CAIRO UNIVERSITY
FACULTY OF ENGINEERING
ECTRONICS AND COMMUNICATIONS DEPT
3RD YEAR

Digital Communications
MODULATION
Project
Due on 10 May 2023, 11:59 PM

Project 3

Submitted to:

Dr/Mohamed Khairy

Eng/Mohamed Wael

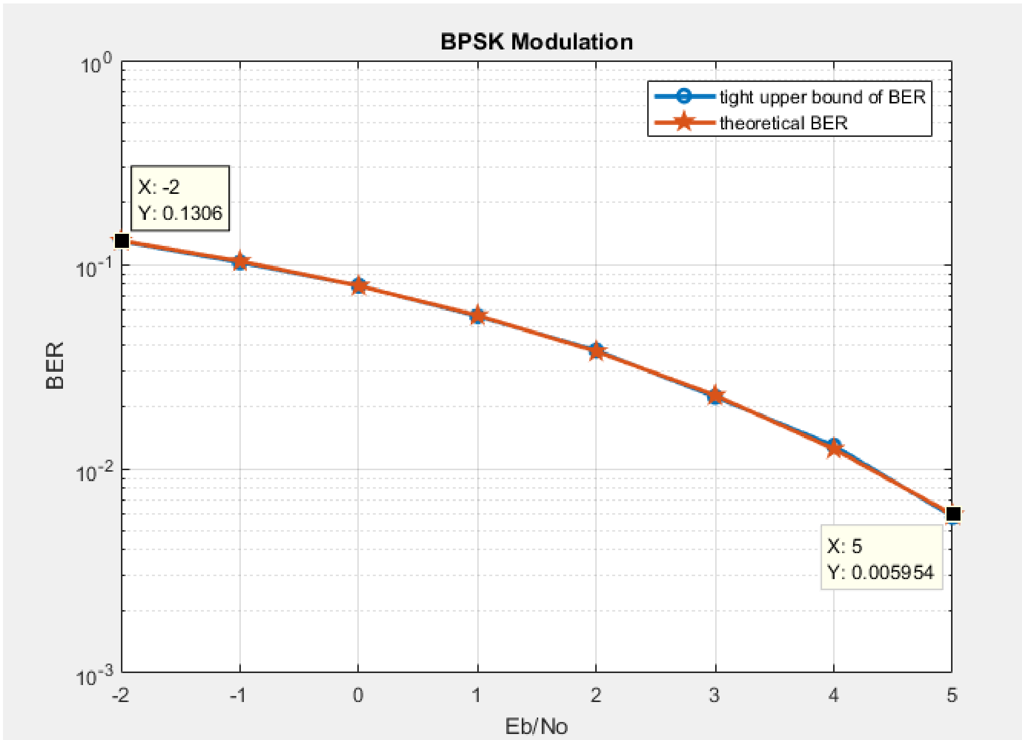
Submitted by:

Name: شهيرة اسامة شفيع محمد	Sec:2	B.N:28	ID:9202725

Contents

1. BPSK Modulation:

MATLAB Results:

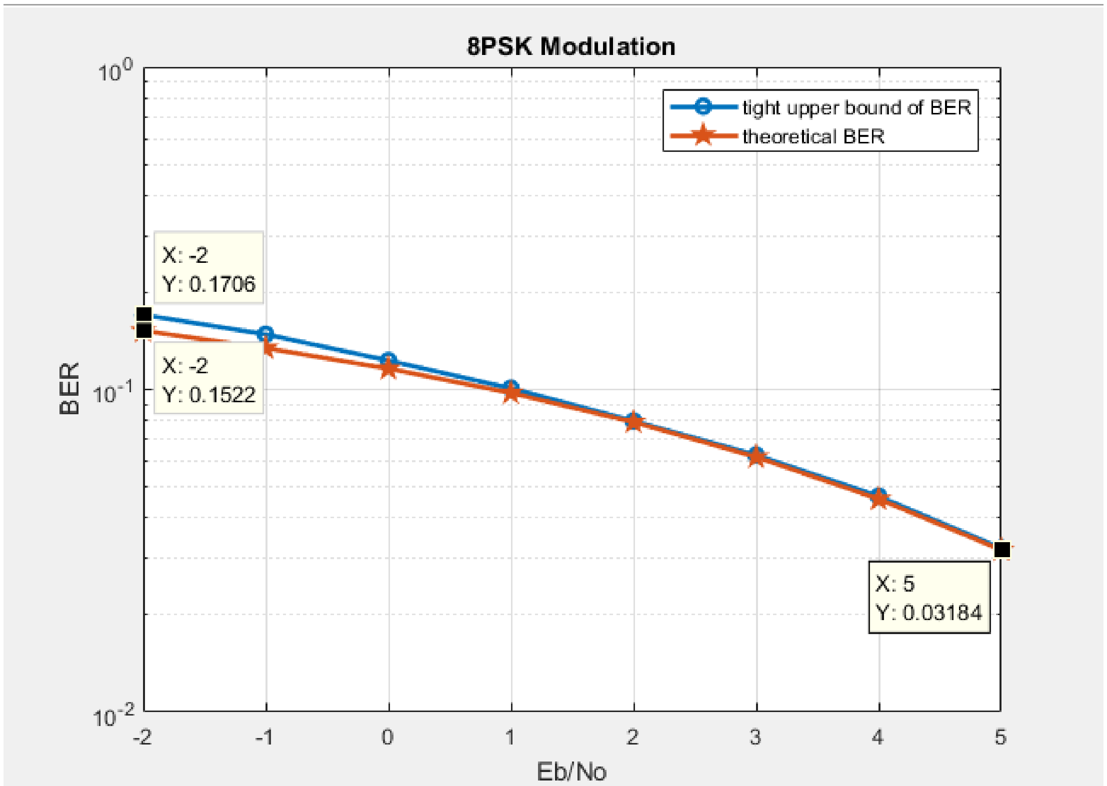


Comments:

- Avg Es $= \frac{Summation\ of\ symbol\ Energy}{Number\ of\ symbols} = \frac{((1)^2+(1)^2)}{2} = 1$
- Bit Energy (Eb)= $\frac{avg\ Es}{Number\ of\ bits} = \frac{1}{1} = 1$
- $No = \frac{Eb}{10^{\frac{snr}{10}}}$
- The range of SNR in dB that I used in the code = [-2:5] as the previous project
- BER=0.5*erfc ($\sqrt{\frac{Eb}{No}}$)

2. 8PSK Modulation:

MATLAB Results:

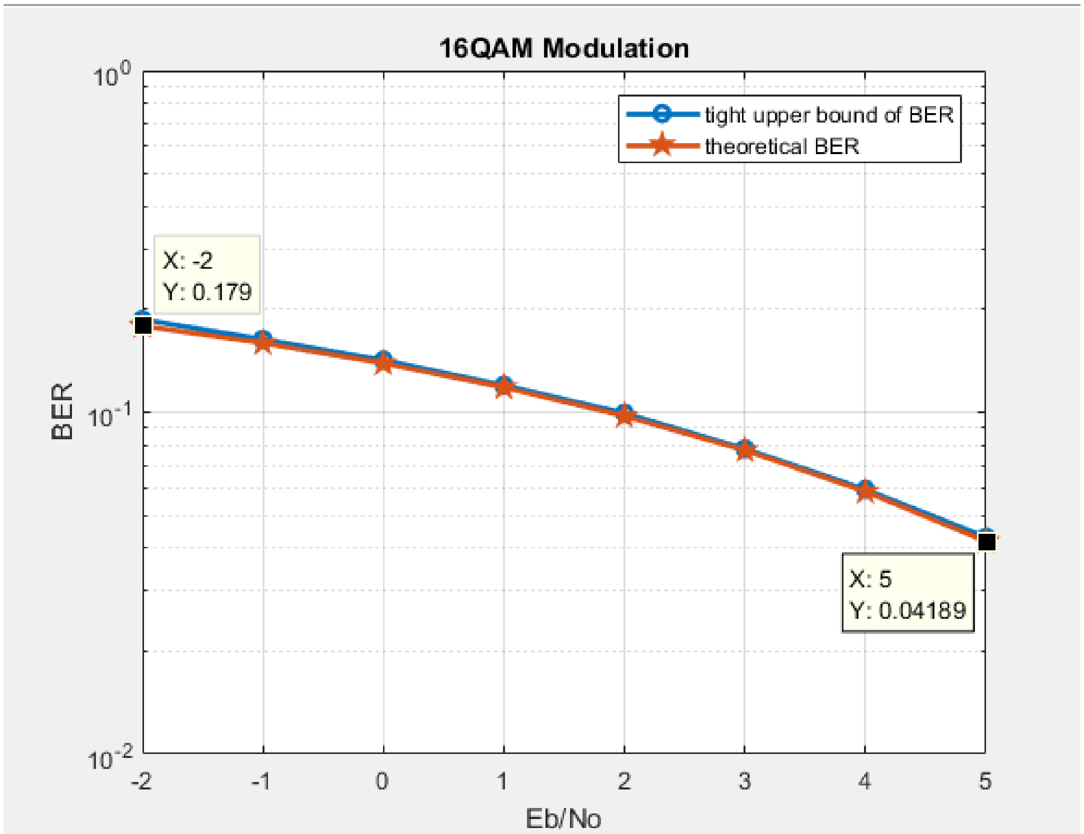


Comments:

- Avg Es $= \frac{Summation\ of\ symbol\ Energy}{Number\ of\ symbols} = \frac{((1)^2+(1)^2+(1)^2+(1)^2+(1)^2+(1)^2+(1)^2+(1)^2)}{8} = 1$
- Bit Energy (Eb)= $\frac{avg\ Es}{Number\ of\ bits} = \frac{1}{3}$
- $No = \frac{Eb}{10^{\frac{snr}{10}}}$
- **The range of SNR in dB that I used in the code = [-2:5] as the previous project**
- BER= $\frac{1}{3}$*erfc $\left(\sqrt{\frac{Eb}{No}} * \sin\left(\frac{\pi}{8}\right)\right)$
- **We can see that at $\frac{Eb}{No} = -2, -1, 0, 1 \rightarrow$ the Theoretical BER and tight upper bound BER aren't matched and that's because in theoretical calculation we make approximation that we work on grey encoding which means that only one bit will flip but actually in small values of snr more than one bit will flip not just one bit as approximated so the upper tight bound will be larger than the theoretical BER in small snr values and then they became matched that's because the approximation is applied and only one bit flipped.**

3. 16QAM Modulation:

MATLAB Results:



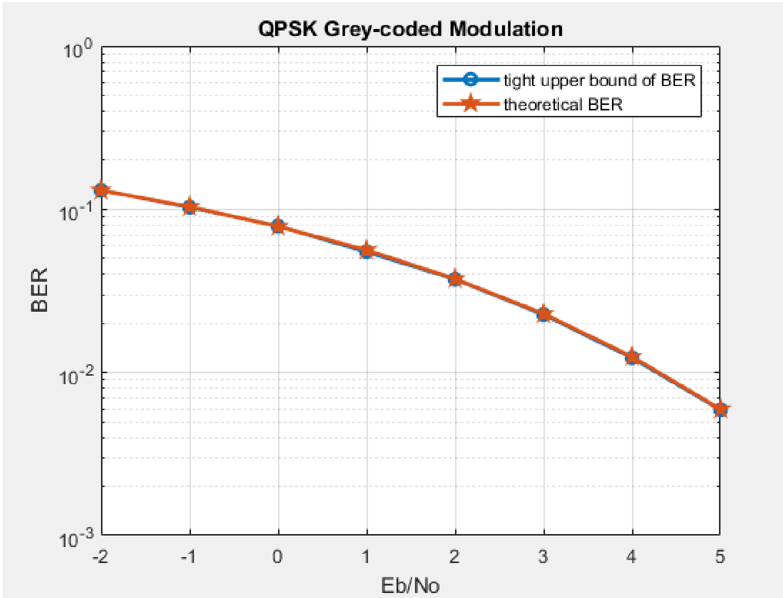
Comments:

- **Avg Es** $= \frac{\boldsymbol{Summation\ of\ symbol\ Energy}}{\boldsymbol{Number\ of\ symbols}} = \frac{(4(\sqrt{2})^2+4(3\sqrt{2})^2+8(\sqrt{10})^2)}{16} = \mathbf{10}$
- **Bit Energy (Eb)**$= \frac{\boldsymbol{avg\ Es}}{\boldsymbol{Number\ of\ bits}} = \frac{10}{4} = \mathbf{2.5}$
- $\boldsymbol{No = \frac{Eb}{10^{\frac{snr}{10}}}}$
- **The range of SNR in dB that I used in the code = [-2:5] as the previous project**
- **BER**$= \frac{3}{8} * \text{erfc}\left(\sqrt{\frac{Eb}{No}}\right)$
- **The theoretical BER and upper tight bound BER are slightly different at small snr values as explained before**

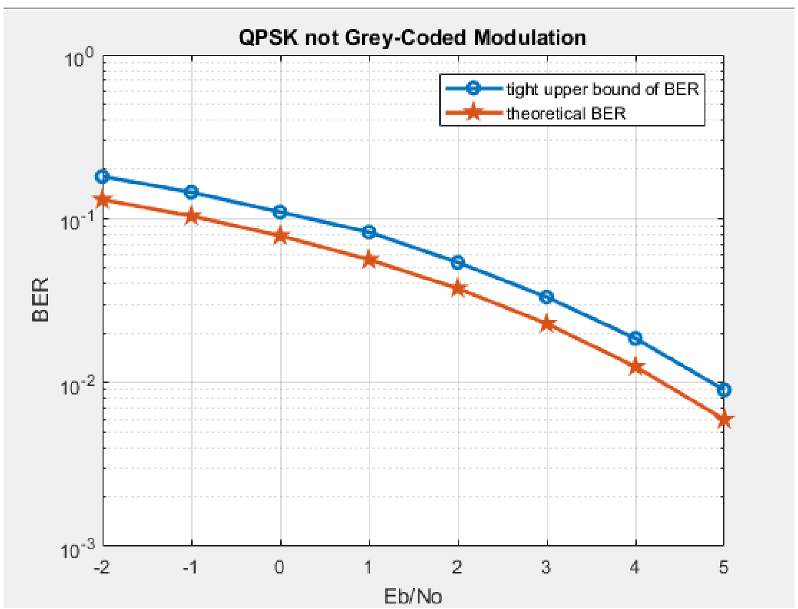
4. QPSK Modulation:

MATLAB Results:

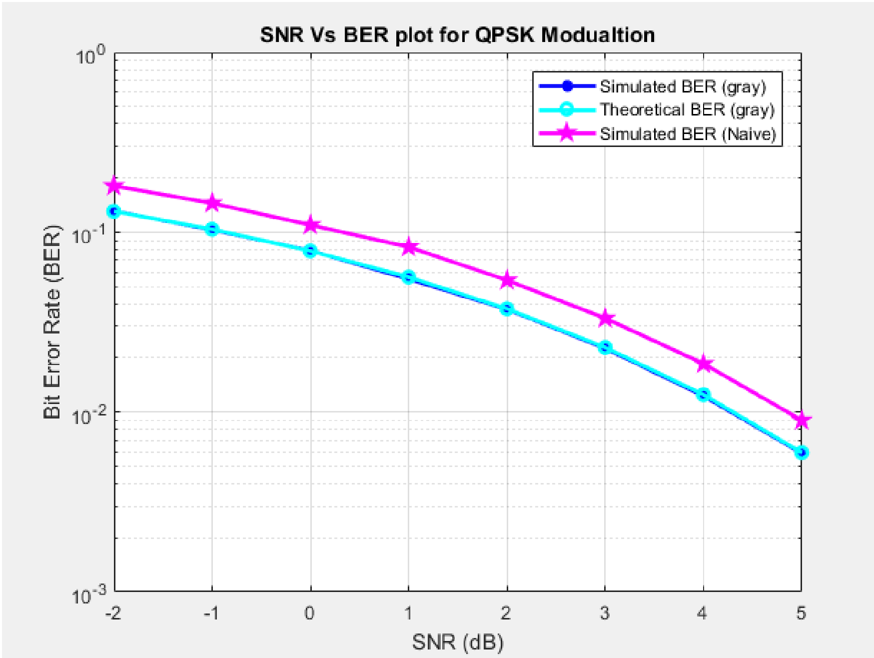
I. Theoretical BER and upper tight bound BER for Gray coded:



II. Theoretical BER and upper tight bound BER for binary coded



III. Theoretical BER and upper tight bound BER for binary coded and Gray Coded on the same graph:



Comments:

- **For case (I):**

- **Avg Es** $= \frac{\textit{Summation of symbol Energy}}{\textit{Number of symbols}} = \frac{(4(\sqrt{2})^2)}{4} = 2$
- **Bit Energy (Eb)**$= \frac{\textit{avg Es}}{\textit{Number of bits}} = \frac{2}{2} = 1$
- $No = \frac{Eb}{10^{\frac{snr}{10}}}$
- **The range of SNR in dB that I used in the code = [-2:5] as the previous project**
- **BER**$=\frac{1}{2}$ * **erfc** $(\sqrt{\frac{Eb}{No}})$
- **The theoretical BER and upper tight bound BER are approximately the same because its gray encoded representation that mean when error occurs only occurs in one bit and BER decreases as SNR increases as expected.**

Comments:

- **For case (II):**

➢ **Avg Es** $= \frac{Summation\ of\ symbol\ Energy}{Number\ of\ symbols} = \frac{(4(\sqrt{2})^2)}{4} = \mathbf{2}$

➢ **Bit Energy (Eb)=** $\frac{avg\ Es}{Number\ of\ bits} = \frac{2}{2} = \mathbf{1}$

➢ $No = \frac{Eb}{10^{\frac{snr}{10}}}$

➢ **The range of SNR in dB that I used in the code = [-2:5] as the previous project**

➢ **BER=** $\frac{1}{2}$ *** erfc (**$\sqrt{\frac{Eb}{No}}$**)**

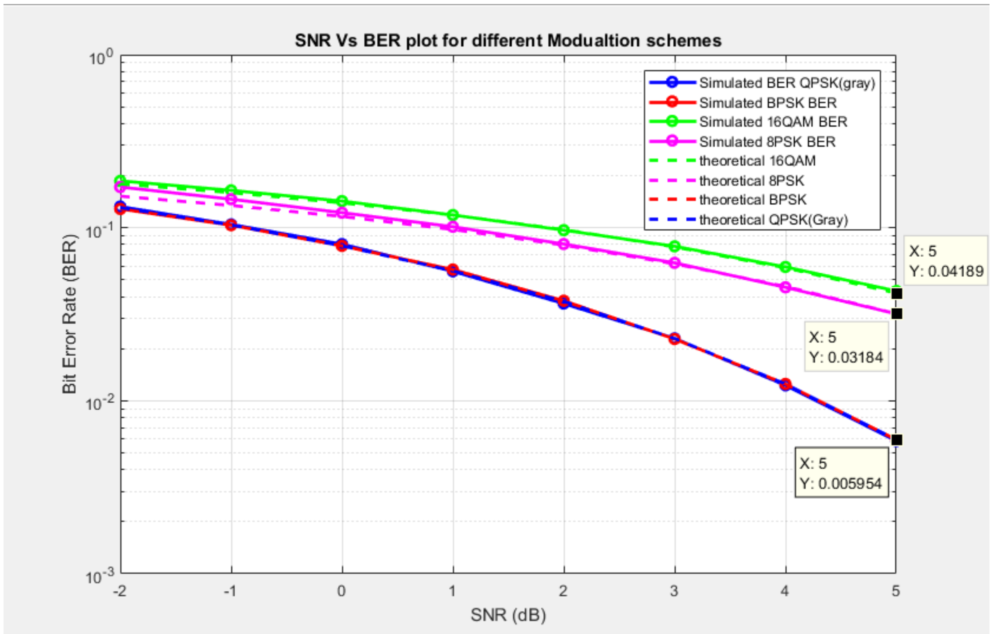
➢ **The theoretical BER and upper tight bound BER aren't the same because its binary encoded representation that mean when error occurs it occurs in more than one bit so the upper tight bound BER will increase than the theoretical BER as there is an error in more than one bit which make sense.**

Comments:

- **For case (III):**
  - ➢ **The upper tight bound BER for Binary encoding QPSK is larger than upper tight bound BER for Gray encoding QPSK because in binary encoded representation when error occurs it occurs in more than one bit but in Gray encoded the error occurs in one bit so the upper tight bound BER in binary encoding will increase than the upper tight bound BER for Gray encoding which make sense.**

5. QPSK(Gray), BPSK, 16QAM, 8PSK Modulations on the same graph:

MATLAB Results:



Comments:

- **Simulated BER and theoretical BER in QPSK are matched together and matched with Simulated BER and theoretical BER in BPSK and it has the smallest BER in the four modulation schemes since they have the same bit Energy and have small number of bits per symbol.**
- **Simulated BER and theoretical BER in 16QAM are matched and it has the largest BER in the four modulation schemes since the BER increases as number of bits per symbol increases and it has the largest value of bit Energy.**
- **Simulated BER and theoretical BER in 8PSK are matched in large SNR values.**
- **The BER decreases when SNR increases.**
- **The distortion or the ripples in the curves is because of finite number of bits and Randomness so if we increased the number of bits the curves will be smoother.**

6. BFSK Modulation:

Consider the BFSK signal given by:

$$S_i(t) = \begin{cases} \sqrt{\frac{2Eb}{Tb}} \cos(2\pi f_i t), & 0 \le t \le Tb \\ 0 & otherwise \end{cases}$$

$$f_i = \frac{nc + i}{Tb} \qquad i = 1, 2$$

1-What are the basis functions of the signal set?

$$\emptyset_1(t) = \sqrt{\frac{2}{Tb}} \cos(2\pi f_1 t) \qquad \rightarrow f_1 = \frac{nc+1}{Tb}$$

$$\emptyset_2(t) = \sqrt{\frac{2}{Tb}} \cos(2\pi f_2 t) \qquad \rightarrow f_2 = \frac{nc+2}{Tb}$$

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

$$f_2 - f_1 = \Delta f$$

$$\Delta f = \frac{1}{Tb}$$

$$@\, f_1 = f_c$$

$$S_1(t) = \sqrt{\frac{2Eb}{Tb}} \cos(2\pi f_c t)$$

$$S_2(t) = \sqrt{\frac{2Eb}{Tb}} \cos(2\pi (f_1 + \Delta f) t)$$

$$S_2(t) = \sqrt{\frac{2Eb}{Tb}} [\cos(2\pi f_1 t) \cos(2\pi \Delta f t) - \sin(2\pi f_1 t) \sin(2\pi \Delta f t)]$$

$$S_2(t) = \sqrt{\frac{2Eb}{Tb}} [\cos(2\pi f_c t) \cos(2\pi \Delta f t) + \sin(2\pi f_c t) \sin(2\pi \Delta f t)]$$

As

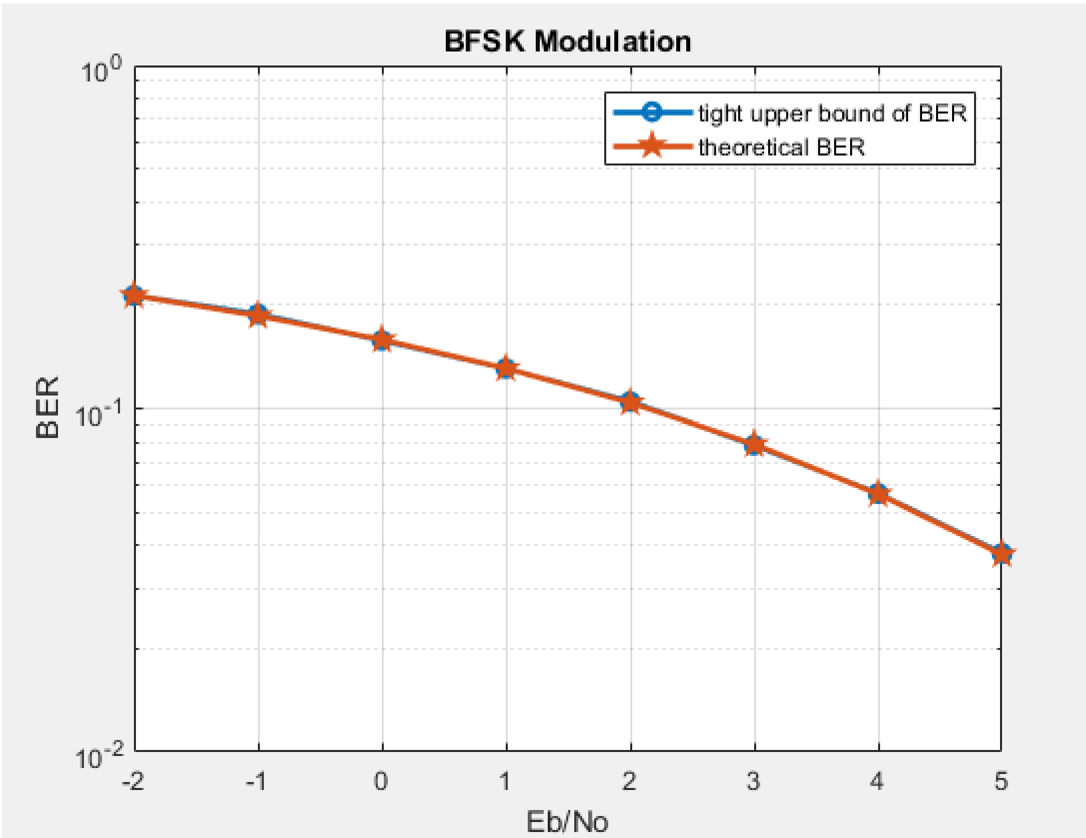
$$S_i(t)_{BB} = Real\, \{S_i(t)_{BB}\, e^{2\pi f_c t}\}$$

$$S_1(t)_{BB} = \sqrt{\frac{2Eb}{Tb}}$$

$$S_2(t)_{BB} = \sqrt{\frac{2Eb}{Tb}} [\cos(2\pi \Delta f t) + j\sin(2\pi \Delta f t)]$$

3-Theoretical BER and BER on the same graph:

MATLAB Results:

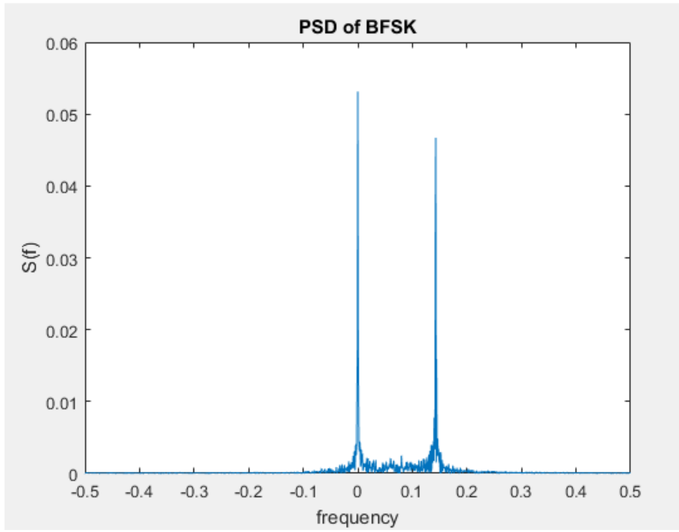


Comments:

- BER and Theoretical BER are matched as BER is a measure of the number of bit errors in a communication system and it's calculated for only one bit.

4.Simulate the PSD of the signal set using the baseband equivalent signal

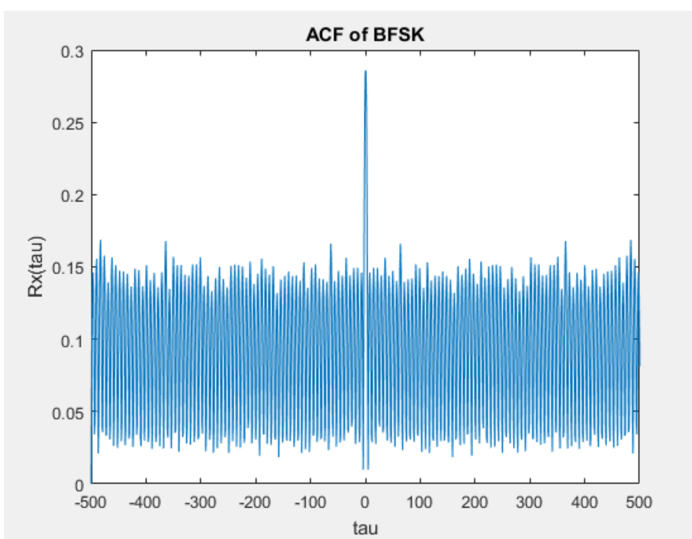
PSD Result from MATLAB:



Comments:

- We have two deltas as shown in PSD figure, one at zero and the other one is at 1/Tb and since I used Tb=7 it's at 0.14285 and the amplitude of the first one actually =0.12*1400 (that we can consider as infinity as we expected)
- Note that PSD figure is normalized to 1400
- At high frequency PSD equals zero

ACF Result from MATLAB:



[Click here for code]

```
clear all
clc

%%%%%%%%%%%%%%%%%% generating bits %%%%%%%%%%%%%%%%%%%%
stream_of_bits = 100000 ;
Random_bits = randi([0 1] , 1 , stream_of_bits) ;
%---------------------- 1)BPSK --------------------------
%%%%%%%%%%%%%%%%% Mapper %%%%%%%%%%%%%%%%%%%%
mapped_symbols = Random_bits .*2 - 1 ; %mapping bits to 1 and -1
%%%%%%%%%%%%%%%%%%%% AWGN channel %%%%%%%%%%%%%%%
BER_BPSK = [] ;
theoritical_BER_BPSK = [] ;
snr = [-2 : 5] ; %range of snr in dB
Eb=1; %bit Energy
No = Eb./(10.^(snr/10));
for i = 1 : length(snr)
AWGN = randn(1,stream_of_bits)*sqrt(No(i)/2); %generating gaussian noise of mean zero and variance 1
recieved_signal = mapped_symbols + AWGN ; %Y=X+N
%%%%%%%%%%%% Demapper %%%%%%%%%%%%%%%%%%%%%%%%
demapped_signal = [] ;
for k = 1 : stream_of_bits
if recieved_signal(k) >= 0 %zero is the threshold (-1+1)/2
demapped_signal = [demapped_signal 1] ;
else
demapped_signal = [demapped_signal 0] ;
end
end
%%%%%%%%%%%% BER calculation %%%%%%%%%%%%%%%%%%%%%
error = abs(demapped_signal - Random_bits);
BER_BPSK = [BER_BPSK sum(error)/stream_of_bits] ;
theoritical_BER_BPSK = [theoritical_BER_BPSK 0.5*erfc(sqrt(1/No(i)))];
end
%%%%%%%%%%%%%% plotting %%%%%%%%%%%%%%%%%%%%
figure(1)
semilogy(snr,BER_BPSK , '-o','linewidth',2 ) ;
hold on
semilogy( snr , theoritical_BER_BPSK ,'-p','linewidth',2) ;
xlabel('Eb/No');
ylabel('BER');
legend('tight upper bound of BER' , 'theoretical BER ') ;
grid on
title('BPSK Modulation');
%---------------------- 2)8PSK -------------------------
%%%%%%%%%%%%%%%%%%%%%%%%% Mapper %%%%%%%%%%%%%%%%%%%%%%%%%%

PSK8_mapped = zeros(1 , (stream_of_bits-1)/3) ;
for i = 1 : (stream_of_bits-1)/3

    if Random_bits (i*3-2 : i*3) == [0 0 0]

        PSK8_mapped(i) = cos(0)+j*sin(0);

    elseif Random_bits (i*3-2 : i*3) == [0 0 1]
        PSK8_mapped(i) = cos(pi/4)+j*sin(pi/4);

    elseif Random_bits(i*3-2 : i*3) ==[0 1 1]

        PSK8_mapped(i) = cos(pi/2)+j*sin(pi/2);

    elseif Random_bits(i*3-2 : i*3) ==[0 1 0]
```

```
        PSK8_mapped(i) = cos(3*pi/4)+j*sin(3*pi/4);

    elseif Random_bits(i*3-2 : i*3) ==[1 1 0]

        PSK8_mapped(i) = cos(pi)+j*sin(pi);

    elseif Random_bits(i*3-2 : i*3) ==[1 1 1]

        PSK8_mapped(i) = cos(5*pi/4)+j*sin(5*pi/4);

    elseif Random_bits(i*3-2 : i*3) ==[1 0 1]

        PSK8_mapped(i) = cos(3*pi/2)+j*sin(3*pi/2);

    elseif Random_bits(i*3-2 : i*3) ==[1 0 0]

        PSK8_mapped(i) = cos(7*pi/4)+j*sin(7*pi/4);
end
end

%%%%%%%%%%%%%%%%%%%%% channel %%%%%%%%%%%%%%%%%%
Eb_8psk=1/3;
No_8PSK = Eb_8psk./(10.^(snr/10));
BER_8PSK = [] ;
theoritical_BER_8PSK = [] ;
for i = 1 : length(snr)
AWGN_8PSKK = randn(1,(stream_of_bits-1)/3)*sqrt(No_8PSK(i)/2)+ j.*randn(1,(stream_of_bits-1)/3)*sqrt(No_8PSK(i)/2);
recieved_8PSK_signal = PSK8_mapped + AWGN_8PSKK ;
%%%%%%%%%%%%%%%%%%% Demapper %%%%%%%%%%%%%%%%
demapped_8PSK = [] ;
for k = 1 : (stream_of_bits-1)/3

    if angle(recieved_8PSK_signal(k)) >= -pi/8 && angle(recieved_8PSK_signal(k)) <= pi/8

        demapped_8PSK = [demapped_8PSK 0 0 0] ;

    elseif angle(recieved_8PSK_signal(k)) >= pi/8 && angle(recieved_8PSK_signal(k)) <= 3*pi/8

        demapped_8PSK = [demapped_8PSK 0 0 1] ;

    elseif angle(recieved_8PSK_signal(k)) >= 3*pi/8 && angle(recieved_8PSK_signal(k)) <= 5*pi/8

        demapped_8PSK = [demapped_8PSK 0 1 1] ;

    elseif angle(recieved_8PSK_signal(k)) >= 5*pi/8 && angle(recieved_8PSK_signal(k)) <= 7*pi/8
        demapped_8PSK = [demapped_8PSK 0 1 0] ;

    elseif angle(recieved_8PSK_signal(k)) >= -7*pi/8 && angle(recieved_8PSK_signal(k)) <= -5*pi/8

        demapped_8PSK = [demapped_8PSK 1 1 1] ;

    elseif angle(recieved_8PSK_signal(k)) >= -5*pi/8 && angle(recieved_8PSK_signal(k)) <= -3*pi/8

        demapped_8PSK = [demapped_8PSK 1 0 1] ;

    elseif angle(recieved_8PSK_signal(k)) >= -3*pi/8 && angle(recieved_8PSK_signal(k)) <= -pi/8

    demapped_8PSK = [demapped_8PSK 1 0 0] ;
    else

        demapped_8PSK = [demapped_8PSK 1 1 0] ;
```

```
end
end
%%%%%%%%%%%%%%%%%%%%%%5 BER calculation %%%%%%%%%%%%%%%%%

error = abs( demapped_8PSK - Random_bits( 1 : (stream_of_bits-1) ) );
BER_8PSK = [BER_8PSK sum(error)/stream_of_bits] ;
theoritical_BER_8PSK = [theoritical_BER_8PSK (1/3)*erfc(sqrt(1/No_8PSK(i))*sin(pi/8) )];
end
% plotting BER of 8PSK
figure(2)
semilogy(snr,BER_8PSK , '-o','linewidth',2 ) ;
hold on
semilogy( snr, theoritical_BER_8PSK ,'-p','linewidth',2) ;
xlabel('Eb/No');
ylabel('BER');
legend('tight upper bound of BER' , 'theoretical BER ');
grid on
title('8PSK Modulation');
%---------------- 3)QPSK Grey Coded-----------------------%
%%%%%%%%%%%%%%%%%%%%%% Mapper %%%%%%%%%%%%%%%%%%%%%%

QPSK_mapped = zeros(1 , (stream_of_bits)/2) ;
for i = 1 : (stream_of_bits-1)/2

    if Random_bits(i*2-1 : i*2) == [1 1]

        QPSK_mapped(i) = (cos(0)+j*sin(pi/2));

    elseif Random_bits (i*2-1 : i*2) == [0 1]
        QPSK_mapped(i) = (cos(pi)+j*sin(pi/2));

    elseif Random_bits(i*2-1 : i*2) ==[1 0]

        QPSK_mapped(i) = (cos(0)+j*sin(3*pi/2));

    elseif Random_bits(i*2-1 : i*2) ==[0 0]

        QPSK_mapped(i) = (cos(pi)+j*sin(3*pi/2));


end
end
%%%%%%%%%%%%%%%%%%%%%% channel %%%%%%%%%%%%%%%%%%%%%%%%%
Eb_Qpsk=1;
No_QPSK = Eb_Qpsk./(10.^(snr/10));
BER_QPSK = [] ;
theoritical_BER_QPSK = [] ;
for i = 1 : length(snr)
AWGN_QPSKK = randn(1,(stream_of_bits)/2)*sqrt(No_QPSK(i)/2)+ j.*randn(1,(stream_of_bits)/2)*sqrt(No_QPSK(i)/2);
recieved_QPSK_signal = QPSK_mapped + AWGN_QPSKK ;

%%%%%%%%%%%%%%%%%%% Demapper %%%%%%%%%%%%%%%%%
demapped_QPSK = [] ;
for k = 1 : (stream_of_bits)/2

    if real(recieved_QPSK_signal(k)) >= 0

        demapped_QPSK = [demapped_QPSK 1] ;

    else

        demapped_QPSK = [demapped_QPSK 0 ] ;
```

```
    end
    if imag(recieved_QPSK_signal(k)) >= 0

        demapped_QPSK = [demapped_QPSK 1] ;

    else
        demapped_QPSK = [demapped_QPSK 0] ;
end
end
%%%%%%%%%%%%%%%%%%%%% BER calculation %%%%%%%%%%%%%%%%
error = abs(demapped_QPSK - Random_bits);
BER_QPSK = [BER_QPSK sum(error)/stream_of_bits] ;
theoritical_BER_QPSK = [theoritical_BER_QPSK (0.5)*erfc(sqrt(1/No_QPSK(i)))];
end
%%%%%%%%%%%%%% plotting BER of QPSK %%%%%%%%%%%%%%%%%
figure(3)
semilogy(snr,BER_QPSK , '-o','linewidth',2 ) ;
hold on
semilogy( snr, theoritical_BER_QPSK ,'-p','linewidth',2) ;
xlabel('Eb/No');
ylabel('BER');
legend('tight upper bound of BER' , 'theoretical BER ') ;
grid on
title('QPSK Grey-coded Modulation');

%---------------- 3)QPSK Naive Coded-----------------------%
%%%%%%%%%%%%%%%%%%%% Mapper %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
QPSK2_mapped = zeros(1, stream_of_bits/2);
for i = 1:(stream_of_bits-1)/2
    if Random_bits(i*2-1 : i*2) == [1 0]
        QPSK2_mapped(i) = (cos(0)+j*sin(pi/2));
    elseif Random_bits(i*2-1 : i*2) == [0 1]
        QPSK2_mapped(i) = (cos(pi)+j*sin(pi/2));
    elseif Random_bits(i*2-1 : i*2) == [1 1]
        QPSK2_mapped(i) = (cos(0)+j*sin(3*pi/2));
    elseif Random_bits(i*2-1 : i*2) == [0 0]
        QPSK2_mapped(i) = (cos(pi)+j*sin(3*pi/2));
    end
end

%%%%%%%%%%%%%%%%%%%%% channel %%%%%%%%%%%%%%%%%%%%%%%%%
Eb_Qpsk_2 = 1;
No_QPSK_2 = Eb_Qpsk_2./(10.^(snr/10));
BER_QPSK_2 = [];
theoretical_BER_QPSK_2 = [];
for i = 1:length(snr)
    AWGN_QPSK_2 = randn(1, stream_of_bits/2)*sqrt(No_QPSK_2(i)/2) + j.*randn(1, stream_of_bits/2)*sqrt(No_QPSK_2(i)/2);
    received_QPSK2_signal = QPSK2_mapped + AWGN_QPSK_2;

    %%%%%%%%%%%%%%%%%%% Demapper %%%%%%%%%%%%%%%%%%%%%%%%%
    demapped_QPSK_2 = [];
    for k = 1:stream_of_bits/2
        if real(received_QPSK2_signal(k)) >= 0 && imag(received_QPSK2_signal(k)) >= 0
            demapped_QPSK_2 = [demapped_QPSK_2 1 0];
        elseif real(received_QPSK2_signal(k)) >= 0 && imag(received_QPSK2_signal(k)) < 0
            demapped_QPSK_2 = [demapped_QPSK_2 1 1];
        elseif real(received_QPSK2_signal(k)) < 0 && imag(received_QPSK2_signal(k)) >= 0
            demapped_QPSK_2 = [demapped_QPSK_2 0 1];
        elseif real(received_QPSK2_signal(k)) < 0 && imag(received_QPSK2_signal(k)) < 0
            demapped_QPSK_2 = [demapped_QPSK_2 0 0];
        end
    end
```

```
 %%%%%%%%%%%%%%%%%%%%%%% BER calculation %%%%%%%%%%%%%%%%
   error = abs(demapped_QPSK_2 - Random_bits);
   BER_QPSK_2 = [BER_QPSK_2 sum(error)/stream_of_bits];
   theoretical_BER_QPSK_2 = [theoretical_BER_QPSK_2 ((0.5)*erfc(sqrt(10.^(snr(i)/10))))];
end
%%%%%%%%%%%%%%% plotting BER of QPSK %%%%%%%%%%%%%%%%%%
figure(4)
semilogy(snr,BER_QPSK_2 , '-o','linewidth',2 ) ;
hold on
semilogy( snr, theoretical_BER_QPSK_2 ,'-p','linewidth',2) ;
xlabel('Eb/No');
ylabel('BER');
legend('tight upper bound of BER' , 'theoretical BER ') ;
grid on
title('QPSK not Grey-Coded Modulation');

figure(5)
semilogy(snr,BER_QPSK,'-*b','linewidth',2)
hold on
semilogy(snr,theoritical_BER_QPSK,'-oc','linewidth',2)
hold on
semilogy(snr,BER_QPSK_2,'-pm','linewidth',2)
% hold on
% semilogy(snr,theoretical_BER_QPSK_2,'-k','linewidth',2)
legend('Simulated BER (gray)', 'Theoretical BER (gray)','Simulated BER (Naive)');
grid on
xlabel('SNR (dB)');
ylabel('Bit Error Rate (BER)');
title('SNR Vs BER plot for QPSK Modualtion');
%---------------- 3)16QAM-----------------------%
%%%%%%%%%%%%%%%%%%%%%Mapper %%%%%%%%%%%%%%%%%%%%%%%%%
mapped_16QAM = zeros(1 , stream_of_bits/4) ;
for i = 1 : stream_of_bits/4

   if Random_bits(4*i-3 : 4*i) == [0 0 0 0]
   mapped_16QAM(i) = -3 + j*-3 ;

   elseif Random_bits(4*i-3 : 4*i) == [0 0 0 1]
mapped_16QAM(i) = -3 + j*-1 ;

   elseif Random_bits(4*i-3 : 4*i) == [0 0 1 0]
mapped_16QAM(i) = -3 + j*3 ;

   elseif Random_bits(4*i-3 : 4*i) == [0 0 1 1]
mapped_16QAM(i) = -3 + j*1 ;

   elseif Random_bits(4*i-3 : 4*i) == [0 1 0 0]
mapped_16QAM(i) = -1 + j*-3 ;

   elseif Random_bits(4*i-3 : 4*i) == [0 1 0 1]
mapped_16QAM(i) = -1 + j*-1 ;

   elseif Random_bits(4*i-3 : 4*i) == [0 1 1 0]
mapped_16QAM(i) = -1 + j*3 ;

   elseif Random_bits(4*i-3 : 4*i) == [0 1 1 1]
mapped_16QAM(i) = -1 + j*1 ;

   elseif Random_bits(4*i-3 : 4*i) == [1 0 0 0]
mapped_16QAM(i) = 3 + j*-3 ;
```

```
    elseif Random_bits(4*i-3 : 4*i) == [1 0 0 1]
mapped_16QAM(i) = 3 + j*-1 ;

    elseif Random_bits(4*i-3 : 4*i) == [1 0 1 0]
mapped_16QAM(i) = 3 + j*3 ;

    elseif Random_bits(4*i-3 : 4*i) == [1 0 1 1]
mapped_16QAM(i) = 3 + j*1 ;

    elseif Random_bits(4*i-3 : 4*i) == [1 1 0 0]
mapped_16QAM(i) = 1 + j*-3 ;

    elseif Random_bits(4*i-3 : 4*i) == [1 1 0 1]
mapped_16QAM(i) = 1 + j*-1 ;

    elseif Random_bits(4*i-3 : 4*i) == [1 1 1 0]
mapped_16QAM(i) = 1 + j*3 ;

    elseif Random_bits(4*i-3 : 4*i) == [1 1 1 1]
mapped_16QAM(i) = 1 + j*1 ;
end
end
%%%%%%%%%%%%%%%%%%%%%%%% channel %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Eb_16QAM=2.5;
No_16QAM =Eb_16QAM ./(10.^(snr/10));
BER_16QAM = [] ;
theoritical_BER_16QAM = [] ;
for i = 1 : length(snr)
    AWGN_16QAM = randn(1,stream_of_bits/4)*sqrt(No_16QAM(i)/2) +j.*randn(1,stream_of_bits/4)*sqrt(No_16QAM(i)/2);
recieved_16QAM_signal = mapped_16QAM + AWGN_16QAM ;
%%%%%%%%%%%%%%%% Demapper %%%%%%%%%%%%%%%%%%%%%%%%%%%
demapped_16QAM = [] ;
for k = 1 : stream_of_bits/4
%%%% Calculating the distance between recieved signal from the channel and constellation points and getting the minimum distance
D1 = abs(recieved_16QAM_signal(k) - ( -3 + j*-3 )) ;
D2 = abs(recieved_16QAM_signal(k) - ( -3 + j*-1 )) ;
D3 = abs(recieved_16QAM_signal(k) - ( -3 + j*3 )) ;
D4 = abs(recieved_16QAM_signal(k) - ( -3 + j*1 )) ;
D5 = abs(recieved_16QAM_signal(k) - ( -1 + j*-3 )) ;
D6 = abs(recieved_16QAM_signal(k) - ( -1 + j*-1 )) ;
D7 = abs(recieved_16QAM_signal(k) - ( -1 + j*3 )) ;
D8 = abs(recieved_16QAM_signal(k) - ( -1 + j*1 )) ;
D9 = abs(recieved_16QAM_signal(k) - ( 3 + j*-3 )) ;
D10 = abs(recieved_16QAM_signal(k) - ( 3 + j*-1 )) ;
D11 = abs(recieved_16QAM_signal(k) - ( 3 + j*3 )) ;
D12 = abs(recieved_16QAM_signal(k) - ( 3 + j*1 )) ;
D13 = abs(recieved_16QAM_signal(k) - ( 1 + j*-3 )) ;
D14 = abs(recieved_16QAM_signal(k) - ( 1 + j*-1 )) ;
D15 = abs(recieved_16QAM_signal(k) - ( 1 + j*3 )) ;
D16 = abs(recieved_16QAM_signal(k) - ( 1 + j*1 )) ;
total_distance = [D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15 D16];
if min(total_distance) == D1
demapped_16QAM = [demapped_16QAM 0 0 0 0];
elseif min(total_distance) == D2
demapped_16QAM = [demapped_16QAM 0 0 0 1];
elseif min(total_distance) == D3
demapped_16QAM = [demapped_16QAM 0 0 1 0];
elseif min(total_distance) == D4
demapped_16QAM = [demapped_16QAM 0 0 1 1];
elseif min(total_distance) == D5
demapped_16QAM = [demapped_16QAM 0 1 0 0];
elseif min(total_distance) == D6
```

```
    demapped_16QAM = [demapped_16QAM 0 1 0 1];
elseif min(total_distance) == D7
demapped_16QAM = [demapped_16QAM 0 1 1 0];
elseif min(total_distance) == D8
demapped_16QAM = [demapped_16QAM 0 1 1 1];
elseif min(total_distance) == D9
demapped_16QAM = [demapped_16QAM 1 0 0 0];
elseif min(total_distance) == D10
demapped_16QAM = [demapped_16QAM 1 0 0 1];
elseif min(total_distance) == D11
demapped_16QAM = [demapped_16QAM 1 0 1 0];
elseif min(total_distance) == D12
demapped_16QAM = [demapped_16QAM 1 0 1 1];
elseif min(total_distance) == D13
demapped_16QAM = [demapped_16QAM 1 1 0 0];
elseif min(total_distance) == D14
demapped_16QAM = [demapped_16QAM 1 1 0 1];
elseif min(total_distance) == D15
demapped_16QAM = [demapped_16QAM 1 1 1 0];
elseif min(total_distance) == D16
demapped_16QAM = [demapped_16QAM 1 1 1 1];
end
end
%%%%%%%%%%%%%%%%%%BER calculation%%%%%%%%%%%%%%%%%%%%%%
error = abs(demapped_16QAM - Random_bits);
BER_16QAM = [BER_16QAM sum(error)/stream_of_bits] ;
theoritical_BER_16QAM = [theoritical_BER_16QAM (3/8)*erfc(sqrt(1/No_16QAM(i)))];
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%% plotting BER of 16QAM %%%%%%%%%%%%%%%%%%%%
figure(6)
semilogy(snr,BER_16QAM , '-o','linewidth',2 ) ;
hold on
semilogy( snr , theoritical_BER_16QAM ,'-p','linewidth',2) ;
xlabel('Eb/No');
ylabel('BER');
legend('tight upper bound of BER' , 'theoretical BER ') ;
grid on
title('16QAM Modulation');
%------------------------ BFSK ----------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%% Mapper %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
PFSK = zeros(1 , stream_of_bits) ;
for i = 1 : (stream_of_bits)

    if Random_bits(i)  == 0

        PFSK(i) = cos(0)+j*sin(0);

    else
        PFSK(i) = cos(pi/2)+j*sin(pi/2);

    end

end

%%%%%%%%%%%%%%%%%%%%%%%% channel %%%%%%%%%%%%%%%%%%%%
Eb_BFSK=1;
No_BFSK = Eb_BFSK./(10.^(snr/10));
BER_BFSK = [] ;
theoritical_BER_BFSK = [] ;
for i = 1 : length(snr)
AWGN_BFSK = randn(1,stream_of_bits)*sqrt(No_BFSK(i)/2)+ j.*randn(1,stream_of_bits)*sqrt(No_BFSK(i)/2);
recieved_BFSK_signal = PFSK + AWGN_BFSK ;
```

```
%%%%%%%%%%%%%%%%%% Demapper %%%%%%%%%%%%%%%%%
demapped_BFSK = [] ;
for k = 1 : stream_of_bits

    if angle(recieved_BFSK_signal(k)) >= -3*pi/4 && angle(recieved_BFSK_signal(k)) < pi/4

        demapped_BFSK = [demapped_BFSK 0] ;

    else

        demapped_BFSK = [demapped_BFSK  1] ;


end
end
%%%%%%%%%%%%%%%%%%%%%5 BER calculation %%%%%%%%%%%%%%%%%

error = abs( demapped_BFSK - Random_bits( 1 : stream_of_bits) );
BER_BFSK = [BER_BFSK sum(error)/stream_of_bits] ;
theoritical_BER_BFSK = [theoritical_BER_BFSK (1/2)*erfc(sqrt(0.5/No_BFSK(i)) )];
end
%%%%%%%%%%%%%%%%%% plotting BER of BFSK %%%%%%%%%%%%%%%%%%%%
figure(7)
semilogy(snr,BER_BFSK , '-o','linewidth',2 ) ;
hold on
semilogy( snr, theoritical_BER_BFSK ,'-p','linewidth',2) ;
xlabel('Eb/No');
ylabel('BER');
legend('tight upper bound of BER' , 'theoretical BER ') ;
grid on
title('BFSK Modulation');

%%%%%%%%%%%% BPSK,QPSK(Grey),QPSK(Binary),16QAM,8PSK,BFSK %%%%%%%%%%%%%%%%%%%%%%
figure(8)
semilogy(snr,BER_QPSK,'-ob','linewidth',2)
hold on
semilogy(snr,BER_BPSK,'-or','linewidth',2)
hold on
semilogy(snr,BER_16QAM,'-og','linewidth',2)
hold on
semilogy(snr,BER_8PSK,'-om','linewidth',2)
hold on
semilogy(snr,theoritical_BER_16QAM,'--g','linewidth',2)
hold on
semilogy(snr,theoritical_BER_8PSK,'--m','linewidth',2)
hold on
semilogy(snr,theoritical_BER_BPSK,'--r','linewidth',2)
hold on
semilogy(snr,theoritical_BER_QPSK,'--b','linewidth',2)

legend('Simulated BER QPSK(gray)', 'Simulated BPSK BER ','Simulated 16QAM BER ','Simulated 8PSK BER','theoretical
16QAM','theoretical 8PSK','theoretical BPSK','theoretical QPSK(Gray)');
grid on
xlabel('SNR (dB)');
ylabel('Bit Error Rate (BER)');
title('SNR Vs BER plot for different Modualtion schemes');
ylim([1e-3 1]);
xlim([-2 5]);
%%%%%%%%%%%%%%%%%%% BFSK PSD %%%%%%%%%%%%%%%%%%%%%%
Realizations = 500;
Data = randi([0, 1], 500, 101);
Data2 = repelem(Data, 1, 7);
```

```
Tb = 7;
t = 0:1:Tb - 1;
Eb = 1;
Delay = randi([1, 7], 1, 500);
S1_BB = sqrt(2 * Eb / Tb); %Complex baseband equivalent '0'
S2_BB = S1_BB * (cos(2 * pi * t * 1 / Tb) + j * sin(2 * pi * t * 1 / Tb)); %complex baseband equivalent '1'

for i = 1:500
    k = 1;
    for j = 1:101
        if Data(i, j) == 1
            Data2(i, k:k + 6) = S2_BB;
        else
            Data2(i, k:k + 6) = S1_BB;
        end
        k = k + 7;
    end
end

Tx2 = zeros(500, 700);

for k = 1:500
    Tx2(k, :) = [Data2(k, 700 - Delay(k) + 1:700) Data2(k, 1:700 - Delay(k))];
end
Tx2=Tx2(1:500, 1:700);

no_of_samples = 7;
ensemble_ACF(1, 7 * 100) = 0;

for i = 1:700
    for j = 1:1:500
        ensemble_ACF(i, j) = sum(conj(Tx2(:, i)) .* Tx2(:, j), 1) / (500);
    end
end

ensemble_ACF_2 = [conj(fliplr(ensemble_ACF(1, :))) ensemble_ACF(1, :)];
%%%%%%%%%%%%%%%%%%%%%%% plotting the PSD %%%%%%%%%%%%%%%%
figure(9)
ACF = (-700:699) / 1400;
x = fftshift(fft(ensemble_ACF_2(1, :)));
plot(ACF, abs(x) / 1400)
title('PSD of BFSK')
ylabel('S(f)');
xlabel('frequency');
%%%%%%%%%%%%%%%%%%%%%%% plotting the ACF %%%%%%%%%%%%%%%%
figure(10)
N= length(ensemble_ACF_2);
plot((-N/2+1:1:N/2), abs(ensemble_ACF_2))
xlim([-500 500])
title('ACF of BFSK')
ylabel('Rx(tau)');
xlabel('tau');
```