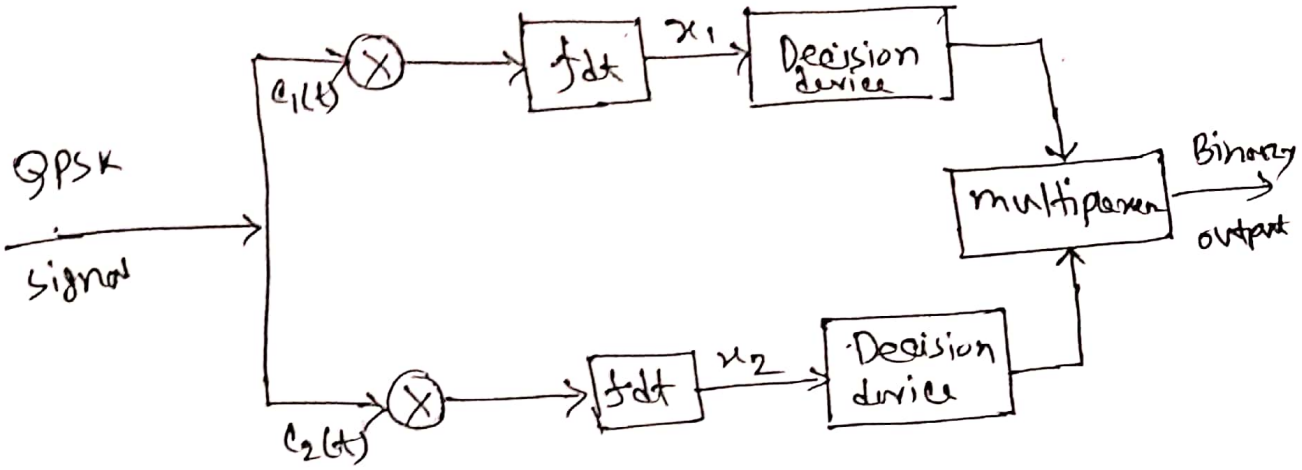



Figure – QPSK receiver.

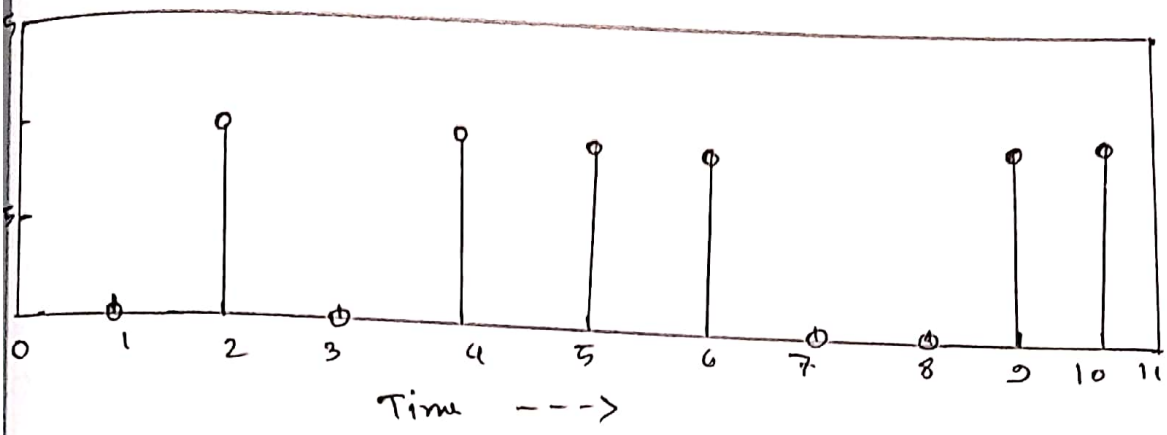
The correlator output $x_1$ and $x_2$ are each compared with a threshold of zero volts. If $x_1 > 0$; decision is made in favour of symbol '1'. Parallely if $x_2 > 0$ decision is made in favour of symbol 1 for lower channel and if $x_2 < 0$, decision is made in favour of symbol 0. These two channels are combined in a multiplexer to get the original binary output.

## Source code in MATLAB:

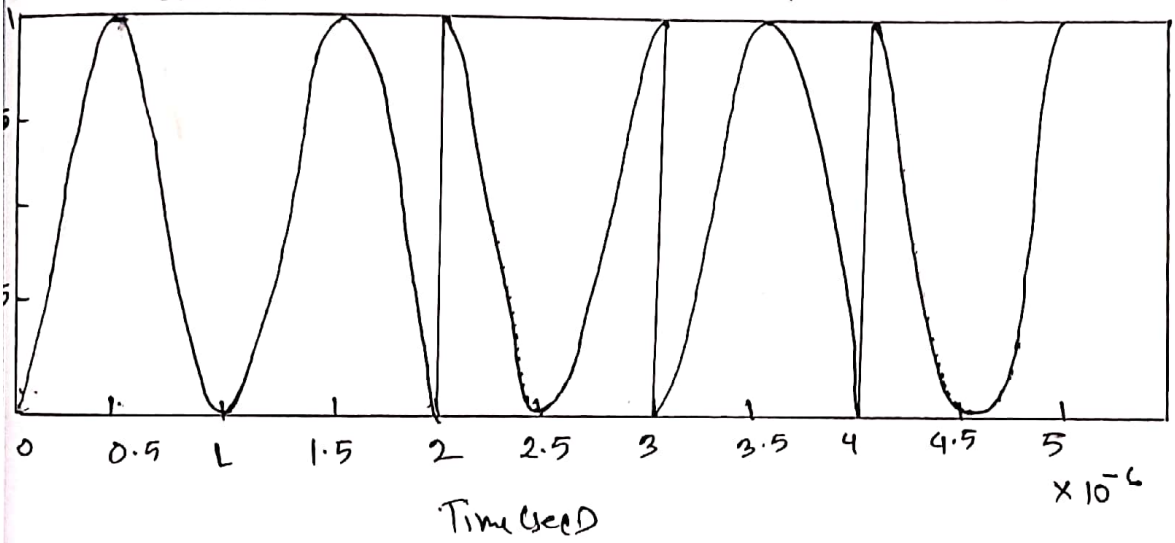
```
%QPSK waveform generation
clc;
clear all;
close all;
%x=[0 1 0 1]; %input bits
x=randi([0 1],1,10)
%Bits to polar
for i=1:length(x)
if x(i)==0
 p(i)=-1;
else
 p(i)=1;
end
end
%Seperation of even and odd sequences
even_seq=p(1:2:length(x));
odd_seq=p(2:2:length(x));
%NRZ polar line coder signal generation
i=1;
t=0:0.01:length(x);
m=2:2:length(x);
for j=1:length(t)
if t(j)<=m(i)
 even_ps(j)=even_seq(i);

else
 even_ps(j)=even_seq(i);
 i=i+1;
end
end
i=1;
m=2:2:length(x);
for j=1:length(t)
if t(j)<=m(i)
 odd_ps(j)=odd_seq(i);
else
 odd_ps(j)=odd_seq(i);
 i=i+1;
end
end
figure(1);
subplot(211);plot(t,even_ps,'r');
subplot(212);plot(t,odd_ps,'r');
%Carrier signals generation
c1=cos(2*pi*1*t);
c2=sin(2*pi*1*t);
figure(2);
subplot(211);plot(t,c1,'r');
subplot(212);plot(t,c2,'b');
%QPSK Wveform generation
r1=even_ps.*c1;
r2=odd_ps.*c2;
qpsk_sig=r1-r2;
figure(3);
subplot(311);plot(t,r1,'r');
subplot(312);plot(t,r2,'b');
subplot(313);plot(t,qpsk_sig,'b');
```

Output :

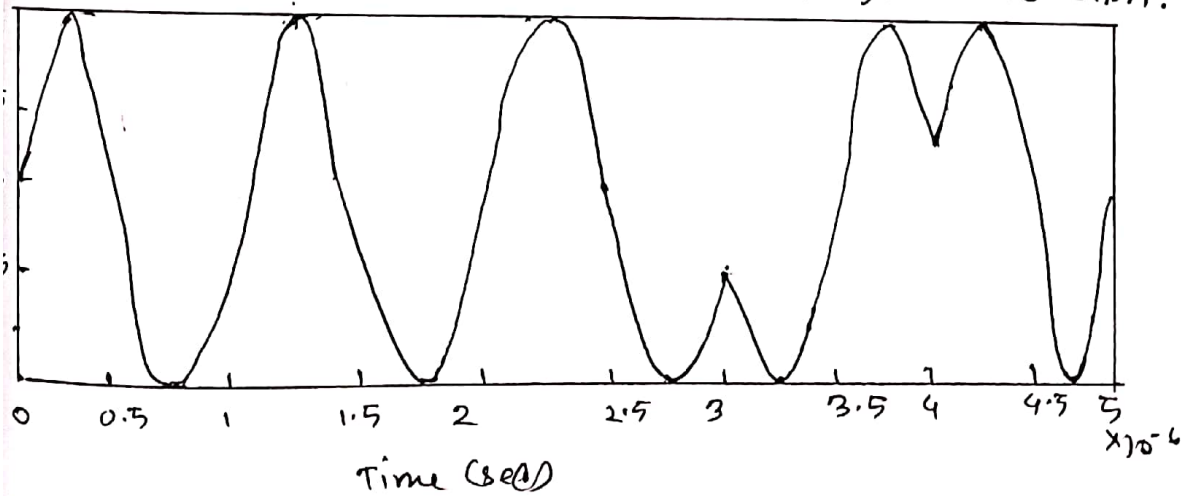
Information before transmitting



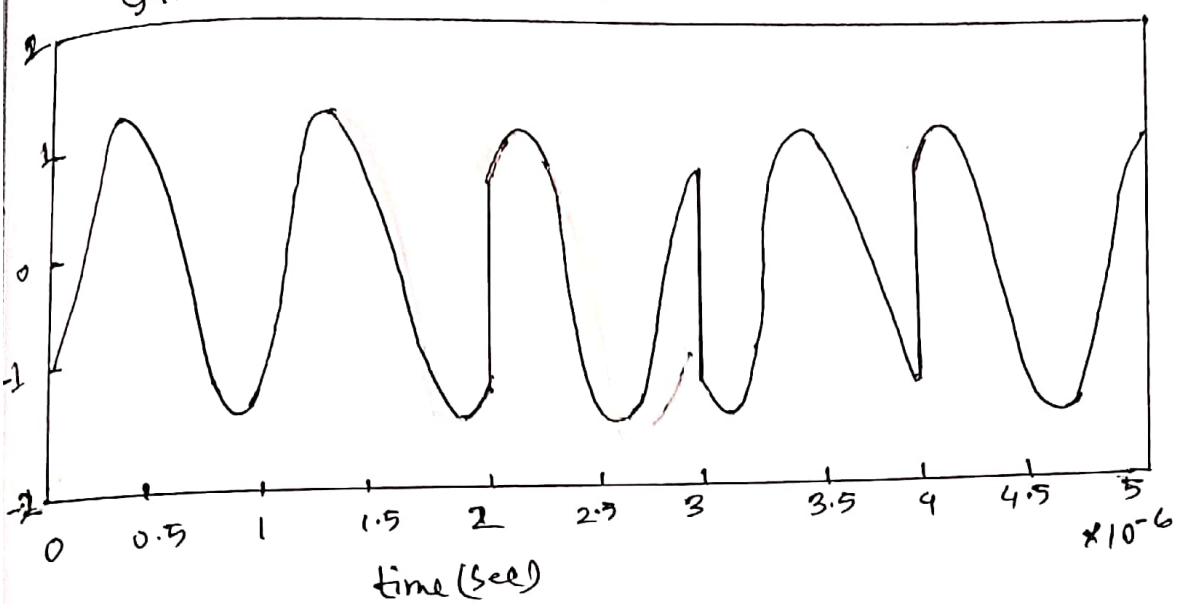

Waveform form for inphase component in QPSK modulation



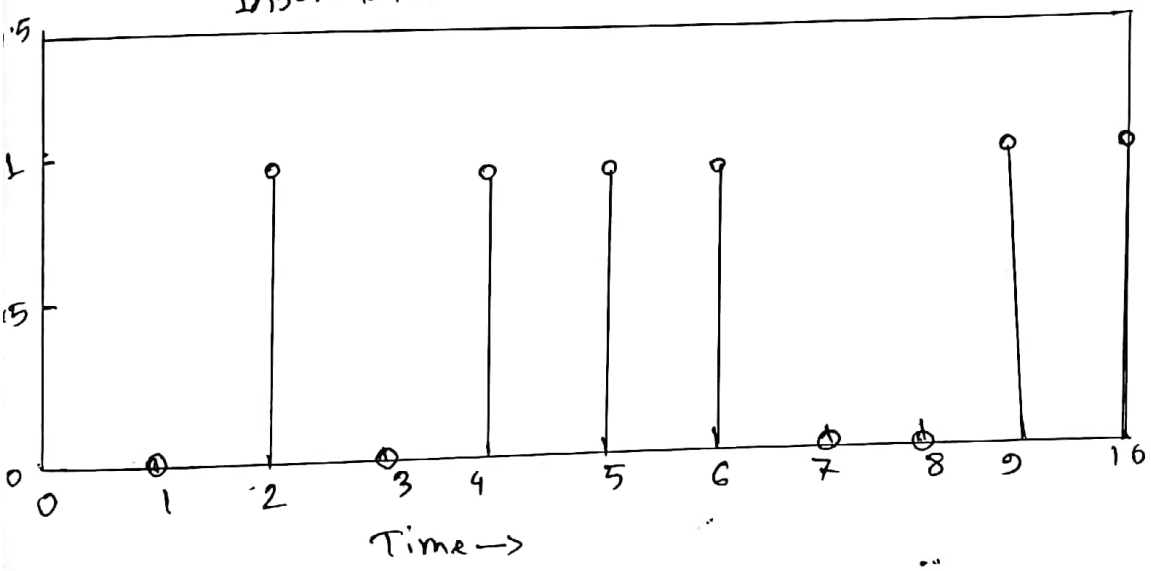

waveform for Quadrature component is QPSK modulation.



QPSK modulated signal (sum of inPhase and qpsignal)




Information after receiving

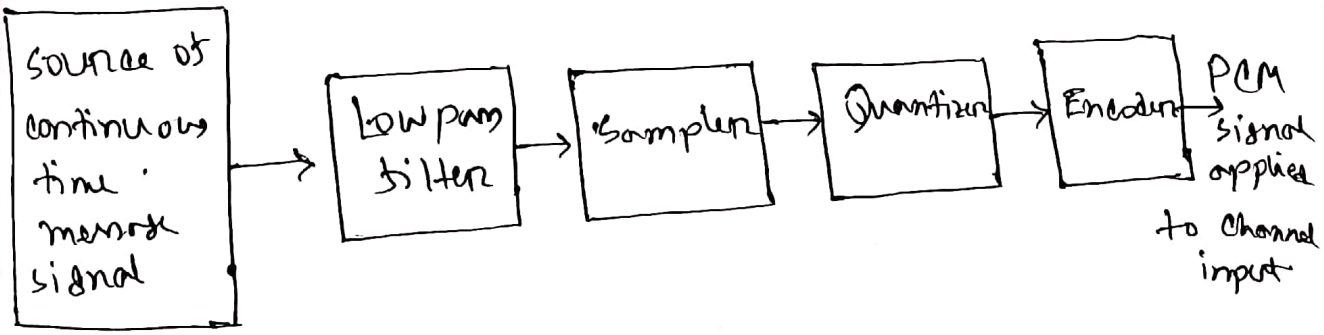


## Experiment No: 10

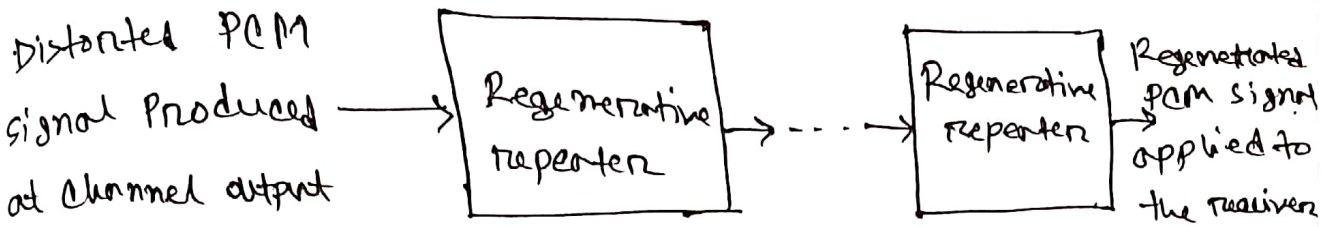
## Experiment Name: Write a MATLAB code for Pulse Code Modulation and Demodulation.

## Theory:

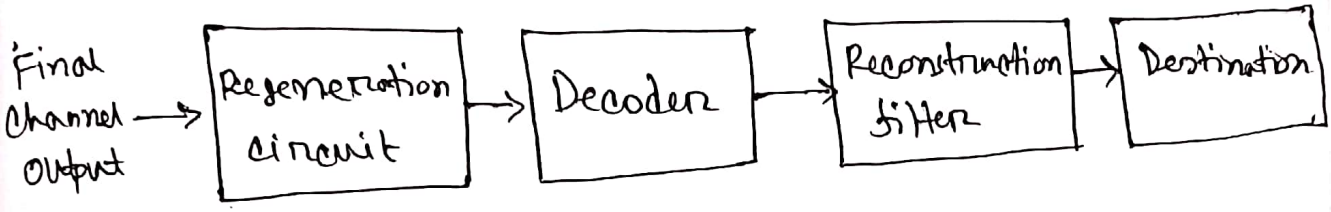
In pulse code Modulation (PCM) a message signal is represented by a sequence of coded pulses, which is accomplished by representing the signal in discrete form in both time and amplitude. The basic operation performed in the transmitter of a PCM system are sampling, quantizing and encoding as shown in figure -11.




(a) Transmitter.




(b) Transmission Path.




(c) Receiver.

Figure-11. The basic element of a PCM system.

In fig 11(a). the low-pass filter prior to sampling is included to prevent aliasing of message signal. The quantizing and encoding operation are usually performed in the same circuit which is called an analog-to-digital converter. The basic operations in the receiver are regeneration of impaired signals decoding, and reconstruction of the train of quantized samples. as shown in 11(c). Regeneration also occurs at intermediate point along the transmission path as necessary as in Figure 11(b).

Sampling: It is a process of measuring the amplitude of a continuous time signal at discrete instants. To ensure perfect reconstruction of msg signal at the receiver, the sampling rate must be greater than twice the highest frequency component w of message signal.

Quantization: The sampled version of the message signal is then quantized and provide a new representation of the signal that is discrete in both time and amplitude.

Encoding: The encoder encodes the quantized samples Each quantized sample is encoded into 8-bit code word by using A-law in encoding process.

Pulse code Demodulation:

Pulse code Demodulation will be doing the some modulation process in reverse. Demodulation starts with decoding process, during the transmission the PCM signal will affected by the noise interference. So before the PCM signal sends into PCM Demodulator we have to recover the signal into the original level for that we using a comparator. After decoding the reconstruction filter 'reconstruct the original signal and pass to destination.

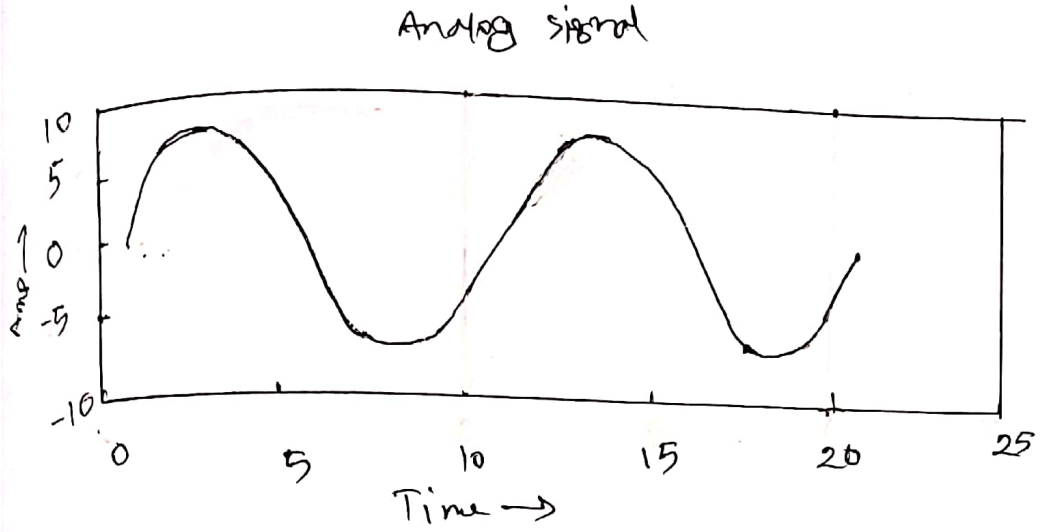
## Source code in MATLAB:

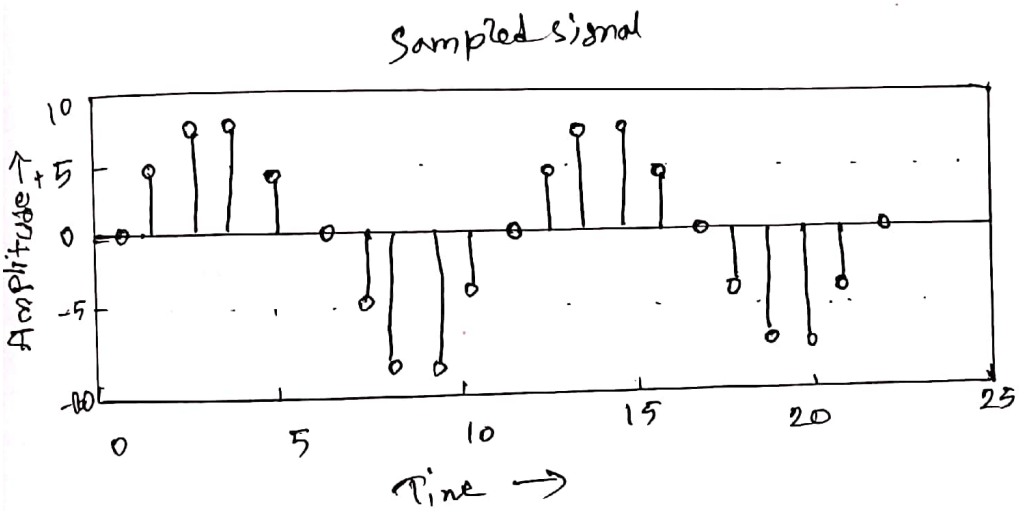
```
% Code for Pulse Code Modulation

clc;
close all;
clear all;
n=input('Enter n value for n-bit PCM system :  ');
n1=input('Enter number of samples in a period : ');
L=2^n;

% % Signal Generation
% x=0:1/100:4*pi;
% y=8*sin(x);                          % Amplitude Of signal is 8v
% subplot(2,2,1);
% plot(x,y);grid on;
% Sampling Operation
x=0:2*pi/n1:4*pi;                     % n1 nuber of samples have tobe selected
s=8*sin(x);
subplot(3,1,1);
plot(s);
title('Analog Signal');
ylabel('Amplitude--->');
xlabel('Time--->');
subplot(3,1,2);
stem(s);
grid on;
title('Sampled Sinal');
ylabel('Amplitude--->');
xlabel('Time--->');

 %  Quantization Process
 vmax=8;
 vmin=-vmax;
 del=(vmax-vmin)/L;
 part=vmin:del:vmax;
% level are between vmin and vmax with difference of del
 code=vmin-(del/2):del:vmax+(del/2);
 % Contaion Quantized valuses
 [ind,q]=quantiz(s,part,code);
% Quantization process

 % ind contain index number and q contain quantized  values
 l1=length(ind);
 l2=length(q);

 for i=1:l1
    if(ind(i)~=0)
% To make index as binary decimal so started from 0 to N
       ind(i)=ind(i)-1;
    end
    i=i+1;
 end
  for i=1:l2                                          % To make quantize
     if(q(i)==vmin-(del/2))
value inbetween the levels
        q(i)=vmin+(del/2);
     end
 end
 subplot(3,1,3);
```

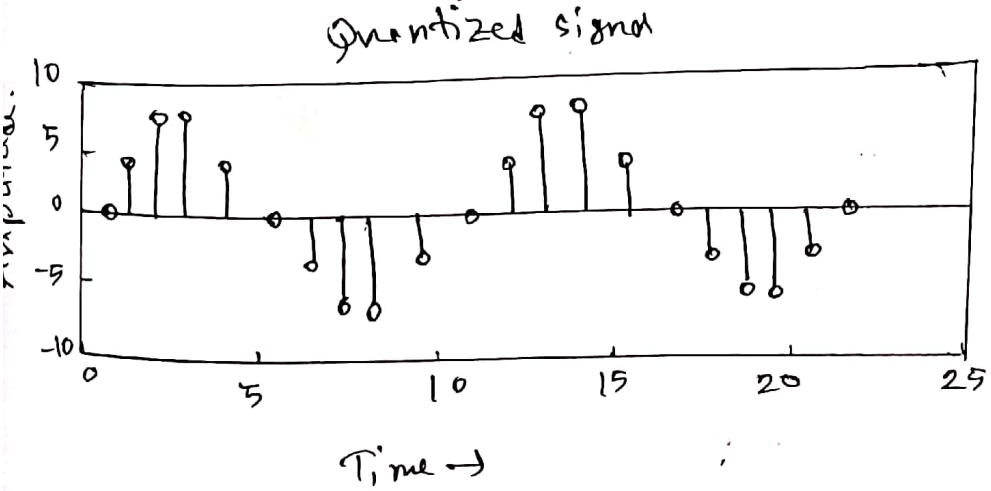
```
stem(q);grid on;                                        % Display the
Quantize values
title('Quantized Signal');
ylabel('Amplitude--->');
xlabel('Time--->');

% Encoding Process
figure
code=de2bi(ind,'left-msb');                % Cnvert the decimal to binary
k=1;
for i=1:11
    for j=1:n
        coded(k)=code(i,j);                % convert code matrix to a
coded row vector
        j=j+1;
        k=k+1;
    end
    i=i+1;
end
subplot(2,1,1); grid on;
stairs(coded);                                  % Display the encoded
signal
axis([0 100 -2 3]);  title('Encoded Signal');
ylabel('Amplitude--->');
xlabel('Time--->');

%   Demodulation Of PCM signal

qunt=reshape(coded,n,length(coded)/n);
index=bi2de(qunt','left-msb');                  % Getback the index in
decimal form
q=del*index+vmin+(del/2);                       % getback Quantized values
subplot(2,1,2); grid on;
plot(q);                                                % Plot
Demodulated signal
title('Demodulated Signal');
ylabel('Amplitude--->');
xlabel('Time--->')
```

Output:

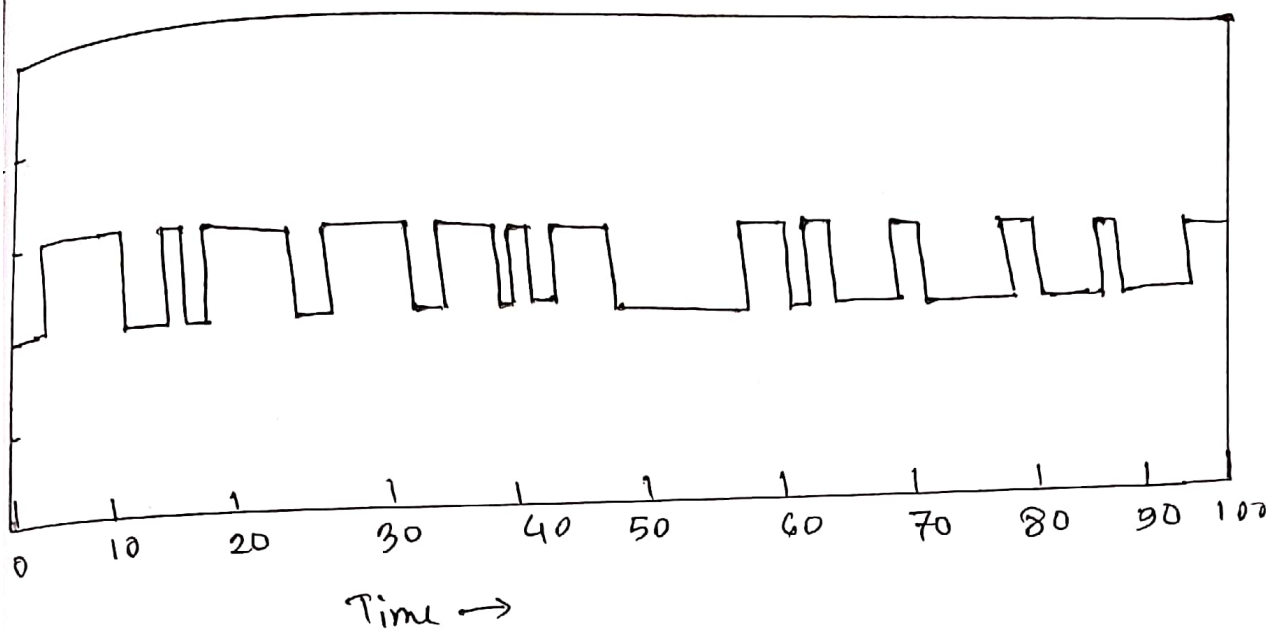
```
Enter n value for n-bit PCM system: 8
Enter numer of samples in a period: 10
```



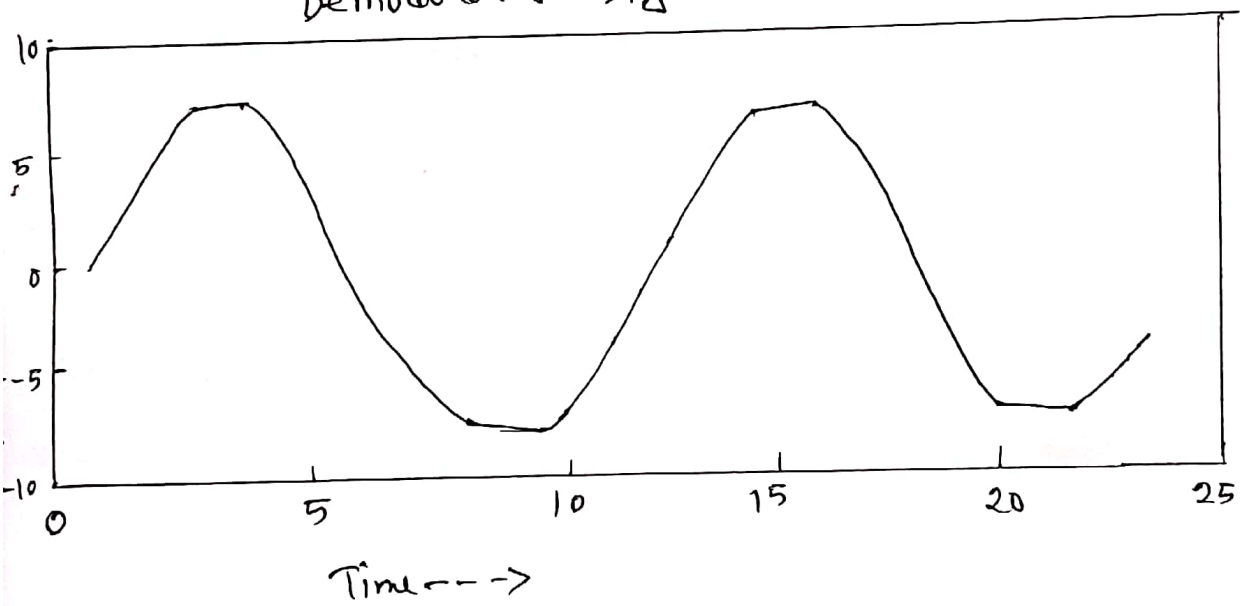



Encoded signal




Demodulated signal



## Experiment No: 11

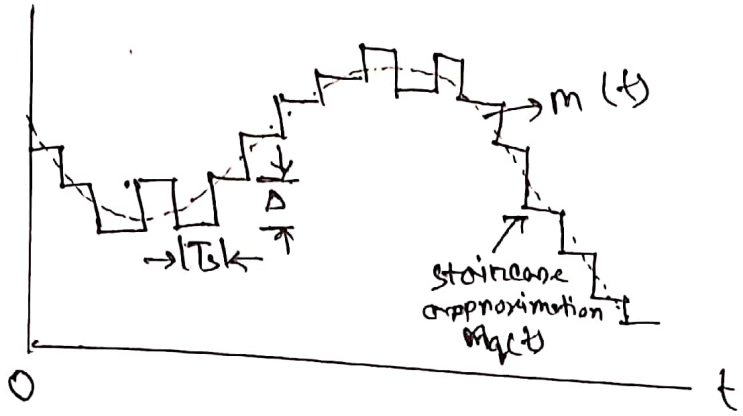
**Experiment Name:** Write a MATLAB code for Delta Modulation (DM) and Demodulation.

**Theory:**

**Delta Modulation:** In DM, an incoming message signal is over-sampled at a rate higher than the Nyquist rate to purposely increase the correlation between adjacent samples of the signal.

In its basic form, DM provides a staircase approximation to the oversampled version of the message signal as in Figure - 12.(a)

The difference between the input and the approximation is quantized into only two levels namely $\pm\Delta$ corresponding to positive and negative differences. If the approximation falls below the signal at any sampling epoch, it is increased by $\Delta$. If the approximation lies above the signal, it is decreased by $\Delta$.




(a)

Binary sequence at modulator output: 0 0 1 0 1 1 1 1 1 0 1 0 0 0

(b)

Figure -12: Illustration of Delta Modulation.

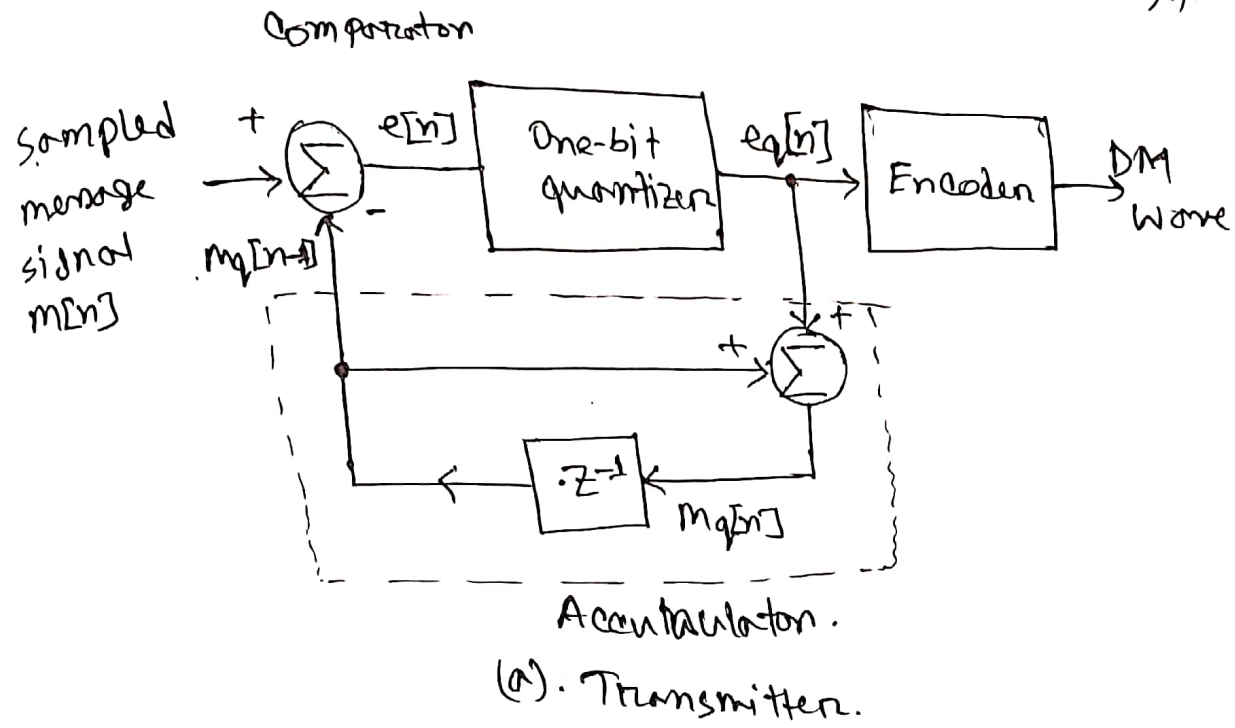
The principle of Delta Modulation is

$$e[n] = m[n] - m_q[n-1] \quad \text{---} (1)$$

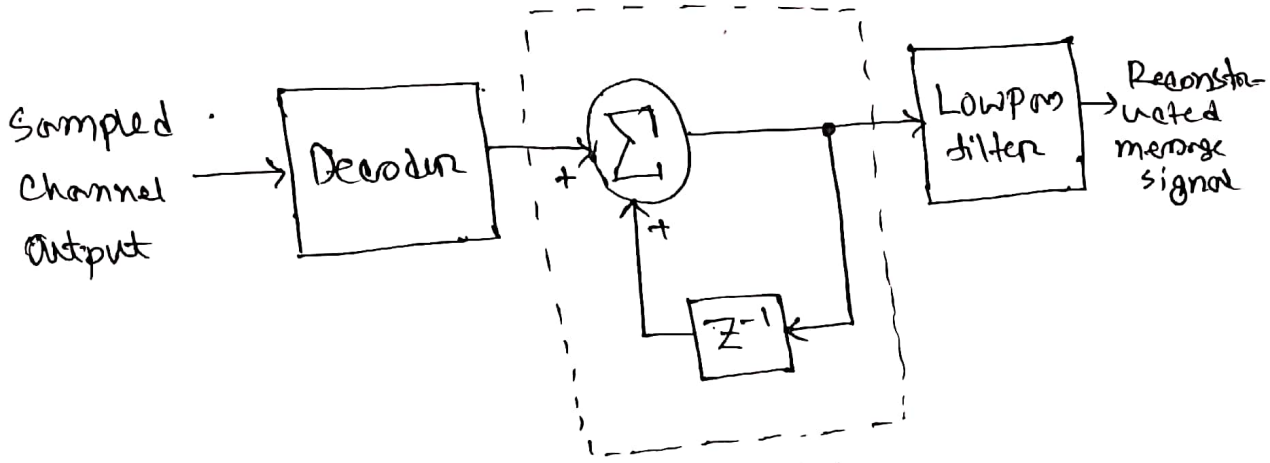
$$e_q = \Delta \, \text{sgn}\,(e[n]) \quad \text{---} (11)$$

$$m_q[n] = m_q[n-1] + e_q[n] \quad \text{---} (111)$$

Here $e[n]$ = error signal representing the difference between the present sample $m[n]$ of the input signal and the latest approximation $m_q[n-1]$ to it. $e_q[n]$ is the quantized version of $e[n]$. and $\text{sgn}(\cdot)$ is signum function. The quantizer output $m_q[n]$ is coded to produce the DM signal.




(a). Transmitter.




(b) Receiver.

Figure -13: DM System

Figure -13 shows the DM system. The DM signal is generated by applying the sampled version of the incoming message signal to a modulator that involves a comparator, quantizer and

accumulator interconnected in as shown in Figure-13(a).

The block labeled $z^{-1}$ inside the accumulator represents a unit delay that is a delay equal to one sampling period. Modulator follow directly from equation (i)-(iii). The comparator computes the difference between its two input. The quantizer output is then applied to an accumulator,

$$m_q[n] = \Delta \sum_{i=1}^{n} \text{sgn}(e[i])$$

$$= \sum_{i=1}^{n} e_q[i]$$

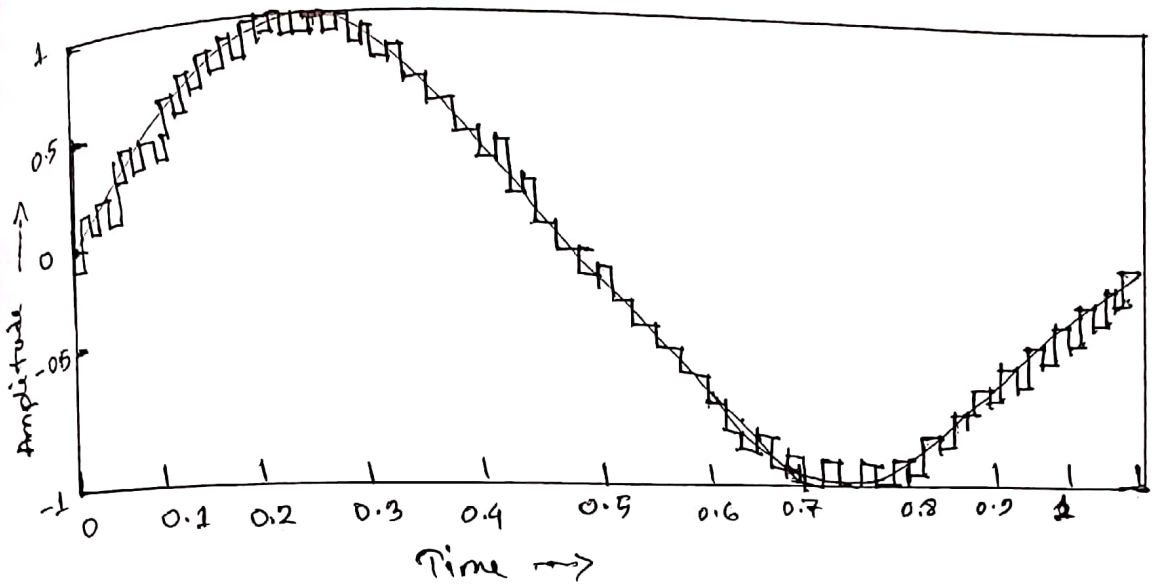
Delta demodulation is same as Modulation but in reverse order.

## Source code in MATLAB:

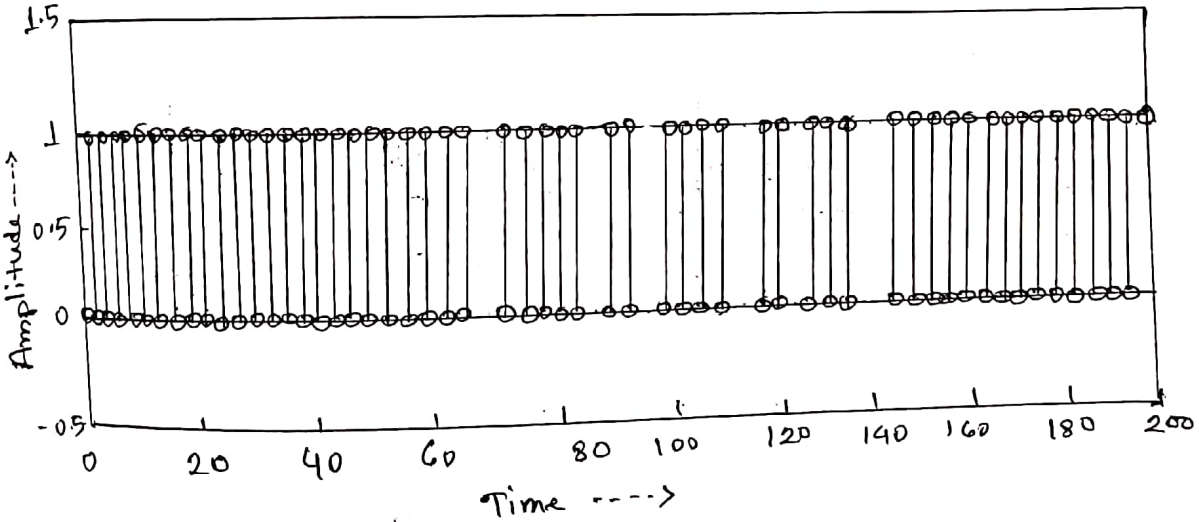
```
clc;
clear all;
close all;

%Signal Parameters
A=1;
fc=1;
t=linspace(0,1,1000);
x_t=A*sin(2*pi*fc*t);
s=.2;

%Delta Modulation
s_t=0;
v=1;
mqc_t=0;
for i=1:5:length(t)
  if(x_t(1,i)>mqc_t(1,v))
  s_t(1,v)=1;
  mqc_t(1,v+1)=mqc_t(1,v)+s;
else
 s_t(1,v)=0;
  mqc_t(1,v+1)=mqc_t(1,v)-s;
end
t1(1,v)=t(1,i);
v=v+1;
end

subplot(3,1,1) %first plotplot(t,x_t);
title('Message Signal');
xlabel('Time Axis');ylabel('Amplitude');
hold on
stairs(t1,mqc_t(2:201)) subplot(3,1,2) %second plot
stairs(s_t)
title('Delta Modulated Signal ');
xlabel('Time Axis');ylabel('Amplitude');
hold on
stem(s_t,'r')axis([0 200 -0.5 1.5])

%Demodulation
mq_t=0;
%Convert bit stream into steps and integrate
for i=1:length(s_t)
  if(s_t(i)==1)
  mq_t(i+1)=mq_t(i)+s;
else
  mq_t(i+1)=mq_t(i)-s;
end
end

%Filtering through LPF
[b,a]=butter(5,.1);
r_t=filter(b,a,mq_t);

subplot(3,1,3) %third plotplot(t1,r_t(2:201))
title('Demodulated Signal ');
xlabel('Time Axis');ylabel('Amplitude');
hold on;
```

Output:

Message Signal




Delta Modulated signal




Demodulated Signal