ASSIGNMENT 7.3

UNDERSTANDING THE OFDM CONCEPT AND ITS IMPLEMENTATION

SUBMITTED

By

PENDOTA AMULYA

EE21MTECH12003

COMMUNICATON AND SIGNAL PROCESSING

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY HYDERABAD

TABLE OF CONTENTS:

1. INTRODUCTION to OFDM

OFDM is abbreviated as Orthogonal Frequency Division Multiplexing, which is a digital multi carrier modulation technique. Multiplexing generally refers to transmitting multiple or more than one signal over the channel/medium. This OFDM modulation technique can be understood as an improved version of FDM (Frequency Division multiplexing) although they differ by implementation.

In FDM, the total bandwidth is split into different frequency bands i.e, each message signal information is transmitted over a different frequency band, such that they don’t overlap. The frequency bands are separated by guard bands to avoid loss of information in the message signal at the receiver due to inter symbol interference or overlaping/aliasing.

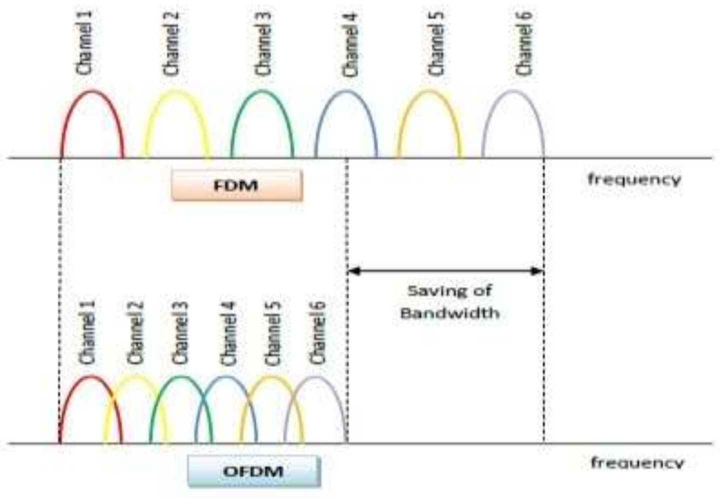
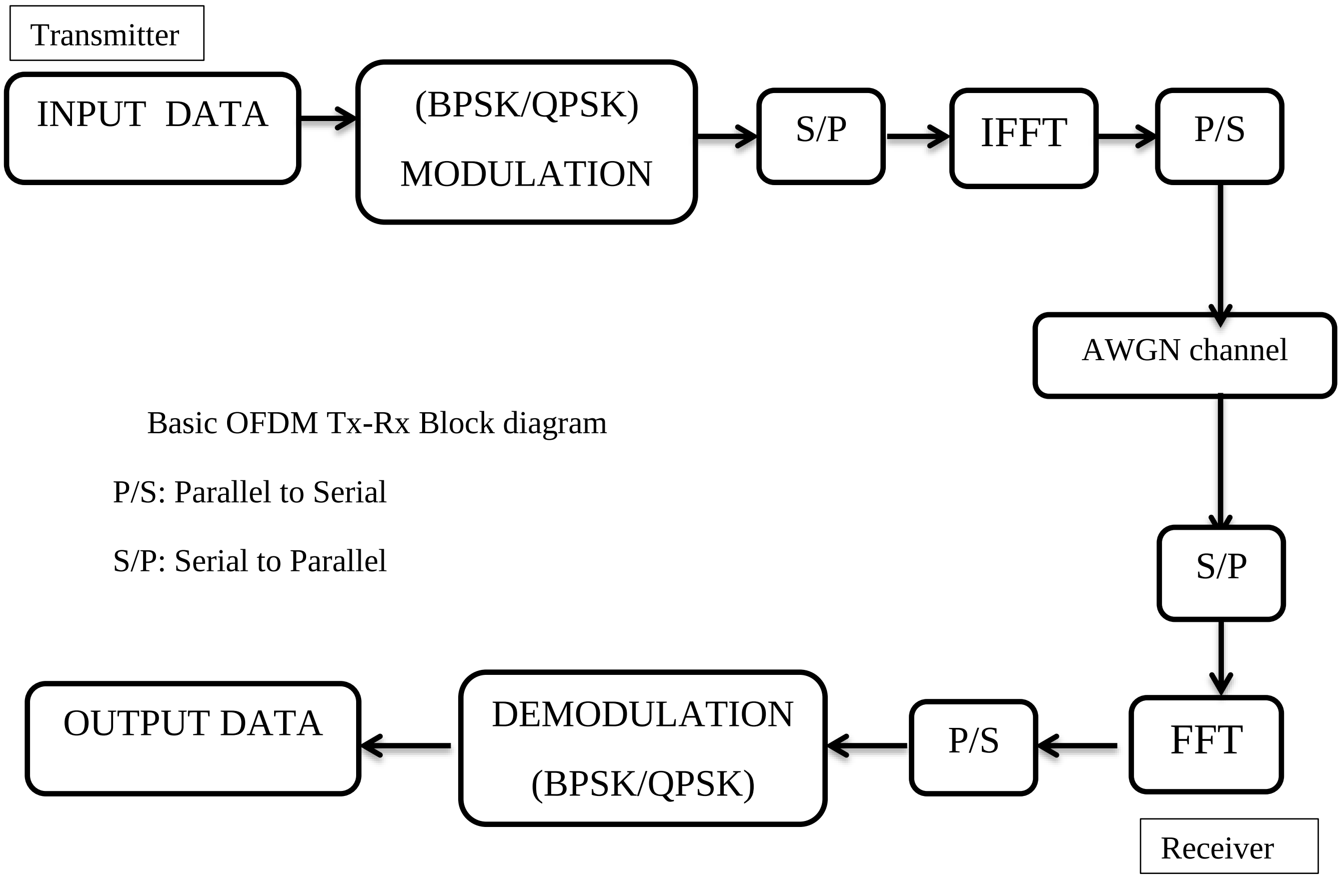


Figure-1: Basic Difference Between FDM and OFDM

Whereas OFDM is a modulation scheme which splits the channel bandwidth into multiple sub-carriers, each of which carries the message signal information independently using modulation techniques like BPSK/QPSK/QAM etc. These sub-carriers are orthogonal to each other and so there won’t be any loss of information occurring at the receiver, as they don’t interference would occur and hence the use of guard bands is also not required in case of OFDM. In OFDM,.the major advantage lies with the better utilization of the bandwidth.

2. BLOCK DIAGRAM

The basic block diagram of OFDM Transmitter and receiver is show below



Basic OFDM Tx-Rx Block diagram

P/S: Parallel to Serial

S/P: Serial to Parallel

Block diagram description:

The input data is initially encoded into bit stream and then modulated using BPSK/QPSK/QAM and to this modulated output IFFT is applied to generate OFDM signals which are then transmitted through channel.

At the receiver, it receives the transmitted data with the AWGN noise added to it. The received data is passed through FFT and then demodulated to get the output data. Detailed explaination of each block of OFDM will be discussed below.

3. IMPORTANCE OF ORTHOGONAL SIGNALS IN OFDM

Orthogonal signals are those signals whose inner product over a time period is zero. In other words the signals are perpendicular to each other. This property of the signals helps in no interference with other signals and hence the retrieving of the transmitted information can be done with out much loss. Due to this orthogonality of signals, even if their side bands overlap no interference occurs in OFDM but this is not the case in FDM and hence guard bands are not required here which results in better utilization of bandwidth.

The below is the matlab code to generate and plot the orthogonal and non orthogonal signals

3.1 MATLAB CODE FOR 2 ORTHOGONAL SIGNALS

```
% proving two signals are orthogonal
output1=0;
t=0:0.0001:0.01;
f1=1000;
w1=2*pi*f1;
x1=sin(w1*t);
x2=sin(2*w1*t);
x3=x1.*x2;          % product of 2 signals
output1=sum(x3);     % finding total sum over time period T
k1=round(output1,1);
if (k1==0)
    fprintf("x1 and x2 are orthogonal sig\n");
else
    fprintf("x1 and x2 are non orthogonal sig\n");
end
% plot of addition of 2 orthogonal signals
x4=x1+x2;  % sum of 2 signals
figure;
plot(t,x4);
xlabel('time T=10ms')
ylabel('Amplitude')
title('sum of 2 orthogonal')
```

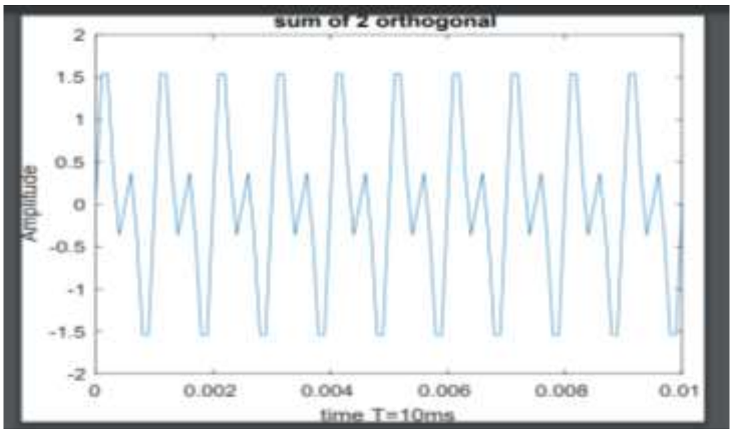


Figure-2: PLOT OF SUM OF TWO ORTHOGONAL SIGNALS

3.2 MATLAB CODE FOR 2 NON ORTHOGONAL SIGNALS

```
% proving two signals are non orthogonal
output2=0;
t=0:0.0001:0.01;
f1=1000;
w1=2*pi*f1;
y1=sin(w1*t);
```

```
y2=sin(1.22*w1*t);
y3=y1.*y2;                      % product of 2 signals
output2=sum(y3);          % finding total sum over time period T
k2=round(output2,1);
if (k2==0)
   fprintf("y1 and y2 are orthogonal sig\n");
else
   fprintf("y1 and y2 are non orthogonal sig\n");
end
% plot of addition of 2 non orthogonal signals
y4=y1+y2;             % sum of 2 signals
plot(t,y4);
xlabel('time T=10ms');
ylabel('Amplitude');
title('sum of 2 non orthogonal');
```

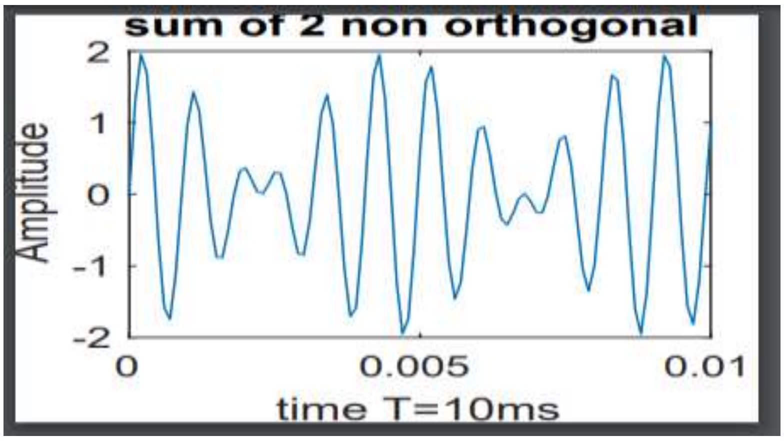


Figure-3: PLOT OF SUM OF TWO NON ORTHOGONAL SIGNALS

4. MODULATION/MAPPING OF BIT STREAM

4.1 BPSK Modulation and its Constellation Diagram

Binary Phase Shift keying is a digital modulation technique. As the name suggests, BPSK uses two phases i.e 0 degrees and 180 degrees. Using BPSK each symbol is capable of carrying one bit information. If the input message bit is 0 is it mapped to a symbol 1 and if the message bit is 1, it is mapped to a symbol -1.

Matlab Code for Bpsk constellation:

```
N=100;                          % no of bits

bbitg=randi([0 1],1,N);    % generating N random binary bits

bx=dig_mod_bpsk(bbitg);          % bpsk modulation (bits to symbols mapping)
```

```
scatterplot(bx)

figure;
% function for bpsk modulation
function bx=dig_mod_bpsk(bbitg)
for i=1:length(bbitg)
    if (bbitg(i)==0)
        bx(i)=1;
    else
        bx(i)=-1;
    end
end
```

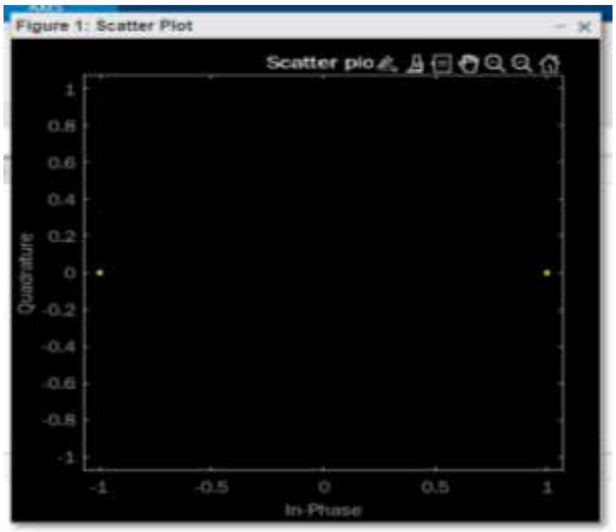


Figure-4: CONSTELLATION DIAGRAM OF BPSK

Using ‘randi’ command of matlab, 100 random binary bits are generated and are mapped to respective BPSK symbols using the BPSK modulation function. Scatter plot is used to display the constellation diagram of the mapped symbols.

4.2 QPSK Modulation and its Constellation Diagram

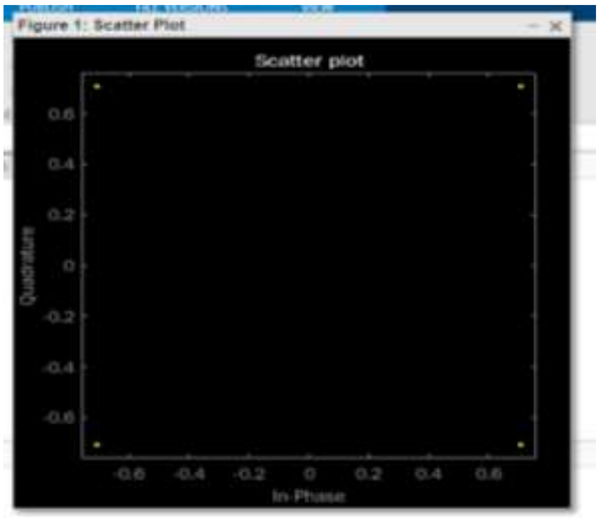
Quadrature Phase Shift keying is a digital modulation technique. As the name suggests, QPSK uses four phases i.e 0 degrees, 90 degrees, 180 degrees and 270 degrees. Using BPSK each symbol is capable of carrying two bit information.

Input message bits	Symbol mapped to
0	0..707+i*0.707
1	-0.707+i*0.707
1	-0.707-i*0.707
0	0.707-i*0.707

The input message bits are mapped to symbols in gray code fashion to minimize the occurrences of the bit error

Matlab Code for Qpsk constellation:

```
N=100;                          % no of bits
bbitg=randi([0 1],1,N);   % generating N random binary bits
qx=dig_mod_qpsk(bbitg);          % qpsk modulation (bits to symbols mapping)
scatterplot(qx)
figure;
% function for qpsk modulation
function qx=dig_mod_qpsk(bbitg)
qx=[ ];
for i=1:2:length(bbitg)
      if (bbitg(i)==0 && bbitg(i+1)==0)
      qxx=1+1i*1;
   elseif (bbitg(i)==0 && bbitg(i+1)==1)
      qxx=-1+1i*1;
   elseif (bbitg(i)==1 && bbitg(i+1)==1)
      qxx=-1-1i*1;
   elseif (bbitg(i)==1 && bbitg(i+1)==0)
      qxx=1-1i*1;
      end
      qx=[qx qxx];
end
end
```



5. ROLE OF IFFT AND FFT IN OFDM

As the OFDM basically uses multiple carriers to modulate the symbols by BPSK or QPSK modulation techniques over different carriers which are orthogonal to each other.

This is equivalent to IFFT and FFT formulae as shown below.

$$DFT(FFT):$$

$$X(k)=\sum_{n=0}^{N-1} x(n)\cdot e^{-j\left(\frac{2\pi}{N}\right)nk} \quad (k=0,1,\ldots,N-1)$$

$$IDFT(IFFT):$$

$$x(n)=\frac{1}{N}\sum_{k=0}^{N-1} X(k)\cdot e^{j\left(\frac{2\pi}{N}\right)nk} \quad (n=0,1,\ldots,N-1)$$

The modulated/mapped symbols are treated to be in frequency domain and are given as inputs to IFFT to obtain time domain OFDM sequence at the transmitter. Similarly at the receiver FFT is applied before demodulation to convert the time domain sequence to frequency domain modulated symbols before ML detection.

The above IFFT formula resembles transmitting the symbols present in x(n) signal over multiple orhtogonal carriers. Similarly FFT is used at receiver to recover back the transmitted data.

6. IMPLEMENTATION OF BER vs SNR CURVE with OFDM

Following steps are followed with respect to block diagram mentioned above to in order to implementt the BER vs SNR curve.

Step1:

Generate N random bits using randi function of the matlab and define the bandwidth, subcarrier bandwidth as shown in the code snippet below

```
clc
bpsk_ber_pr=[ ];
bpsk_ber_thry=[ ];
qpsk_ber_pr=[ ];
qpsk_ber_thry=[ ];
B=100; %bandwidth in kHz
f=1;   %subcarrier bandwidth in kHz
scb=B/f;% no of sub carriers
N=100; % no of bits
% generating N random bits
bbitg=randi([0 1],1,N);
```

Step 2:

Map the generated bits to symbols using a digital modulation technique such as BPSK or QPSK. Code snippets for mapping i.e modulation of bits to symbols using BPSK and QPSK is show separately above under section 4 i.e Modulation/Mapping using bit stream and the respective constellation diagrams are also plotted for BPSK and QPSK under figure 4 and 5 respectively. Plot the Frequency plot of the modulated symbols separately for BPSK and QPS as shown in the code snippet below

```
% bpsk and qpsk modulation (bits to symbols mapping)
bx=dig_mod_bpsk(bbitg);
qx=dig_mod_qpsk(bbitg);
scatterplot(qx)
figure;
%subplot(2,2,1);
%bx_shift=ifftshift(bx);
freq1=-49:1:50
stem(freq1,(abs(bx)));
 %axis([-50 50 -1 1]);
xlabel('frequency');
ylabel('bpsk symbols')
title('frequency plot before ofdm with bpsk mod');
figure;
```

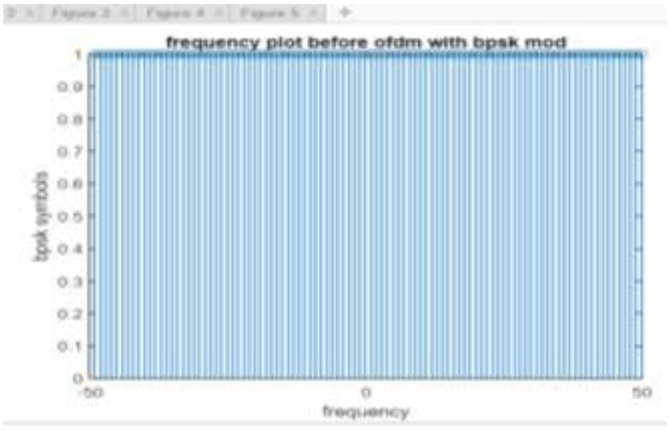


Figure-6: Frequency plot for BPSK symbols before OFDM

```
%subplot(2,2,2);
%qx_shift=ifftshift(qx);
freq=-49:2:50
stem(freq,(abs(qx)));
 xlabel('freq');
ylabel('qpsk symbols');
title('frequency plot before ofdm with qpsk mod');
figure;
```

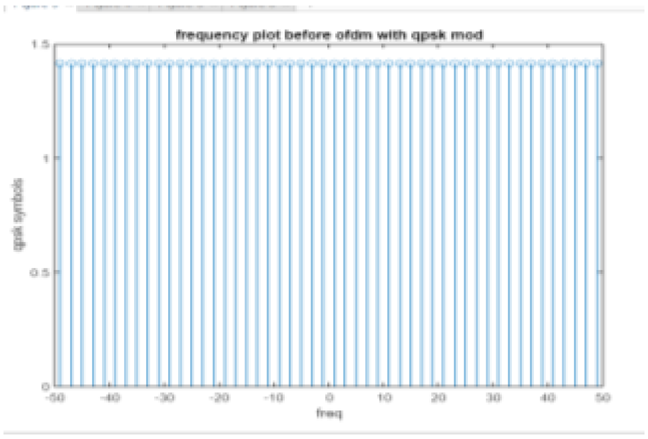


Figure-7: Frequency plot for QPSK symbols before OFDM

Step 3:

Conversion of Serial data in to parallel:. This is basically done to allot a subcarrier frequency differently to each set of symbols that are to be sent in a single sub carrier. This is done using IFFT command which allocates different/separate frequencies to each parallel transformed data. Generated using The resultant data is said to be OFDM data. Reshape this data to original length after application of IFFT i.e before transmitting over AWGN channel

The time plots of OFDM data are plotted for BPSK and QPSK techniques using the matlab code snippet is shown below .

```
% serial to parallel
bsp1=reshape(bx,scb,[ ]);
a=N/(2*scb);
qsp1=reshape(qx,scb,[ ]);

% ifft application on bpsk and qpsk modulated symbols
bifft=ifft(bsp1);
qifft=ifft(qsp1);

%parallel to serial
bps1=reshape(bifft,1,[ ]);
qps1=reshape(qifft,1,[ ]);

%subplot(2,2,1);
plot(real(bps1));
xlabel('time');
ylabel('ofdm symbols')
title('ofdm Time axis plot with bpsk mod');
figure;
```

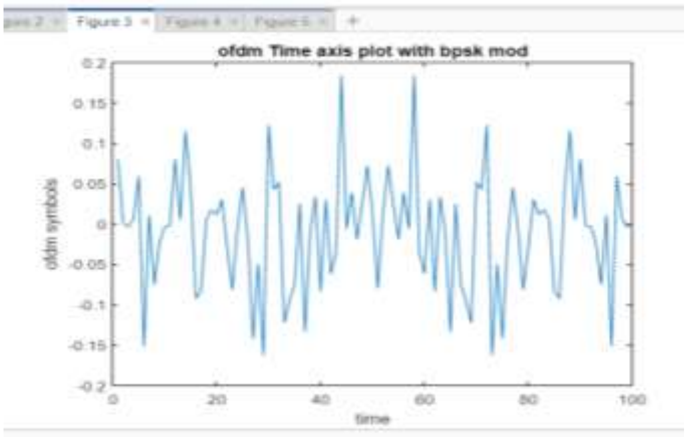


Figure-8: OFDM Time plot with BPSK modulation

```
%subplot(2,2,2);
plot(real(qps1));
xlabel('time');
ylabel('ofdm symbols')
title('ofdm Time axis plot with qpsk mod');
figure;
```

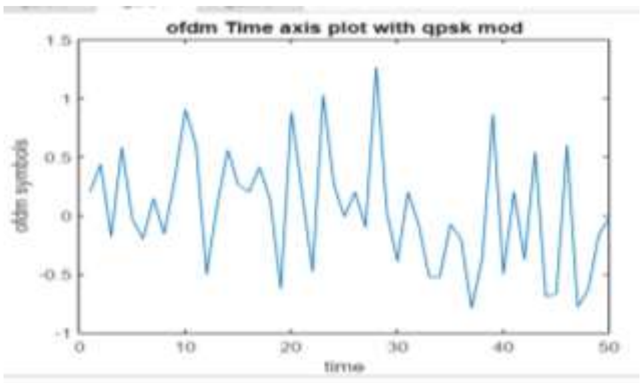


Figure-8: OFDM Time plot with QPSK modulation

Step 4 :

The data transmitted over AWGN channel where the channel noise gets added to the transmitted data.

```
for EbNo_dB=0:1:10
EbNo_L=10^(EbNo_dB/10); %converting Eb/N dB toEb/No linear
bpsk_snr_L=EbNo_L;    %defining bpsk snr linear
qpsk_snr_L=(2*EbNo_L);  %defining qpsk snr linear
bpsk_ser_count=0; %initialising error count to zero
qpsk_ber_count=0; %initialising error count to zero
```

Addition of AWGN Channel noise :AWGN is an additive in nature. It is calculated using the average energy of the generated OFDM bits and is used in the standard deviation formula, which is multiplied with the normal/Gaussian noise generated. This is done in order to add the noise to the transmitted bits.

The below code snippet shows the energy calculation and addition of noise to the transmitted data

```
% calucating avg symbol energy for ofdm symbols generated by bpsk and qpsk modulation
Ebpsk_ofdm=energybpsk(bps1);
Eqpsk_ofdm=energyqpsk(qps1);
% function to calculate avg symbol Energy of ofdm symbols with bpsk modulation
function Ebpsk_ofdm=energybpsk(bps1)
  E_ofdm=0;
  for i=1:length(bps1)
      E_ofdm=E_ofdm+(abs(bps1(i)))*(abs(bps1(i)));
  end
  Ebpsk_ofdm=E_ofdm/length(bps1);
end
% function to calculate avg symbol Energy of ofdm symbols with bpsk modulation
function Eqpsk_ofdm=energyqpsk(qps1)
  Eq_ofdm=0;
  for i=1:length(qps1)
      Eq_ofdm=Eq_ofdm+(abs(qps1(i)))*(abs(qps1(i)));
  end
  Eqpsk_ofdm=Eq_ofdm/length(qps1);
end
end

 % bpsk standard deviation
sd_b=sqrt(Ebpsk_ofdm/(2*bpsk_snr_L));
sd_qpsk=sqrt(Eqpsk_ofdm/(2*qpsk_snr_L));
for iter=1:1000
% generating AWGN noise
bn=sd_b*(randn(1,N)+1i*randn(1,N));
qn=sd_qpsk*(randn(1,N/2)+1i*randn(1,N/2));
% addition of AWGN noise
by=bps1+bn; %for bpsk symbols
qy=qps1+qn; %for qpsk symbols
```

Step 5:

At the receiver to retrieve the original data, initially the received data is converted from time domain to frequency domain using FFT command with proper serial to parallel and parallel to serial conversion before and after FFT application respectively.

```
% serial to parallel
bsp2=reshape(by,scb,[]);
```

```
qsp2=reshape(qy,scb,[]);
% fft on received data
bfft=fft(bsp2);
qfft=fft(qsp2);
%parallel to serial
bps2=reshape(bfft,1,[ ]);
qps2=reshape(qfft,1,[ ]);
```

Step 6:
At this stage, the data is in frequency domain. Now using ML Detector the estimated symbols are found separately for BPSK and QPSK. The concept of ML detection is as follows:
The nearest neighbour of the received symbol is considered to be the estimated symbol that has been transmitted .
The code snippets for ML detection for BPSK nad QPSK are show separately below;

```
% ML detection
bx_cp=MLdetcbpsk(bps2);
qx_cp=MLdetcqpsk(qps2);
% function for bpsk MLdetection
function bx_cp=MLdetcbpsk(by)

        for i=1:length(by)
        d1=sqrt(((real(by(i))-1))^2+(imag(by(i)))^2);
        d2=sqrt(((real(by(i))+1))^2+(imag(by(i)))^2);
        if (d1<d2)
          bx_cp(i)=1;
        else
          bx_cp(i)=-1;
        end
        end

end
% function for qpsk MLdetection
function qx_cp=MLdetcqpsk(qps2)
qx_cp=[ ];
   for i=1:length(qps2)
      d1=sqrt((real(qps2(i))-1)^2+(imag(qps2(i))-1)^2);
      d2=sqrt((real(qps2(i))+1)^2+(imag(qps2(i))-1)^2);
      d3=sqrt((real(qps2(i))+1)^2+(imag(qps2(i))+1)^2);
      d4=sqrt((real(qps2(i))-1)^2+(imag(qps2(i))+1)^2);
      x=[d1 d2 d3 d4];
      min_dis=min(x);
      if (min_dis==d1)
        qxx_cp=1+1i*1;
      elseif (min_dis==d2)
```

```
            qxx_cp=-1+1i*1;
         elseif (min_dis==d3)
            qxx_cp=-1-1i*1;
          elseif (min_dis==d4)
            qxx_cp=1-1i*1;
        end
        qx_cp=[qx_cp qxx_cp];
    end
end
```

Step 7:
The estimated symbols are then demodulated to estimated bits using the demodulation techniques BPSK and QPSK. This demodulated bits are then checked/compared with the original message bits that are generated at the transmitter to find the bit error rate.

```
% bpsk and qpsk demodulation
bpskbx_cp=dig_demod_bpsk(bx_cp);
qpskqx_cp=dig_demod_qpsk(qx_cp);
% function for bpsk demodulation

function bpskbx_cp=dig_demod_bpsk(bx_cp)

for i=1:length(bx_cp)

    if (bx_cp(i)==1)

       bpskbx_cp(i)=0;

    else

       bpskbx_cp(i)=1;
    end
end
end
% function for qpsk demodulation
function qpskqx_cp=dig_demod_qpsk(qx_cp)
qpskqx_cp=[ ];
for i=1:length(qx_cp)
    if (qx_cp(i)==1+1i*1)
           qpskqxx_cp=[0 0];
    elseif (qx_cp(i)==-1+1i*1)
      qpskqxx_cp=[0 1];
    elseif (qx_cp(i)==-1-1i*1)
        qpskqxx_cp=[1 1];
    elseif (qx_cp(i)==1-1i*1)
        qpskqxx_cp=[1 0];
        end
    qpskqx_cp=[qpskqx_cp qpskqxx_cp];
end
```

Step 8:

Find the bit error rate for BPSK and QPSK seperatly and compare the performance using the theorical BER vs SNR and is is plotted using semilogy function of matlb and lengend is used to differentiate the curves from one another.

```
% calculating no of bits in error
bpsk_ber_count=bpsk_ser_count+sum(bbitg~=bpskbx_cp);
qpsk_ber_count=qpsk_ber_count+sum(bbitg~=qpskqx_cp);
end
bpsk_ber=bpsk_ber_count/(N);
bpsk_ber_pr=[bpsk_ber_pr bpsk_ber];% ber of bpsk simulated
bpsk_ber_try1=qfunc(sqrt(bpsk_snr_L*2));
bpsk_ber_thry=[bpsk_ber_thry bpsk_ber_try1]; % ber of bpsk theoritical
qpsk_ber=qpsk_ber_count/(N/2);
qpsk_ber_pr=[qpsk_ber_pr qpsk_ber]; % ber of qpsk simulated
qpsk_ber_try1=qfunc(sqrt(bpsk_snr_L*2));
qpsk_ber_thry=[qpsk_ber_thry qpsk_ber_try1];  % ber of qpsk theoritical
end
%BER vs snr plot
EbNo_dB=0:1:10;
semilogy(EbNo_dB,qpsk_ber_pr,'-k',EbNo_dB,qpsk_ber_thry,'-v',EbNo_dB,bpsk_ber_pr,'-
o',EbNo_dB,bpsk_ber_thry,'-x');
xlabel('EbNodB');
ylabel('BER/SER');
title('BER vs SNR');
legend('qpskber prac','qpskber thry','bpskber prac','bpskbersim thry');
grid on;
```

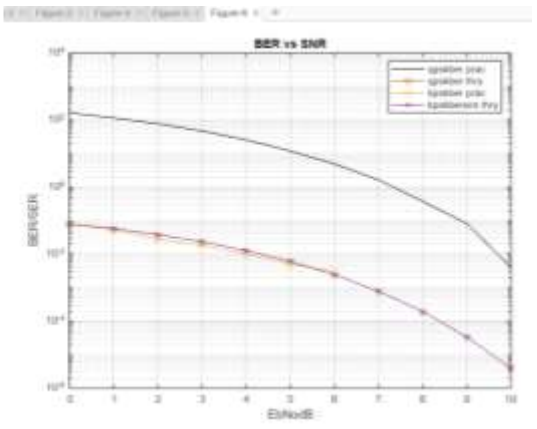


Figure-9: BER vs SNR curve