CAIRO UNIVERSITY
FACULTY OF ENGINEERING
EECE 3RD YEAR

COMMUNICATION PROJECT 3

ID	SEC	BN	Name
9203221	3	30	محمد خالد الرفاعي عبد اللطيف
9202806	2	37	عبدالرحمن هشام السيد محمود

1.BPSK

General Rule: $S_i(t) = \sqrt{\frac{2E}{T}}\cos\left(2\pi F_c t + (i-1)\frac{2\pi}{M}\right) \rightarrow$ case BPSK $\rightarrow M = 2$

$$S_i(t) = \sqrt{\frac{2E}{T}}\cos(2\pi F_c t + (i-1)\pi) \quad \rightarrow \quad \text{Basis functions} \quad \rightarrow \quad \phi = \sqrt{\frac{2}{T}}\cos(2\pi F_c t)$$

Our symbols in BPSK:

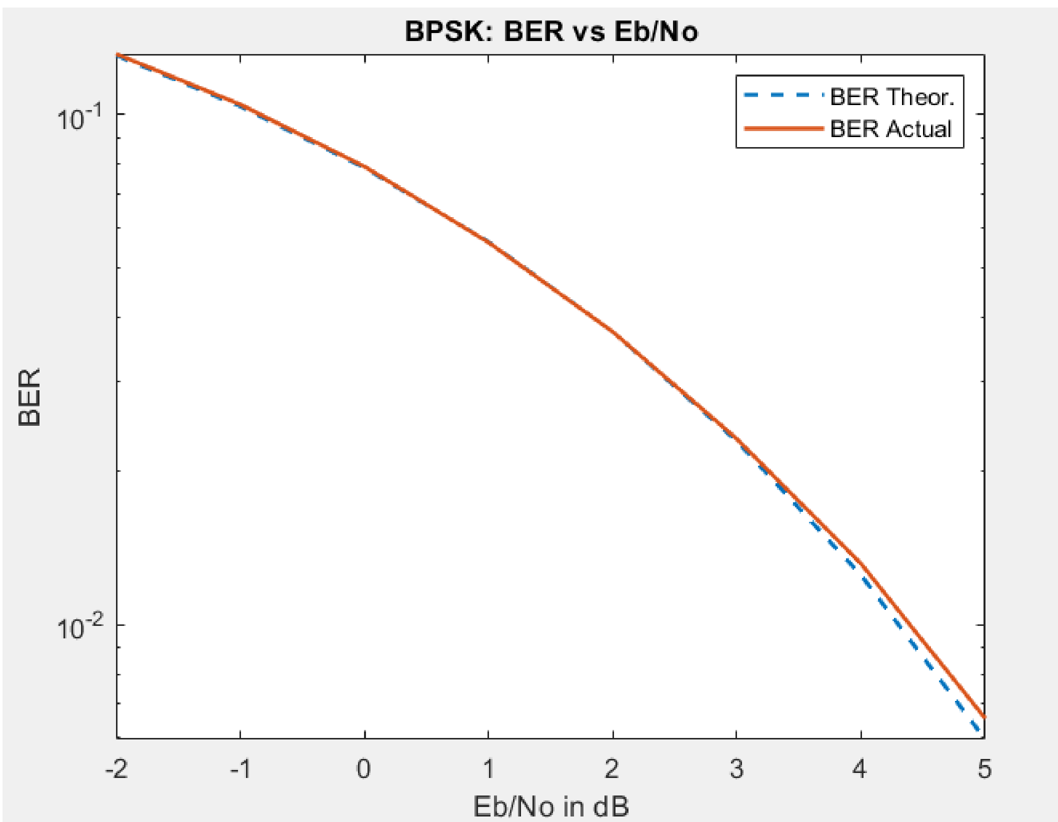
$S_1(t) = \sqrt{E}\,\phi$ $\qquad\qquad$ $S_2(t) = \sqrt{E}\,\phi$

Where $E = \frac{(1)^2 + (1)^2}{2} = 1$

Our symbols encoded in code:

$S_1(t) = 1$ $\qquad\qquad$ $S_2(t) = -1$

Plotting BER:



2.QPSK

General Rule: $S_i(t) = \sqrt{\frac{2E}{T}} \cos\left(2\pi F_c t + (i-1)\frac{2\pi}{M}\right) \rightarrow$ case BPSK $\rightarrow \quad M = 4$

$$S_i(t) = \sqrt{\frac{2E}{T}} \cos\left(2\pi F_c t + (i-1)\frac{\pi}{2}\right)$$

Basis functions $\rightarrow \qquad \phi_1 = \sqrt{\frac{2}{T}}\,(2\pi F_c t) \quad \rightarrow \qquad \phi_2 = \sqrt{\frac{2}{T}} \cos(2\pi F_c t)$

Our symbols in BPSK:

$S_1(t) = -\sqrt{E}\,\phi_1 - \sqrt{E}\,\phi_2 \qquad\qquad S_2(t) = -\sqrt{E}\,\phi_1 + \sqrt{E}\,\phi_2$

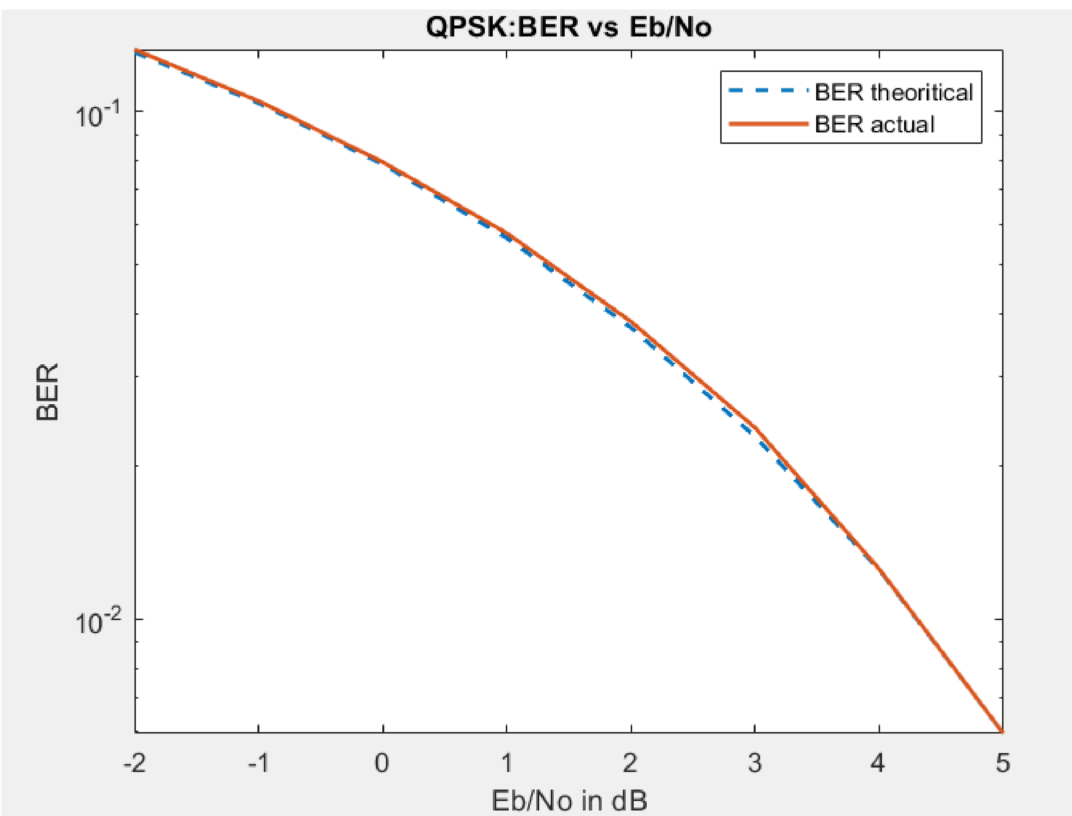
$S_3(t) = +\sqrt{E}\,\phi_1 + \sqrt{E}\,\phi_2 \qquad\qquad S_4(t) = +\sqrt{E}\,\phi_1 - \sqrt{E}\,\phi_2$

Where $E = \frac{4*((1)^2 + (1)^2)}{4*2} = 1$

Our symbols encoded in code:

$S_1(t) = -1 - i\,, \qquad S_2(t) = -1 + i\,, \qquad S_3(t) = 1 + i\,, \qquad S_4(t) = 1 - i$

Plotting BER:



3.8PSK

General Rule: $S_i(t) = \sqrt{\frac{2E}{T}} \cos\left(2\pi F_c t + (i-1)\frac{2\pi}{M}\right) \rightarrow$ case BPSK $\rightarrow M = 8$

$$S_i(t) = \sqrt{\frac{2E}{T}} \cos\left(2\pi F_c t + (i-1)\frac{\pi}{4}\right)$$

Basis functions $\rightarrow \quad \phi_1 = \sqrt{\frac{2}{T}}\,(2\pi F_c t) \quad \rightarrow \quad \phi_2 = \sqrt{\frac{2}{T}} \cos(2\pi F_c t)$

Our symbols in BPSK:

$S_1(t) = -\frac{\sqrt{E}}{2}\,\phi_1 - \frac{\sqrt{E}}{2}\,\phi_2$ $\qquad$ $S_2(t) = -\frac{\sqrt{E}}{2}\,\phi_1 + \frac{\sqrt{E}}{2}\,\phi_2$

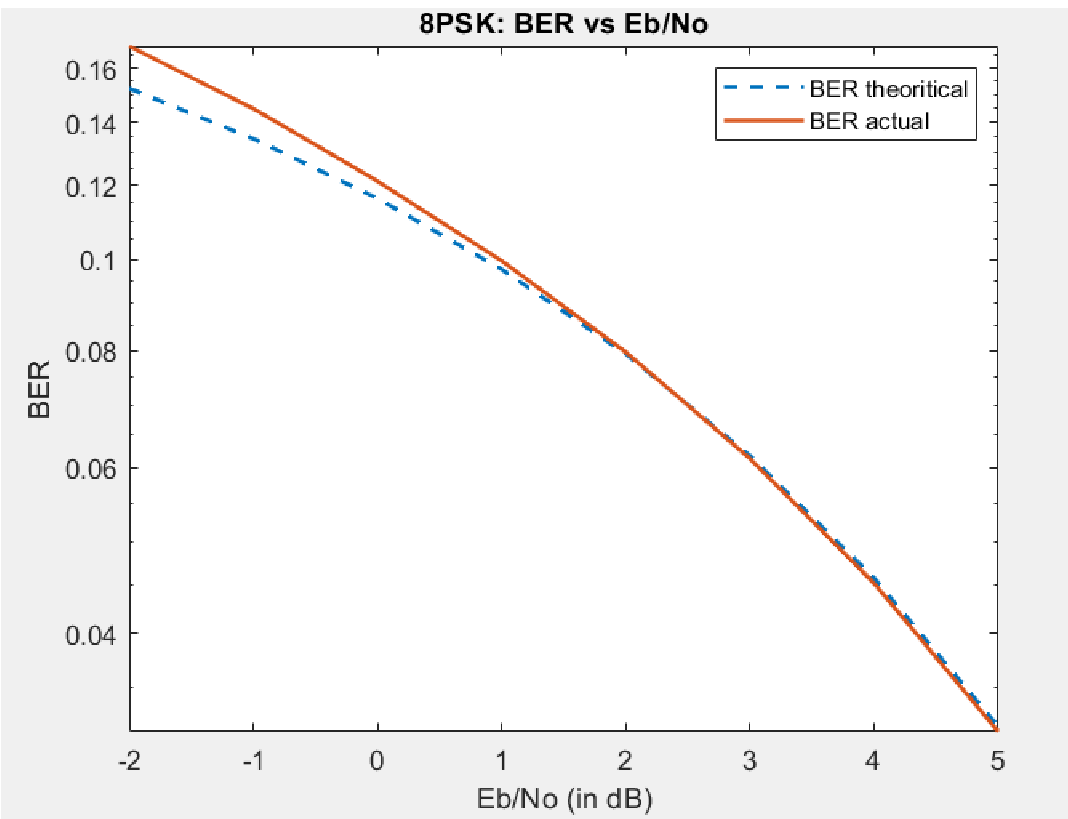
$S_1(t) = -\frac{\sqrt{E}}{2}\,\phi_1 - \frac{\sqrt{E}}{2}\,\phi_2$ $\qquad$ $S_2(t) = -\frac{\sqrt{E}}{2}\,\phi_1 + \frac{\sqrt{E}}{2}\,\phi_2$

Where $E = \frac{4*((1)^2 + (1)^2)}{3*8} = \frac{1}{3}$

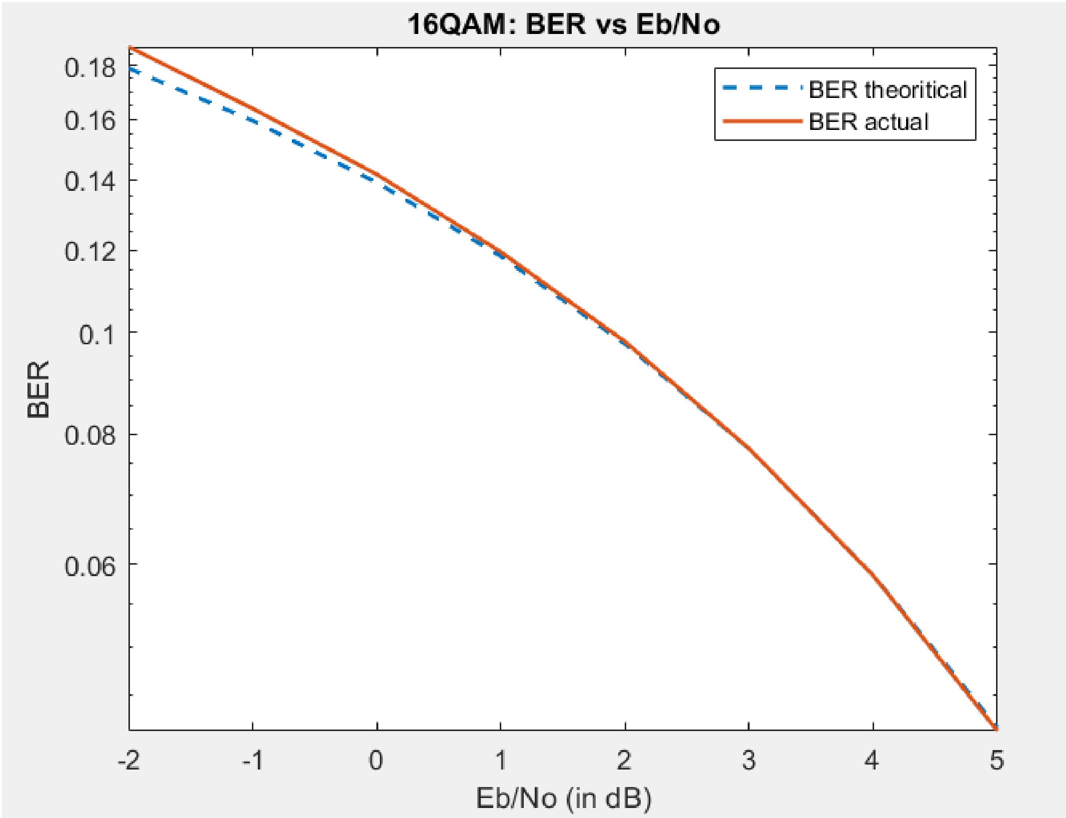
Our symbols encoded in code:

$S_1(t) = 1, S_2(t) = -1, S_3(t) = -i, S_4(t) = i, S_5(t) = \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i, S_6(t) = -\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}i, S_7(t) = -\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i, S_8(t) = \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}i$

Plotting BER:



4.16QAM

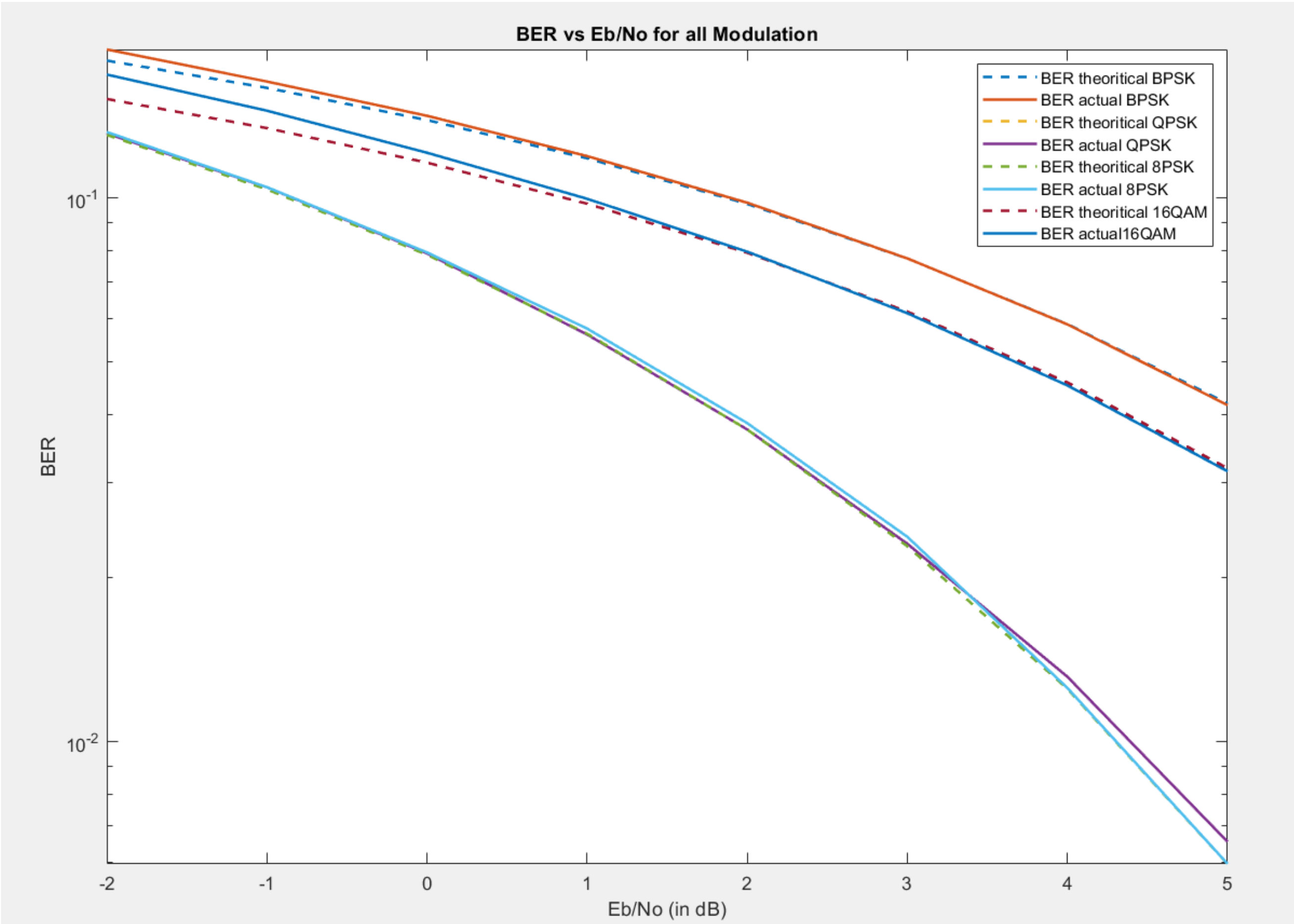


5.COMPARING ALL MODULATIONS

General theoretical BER Rules:

1. BPSK: $BER = \frac{1}{2} * erfc\left(\sqrt{\frac{E_b}{N_0}}\right)$ $\quad$ $where\ E_b = 1$
2. $QPSK$: $BER = \frac{1}{2} * erfc\left(\sqrt{\frac{E_b}{N_0}}\right)$ $\quad$ $where\ E_b = 1$
3. 8PSK: $BER = \frac{1}{log_2 M} * erfc\left(\sqrt{\frac{E_b\, log_2 M}{N_0}}\sin\left(\frac{\pi}{M}\right)\right)$ $\quad$ $where\ E_b = 1, M = 8$
4. 16QAM: $BER = \frac{1.5}{log_2 M} * erfc\left(\sqrt{\frac{E_b}{2.5\, N_0}}\sin\left(\frac{\pi}{M}\right)\right)$ $\quad$ $where\ E_b = 1$

Plotting BER:



Comments on all modulations:

1- By increasing Number of bits, BER Theoretical & BER Actual are almost identical in all modulation cases.

2-In BPSK & QPSK Actual are almost identical as expected from the theoretical but as we see the QPSK have double rate compare to BPSK for same BW.

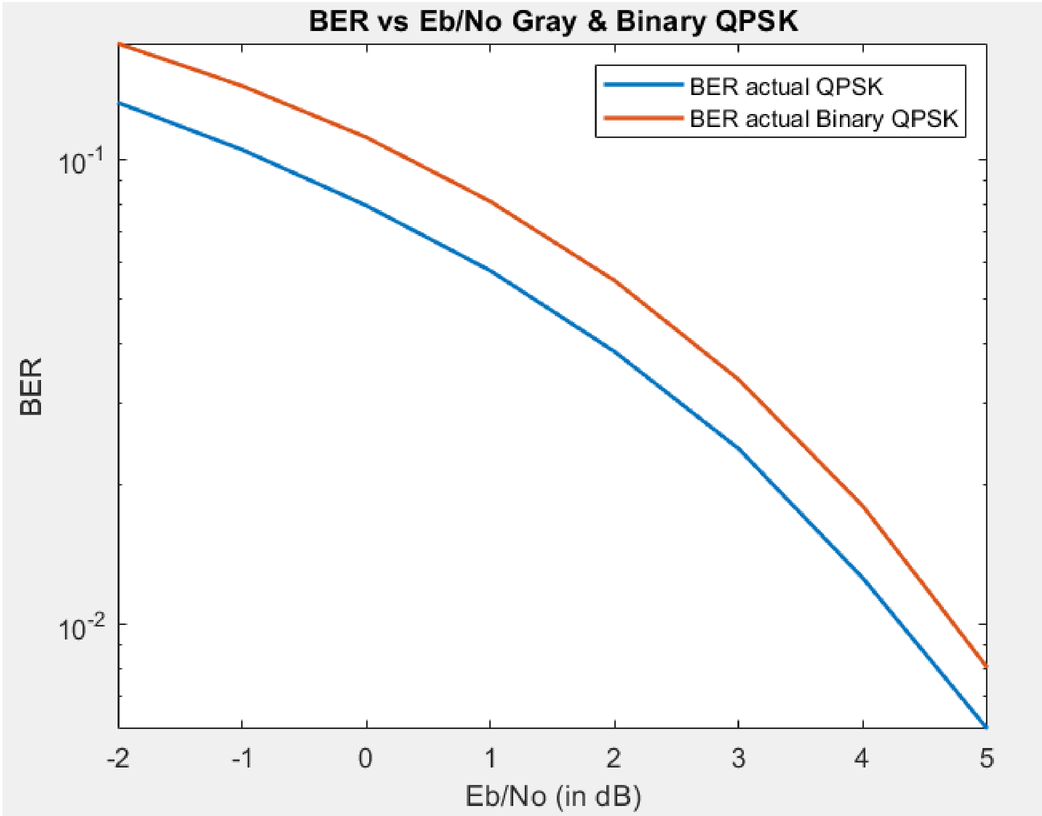
3-BER 16QAM>BER of 8PSK > BER of BPSK & QPSK as by increasing number of symbol that distance between symbols decrease and their decision region become narrower so the probability of getting wrong symbol increase.

4- As No. of bits/symbol increase → No. of symbols increase → as No. of symbols increase → BW decrease & Energy increase

5- To maintain same BER, we have to maintain the distance between as BER

depends on the distance between the symbols in any modulation technique.

6.COMPARING GRAY QPSK AND BINARY QPSK MODULATIONS



7.THE FIRST CODE

```
clc
clear all
close
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Inlialzation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
No_bits = 120000;
Bits = randi([0 1],1,No_bits);
SNR = -2:5; %in dB
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BPSK
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Eb_BPSK = 1;
BPSK_Bits_Tobe_Transmitted=zeros(1,No_bits);
constellation_points_Bpsk=[1 -1];
%%%% BPSK Mapper %%%%
%Encoding bits into 1 and -1:
for n = 1: No_bits
if Bits(n)==1
BPSK_Bits_Tobe_Transmitted(n)=constellation_points_Bpsk(1);
else
BPSK_Bits_Tobe_Transmitted(n)=constellation_points_Bpsk(2);
end
end
%%% BPSK Channel %%%%
AWGN = randn(1,No_bits);
No_BPSK = Eb_BPSK./(10.^(SNR./10));
%%%% BPSK DeMapper %%%%
BER_actual_BPSK = zeros(1,length(No_BPSK));
Estimated_bits_BPSK=zeros(1,No_bits);
%Generating Noise:
for n = 1: length(No_BPSK)
Noise_vector_BPSK = sqrt(No_BPSK(n)/2)*AWGN;
Rx_Symbols_BPSK_After_Noise = BPSK_Bits_Tobe_Transmitted + Noise_vector_BPSK;
```

```
%%%Estimate bits and BER calculation:%%%
for ik = 1: No_bits
if Rx_Symbols_BPSK_After_Noise(ik)>0
Estimated_bits_BPSK(ik)=1;
else
Estimated_bits_BPSK(ik)=0;
end
end
[Number_of_Error_Bits_BPSK, BER_actual_BPSK(n)] = symerr(Estimated_bits_BPSK,Bits);
end
%%% BER calculations %%%
BER_theoritical_BPSK = 0.5*erfc(sqrt(Eb_BPSK./No_BPSK));
%%%% plots %%%%

figure(1)
semilogy(SNR,BER_theoritical_BPSK,'--','linewidth',1.5)
hold on
semilogy(SNR,BER_actual_BPSK,'linewidth',1.5)
legend('BER Theor.','BER Actual')
title(' BPSK: BER vs Eb/No ')
xlabel('Eb/No in dB')
ylabel('BER')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% QPSK
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Eb_QPSK = 1;
%%% QPSK Mapper %%%
Bits_QPSK = transpose(reshape(Bits,[2 No_bits/2]));
QPSK_Bits_Tobe_Transmitted=zeros(No_bits/2,2);
for n = 1: No_bits/2
for j=1:2
if Bits_QPSK(n,j)==0
QPSK_Bits_Tobe_Transmitted(n,j)=-1;
else
QPSK_Bits_Tobe_Transmitted(n,j)=1;
end
end
end

QPSK_Bits_Tobe_Transmitted =QPSK_Bits_Tobe_Transmitted(:,1)+1i*QPSK_Bits_Tobe_Transmitted(:,2);
%%% QPSK Channel %%%
AWGN = transpose(randn(1,No_bits/2)+1i*randn(1,No_bits/2));
No_QPSK = Eb_QPSK./10.^(SNR./10);
%%% QPSK DeMapper %%%
BER_Actual_QPSK = zeros(1,length(No_QPSK));
stimated_bits_QPSK=zeros(No_bits/2,2);
for n = 1: length(No_QPSK)
Noise_vector_QPSK = sqrt(No_QPSK(n)/2)*AWGN;
Rx_Symbols_QPSK_Ater_Noise = QPSK_Bits_Tobe_Transmitted + Noise_vector_QPSK;
for ik = 1: No_bits/2
if real(Rx_Symbols_QPSK_Ater_Noise(ik))>0
Estimated_bits_QPSK(ik,1)=1;
else
Estimated_bits_QPSK(ik,1) = 0;
end
if imag(Rx_Symbols_QPSK_Ater_Noise(ik))>0
Estimated_bits_QPSK(ik,2)=1;
else
Estimated_bits_QPSK(ik,2)=0;
end
end
[Number_of_Error_Bits_QPSK, BER_Actual_QPSK(n)] = symerr(Estimated_bits_QPSK,Bits_QPSK);
end
%### BER calculations ###%
BER_theoritical_QPSK = 0.5*erfc(sqrt(Eb_QPSK./No_QPSK));
%### BER plots ###%
figure(2)
semilogy(SNR,BER_theoritical_QPSK,'--','linewidth',1.5)
hold on
semilogy(SNR,BER_Actual_QPSK,'linewidth',1.5)
```

```
legend('BER theoritical','BER actual')
title('QPSK:BER vs Eb/No')
xlabel('Eb/No in dB')
ylabel('BER')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 8PSK
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Eb_8PSK = 1/3;
Bits_8PSK = transpose(reshape(Bits,[3 No_bits/3]));
%%% 8 PSK Mapper %%%
MPSK8_Bits_Tobe_Transmitted = zeros(1,No_bits/3);
encoding_Signal = [1 + 0i; cos(pi/4) + sin(pi/4)*1i;-cos(pi/4) + sin(pi/4)*1i; 1i;
cos(pi/4) - sin(pi/4)*1i;-1i; -1; -cos(pi/4) - sin(pi/4)*1i];
for m = 1:(No_bits/3)
if Bits_8PSK(m,:) == [0 0 0]
MPSK8_Bits_Tobe_Transmitted(m) = encoding_Signal(1);
elseif Bits_8PSK(m,:) == [0 0 1]
MPSK8_Bits_Tobe_Transmitted(m) = encoding_Signal(2);
elseif Bits_8PSK(m,:) == [0 1 0]
MPSK8_Bits_Tobe_Transmitted(m) = encoding_Signal(3);
elseif Bits_8PSK(m,:) == [0 1 1]
MPSK8_Bits_Tobe_Transmitted(m) = encoding_Signal(4);
elseif Bits_8PSK(m,:) == [1 0 0]
MPSK8_Bits_Tobe_Transmitted(m) = encoding_Signal(5);
elseif Bits_8PSK(m,:) == [1 0 1]
MPSK8_Bits_Tobe_Transmitted(m) = encoding_Signal(6);
elseif Bits_8PSK(m,:) == [1 1 0]
MPSK8_Bits_Tobe_Transmitted(m) = encoding_Signal(7);
else
MPSK8_Bits_Tobe_Transmitted(m) = encoding_Signal(8);
end
end
%%% 8 PSK Channel %%%
AWGN = randn(1,No_bits/3)+1i*randn(1,No_bits/3);
No_8PSK = Eb_8PSK./10.^(SNR./10);
Distance_between_Bits_Symbol = zeros(1,8);
Estimated_bits_8PSK = zeros(No_bits/3,3);
%### DeMapper ###%
BER_Actual_8PSK = zeros(1,length(No_8PSK));
for k = 1: length(No_8PSK)
Noise_vector_8PSK = sqrt(No_8PSK(k)/2)*AWGN;
Rx_Symbols_8PSK = MPSK8_Bits_Tobe_Transmitted + Noise_vector_8PSK;
for j = 1:(No_bits/3)
for n=1:8
Distance_between_Bits_Symbol(n)=abs(Rx_Symbols_8PSK(j)-encoding_Signal(n));
end
[distance,index] = min(Distance_between_Bits_Symbol);
if index == 1
Estimated_bits_8PSK(j,:) = [0 0 0];
elseif index == 2
Estimated_bits_8PSK(j,:) = [0 0 1];
elseif index == 3
Estimated_bits_8PSK(j,:) = [0 1 0];
elseif index == 4
Estimated_bits_8PSK(j,:) = [0 1 1];
elseif index == 5
Estimated_bits_8PSK(j,:) = [1 0 0];
elseif index == 6
Estimated_bits_8PSK(j,:) = [1 0 1];
elseif index == 7
Estimated_bits_8PSK(j,:) = [1 1 0];
elseif index == 8
Estimated_bits_8PSK(j,:) = [1 1 1];
end
end
[Number_of_Error_Bits_8PSK, BER_Actual_8PSK(k)] = symerr(Estimated_bits_8PSK,Bits_8PSK);
end
X_8PSK= [1 , cos(pi/4) , -cos(pi/4), 0 , cos(pi/4), 0, -1, -cos(pi/4)];
Y_8PSK= [0, sin(pi/4), sin(pi/4) , 1 , - sin(pi/4) , -1 , 0, - sin(pi/4)];
```

```
%### BER calculations ###%
BER_theoritical_8PSK = (1/3)*erfc(sin(pi/8).*sqrt(3*Eb_8PSK./No_8PSK));
%### BER plots ###%
figure(3)
semilogy(SNR,BER_theoritical_8PSK,'--','linewidth',1.5)
hold on
semilogy(SNR,BER_Actual_8PSK,'linewidth',1.5)
legend('BER theoritical','BER actual')
title('8PSK: BER vs Eb/No')
xlabel('Eb/No (in dB)')
ylabel('BER')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 16-QAM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Eb_16QAM = 2.5;
Bits_16QAM = transpose(reshape(Bits,[4 No_bits/4]));
%%% 16QAM Mapper %%%
QAM16_Bits_Tobe_Transmitted = zeros(1,No_bits/4);
for L = 1:(No_bits/4)
if Bits_16QAM(L,1:2) == [1 1]
real_part = 1;
elseif Bits_16QAM(L,1:2) == [1 0]
real_part = 3;
elseif Bits_16QAM(L,1:2) == [0 1]
real_part = -1;
elseif Bits_16QAM(L,1:2) == [0 0]
real_part = -3;
end
if Bits_16QAM(L,3:4) == [1 1]
imag_part = 1;
elseif Bits_16QAM(L,3:4) == [1 0]
imag_part = 3;
elseif Bits_16QAM(L,3:4) == [0 1]
imag_part = -1;
elseif Bits_16QAM(L,3:4) == [0 0]
imag_part = -3;
end
QAM16_Bits_Tobe_Transmitted(L) = real_part + imag_part*1i;
end
%%% 16QAM Channel %%%
Noise = randn(1,No_bits/4)+1i*randn(1,No_bits/4);
No_16QAM = Eb_16QAM./10.^(SNR./10);
%%% 16QAM DeMapper %%%
BER_Actual_16QAM = zeros(1,length(No_16QAM));
for F = 1: length(No_16QAM)
noise_vector_16QAM = sqrt(No_16QAM(F)/2)*Noise;
Rx_Symbols_16QAM_After_Noise = QAM16_Bits_Tobe_Transmitted + noise_vector_16QAM;
Estimated_bits_16QAM = zeros(No_bits/4,4);
for j = 1:(No_bits/4)
real_part = real(Rx_Symbols_16QAM_After_Noise (j));
imag_part = imag(Rx_Symbols_16QAM_After_Noise (j));
if (real_part >= 0) && (real_part < 2)
bit_1_2 = [1 1];
elseif real_part >= 2
bit_1_2 = [1 0];
elseif (real_part < 0) && (real_part >= -2)
bit_1_2 = [0 1];
elseif real_part < -2
bit_1_2 = [0 0];
end
if (imag_part >= 0) && (imag_part < 2)
bit_3_4 = [1 1];
elseif imag_part >= 2
bit_3_4 = [1 0];
elseif (imag_part < 0) && (imag_part >= -2)
bit_3_4 = [0 1];
elseif imag_part < -2
bit_3_4 = [0 0];
end
```

```matlab
Estimated_bits_16QAM(j,:) = [bit_1_2, bit_3_4];
end
[Number_of_Error_Bits_16QAM, BER_Actual_16QAM(F)] = symerr(Estimated_bits_16QAM,Bits_16QAM);
end
%### BER calculations ###%
BER_theoritical_16QAM = (3/8)*erfc(sqrt((Eb_16QAM/2.5)./No_16QAM));
X_16QAM= [1 , 1 , 3, 3 ,-1, -1, -3,-3 , 1 , 1 , 3, 3, -1, -1 -3, -3];
Y_16QAM= [1, 3, 1 , 3 , 1 , 3 , 1, 3 ,-1, -3, -1, -3, -1, -3, -1,-3] ;
%### BER plots ###%
figure(4)
semilogy(SNR,BER_theoritical_16QAM,'--','linewidth',1.5)
hold on
semilogy(SNR,BER_Actual_16QAM,'linewidth',1.5)
legend('BER theoritical','BER actual')
title('16QAM: BER vs Eb/No')
xlabel('Eb/No (in dB)')
ylabel('BER')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plotting all modulation BER:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(5)
semilogy(SNR,BER_theoritical_16QAM,'--','linewidth',1.5)
hold on
semilogy(SNR,BER_Actual_16QAM,'linewidth',1.5)
hold on
semilogy(SNR,BER_theoritical_BPSK,'--','linewidth',1.5)
hold on
semilogy(SNR,BER_actual_BPSK,'linewidth',1.5)
hold on
semilogy(SNR,BER_theoritical_QPSK,'--','linewidth',1.5)
hold on
semilogy(SNR,BER_Actual_QPSK,'linewidth',1.5)
hold on
semilogy(SNR,BER_theoritical_8PSK,'--','linewidth',1.5)
hold on
semilogy(SNR,BER_Actual_8PSK,'linewidth',1.5)
legend('BER theoritical BPSK','BER actual BPSK','BER theoritical QPSK','BER actual QPSK','BER
theoritical 8PSK','BER actual 8PSK','BER theoritical 16QAM','BER actual16QAM');
title('BER vs Eb/No for all Modulation')
xlabel('Eb/No (in dB)')
ylabel('BER')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Binary_QPSK
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Eb_Binary_QPSK = 1;
%%% Binary_QPSK Mapper %%%
Bits_Binary_QPSK = transpose(reshape(Bits,[2 No_bits/2]));
Binary_QPSK_Bits_Tobe_Transmitted=zeros(No_bits/2,1);
constellation_points_Binary_Qpsk=[1+i 1-i -1+i -1-i];
for n = 1: No_bits/2

if (Bits_Binary_QPSK(n,1)==0 && Bits_Binary_QPSK(n,2)==0)
Binary_QPSK_Bits_Tobe_Transmitted(n,1)= constellation_points_Binary_Qpsk(4);
end
    if (Bits_Binary_QPSK(n,1)==0 && Bits_Binary_QPSK(n,2)==1)
Binary_QPSK_Bits_Tobe_Transmitted(n,1)= constellation_points_Binary_Qpsk(3);
    elseif (Bits_Binary_QPSK(n,1)==1 && Bits_Binary_QPSK(n,2)==1)
Binary_QPSK_Bits_Tobe_Transmitted(n,1)= constellation_points_Binary_Qpsk(2);
    elseif (Bits_Binary_QPSK(n,1)==1 && Bits_Binary_QPSK(n,2)==0)
Binary_QPSK_Bits_Tobe_Transmitted(n,1)= constellation_points_Binary_Qpsk(1);
end
end
%%% Binary_QPSK Channel %%%
AWGN = transpose(randn(1,No_bits/2)+1i*randn(1,No_bits/2));
No_Binary_QPSK = Eb_Binary_QPSK./10.^(SNR./10);
%%% Binary_QPSK DeMapper %%%
BER_Actual_Binary_QPSK = zeros(1,length(No_Binary_QPSK));
stimated_bits_Binary_QPSK=zeros(No_bits/2,2);
```

```
for n = 1: length(No_Binary_QPSK)
Noise_vector_Binary_QPSK = sqrt(No_Binary_QPSK(n)/2)*AWGN;
Rx_Symbols_Binary_QPSK_Ater_Noise = Binary_QPSK_Bits_Tobe_Transmitted + Noise_vector_Binary_QPSK;

for ik = 1: No_bits/2
if real(Rx_Symbols_Binary_QPSK_Ater_Noise(ik))>0
   Estimated_bits_Binary_QPSK(ik,1)=1;
   if imag(Rx_Symbols_Binary_QPSK_Ater_Noise(ik))>0
      Estimated_bits_Binary_QPSK(ik,2)=0;
   else
      Estimated_bits_Binary_QPSK(ik,2)=1;
   end
end
if real(Rx_Symbols_Binary_QPSK_Ater_Noise(ik))<0
   Estimated_bits_Binary_QPSK(ik,1)=0;
   if imag(Rx_Symbols_Binary_QPSK_Ater_Noise(ik))>0
      Estimated_bits_Binary_QPSK(ik,2)=1;
   else
      Estimated_bits_Binary_QPSK(ik,2)=0;
   end
end
end
[Number_of_Error_Bits_Binary_QPSK, BER_Actual_Binary_QPSK(n)] =
symerr(Estimated_bits_Binary_QPSK,Bits_Binary_QPSK);
end
%### BER calculations ###%
BER_theoritical_Binary_QPSK = 0.5*erfc(sqrt(Eb_Binary_QPSK./No_Binary_QPSK));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plotting Gray vs Binary QPSK BER:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(6)
semilogy(SNR,BER_Actual_QPSK,'linewidth',1.5)
hold on
semilogy(SNR,BER_Actual_Binary_QPSK,'linewidth',1.5)
legend('BER actual QPSK','BER actual Binary QPSK')
title('BER vs Eb/No Gray & Binary QPSK')
xlabel('Eb/No (in dB)')
ylabel('BER')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

8.BFSK

1. the basis functions of the signal set

$$S_1(t) = \sqrt{\frac{2E_b}{T_b}} \cos(2\pi f_1 t) \quad ; \quad S_2(t) = \sqrt{\frac{2E_b}{T_b}} \cos(2\pi f_2 t)$$

$$\phi_1(t) = \sqrt{\frac{2}{T_b}} \cos(2\pi f_1 t) \quad ; \quad \phi_2(t) = \sqrt{\frac{2}{T_b}} \cos(2\pi f_2 t)$$

If S_1 was Transmitted,

$$\therefore X_1 \sim N\left(\sqrt{E_b}, \frac{N_o}{2}\right) \quad ; \quad X_2 \sim N\left(0, \frac{N_o}{2}\right)$$

If S_2 was Transmitted,

$$\therefore X_1 \sim N\left(0, \frac{N_o}{2}\right) \quad ; \quad X_2 \sim N\left(\sqrt{E_b}, \frac{N_o}{2}\right)$$

2.Write an expression for the baseband equivalent signals for this set, indicating the carrier frequency used.

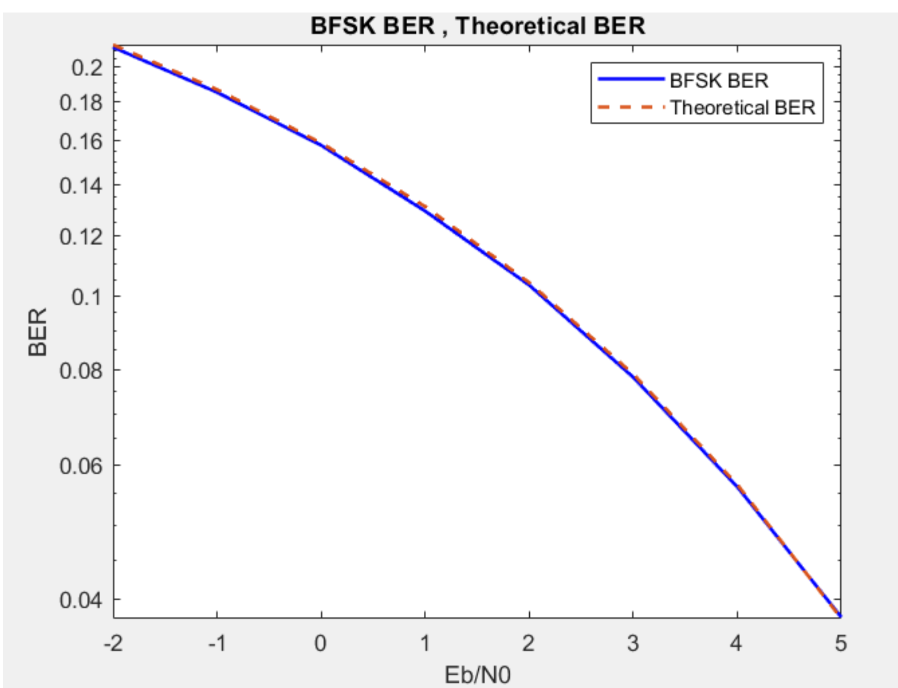
Assume $f_1 = f_c$ and $f_1 = f_c + \triangle f$

In this case $S_1(t) = \sqrt{\frac{2E_b}{T_b}} \cos(2\pi f_c t)$

$$S_2(t) = \sqrt{\frac{2E_b}{T_b}} \cos(2\pi(f_c + \triangle f)t) = \sqrt{\frac{2E_b}{T_b}} (\cos(2\pi f_c t)\ \cos(2\pi \triangle f t) - \sin(2\pi f_c t) \sin(2\pi \triangle f t))$$

$$\therefore S_{1BB}(t) = \sqrt{\frac{2E_b}{T_b}} \qquad \therefore S_{2BB}(t) = \sqrt{\frac{2E_b}{T_b}} (\cos(2\pi \triangle f t) + j\ \sin(2\pi \triangle f t))$$

3.Graph of BFSK BER.

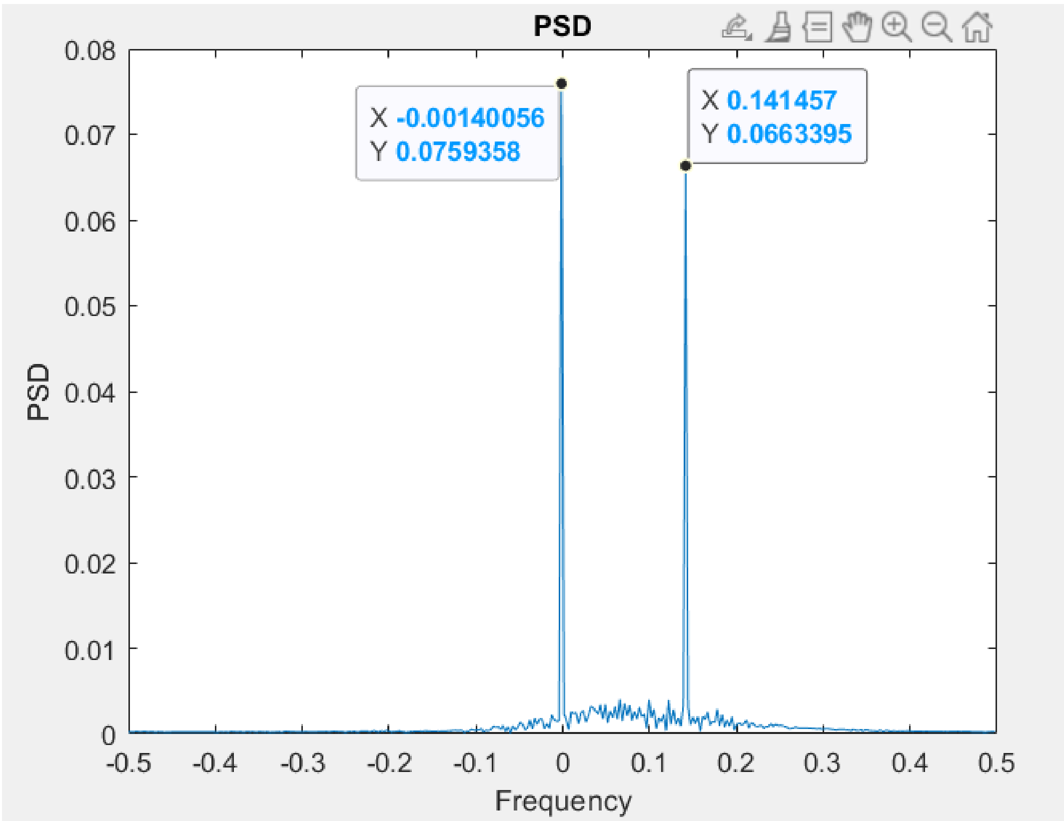


Comment on BFSK:

1-as expected Theoretical and Practical BER are the same.

2- as SNR increases BER decreases as shown in the figure.

4.Graph of BFSK PSD.



5.Comment on BFSK PSD:

two deltas (one at zero) with difference $1/T_b$ (T_b = 7 in our code) with high peak of delta (theoretically is infinity) as expected.

9.THE SECOND CODE

```
clear all;
clc;
close;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Generating random signal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
numofbits = 120000;
randomBits = randi([0,1],1,numofbits); %generating 120000 random bits
Eb = 1;
for(j=1:1:numofbits)
if(randomBits(1,j)==0)
mapping(j)=Eb;
elseif(randomBits(1,j)==1)
mapping(j)=Eb*1i;
end
end
range=-2:5;
BER=zeros(1,length(range));
theoretical_BER=zeros(1,length(range));
noise_vector = randn(1,size(randomBits,2))+randn(1,size(randomBits,2))*1i;
No=zeros(1,length(range));
for range=-2:1:5
No(1,range+3)=Eb/(10^(range/10));
noise_scaled = noise_vector.*sqrt(No(1,range+3)/2);
RxSig(range+3,:)=noise_scaled+mapping;
end
for(range=-2:1:5)
for(j=1:1:numofbits)
if(real(RxSig(range+3,j))>imag(RxSig(range+3,j)))
demapping(range+3,j)=0;
if(demapping(range+3,j)~=randomBits(1,j))
BER(1,range+3)=BER(1,range+3)+1;
end
elseif(real(RxSig(range+3,j))<imag(RxSig(range+3,j)))
demapping(range+3,j)=1;
if(demapping(range+3,j)~=randomBits(1,j))
BER(1,range+3)=BER(1,range+3)+1;
end
end
end
BER(1,range+3)=BER(1,range+3)/numofbits;

end
%Getting theoretical Bit error rate
for j=1:length(No)
theoretical_BER(1,j)=0.5*erfc(1/sqrt(No(1,j)*2));
end
%plotting
figure(7);
SNR=-2:5;
semilogy(SNR,BER,'b','linewidth',1.5)
hold on
semilogy(SNR,theoretical_BER,'--','linewidth',1.5)
xlabel('Eb/N0');
ylabel('BER');
legend('BFSK BER' , 'Theoretical BER');
title('BFSK BER , Theoretical BER');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PSD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
Realization_num = 500;
samples_num = 7;
Tb = 7;
Eb=1;
num_Bits = 51;
received = zeros(Realization_num,samples_num*num_Bits) ;
%receivedafterdelay = zeros(Realization_num,samples_num*num_Bits) ;
Tx = randi([0,1],Realization_num,num_Bits);
COS_value (1,:) = [cos(2*pi*0/Tb) ,cos(2*pi*1/Tb) , cos(2*pi*2/Tb) , cos(2*pi*3/Tb) ,
cos(2*pi*4/Tb) , cos(2*pi*5/Tb) , cos(2*pi*6/Tb) ] ;
Sin_value (1,:) = [sin(2*pi*0/Tb) ,sin(2*pi*1/Tb) , sin(2*pi*2/Tb) , sin(2*pi*3/Tb) ,
sin(2*pi*4/Tb) , sin(2*pi*5/Tb) , sin(2*pi*6/Tb) ] ;
SBB1(1,:) = [sqrt(2*Eb/Tb)  , sqrt(2*Eb/Tb) , sqrt(2*Eb/Tb) , sqrt(2*Eb/Tb) ,
sqrt(2*Eb/Tb) , sqrt(2*Eb/Tb) , sqrt(2*Eb/Tb)   ] ;
SBB2 =  COS_value * sqrt(2*Eb/Tb) + 1*i*Sin_value*sqrt(2*Eb/Tb) ;
for (k=1:1:Realization_num )

for (j=1:1:num_Bits )

if(Tx(k,j)==1)
for m=1:1:7
   received(k,(7*(j-1))+m)=SBB1(1,m);
    end
elseif(Tx(k,j)==0)
    for m=1:1:7
   received(k,(7*(j-1))+m)=SBB2(1,m);
    end
end
end
end
randomDelay = randi([0,6],Realization_num,1);
randombitdelay = randi([0,1],Realization_num,1);

for (j=1:1:Realization_num )

if(randombitdelay(j,1)==1)
delaysample = SBB1(1,1:randomDelay(j)) ;
elseif(randombitdelay(j,1)==0)
delaysample = SBB2(1,1:randomDelay(j)) ;
end
if(j==1)
receivedafterdelay =[delaysample received(j,1:end-randomDelay(j))];
elseif(j~=1)
receivedafterdelay = [receivedafterdelay;delaysample received(j,1:end-randomDelay(j))];

end
end
autocor=(conj(receivedafterdelay(:,179)).*receivedafterdelay(:,1));
for (k=2:1:num_Bits*samples_num)
autocor = [autocor (conj(receivedafterdelay(:,179)).*receivedafterdelay(:,k))];
end
% x = (conj(autocor(:,2:num_Bits*samples_num)))
% autocor_Symm = [x autocor];
datacor = sum(autocor)/Realization_num;
psd = fftshift(fft(datacor(1,:)))/357;
freqAxis = (-0.5*357:0.5*357-1)/357;
figure(8)
plot(freqAxis,abs(psd));
xlabel('Frequency');
ylabel('PSD');
title('PSD');
```