Experiment 1a:

Aim: To develop a MATLAB code for computation of Pathloss between the transmitter and receiver in wireless environment.

Theory:

The path loss, which represents signal attenuation as a positive quantity measured in dB, is defined as the difference between the effective transmitted power and the received power, and may or not include the antenna gains. The path loss for the free space model when antenna gains are included is given by

$$\mathbf{PL(dB)=10log(P_t/P_r)=-10log[G_tG_r\lambda^2/(4\pi)^2d^2]}$$

When antenna gains are excluded, the antennas are assumed to have unity gain, and path loss is given by

$$\mathbf{PL(dB)=10log(P_t/P_r)=-10log[\lambda^2/(4\pi)^2d^2]}$$

The Friis free space model is only a valid predictor for P_r for values of d which are in the far-field region of the transmitting antenna i.e the region beyond the far field distance df, which is related to the largest linear dimension(D) of the transmitting antenna aperture and the carrier wavelength.

$$\mathbf{d_f=2D^2/\lambda}$$

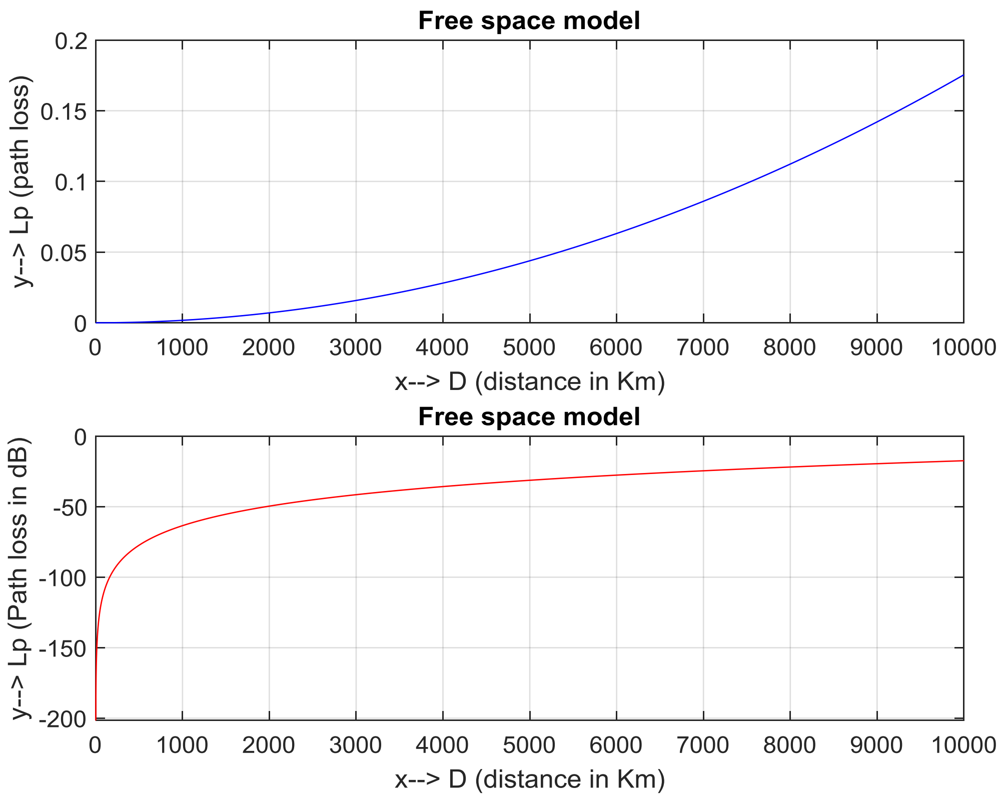
Program:

```
% This program calculate the Free Space Propagation Loss %with assumption of unity antenna gain for transmit and %receive antenna
clc;
close all;
clear all;
f = input('Enter carrier frequency:');
c = 3*10^8;
%Calculate Wavelength
lamda=c/f;
D=input('Enter the diameter of the Transmitting Antenna in meter:');
%calculate fraunhofer(far-field region) distance
df=2*D^2/lamda;
%Calculate pathloss when distance between the antennas is greater than df
d = 0:1:10000;
if d>=df
   Lp =(4*pi*d/lamda).^2;
else
   Lp=0;
end
subplot(2,1,1);
plot(d,Lp,'b');
xlabel('x--> d (distance in Km)');
ylabel('y--> Lp (path loss)');
title('Free space model');
```

```
grid on;
subplot(2,1,2);
plot(d,10*log(Lp),'r');
xlabel('x--> d (distance in Km)');
ylabel('y--> Lp (Path loss in dB)');
title('Free space model');
grid on;
```

Sample Input and Output:

```
Enter carrier frequency:5*10^6

Enter the diameter of the Transmitting Antenna in meter:2
```



Result:

Thus, the pathloss for the given transmitting antenna diameter and operating wavelength is calculated for various distance and graphs are plotted for Pathloss vs Distance.

Experiment 1b:

Aim: To write a MATLAB code for the computation of Link Budget with relevant plot.

Theory:

A link budget is an accounting of all of the power gains and losses that a communication signal experiences in a telecommunication system; from a transmitter, through a communication medium such as radio waves, cable, waveguide, or optical fiber, to the receiver. It is an equation giving the received power from the transmitter power, after the attenuation of the transmitted signal due to propagation, as well as the antenna gains and feedline and other losses, and amplification of the signal in the receiver or any repeaters it passes through. A link budget is a design aid, calculated during the design of a communication system to determine the received power, to ensure that the information is received intelligibly with an adequate signal-to-noise ratio. Randomly varying channel gains such as fading are taken into account by adding some margin depending on the anticipated severity of its effects. The amount of margin required can be reduced by the use of mitigating techniques such as antenna diversity or frequency hopping.

A simple link budget equation looks like this:

Received power (dB) = Transmitted power (dB) + gains (dB) − losses (dB)

Power gains and losses are usually expressed in decibels (dB), which is a logarithmic measurement, so adding decibels is equivalent to multiplying the actual power ratios.

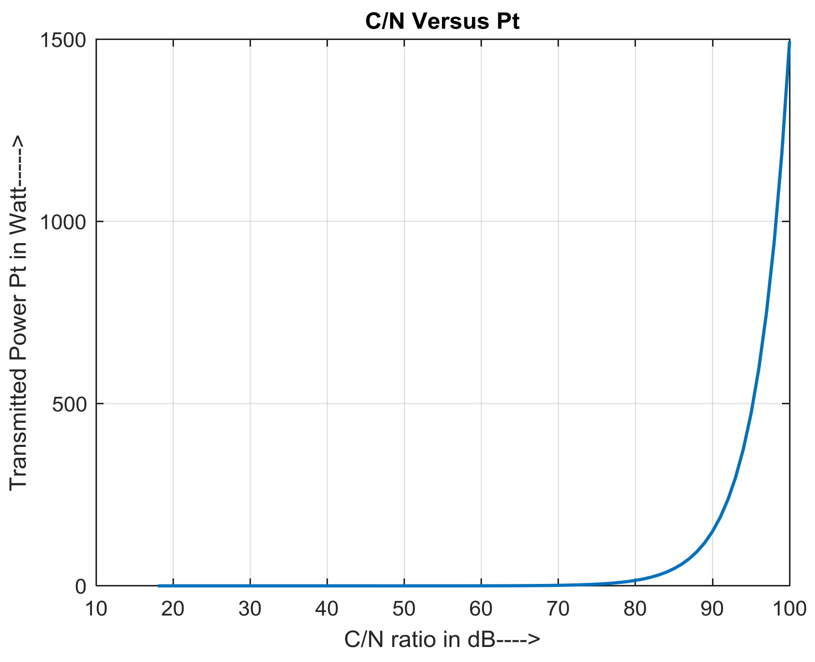
Program:

```
clc;
close all;
clear all;
R=input('Enter the distence between the transmitter and receiver in km');
D=input ('Enter the Antenna Diameter in meter');
Ae=input('Enter the Aperture Efficiency of the antenna, acceptable value in the range(0 to 1)');   %
Aperture Efficiency
f=input('Enter the Operating Frequency');
T=input('Enter system Noise Temperature in Kelvin');
CN=0:1:100;    %C/N required at receiver
c=3*10^8;
% link Noise Power Budget
k=-228.6;  % Boltzman Constant K = -228dBW/K/hz
Ts=10*log10(T);     % noise temp in dBK
B=10*log10(25*10^6);    % noise BW 25 Mhz in dBhz
N=k+Ts+B;    % Thermal Noise power N=kTsB in dBW
Pr=N+CN;   % received power must be  greater than noise power
% Link Power Budget
lamda=c/f; % operating wavelength
Gt=10*log10(Ae*(pi*D/lamda)^2);  %  Antenna gain
Lp=10*log10((4*pi*R/lamda)^2);   % pathlosss in dB
Gr=-31;     %  Receiving Antenna Gain in dB
Lp=-Lp;    % pathloss is negative
```

```
Margin=6;%power margin generally will be 6 dB to account for unexpected loss
Pt=Pr-(Gt+Gr+Lp)+Margin; % As all in dB so simply addition
Ptw =10.^(Pt/10);
plot(CNup,Ptw,'LineWidth',1.5);
xlabel('C/N ratio in dB---->');
ylabel('Transmitted Power Pt in Watt----->');
title('C/N Versus Pt');
grid on;
```

Sample Input and Output:

```
Enter the distance between the transmitter and receiver in km6

Enter the Antenna Diameter2

Enter the Aperture Efficiency of the antenna, acceptable value in the range (0 to 1)0.5

Enter the Operating Frequency2000000

Enter Minimum Required C/N at Receiver in dB:18

Enter system Noise Temperature in Kelvin300
```



Result:

Thus, the Link Budget calculation is done for the given data and plot between the minimum power needed at transmitter for required carrier to noise ratio at the receiver is plotted.

Experiment 2:

Aim: To develop a MATLAB code to represent the different channel models for wireless networks.

Theory:

Propagation models have traditionally focused on predicting the average received signal strength at a given distance from the transmitter, as well as the variability of the signal strength in close spatial proximity to a particular location. Propagation models that predict the mean signal strength for an arbitrary transmitter-receiver (T-R) separation distance are useful in estimating the radio coverage area of a transmitter and are called large-scale propagation models, since they characterize signal strength over large T-R separation distances (several hundreds or thousands of meters). On the other hand, propagation models that characterize the rapid fluctuations of the received signal strength over very short travel distances (a few wavelengths) or short time durations (on the order of seconds) are called small-scale or fading models.

1.Free Space Propagation Model
The free space propagation model is used to predict received signal strength when the transmitter and receiver have a clear, unobstructed line-of-sight path between them. Satellite communication systems and microwave line-of-sight radio links typically undergo free space propagation.

$$\mathbf{Pr(dB)=10logPt(watts)+10log(Gt)+10log(Gr)-20log(4\pi d/\lambda)}$$

2.Ground Reflection (2-ray) Model

The 2-ray ground reflection model is a useful propagation model that is based on geometric optics, and considers both the direct path and a ground reflected propagation path between transmitter and receiver. This node has been found to be reasonably accurate for predicting the large-scale signal strength over distances of several kilometers for mobile radio systems that use tall towers (heights which exceed 50 m), as well as for line-of-sight microcell channels in urban environments

$$P_{r=}P_tG_tG_r(h_t^2h_r^2)/\lambda^4$$

$$PL(db)=40\log d-(10\log G_t+10\log G_r+20\log h_t+20\log h_r)$$

$$Pr(dB)=Pt(dB)-PL(dB)$$

3.Okumura Model
Okumura's model is one of the most widely used models for signal prediction in urban areas. This model is applicable for frequencies in the range 150 MHz to 1920 MHz (although it is typically extrapolated up to 3000 MHz) and distances of 1 km to 100 km. It can be used for base station antenna heights ranging from 30 m to 1000 m.

$$L_{50}(dB)=L_F+A_{mu}(f,d)-G(h_{te})-G(h_{re})-G_{AREA}$$

$G(h_{te})=20\log(h_{te})/200$ $\quad$ $1000\text{ m} > h_{te} > 30\text{ m}$

$G(_{hre})=10\log(h_{re})/3$ $\quad$ $h_{re}\leq 3\text{m}$

$$G(h_{re})=20\log(h_{re})/3 \qquad 10\ m > h_{re} > 3\ m$$

$$Pr(dB)=Pt(dB)-L(dB)$$

4. Hata Model

The Hata model is an empirical formulation of the graphical path loss data provided by Okumura, and is valid from 150 MHz to 1500 MHz. Hata presented the urban area propagation loss as a standard formula and supplied correction equations for application to other situations. The standard formula for median path loss in urban areas is given by

$$L_{50}(urban)(dB)=69.55+26.16\log f_c -13.82\log h_{te} - a(h_{re})+(44.9 - 6.55\log h_{te}) \log d$$

For a small to medium sized city, the mobile antenna correction factor is given by

$$a(h_{re})=(1.1\log f_c - 0.7)h_{re} - (1.56\log f_c - 0.8)dB$$

and for a large city, it is given by

$$a(h_{re})=8.29(\log 1.54h_{re})^2 - 1.1\ dB \quad for \quad f_c \leq 300\ MHz$$

$$a(h_{re})=3.2(\log 11.75h_{re})^2 - 4.97\ dB \quad for \quad f_c \geq 300\ MHz$$

To obtain the path loss in a suburban area the standard Hata formula is modified as

$$L_{50}(dB) = L_{50}(urban) -2 [\log(f_c / 28)]^2 -5.4$$

and for path loss in open rural areas, the formula is modified as

$$L50(dB) = L50(urban) – 4.78 (\log fc)2 – 18.33 \log fc – 40.98$$

$$Pr(dB)=Pt(dB)-L(dB)$$

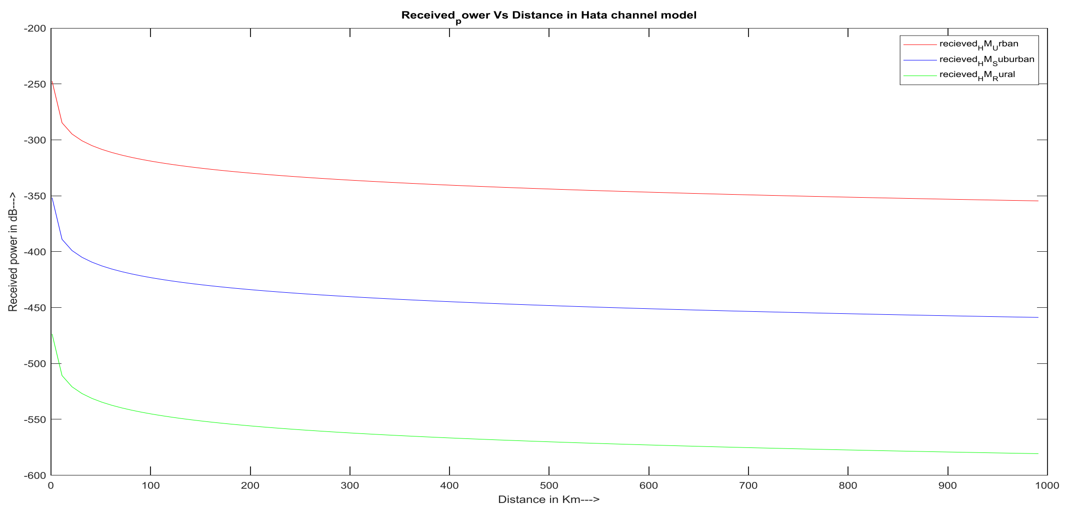
Program:

```
clc;
clear all;
close all;
 d=1:10:1000;

while(1)

   Model_Type=input('Enter Propagation model Type:-1:Free space model\n 2:Two-ray model\n 
3:Okumura model\n 4: Hata model\n 5:exit\n');

   if Model_Type == 1
      fc=input('Enter the operating frequency');
      Gt=input('Enter the transmitting antenna gain');
      Gr=input('Enter the receiving antenna gain');
      pt=input('Enter the Transmitted Power in watts');
      lamda=3*10^8/fc;
      recieved_power_FSM=10*log10(pt)+10*log10(Gt*Gr)-20*log10((4*pi*d/lamda));
      figure(1);
      plot(d,recieved_power_FSM);
      title(‘Free space channel model');
      ylabel('Received power in dB--->');
      xlabel('Distance in Km--->');

   elseif Model_Type == 2
     fc=input('Enter the operating frequency');
     Gt=input('Enter the transmitting antenna gain');
     Gr=input('Enter the receiving antenna gain');
     pt=input('Enter the Transmitted Power in watts');
     ht=input('Enter the hight of transmitting antenna');
     hr=input('Enter the hight of receiving antenna');

     received_power_TRM=10*log10(pt)+10*log10(Gt*Gr)+20*log10(ht*hr)-40*log10(d);
     figure(2);
     plot(d,received_power_TRM);
     title('Two-Ray channel model');
     ylabel('Received power in dB--->');
     xlabel('Distance in Km--->');

   elseif Model_Type == 3
      fc=input('Enter the operating frequency(Range:150-1500MHz)');
      Gt=input('Enter the transmitting antenna gain');
      Gr=input('Enter the receiving antenna gain');
      pt=input('Enter the Transmitted Power in watts');
      ht=input('Enter the hight of transmitting antenna(range applicable:30-100m)');
      hr=input('Enter the hight of receiving antenna(applicable range:3-10m)');
      A=input('Enter the median attenuation(Applicable range: 15-60dB'));
      GAREA=input('Enter the Gain from Environment(applicable range:5-30dB)');
      lamda=3*10^8/fc;
```

```
        Ght=20*log10(ht/200);
       if hr<=3
          Ghr=10*log10(hr/3);
       else
          Ghr=20*log10(hr/3);
       end
       received_power_OKM=10*log10(pt)+Ght+Ghr+10*log10(GAREA)-
20*log10((4*pi*d/lamda))-A;
       figure(3);
       plot(d,received_OKM);
       title('Received_power Vs Distance in Okumura channel model');
       ylabel('Received power in dB--->');
       xlabel('Distance in Km--->');

    elseif Model_Type==4
        fc=input('Enter the operating frequency(Range:150-1500MHz)');
        Gt=input('Enter the transmitting antenna gain');
        Gr=input('Enter the receiving antenna gain');
        pt=input('Enter the Transmitted Power in watts');
        ht=input('Enter the hight of transmitting antenna');
        hr=input('Enter the hight of receiving antenna');
        K=input('Enter the value of K(35.94 (countryside) to 40.94 (desert)');
        ahr=3.2*(log10(11.75*hr))^2-4.97;
        received_HM_Urban=10*log10(pt)+10*log10(Gt)+10*log10(Gr)-(69.55+26.16*log10(fc)-
13.82*log10(ht)-ahr+(44.9-6.55*log10(ht))*log10(d));
        received_HM_Suburban=received_HM_Urban-2*(log10(fc/28))^2-5.4;
        received_HM_Rural=received_HM_Urban-4.78*(log10(fc))^2 + 18.33*log10(fc)-K;
        figure(4);
        plot(d,received_HM_Urban,'r',d,received_HM_Suburban,'b',d,received_HM_Rural,'g');
        title('Received_power Vs Distance in Hata channel model');
        ylabel('Received power in dB--->');
        xlabel('Distance in Km--->');
        legend('recieved_HM_Urban','recieved_HM_Suburban','recieved_HM_Rural');

    elseif Model_Type==5
        break;
    else

        disp('Enter the right choice');
    end
end
```

Sample Input/Output:

```
Enter Propagation model Type:
 1:Free space model
 2:Two-ray model
 3:Okumura model
 4:Hata model
 5:exit
4
Enter the operating frequency(Range:150-1500MHz)300e6
Enter the transmitting antenna gain0.9
Enter the receiving antenna gain0.7
Enter the Transmitted Power in watts60
Enter the hight of transmitting antenna25
Enter the hight of receiving antenna10
Enter the value of K(35.94 (countryside) to 40.94 (desert)38
Enter Propagation model Type:
 1:Free space model
 2:Two-ray model
 3:Okumura model
 4:Hata model
 5:exit
5
```



Result:

Thus, the MATLAB Code for Chennel models is developed and Characteristics of the channel models for different values of distance are compared.

Experiment 3:

Aim: Analysis of Cellular concepts like cell-sectoring, splitting (using Qualnet /NS3/ any other tool)

Theory:

Cell splitting is a method of subdividing cell into the smaller sized cell. ... Cell sectoring is another method to increase capacity. It keeps the radius of the cell constant and decreases the co-channel reuse ratio D/R to reduce the cluster size N.

Diagram…..

Program:

```
clc;
clear all;
H =2/60;
GOS =0.01;
C1 =57;
A =44.2;
calls1 =1326;
C2 =C1 /3;
calls2 =336;
Ns1 =3;
capacity = Ns1 * calls2 ;
dif= calls1 - capacity ;
percentdif =( dif/ calls1 )*100;
printf('\n cell capacity of uinsectored system=0.0%f',calls1);
printf('\n cell capacity of 120 degree sectored system=0.0%f calls/hours',capacity);
printf('\n decrease in cell capacity of 120 degree sectored system=0.0%f calls/hours',percentdif);
```

Result:

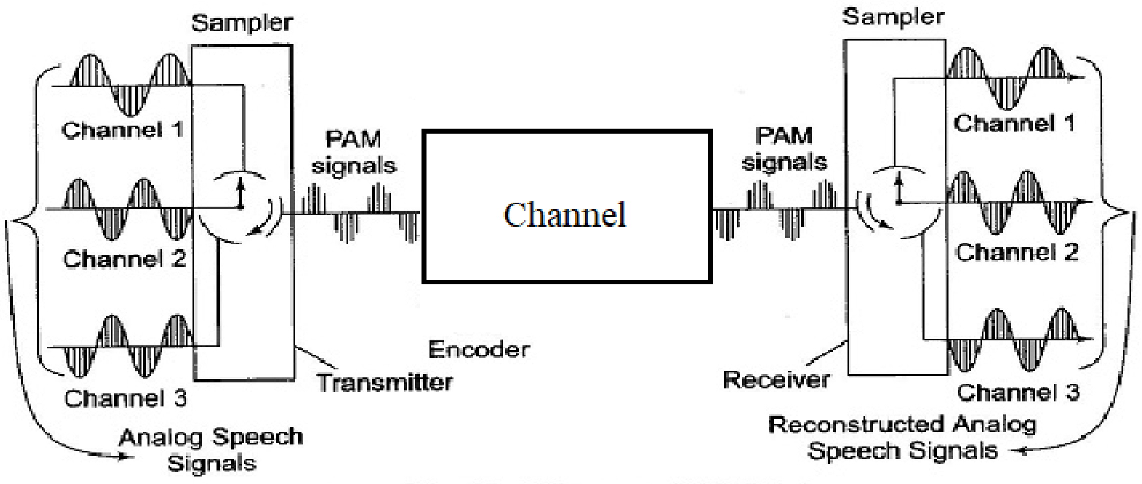
```
calls1 =1326
capacity=1008
percentdif = 23.9819
```

Experiment 4:

Aim: To demonstrate Time-Division Multiplexing and de-multiplexing techniques using MATLAB with necessary waveforms in time and frequency domain.

Theory:

Time-division multiplexing (**TDM**) is a method of transmitting and receiving independent signals over a common signal path by means of synchronized switches at each end of the transmission line so that each signal appears on the line only a fraction of time in an alternating pattern. This method transmits two or more digital signals or analog signals over a common channel. It can be used when the bit rate of the transmission medium exceeds that of the signal to be transmitted. Time-division multiplexing is used primarily for digital signals, but may be applied in analog multiplexing in which two or more signals or bit streams are transferred appearing simultaneously as sub-channels in one communication channel, but are physically taking turns on the channel.The time domain is divided into several recurrent *time slots* of fixed length, one for each sub-channel. A sample byte or data block of sub-channel 1 is transmitted during time slot 1, sub-channel 2 during time slot 2, etc. One TDM frame consists of one time slot per sub-channel plus a synchronization channel and sometimes error correction channel before the synchronization. After the last sub-channel, error correction, and synchronization, the cycle starts all over again with a new frame, starting with the second sample, byte or data block from sub-channel 1, etc.



Simplified Diagram of TDM System

Program:

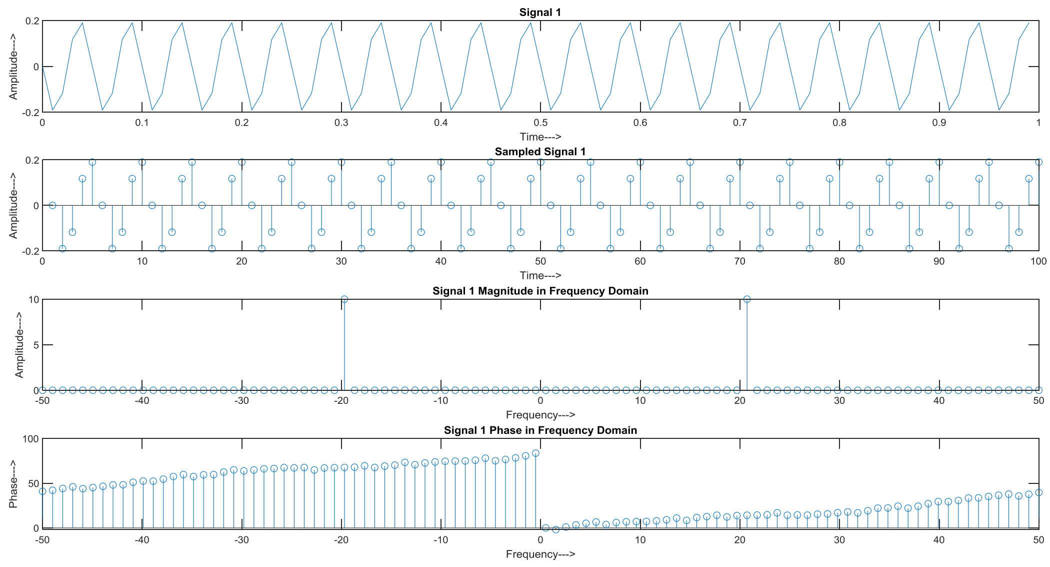
```
clc;
close all;
clear all;
% Signal generation
fs=input('Enter the sampling frequency in Hz:');
%am=input('Enter the ampitude of the signal');
t=0:1/fs:1-(1/fs);
Freq = linspace(-fs/2, fs/2, numel(t));
sig1=0.9*sin(2*pi*100*t)-0.2*sin(2*pi*20*t);      % Generate 1st signal
%l=length(sig1);
f1=150;
sig2=0.5*sin(2*pi*f1*t)+0.9*sin(2*pi*130*t);      % Generate 2nd Sigal
fft_sig1=fft(sig1);
fft_sig2=fft(sig2);
% Display of sinusoidal Signal
figure(1);
subplot(4,1,1);
plot(t,sig1);
title('Signal 1');
ylabel('Amplitude--->');
xlabel('Time--->');
subplot(4,1,2);
stem(sig1);
title('Sampled Signal 1');
ylabel('Amplitude--->');
xlabel('Time--->');
subplot(4,1,3);
stem(Freq,fftshift(abs(fft_sig1)));           % Magnitude of Fourier
title('Signal 1 Magnitude in Frequency Domain');
ylabel('Amplitude--->');
xlabel('Frequency--->');
subplot(4,1,4);
stem(Freq,fftshift(unwrap(angle(fft_sig1))));    % Phase of Fourier
title('Signal 1 Phase in Frequency Domain');
ylabel('Phase--->');
xlabel('Frequency--->');
figure(2);
subplot(4,1,1);
plot(t,sig2);
title('Signal 2');
ylabel('Amplitude--->');
xlabel('Time--->');
subplot(4,1,2);
stem(sig2);
title('Sampled Signal 2');
ylabel('Amplitude--->');
xlabel('Time--->');
subplot(4,1,3);
```

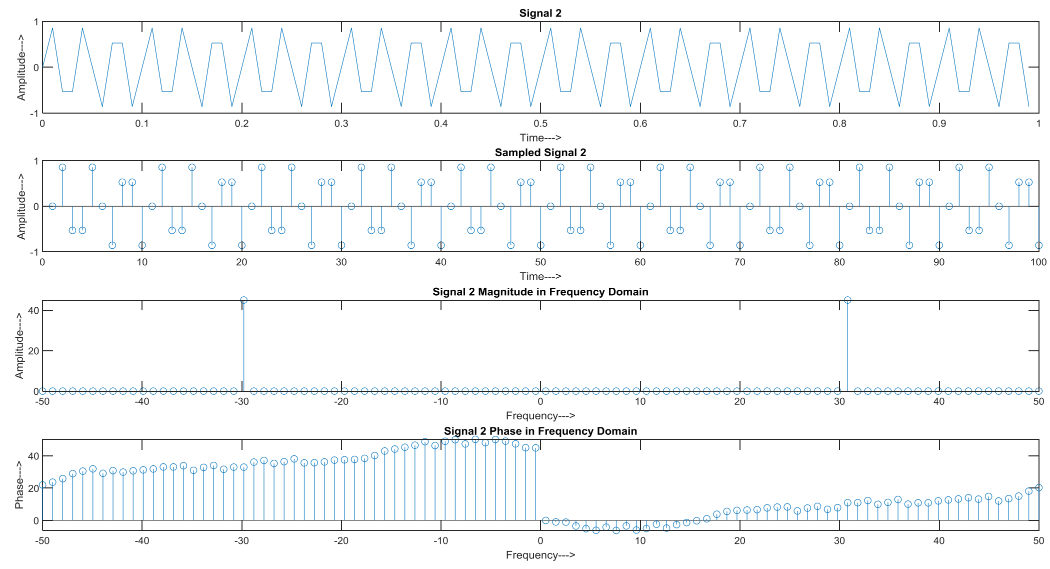
```
stem(Freq,fftshift(abs(fft_sig2)));           % Magnitude of Fourier
title('Signal 2 Magnitude in Frequency Domain');
ylabel('Amplitude--->');
xlabel('Frequency--->');
subplot(4,1,4);
stem(Freq,fftshift(unwrap(angle(fft_sig2))));    % Phase of Fourier
title('Signal 2 Phase in Frequency Domain');
ylabel('Phase--->');
xlabel('Frequency--->');
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
fft_tdmsig=fft(tdmsig);
figure(3);
subplot(3,1,1);
stem(tdmsig);
title('TDM Signal');
ylabel('Amplitude--->');
xlabel('Time--->');
subplot(3,1,2);
stem(fftshift(abs(fft_tdmsig)));           % Magnitude of Fourier
title('TDM signal Magnitude in Frequency Domain');
ylabel('Amplitude--->');
xlabel('Frequency--->');
subplot(3,1,3);
stem(fftshift(unwrap(angle(fft_tdmsig))));
title('TDM signal Phase in Frequency Domain');
ylabel('Phase--->');
xlabel('Frequency--->');
% 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(4);
 subplot(2,1,1);
 plot(sig3);
 title('Recovered  Signal 1');
 ylabel('Amplitude--->');
 xlabel('Time--->');
 subplot(2,1,2);
```

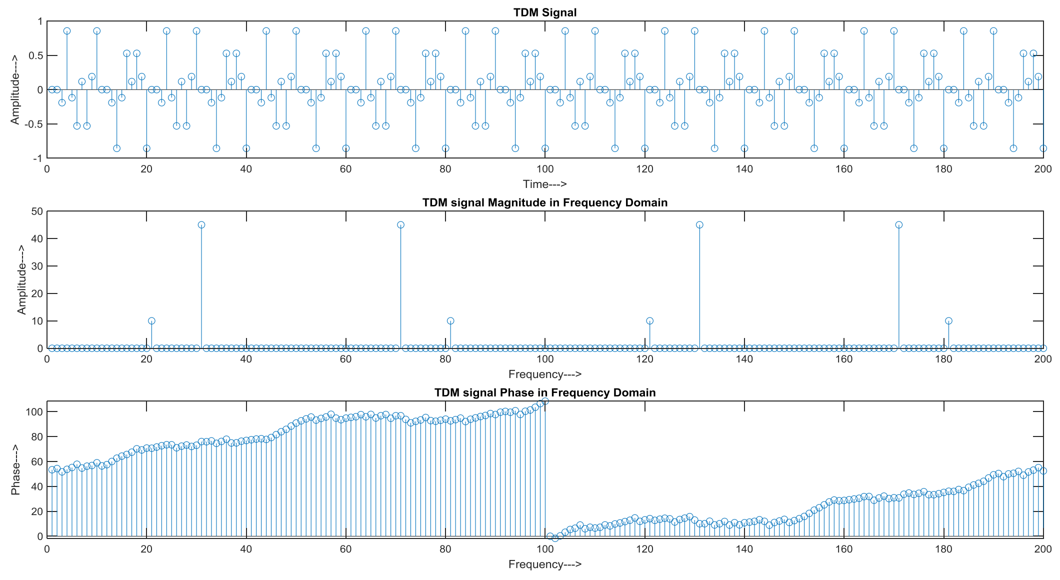
```
plot(sig4);
title('Recovered Signal 2');
ylabel('Amplitude--->');
xlabel('Time--->');
```

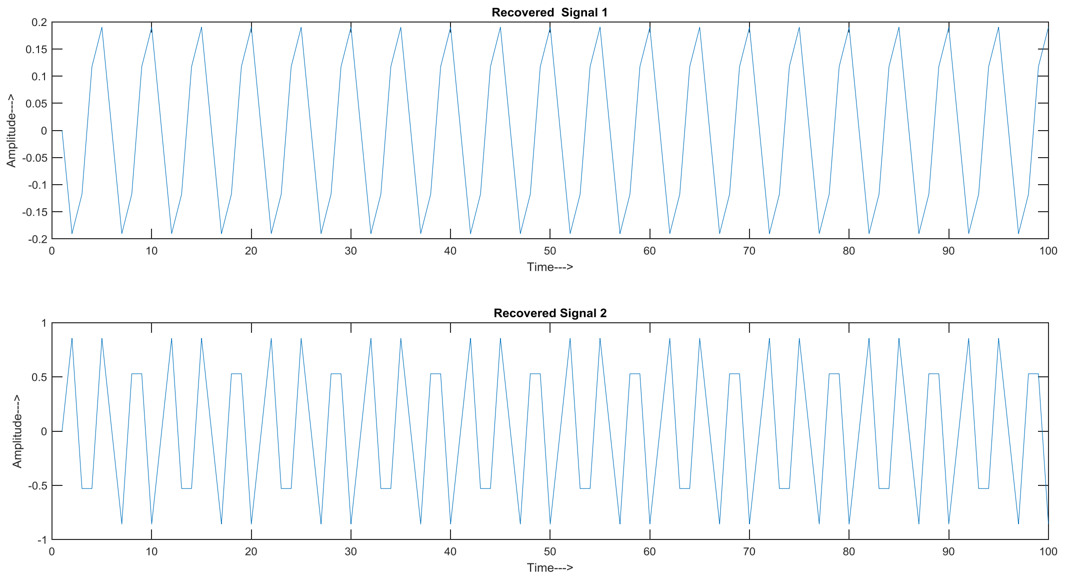
Sample Input/Output:

Enter the sampling frequency in Hz:100









Result:

Thus, the demonstration of Time Division Multiplexing and de-multiplexing are done using MATLAB and verified with necessary waveforms.

Experiment 5:

Aim: To generate T1 Carrier bit stream using MATLAB

Theory:

T1 digital carrier system is a North American digital multiplexing standard since 1963. T1 stands for transmission one and specifies a digital carrier system using PCM encoded analog signal. A T1 carrier system is time division multiplexes PCM encoded samples from 24 voice band channels for transmission over a single metallic wire pair or optical fiber transmission line. Each voice band channel has BW around 300Hz to 3000KHz.

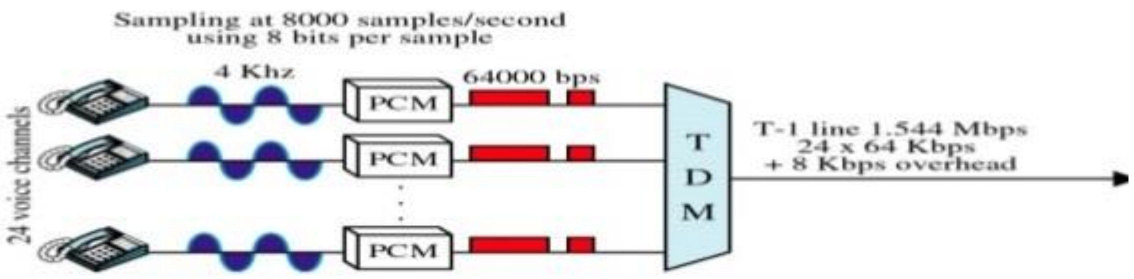


Fig1 : T1 digital system

A multiplexer is simply a digital switch with 24 independent inputs and one time division multiplexed output. The PCM output signals from 24 voice band channels are sequentially selected and connected through the multiplexer to the transmission line. With T1 carrier system, there is sampling, encoding and multiplexing of 24 voice band channels. Each channel contains an 8-bit PCM code and sampled 8000 times a second. Each channel is sampled at same rate but not at same time. The figure shows that, each channel is sampled once in each frame, but not at same time. Each channel's sample is offset from previous channel's sample by 1/24 of total frame time. Therefore one 64Kbps PCM encoded sample is transmitted for each voice band channel during each frame. The line Speed is calculated as:

$$\frac{24\ Channels}{frame} \times \frac{8Bits}{Channel} = 192\text{bits per frame}$$

$$Thus\ \frac{192\ bits}{frame} \times \frac{8000\ frames}{second} = 1.536Mbps$$

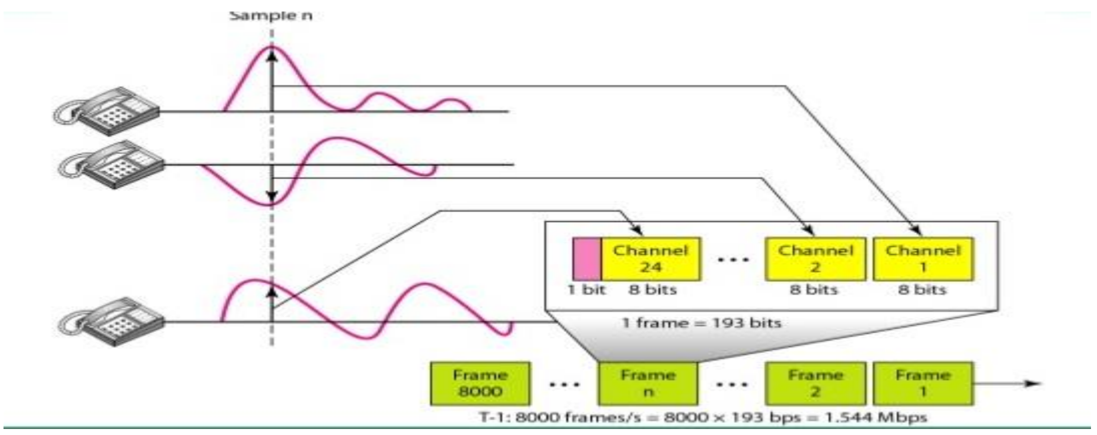


Fig2 : T1 frame structure

An additional bit (called framing bit) is added to each frame. The framing bit occurs once per frame (8000bps rate) and recovered in receiver, where it is used to maintain frame and sample synchronization between TDM transmitter and receiver. So each frame contains 193 bits and line speed for T1 digital carrier system is

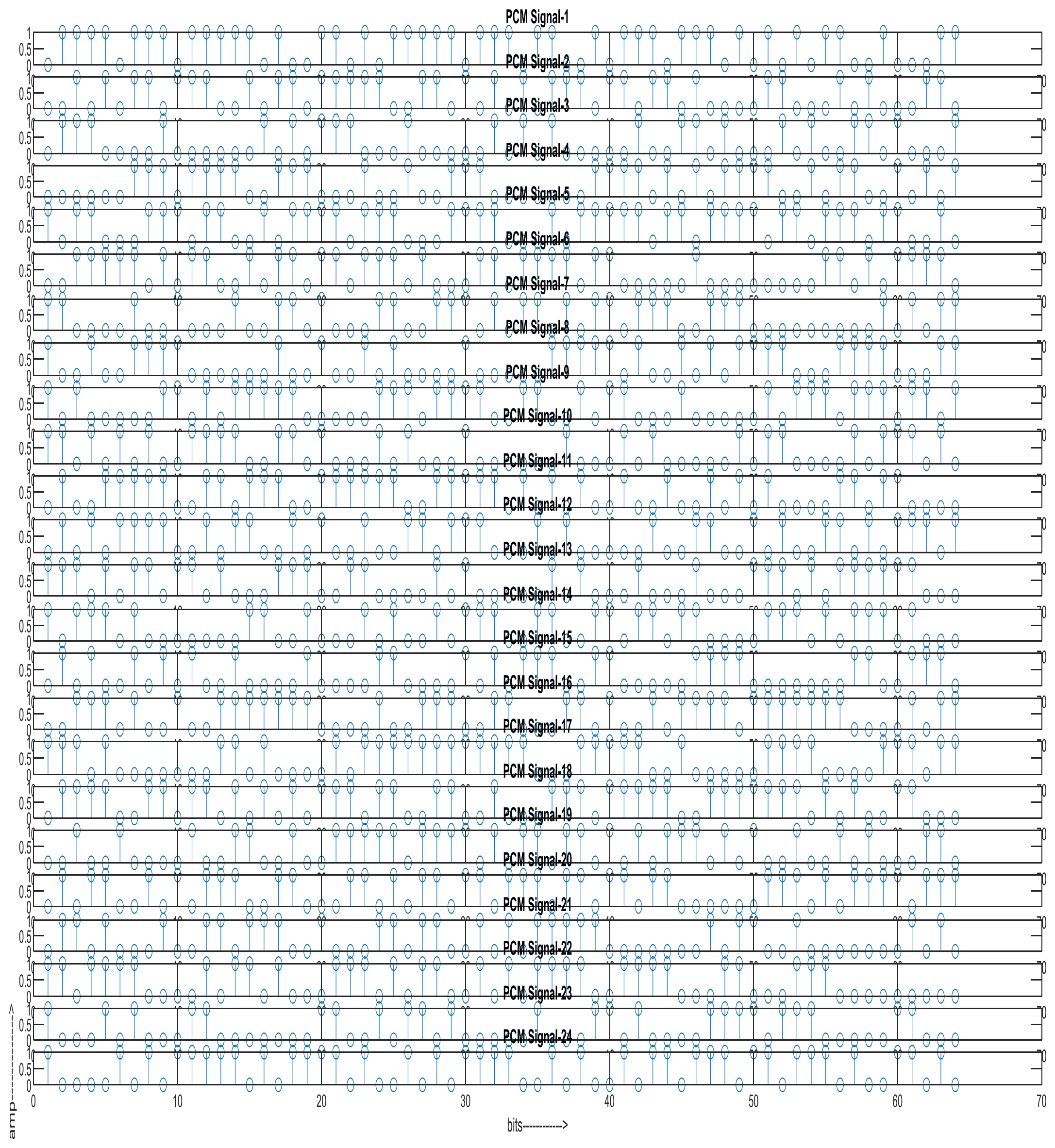
$$\frac{193\ bits}{frame} \times \frac{8000\ frames}{second} = 1.544Mbps$$

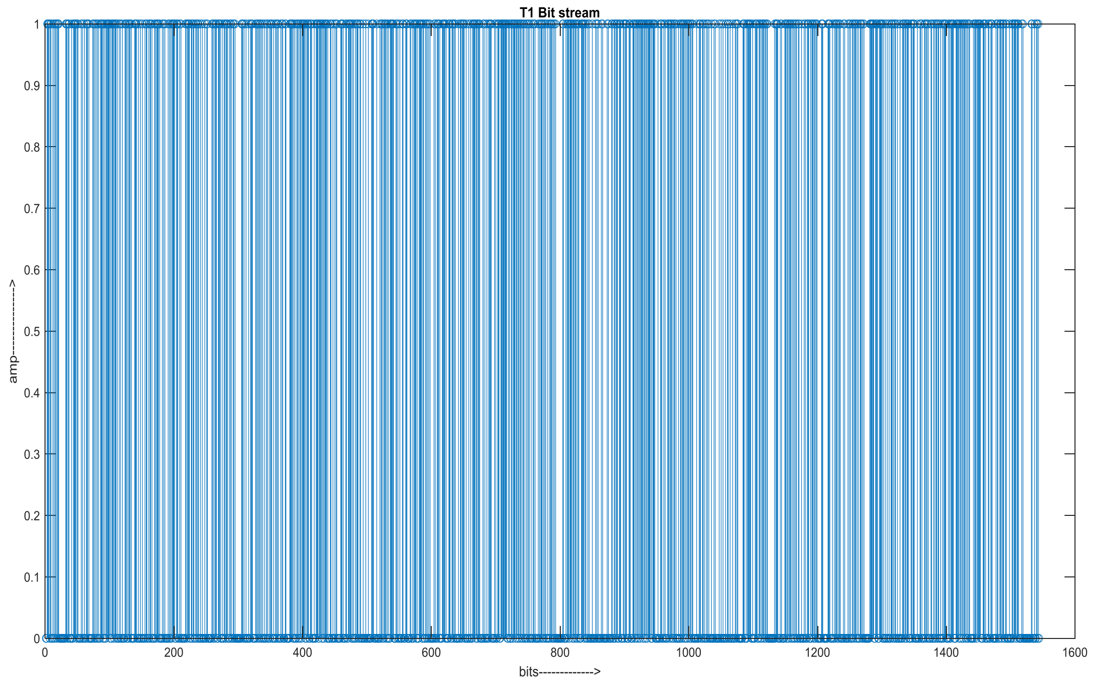
Program:

```
clc;
clear all;
close all;
%PCM of 24 signals
No_samples=input('Enter no of samples to be considered/signal for display:');
Total_BitsperSignal=No_samples*8;%each sample represented in 8 bits
PCM_signals=randi([0,1],24,Total_BitsperSignal);
figure(1);
for i=1:24
   subplot(24,1,i);
   stem(PCM_signals(i,:));
   title(sprintf('PCM Signal-%d',i));
end
   xlabel('bits------------>');
   ylabel('                    amp------------->                    ');

%Multiplxing of 24 PCM signals
T1frame=[];
for i=1:8:Total_BitsperSignal
   multiplexed_signal=[];
   for j=1:24
      multiplexed_signal=[multiplexed_signal PCM_signals(j,i:i+7)];
   end
   T1frame=[T1frame 0 multiplexed_signal ];
end
figure(2);
   stem(T1frame);
   title('T1 Bit stream');
   xlabel('bits------------->');
   ylabel('amp-------------->');
```

Sample Input/Output:

Enter no of samples to be considered/signal for display: 8





Result:

Thus, the MATLAB code is developed for the generation of the T1 bit stream and verified with the necessary waveforms.

Experiment 6:

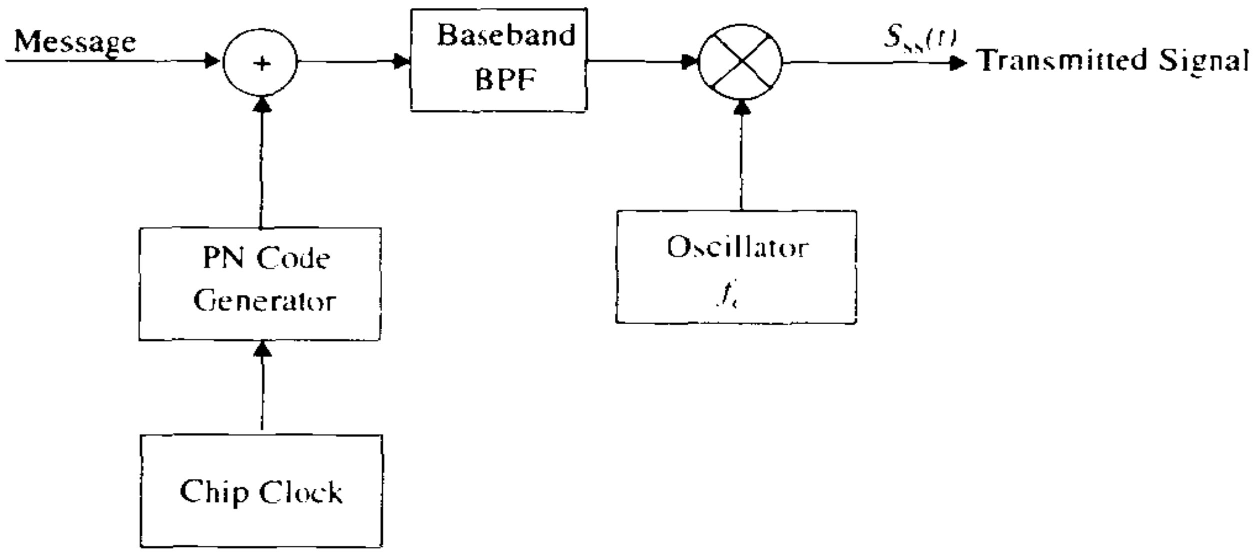
Aim: To Generate PN sequence of length 2^N-1 and build the Spread Spectrum modulation for CDMA system. Where N is order of the Generator Polynomial.

Theory:

A direct sequence spread spectrum (DS-SS) system spreads the baseband data by directly multiplying the baseband data pulses with a pseudo-noise sequence that is produced by a pseudo-noise code generator. A single pulse or symbol of the PN waveform is called a chip. This system is one of the most widely used direct sequence implementations. Synchronized data symbols, which may be information bits or binary channel code symbols, are added in modulo-2 fashion to the chips before being phase modulated. A coherent or differentially coherent phase-shift keying (PSK) demodulation may be used in the receiver. The received spread spectrum signal for a single user can be represented as

$$S_{ss}(t) = \sqrt{\frac{2E_s}{T_s}} m(t) p(t) \cos(2\pi f_c t + \theta)$$

where m(t) is the data sequence, p(t) is the PN spreading sequence, fc is the carrier frequency, and B is the carrier phase angle at t = 0. The data waveform is a time sequence of nonoverlapping rectangular pulses, each of which has an amplitude equal to +1 or -1. Each symbol in m(t) represents a data symbol and has duration Each pulse in p(t) represents a chip, is usually rectangular with an amplitude equal to +1 or -1, and has a duration of Tc. The transitions of the data symbols and chips coincide such that the ratio Ts to Tc is an integer. If Wss is the bandwidth of Sss(t) and B is the bandwidth of m(t) cos(2пfct), the spreading due to p(t) gives Wss >> B.



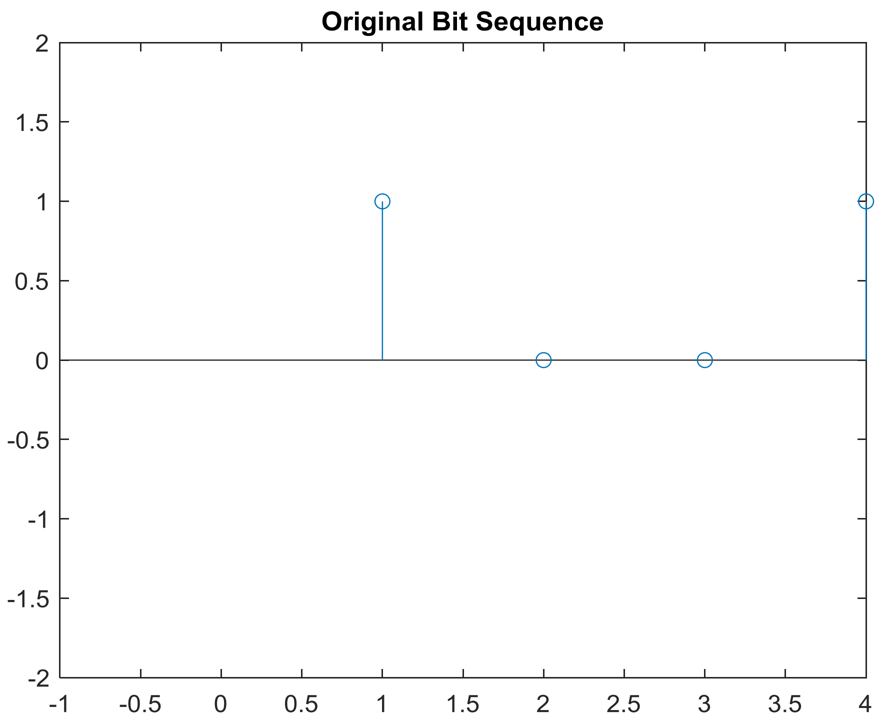
Program:

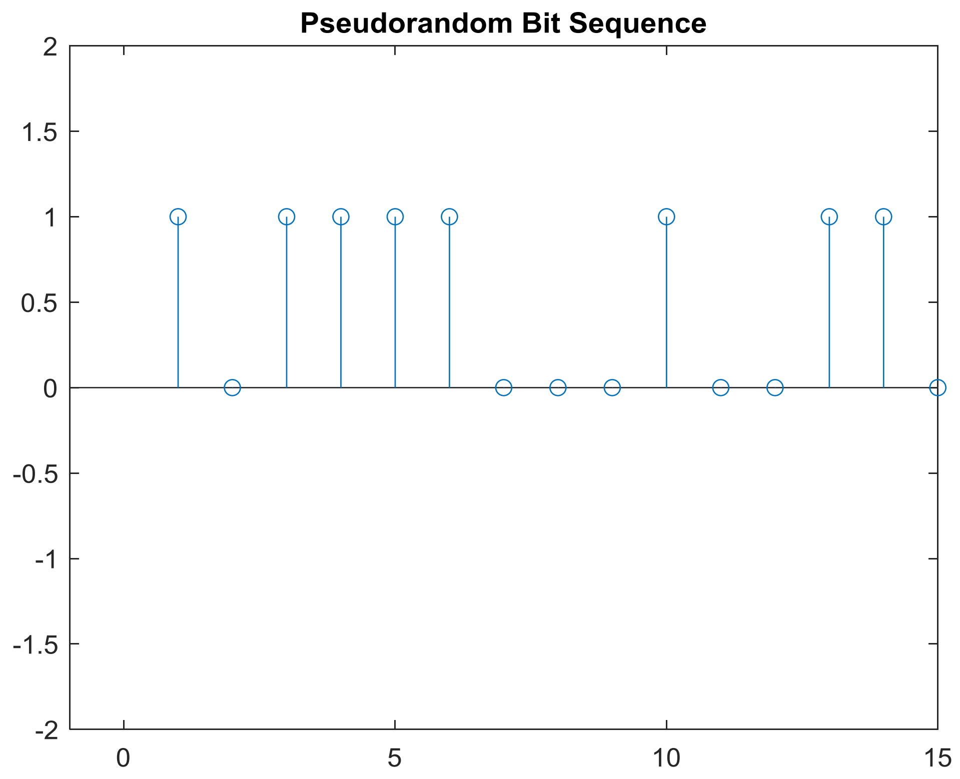
```
% Direct Sequence Spread Spectrum
clc;
clear all;
close all;
% Generating the bit pattern
DATA_pattern=randi([0,1],1,4);
% Generating the pseudo random bit pattern for spreading
M=4; %number of flipflops in LFSR=degree of the generator polynomial
N=2^M-1; %period of PN sequence
%initial states of LFSR
x=input('Enter the initial states of LFSR: ');
x1=x(1);
x2=x(2);
x3=x(3);
x4=x(4);
states=[];
PN_sequence=[];
   for i=1:N
   states=[states;x1(i) x2(i) x3(i) x4(i)];
   x1(i+1)=xor(x3(i),x4(i));
   x2(i+1)=x1(i);
   x3(i+1)=x2(i);
   x4(i+1)=x3(i);
   PN_sequence=[PN_sequence x4(i)];
   end
figure(1);
stem(DATA_pattern);
axis([-1 4 -2 2]);
title('Original Bit Sequence');
figure(2);
stem(PN_sequence);
axis([-1 15 -2 2]);
title('Pseudorandom Bit Sequence');
for i=1:M
  Newpattern(i,:)=repmat(DATA_pattern(i),1,15);
end
% XORing the DATA pattern with the PN sequence
for i=1:M
dsss_bitsequence(i,:)=xor(Newpattern(i,:),PN_sequence);
end
Length=size(dsss_bitsequence);
dsss_bitsequence=reshape(dsss_bitsequence',1,Length(1)*Length(2));
figure(3);
stem(dsss_bitsequence);
axis([-1 length(dsss_bitsequence) -2 2]);
title('DSSS Bit Sequence');
% Modulating the DSSS Bit Sequence
dsss_sig=[];
```

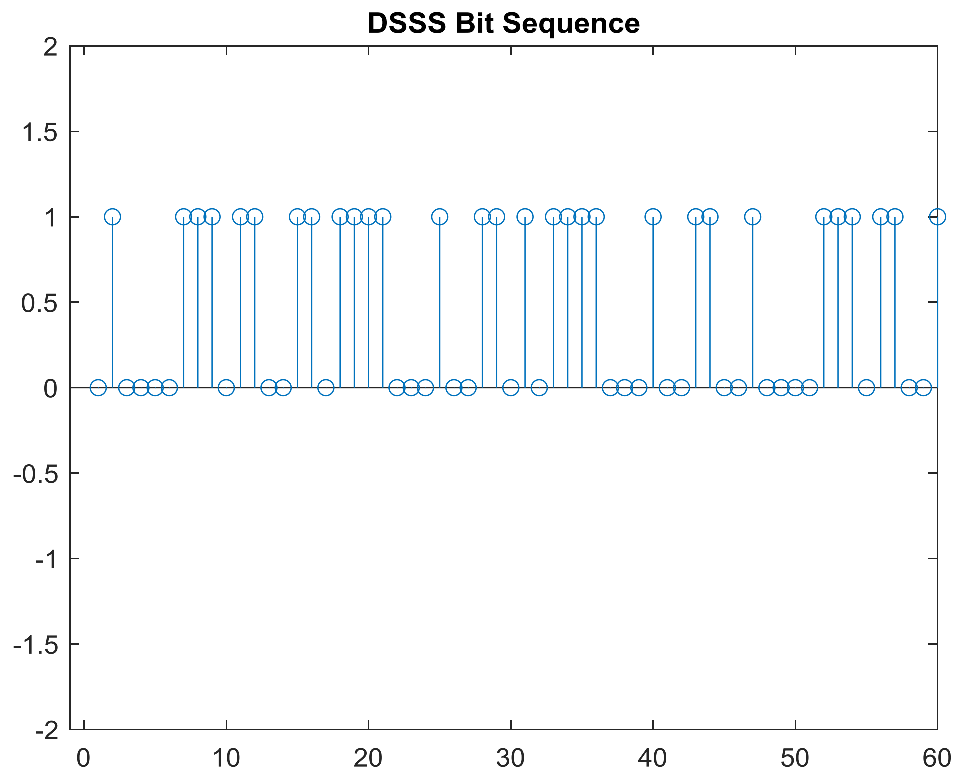
```
t=[0:2*pi/256:2*pi-(pi/256)]; % Creating 256 samples for one cosine
 % Bpsk symbols
c1=cos(t);
c2=cos(t+pi);
for k=1:length(dsss_bitsequence)
    if dsss_bitsequence(1,k)==0
        sig=c1;
    else
        sig=c2;
    end
    dsss_sig=[dsss_sig sig];
end
figure(4);
plot([1:length(dsss_sig)],dsss_sig);
axis([-1 length(dsss_sig) -2 2]);
title('DSSS Signal');
% Plotting the FFT of DSSS signal
figure(5);
plot([1:length(dsss_sig)],abs(fft(dsss_sig)));
title('Magnitude Spectrum');
```

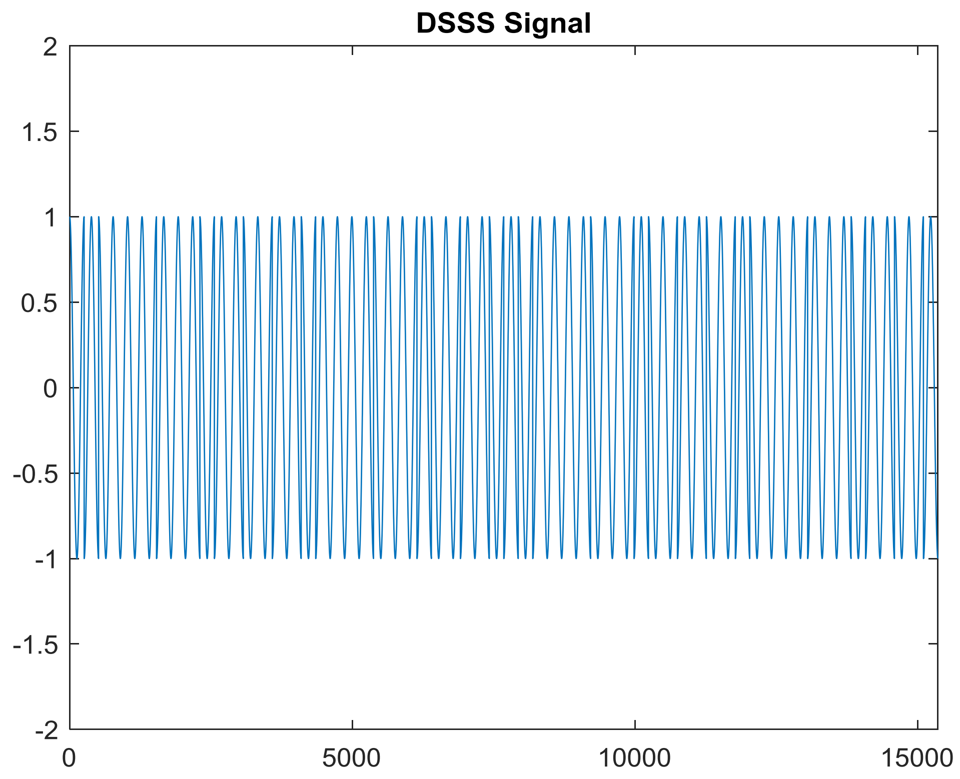
<u>Sample Input/Output:</u>

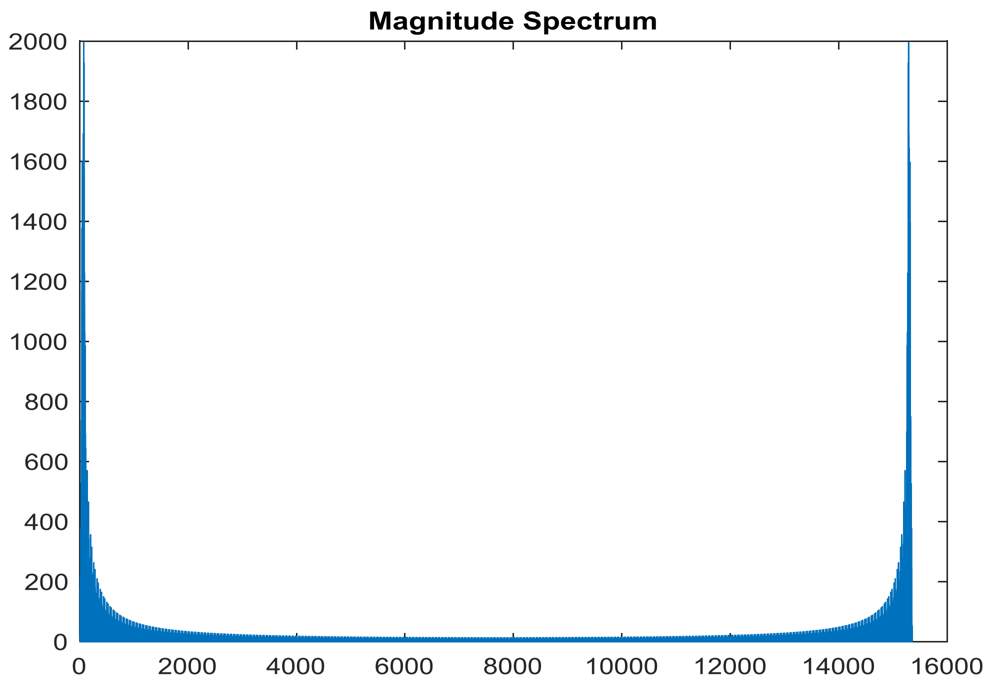
Enter the initial states of LFSR: [1 1 0 1]




Pseudorandom Bit Sequence
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
5
10
15


DSSS Bit Sequence
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
10
20
30
40
50
60





<u>Result:</u>

Thus, the generation of PN sequence of length 2^N-1 and implementation of Direct Spread Spectrum modulation system using randomly choosing one of the generated PN sequence for CDMA system is successfully carried out using MATLAB and results are verified through necessary waveforms.

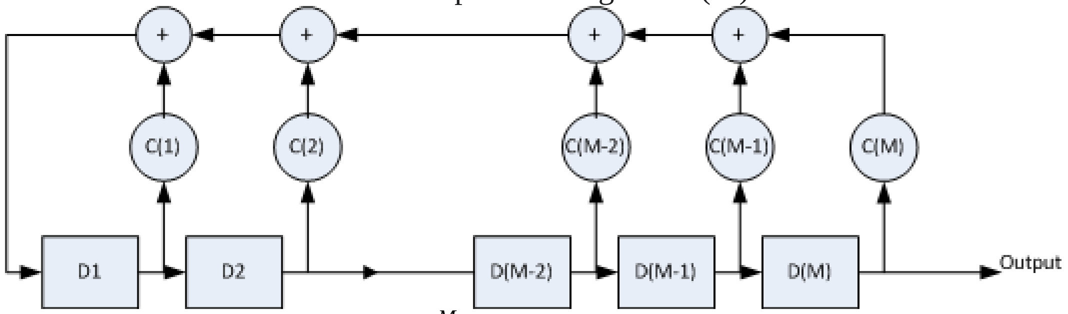
Experiment 7:

Aim: To verify the correlation propoerties of the PN sequence codes devloped for the CDMA.

Theory:

PN Sequence Generation

A PN data sequence is an M-sequence that is generated using a linear feedback shift-register circuit, as illustrated below. M is the number of shift registers. D(M) is the mth shift register, and {c1,c2,…,cM} are the coefficients of them. At each clock pulse, the data in the registers will right shift once and one PN datum is output from register D(M).



The PN sequence, $\{a_n\}$, $0 \le n < 2^M$, is generated by the equation below.

$$a_n = \begin{cases} \text{initial state in } D^{M-n}, & 0 \le n < M \\ c_M a_{n-M} + c_{M-1} a_{n-(M-1)} + \cdots + c_2 a_{n-2} + c_1 a_{n-1}, & M \le n < 2^M \end{cases}$$

where, the initial state of registers {D1,D2…DM} is the seed. D(M) stores the LSB of the seed, and D1 stores the MSB of the seed. For example, if the seed is 10 (binary form 1010), the initial state in register {D1,D2…DM} is {0 0 0 0 0 1 0 1 0}. The figure below shows the initial state of each register for PN9. It’s obvious that LSB of the seed comes out first.

Properties:

1.Balance property

The occurrence of 0 and 1 in the sequence should be approximately the same. More precisely, in a maximum length sequence of length 2^n-1 there are 2^{n-1} ones and 2^{n-1} -1 zeros. The number of ones equals the number of zeros plus one, since the state containing only zeros cannot occur.

2.Run property

A "run" is a sub-sequence of consecutive "1"s or consecutive "0"s within the MLS concerned. The number of runs is the number of such sub-sequences of all the "runs" (consisting of "1"s or "0"s) in the sequence:

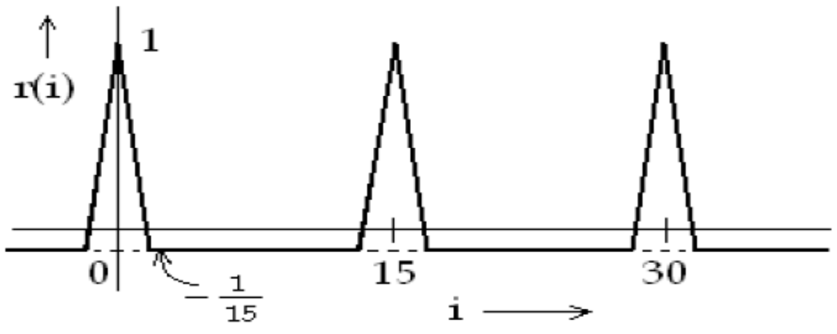
- One half of the runs are of length 1.
- One quarter of the runs are of length 2.
- One eighth of the runs are of length 3.
- ... etc. ...

3. Correlation property

The circular autocorrelation of an MLS is a Kronecker delta function (with DC offset and time delay, depending on implementation). For the ±1 convention, i.e., bit value 1 is assigned to +1 and bit value 0 is assigned to -1, mapping XOR to the negative of the product:

$$R(n) = \frac{1}{N}\sum_{m=1}^{N} s[m]\, s^*[m+n]_N = \begin{cases} 1 & \text{if } n = 0, \\ -\frac{1}{N} & \text{if } 0 < n < N. \end{cases}$$

where s* represents the complex conjugate and $[m+n]_N$ represents a circular shift.



The ACF of the PN sequence obtained

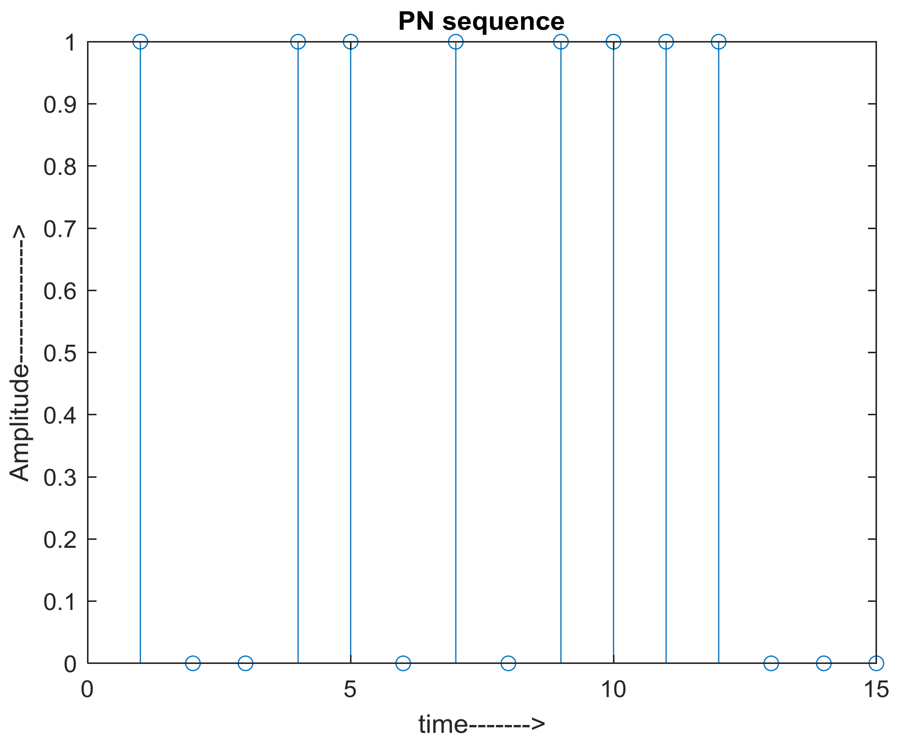
Program:

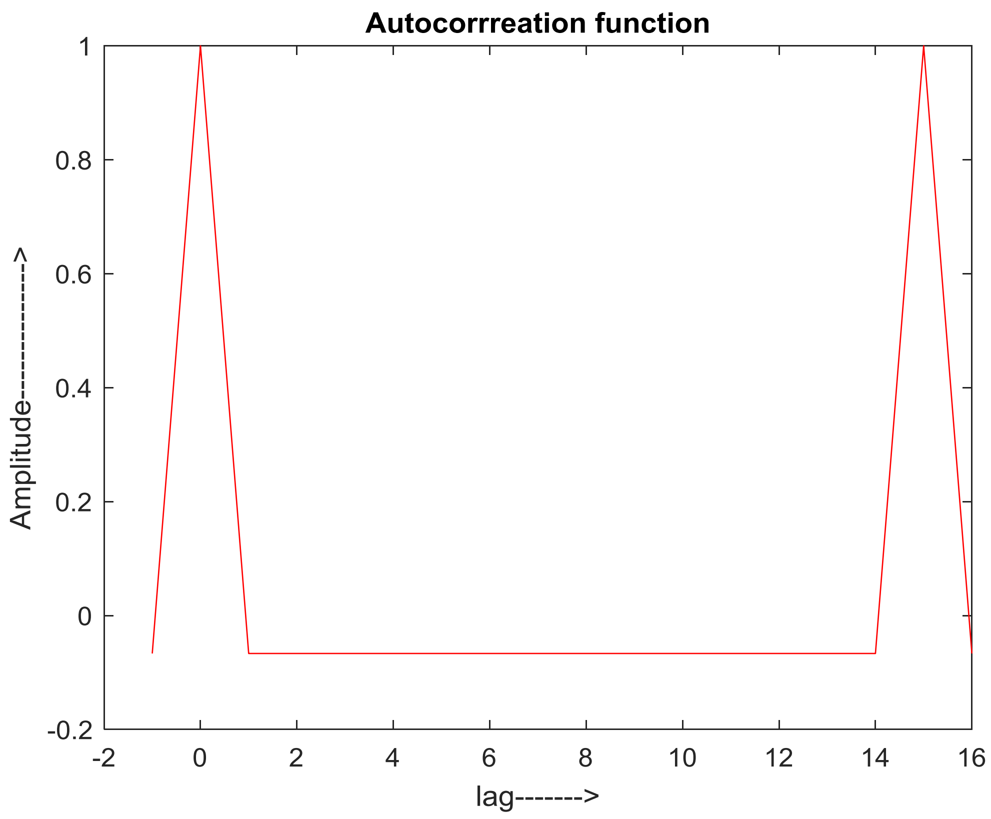
```
%PN sequence generation
clc;
clear all;
close all;
M=4; %number of flipflops in LFSR(length of PN codes)
N=2^M-1; %Period of PN sequence
%initial states of LFSR
x=input('Enter the initial states of LFSR: ');
x1=x(1);
x2=x(2);
x3=x(3);
x4=x(4);
states=[];
PN_sequence=[];
for j=1:N
   states=[states;x1(j) x2(j) x3(j) x4(j)];
   x1(j+1)=xor(x3(j),x4(j));
   x2(j+1)=x1(j);
   x3(j+1)=x2(j);
   x4(j+1)=x3(j);
   PN_sequence=[PN_sequence x4(j)];
end
figure(1);
stem(PN_sequence);
title('PN sequence');
xlabel('time------->');
ylabel('Amplitude-------------->');
%Autocorrelation Properties verification
for i=1:N
```

```
   if PN_sequence(i)==1
      PN_sequence(i)=-1;
   elseif PN_sequence(i)==0
      PN_sequence(i)=1;
   end
end
ac=[];
for lag=-1:N+1
   p=circshift(PN_sequence,lag);
   acf=sum(PN_sequence.*p)/N;
   ac=[ac acf];
end
figure(2);
plot(-1:N+1,ac,'r-');
title('Autocorrreation function');
xlabel('lag------->');
ylabel('Amplitude-------------->');
```

Sample Input/Output:

Enter the initial states of LFSR: [1 0 0 1]





Result:

Thus, the MATLAB code for verification of correlation properties of PN sequence used for CDMA is written and results are verified through necessary graphs.

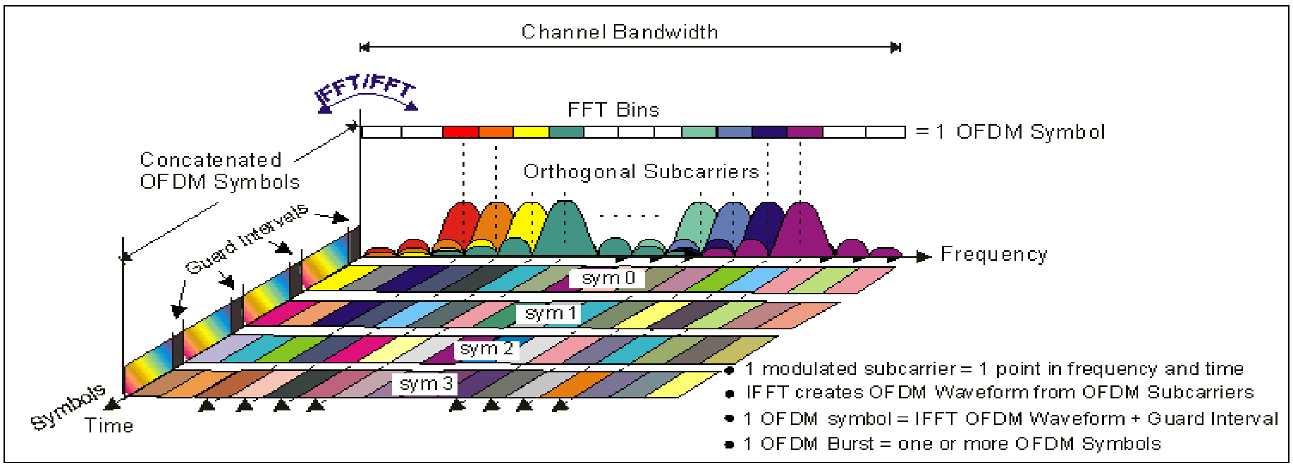
Experiment 8: Generation of BPSK-OFDM Signal

Aim: To build BPSK-OFDM system in MATLAB and verify it function through necessary waveforms.

Theory:

Orthogonal Frequency Division Multiplexing (OFDM) is a digital multi-carrier modulation scheme that extends the concept of single subcarrier modulation by using multiple subcarriers within the same single channel. Rather than transmit a high-rate stream of data with a single subcarrier, OFDM makes use of a large number of closely spaced orthogonal subcarriers that are transmitted in parallel. Each subcarrier is modulated with a conventional digital modulation scheme at low symbol rate. However, the combination of many subcarriers enables data rates similar to conventional single-carrier modulation schemes within equivalent bandwidths.

The following figure illustrates the main concepts of an OFDM signal and the inter-relationship between the frequency and time domains. In the frequency domain, multiple adjacent tones or subcarriers are each independently modulated with complex data. An Inverse FFT transform is performed on the frequency-domain subcarriers to produce the OFDM symbol in the time-domain. Then in the time domain, guard intervals are inserted between each of the symbols to prevent inter-symbol interference at the receiver caused by multi-path delay spread in the radio channel. Multiple symbols can be concatenated to create the final OFDM burst signal. At the receiver an FFT is performed on the OFDM symbols to recover the original data bits.


Channel Bandwidth
FFT/IFFT
FFT Bins
= 1 OFDM Symbol
Concatenated OFDM Symbols
Orthogonal Subcarriers
Guard Intervals
Frequency
sym 0
sym 1
sym 2
sym 3
Symbols
Time
• 1 modulated subcarrier = 1 point in frequency and time
• IFFT creates OFDM Waveform from OFDM Subcarriers
• 1 OFDM symbol = IFFT OFDM Waveform + Guard Interval
• 1 OFDM Burst = one or more OFDM Symbols

Frequency-Time Representative of an OFDM signal

An 802.11a OFDM carrier signal (burst type) is the sum of one or more OFDM symbols each comprised of 52 orthogonal subcarriers, with baseband data on each subcarrier being independently modulated (available formats: BPSK, QPSK, 16-QAM, or 64-QAM). This composite baseband signal is used to modulate a main RF carrier.

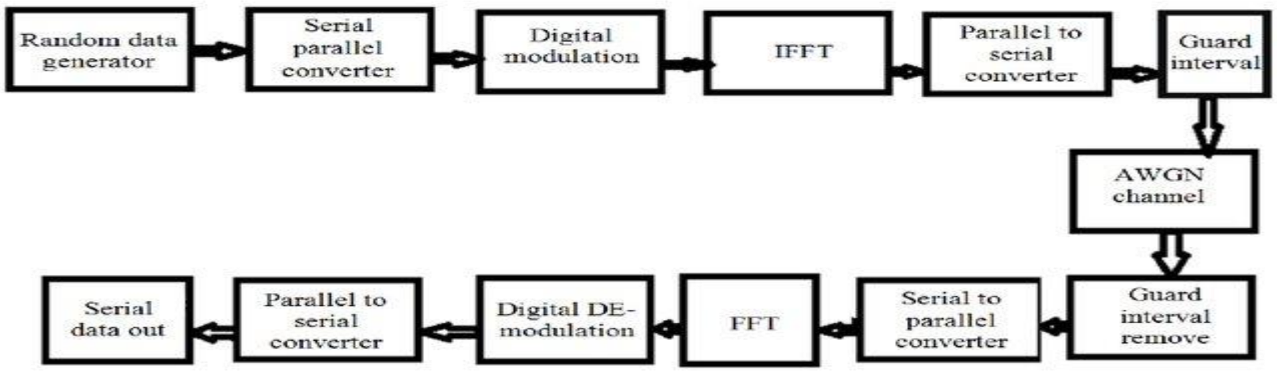
To begin the OFDM signal creation process, the input data bit stream is encoded with convolutional coding and Interleaving. Each data stream is divided into groups of "n" bits (1 bit -BPSK, 2 bits -

QPSK, 4 bits -16QAM, or 6 bits -64QAM) and converted into complex numbers (I+jQ) representing the mapped constellation point.

Then 52 bins of the IFFT block are loaded. 48 bins contain the constellation points which are mapped into frequency offset indexes ranging from -26 to +26, skipping the 4 Pilot and zero bins. There are 4 Pilot subcarriers inserted into frequency offset index locations -21, -7, +7, and +21. The zero bin is the Null or DC subcarrier and is not used; it contains a 0 value (0+j0).

When the IFFT block is completely loaded, the Inverse FFT is computed, giving a set of complex time-domain samples representing the combined OFDM subcarrier waveform. To complete the OFDM symbol, a 0.8 us duration Guard Interval (GI) is then added to the beginning of the OFDM waveform. This produces a "single" OFDM symbol with a time duration of 4 us in length, (3.2 us + 0.8 us). The process is repeated to create additional OFDM symbols for the remaining input data bits.

To complete the OFDM frame structure, the single OFDM symbols are concatenated together and then appended to a 16 us Preamble (used for synchronization) and a 4 us SIGNAL symbol (provides Rate and Length information). This completes the OFDM frame and is ready to be transmitted as an OFDM Burst.



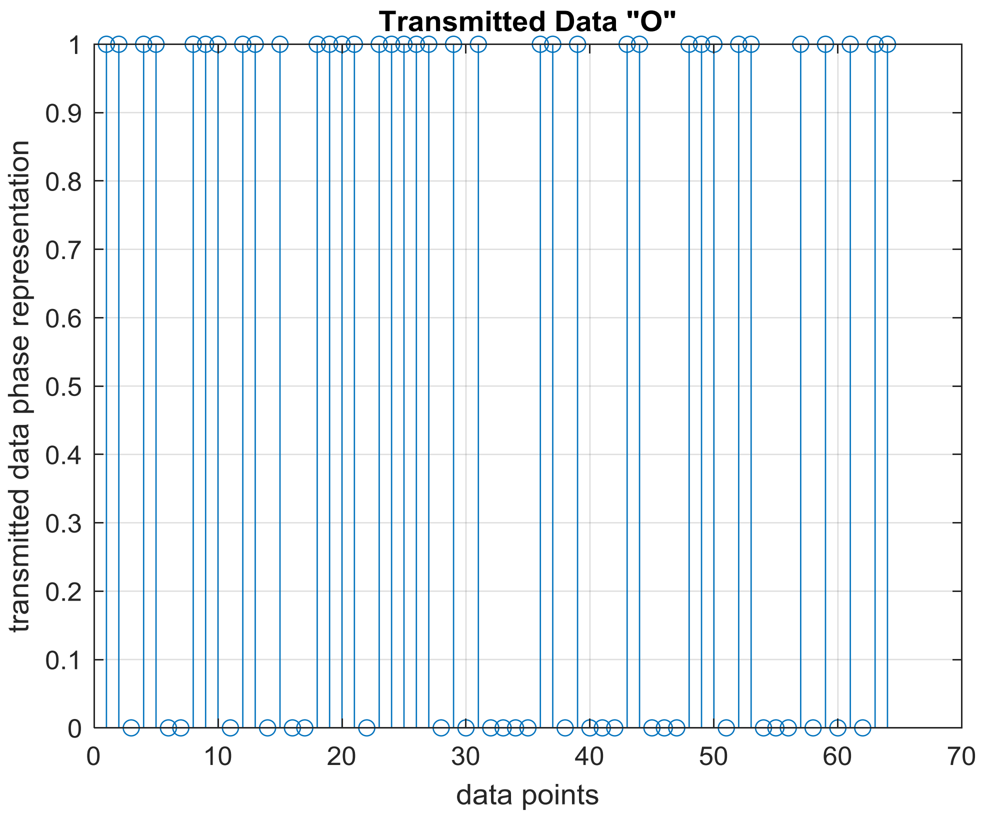
Program:

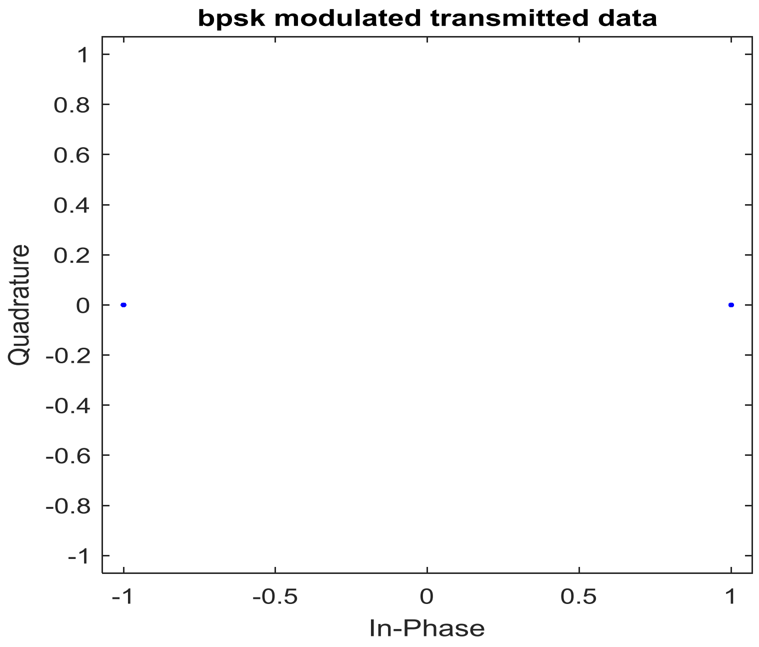
```
clc;
clear all;
close  all;
%Parameters setting
M = 2;                          %   BPSK signal constellation
no_of_data_points = 64;         %   have 64 data points
block_size = 8;                 %   size of each ofdm block
cp_len = ceil(0.1*block_size);  %   length of cyclic prefix
no_of_ifft_points = block_size; %   8 points for the FFT/IFFT
no_of_fft_points = block_size;
%Transmitter
%  Generate data points
data_source = randsrc(1, no_of_data_points, 0:M-1);
figure(1)
stem(data_source);
grid on;
```

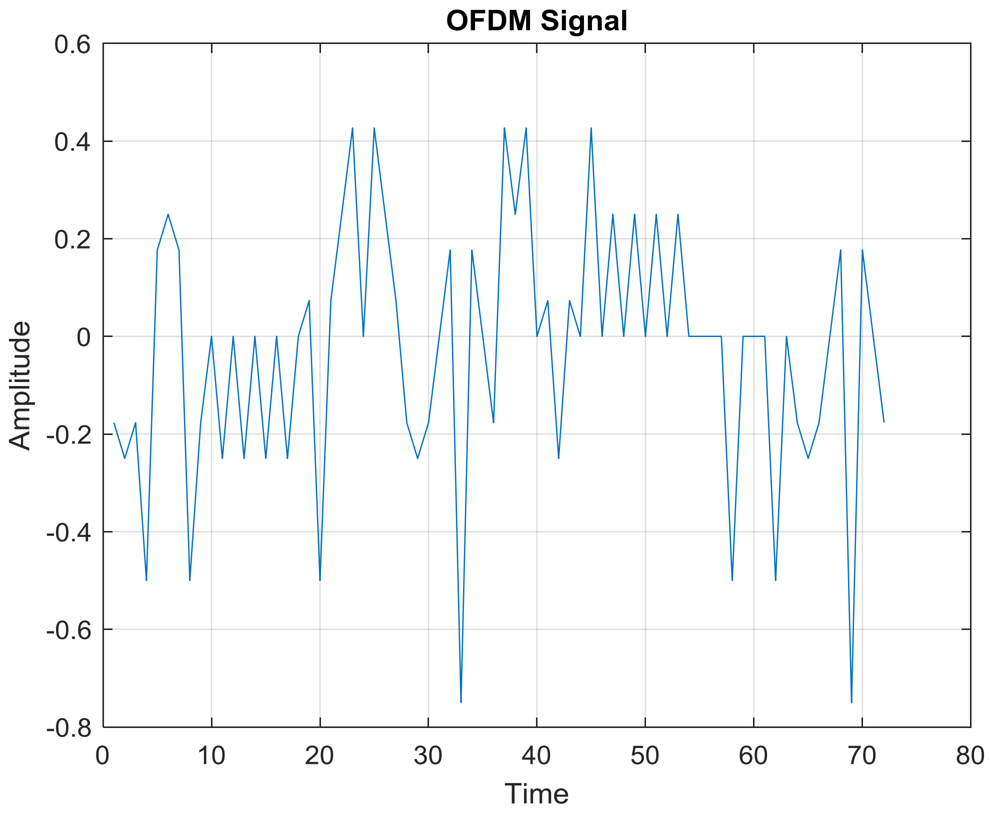
```
xlabel('data points');
ylabel('transmitted data phase representation')
title('Transmitted Data "O"')
%  Perform BPSK modulation
Bpsk_modulated_data = pskmod(data_source, M);
scatterplot(Bpsk_modulated_data);
title('bpsk modulated transmitted data')
%  Do IFFT on each block
num_cols=length(Bpsk_modulated_data)/block_size;
data_matrix = reshape(Bpsk_modulated_data, block_size, num_cols);
% Create empty matix to put the IFFT'd data
cp_start = block_size-cp_len;
cp_end = block_size;
% Operate columnwise & do CP
for i=1:num_cols
   ifft_data_matrix(:,i) = ifft((data_matrix(:,i)),no_of_ifft_points);
   %   Compute and append Cyclic Prefix
   for j=1:cp_len
      actual_cp(j,i) = ifft_data_matrix(j+cp_start,i);
   end
   %   Append the CP to the existing block to create the actual OFDM block
   ifft_data(:,i) = vertcat(actual_cp(:,i),ifft_data_matrix(:,i));
end
%  Convert to serial stream for transmission
[rows_ifft_data cols_ifft_data]=size(ifft_data);
len_ofdm_data = rows_ifft_data*cols_ifft_data;
%  OFDM signal to be transmitted
ofdm_signal = reshape(ifft_data, 1, len_ofdm_data);
figure(3)
plot(real(ofdm_signal));
xlabel('Time');
ylabel('Amplitude');
title('OFDM Signal');
grid on;
%Receiver
%Pass the ofdm signal through the channel
recvd_signal = ofdm_signal;
%Convert Data back to "parallel" form to perform FFT
recvd_signal_matrix = reshape(recvd_signal,rows_ifft_data, cols_ifft_data);
%Remove CP
recvd_signal_matrix(1:cp_len,:)=[];
%Perform FFT
for i=1:cols_ifft_data
   fft_data_matrix(:,i) = fft(recvd_signal_matrix(:,i),no_of_fft_points);
end
%Convert to serial stream
recvd_serial_data = reshape(fft_data_matrix, 1,(block_size*num_cols));
%Demodulate the data
Bpsk_demodulated_data = pskdemod(recvd_serial_data,M);
scatterplot(Bpsk_modulated_data);
```

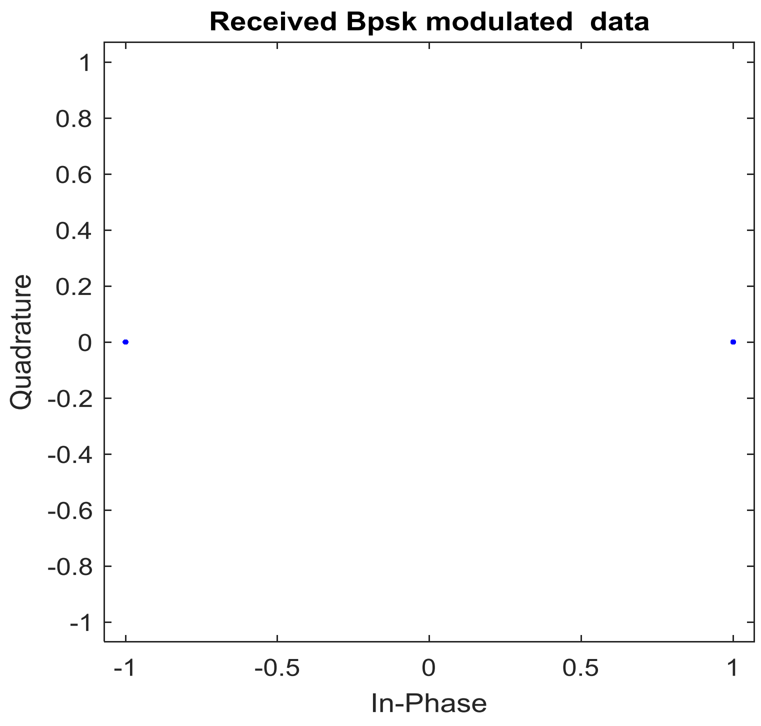
```
title('Received Bpsk modulated  data')
figure(5)
stem(Bpsk_demodulated_data,'rx');
grid on;xlabel('data points');
ylabel('received data phase representation');
title('Received Data "X"')
```

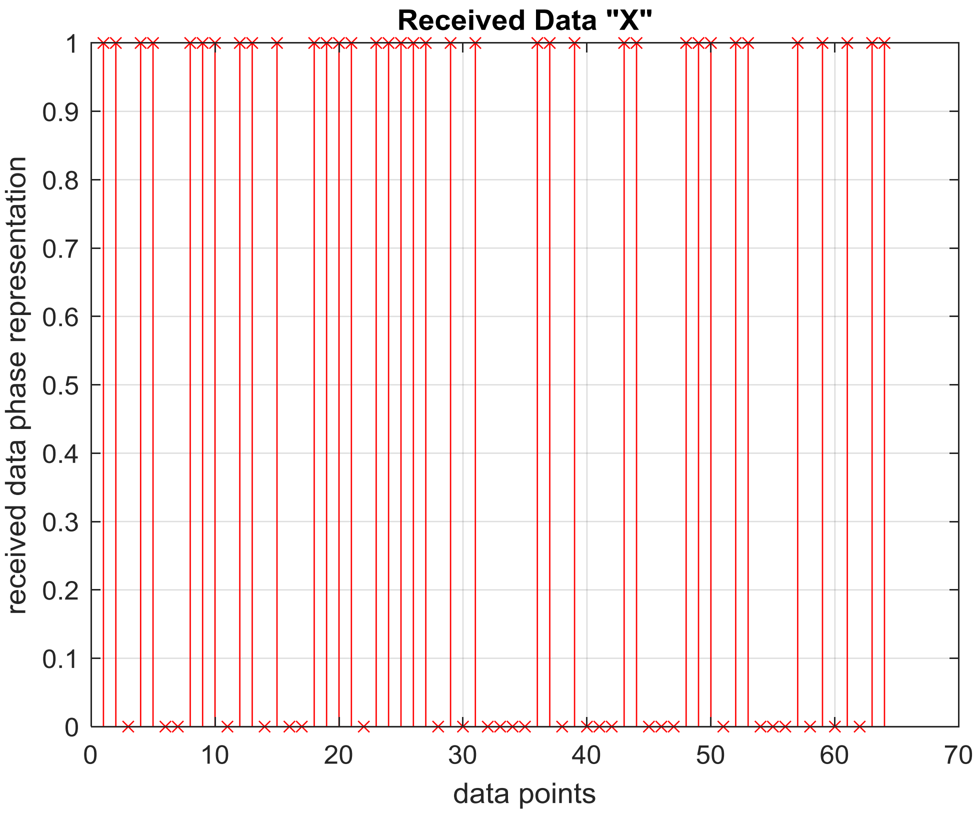
Sample Input/Output:




bpsk modulated transmitted data
Quadrature
In-Phase
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1


OFDM Signal
Amplitude
Time
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
10
20
30
40
50
60
70
80





Result:

Thus, the BPSK-OFDM system implemented in MATLAB and graphs for modulated waveform of binary input data stream, recovered input data stream are simulated and the results are verified.

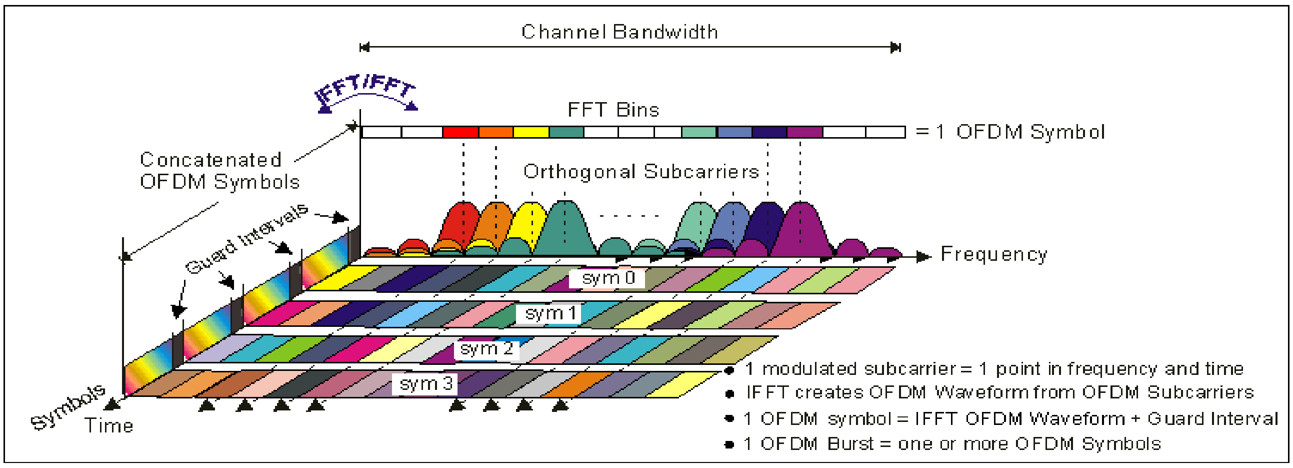
Experiment 9: Generation of QPSK-OFDM Signal

Aim: To build QPSK-OFDM system in MATLAB and verify it function through necessary waveforms.

Theory:

Orthogonal Frequency Division Multiplexing (OFDM) is a digital multi-carrier modulation scheme that extends the concept of single subcarrier modulation by using multiple subcarriers within the same single channel. Rather than transmit a high-rate stream of data with a single subcarrier, OFDM makes use of a large number of closely spaced orthogonal subcarriers that are transmitted in parallel. Each subcarrier is modulated with a conventional digital modulation scheme at low symbol rate. However, the combination of many subcarriers enables data rates similar to conventional single-carrier modulation schemes within equivalent bandwidths.

The following figure illustrates the main concepts of an OFDM signal and the inter-relationship between the frequency and time domains. In the frequency domain, multiple adjacent tones or subcarriers are each independently modulated with complex data. An Inverse FFT transform is performed on the frequency-domain subcarriers to produce the OFDM symbol in the time-domain. Then in the time domain, guard intervals are inserted between each of the symbols to prevent inter-symbol interference at the receiver caused by multi-path delay spread in the radio channel. Multiple symbols can be concatenated to create the final OFDM burst signal. At the receiver an FFT is performed on the OFDM symbols to recover the original data bits.


Channel Bandwidth
FFT/IFFT
FFT Bins
= 1 OFDM Symbol
Concatenated OFDM Symbols
Orthogonal Subcarriers
Guard Intervals
Frequency
sym 0
sym 1
sym 2
sym 3
Symbols
Time
1 modulated subcarrier = 1 point in frequency and time
IFFT creates OFDM Waveform from OFDM Subcarriers
1 OFDM symbol = IFFT OFDM Waveform + Guard Interval
1 OFDM Burst = one or more OFDM Symbols

Frequency-Time Representative of an OFDM signal

An 802.11a OFDM carrier signal (burst type) is the sum of one or more OFDM symbols each comprised of 52 orthogonal subcarriers, with baseband data on each subcarrier being independently modulated (available formats: BPSK, QPSK, 16-QAM, or 64-QAM). This composite baseband signal is used to modulate a main RF carrier.

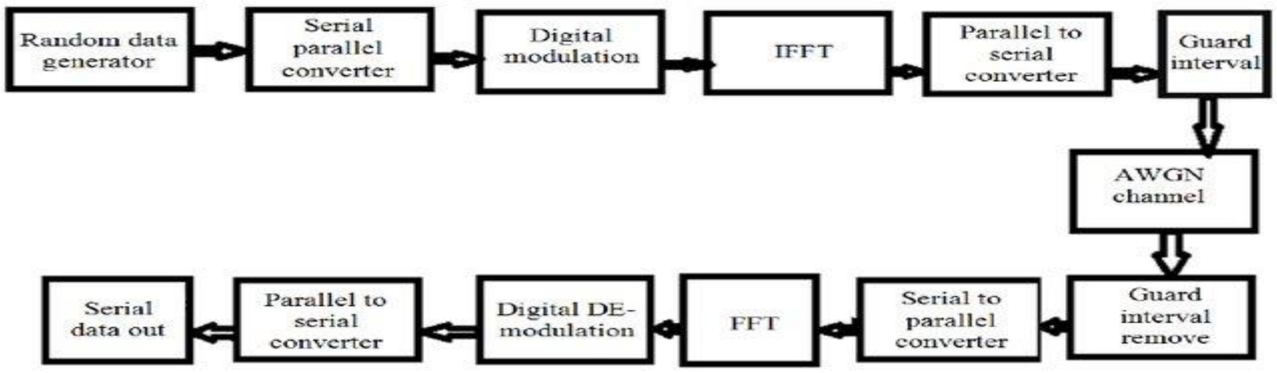
To begin the OFDM signal creation process, the input data bit stream is encoded with convolutional coding and Interleaving. Each data stream is divided into groups of "n" bits (1 bit -BPSK, 2 bits -

QPSK, 4 bits -16QAM, or 6 bits -64QAM) and converted into complex numbers (I+jQ) representing the mapped constellation point.

Then 52 bins of the IFFT block are loaded. 48 bins contain the constellation points which are mapped into frequency offset indexes ranging from -26 to +26, skipping the 4 Pilot and zero bins. There are 4 Pilot subcarriers inserted into frequency offset index locations -21, -7, +7, and +21. The zero bin is the Null or DC subcarrier and is not used; it contains a 0 value (0+j0).

When the IFFT block is completely loaded, the Inverse FFT is computed, giving a set of complex time-domain samples representing the combined OFDM subcarrier waveform. To complete the OFDM symbol, a 0.8 us duration Guard Interval (GI) is then added to the beginning of the OFDM waveform. This produces a "single" OFDM symbol with a time duration of 4 us in length, (3.2 us + 0.8 us). The process is repeated to create additional OFDM symbols for the remaining input data bits.

To complete the OFDM frame structure, the single OFDM symbols are concatenated together and then appended to a 16 us Preamble (used for synchronization) and a 4 us SIGNAL symbol (provides Rate and Length information). This completes the OFDM frame and is ready to be transmitted as an OFDM Burst.



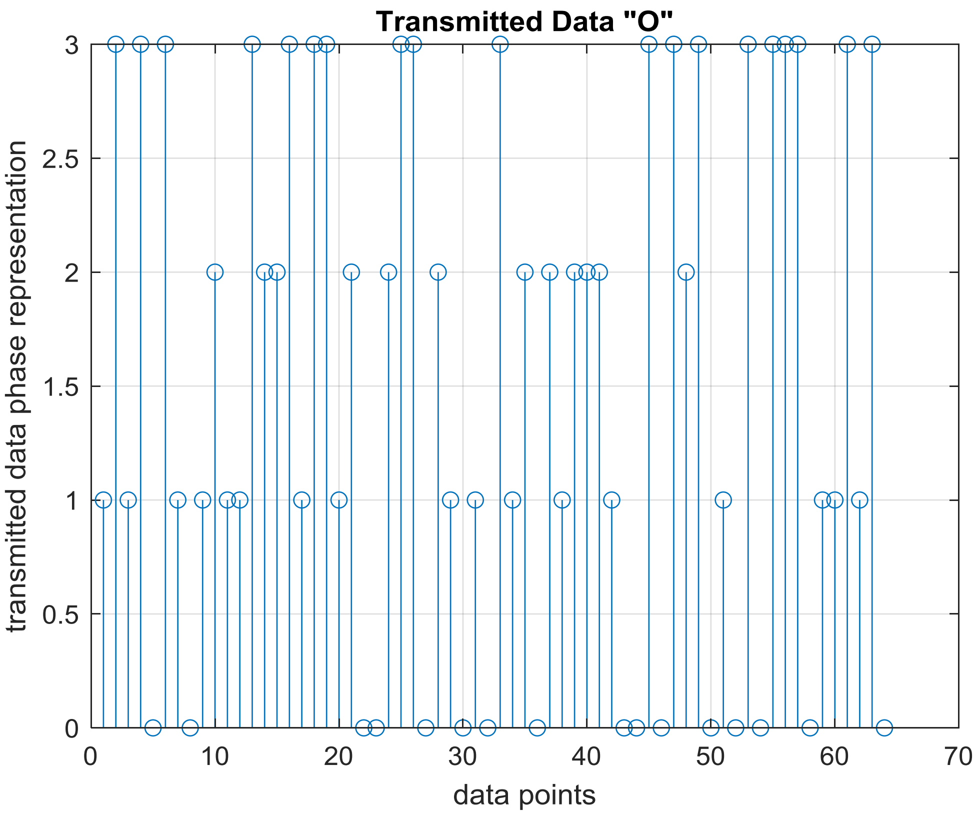
Program:

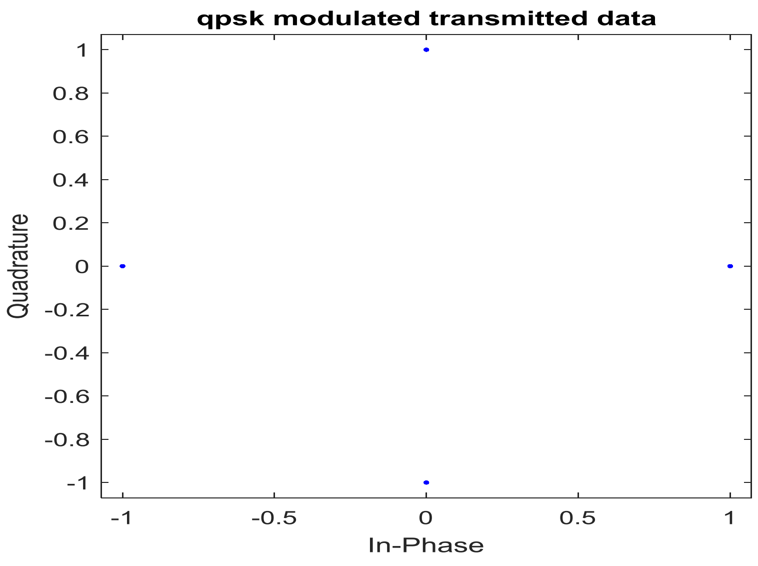
```
clc;
clear all;
close  all;
%Parameters setting
M = 4;                          %   QPSK signal constellation
no_of_data_points = 64;         %   have 64 data points
block_size = 8;                 %   size of each ofdm block
cp_len = ceil(0.1*block_size);  %   length of cyclic prefix
no_of_ifft_points = block_size;  %   8 points for the FFT/IFFT
no_of_fft_points = block_size;
%Transmitter
%  Generate data points
data_source = randsrc(1, no_of_data_points, 0:M-1);
figure(1)
stem(data_source);
grid on;
```

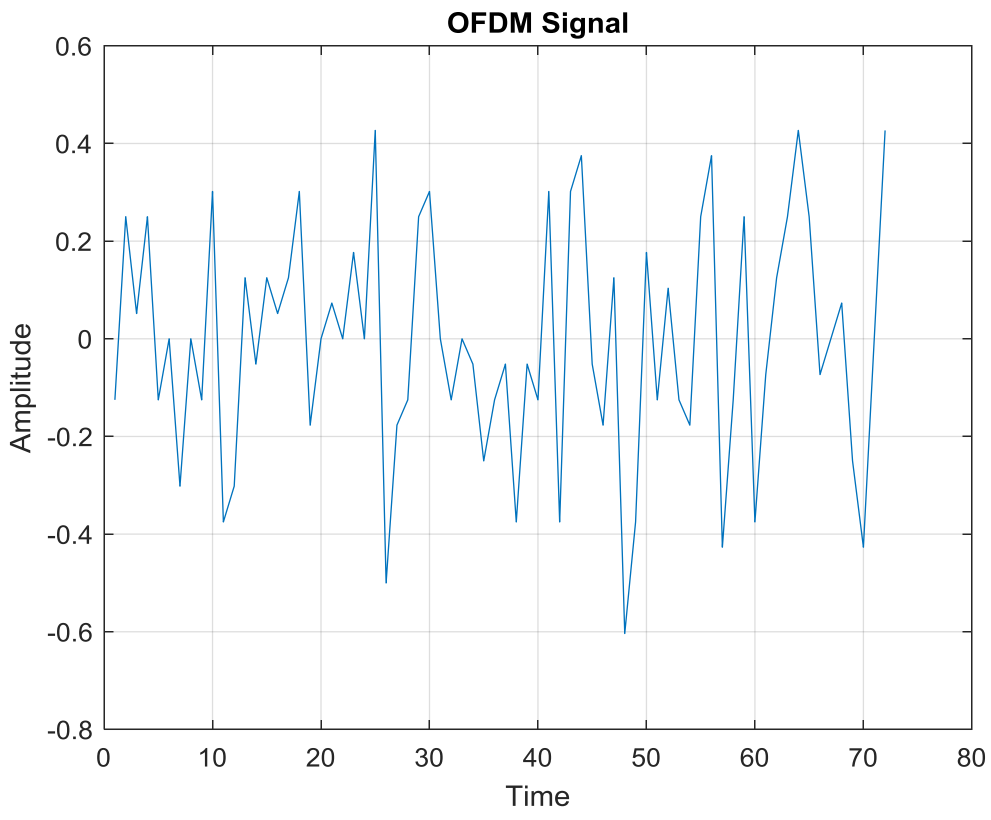
```
xlabel('data points');
ylabel('transmitted data phase representation')
title('Transmitted Data "O"')
%  Perform BPSK modulation
Bpsk_modulated_data = pskmod(data_source, M);
scatterplot(Bpsk_modulated_data);
title('bpsk modulated transmitted data')
%  Do IFFT on each block
num_cols=length(Bpsk_modulated_data)/block_size;
data_matrix = reshape(Bpsk_modulated_data, block_size, num_cols);
% Create empty matix to put the IFFT'd data
cp_start = block_size-cp_len;
cp_end = block_size;
% Operate columnwise & do CP
for i=1:num_cols
   ifft_data_matrix(:,i) = ifft((data_matrix(:,i)),no_of_ifft_points);
   %   Compute and append Cyclic Prefix
   for j=1:cp_len
      actual_cp(j,i) = ifft_data_matrix(j+cp_start,i);
   end
   %   Append the CP to the existing block to create the actual OFDM block
   ifft_data(:,i) = vertcat(actual_cp(:,i),ifft_data_matrix(:,i));
end
%  Convert to serial stream for transmission
[rows_ifft_data cols_ifft_data]=size(ifft_data);
len_ofdm_data = rows_ifft_data*cols_ifft_data;
%  OFDM signal to be transmitted
ofdm_signal = reshape(ifft_data, 1, len_ofdm_data);
figure(3)
plot(real(ofdm_signal));
xlabel('Time');
ylabel('Amplitude');
title('OFDM Signal');
grid on;
%Receiver
%Pass the ofdm signal through the channel
recvd_signal = ofdm_signal;
%Convert Data back to "parallel" form to perform FFT
recvd_signal_matrix = reshape(recvd_signal,rows_ifft_data, cols_ifft_data);
%Remove CP
recvd_signal_matrix(1:cp_len,:)=[];
%Perform FFT
for i=1:cols_ifft_data
   fft_data_matrix(:,i) = fft(recvd_signal_matrix(:,i),no_of_fft_points);
end
%Convert to serial stream
recvd_serial_data = reshape(fft_data_matrix, 1,(block_size*num_cols));
%Demodulate the data
Bpsk_demodulated_data = pskdemod(recvd_serial_data,M);
scatterplot(Bpsk_modulated_data);
```

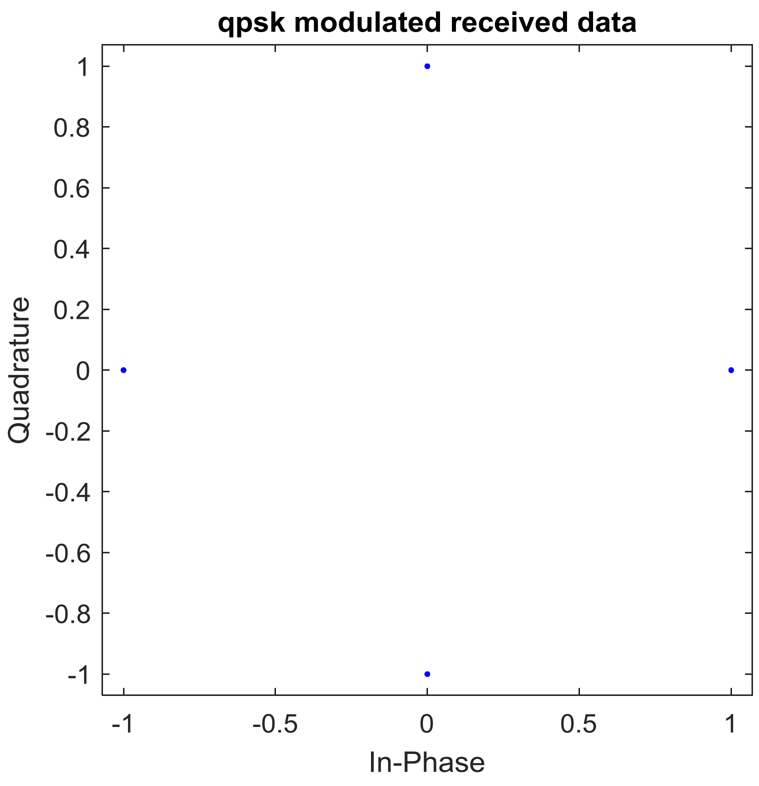
```
title('Received Bpsk modulated  data')
figure(5)
stem(Bpsk_demodulated_data,'rx');
grid on;xlabel('data points');
ylabel('received data phase representation');
title('Received Data "X"')
```

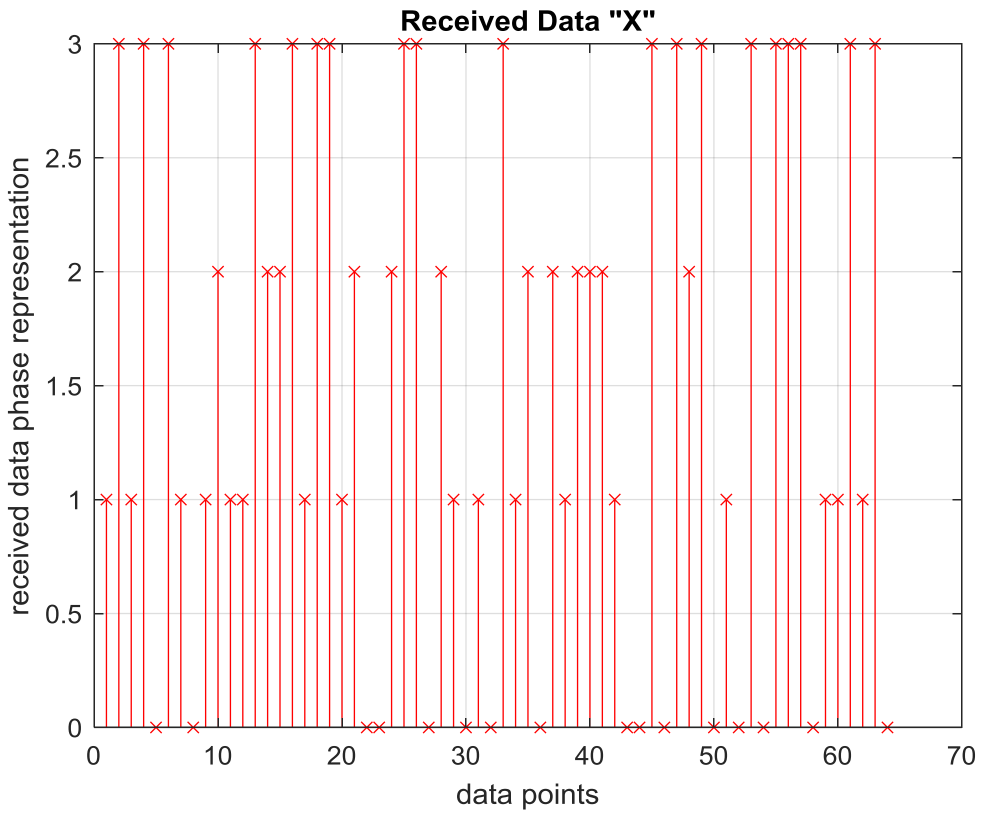
Sample Input/Output:






OFDM Signal
Amplitude
Time
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
10
20
30
40
50
60
70
80


qpsk modulated received data
Quadrature
In-Phase
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1



<u>Result:</u>

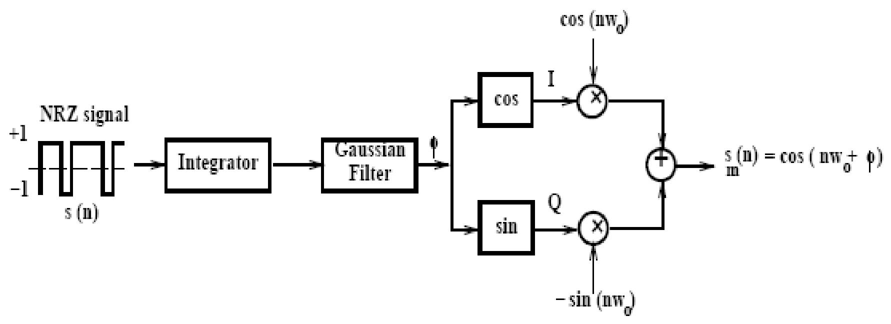
Thus, the QPSK-OFDM system implemented in MATLAB and graphs for modulated waveform of binary input data stream, recovered input data stream are simulated and the results are verified.

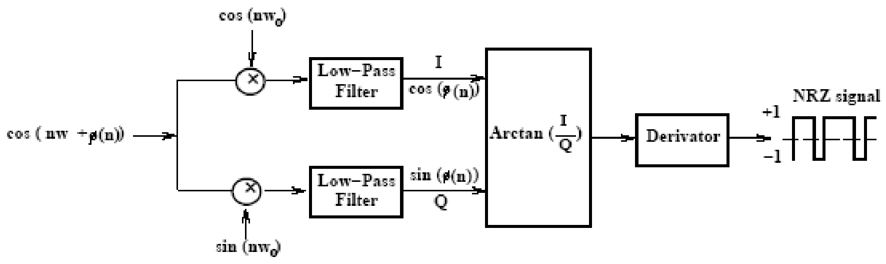
Experiment 10: Generation of GMSK signal

<u>Aim:</u> To build GMSK system in MATLAB and verify it function through necessary waveforms.

<u>Theory:</u>

Gaussian Minimum Shift Keying (GMSK) is one of the most widely used digital modulation technique. In GMSK, the phase of the carrier signal is continuously varied by a contrary signal, which has been shaped by the Gaussian shaping filter. Since it is kind of MSK, it has modulation index of value 0.5. In GMSK, the side lobe levels of the spectrum which were introduced in MSK are further minimized by passing the modulating NRZ data waveform through a pre modulation Gaussian pulse-shaping filter. The Gaussian filter width is determined by the bandwidth-time product BT (e.g., BT = 0.3 for GSM and BT = 0.5 for CDPD). As the BT product decreases, the bandwidth of the spectrum becomes narrow but this increases the inter symbol interference.
The integral of the impulse response of a Gaussian filter output is quite smooth which causes the phase of the modulated signal to vary in a continuous manner. GMSK has a low out of band power characteristic and a constant envelope which makes it a desirable choice for usage in the wireless mobile communications. The effect of the filter brings out the suppression of the out of band power by its sharp cut-off property. The impulse response of the filter function realizing the Gaussian pulse shape is given as
I/Q Modulator/Demodulator:





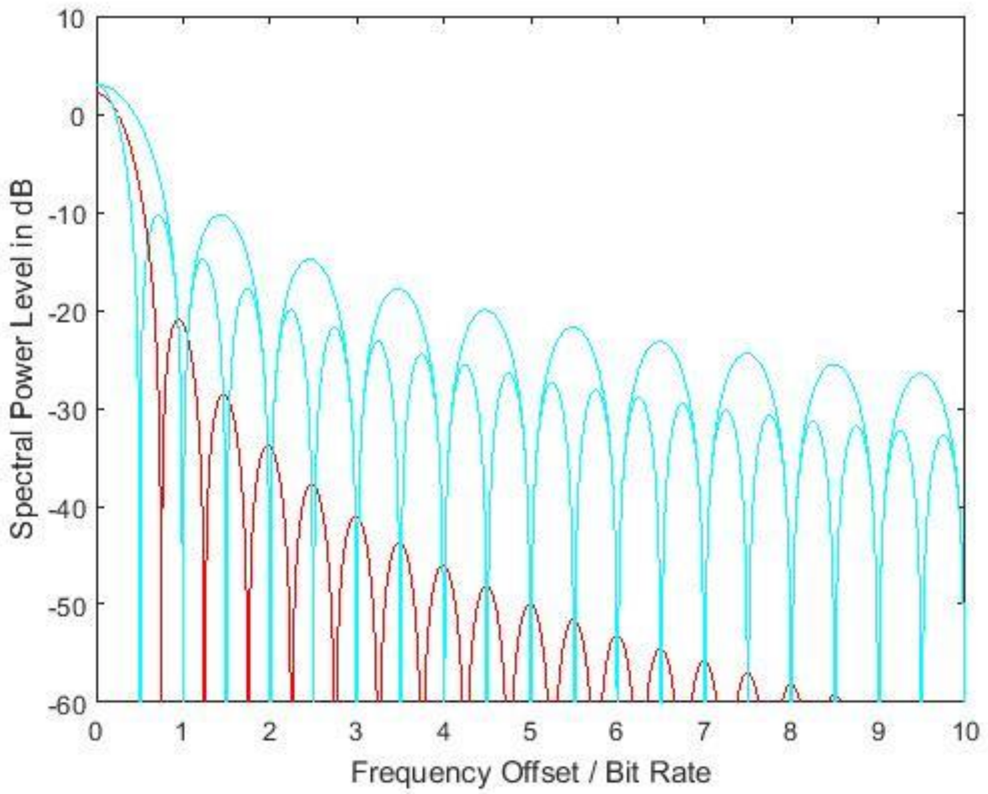
Advantages of GMSK:

- Improved spectral efficiency
- Reduced mail lobe over MSK
- Self-synchronizing capability
- Constant envelope over entire bandwidth
- Good BER performance
- High noise immunity
- Remain undistorted even when amplifies by a non- linear amplifier.

Program:

```
pi = 3.141592;
sqrpi = pi^2;

GMSK  = [];
GBPSK = [];
GQPSK = [];
xaxis = [];

for i=1:1000
    f = i/100;
    % f is frequency normalized to to 1/(bit duration)
    xaxis = [xaxis, f ];
    ymsk = 16/sqrpi * (cos(6.2832 * f))^2/ (1- 16 * f^2)^2;
    GMSK = [GMSK, 10 * log10(ymsk)];

    ybpsk = 2 * (sin(pi*f)/(pi*f))^2;
    GBPSK = [GBPSK, 10 * log10(ybpsk)];
    yqpsk = 2 * (sin(2*pi* f)/(2*pi*f))^2;
    GQPSK = [GQPSK, 10 * log10(yqpsk)];

end

plot(xaxis,GMSK, 'y-',xaxis,GBPSK, 'c.',xaxis,GQPSK, 'c-.');

axis([0 10 -60 10]);
ylabel('Spectral Power Level in dB');
xlabel('Frequency Offset / Bit Rate');
```

Sample Input/Output:



Result:

Thus, the GMSK system implemented in MATLAB and graphs for modulated waveform of binary input data stream, recovered input data stream are simulated and the results are verified.

Extra Program

```
To determine the maximum theoretical data rate and to compare this rate to
digital cellular standards.
clc;
close all;
SNR=20;
B = 30 * 10 ^ 3 ;
SNR =10^( SNR /10);
C=B*(log10(1+ SNR )/ log10 (2));
A=C *10^ -3;
```