Department of Electronics and
Electrical Communications Engineering
Faculty of Engineering - Cairo University

Faculty of Engineering – Cairo University

Electronics and Electrical Communications Engineering Department

Third Year – Mainstream

(Modulation)

Digital Communication (ELC3070)

Submitted to: Dr. Mohamed Nafie

Submitted to: Dr. Mohamed Khairy

Submitted to: TA. Mohamed Khaled

Team:53

Names	Sec	ID
Ahmed Mostafa Ahmed AbuBakr	1	9210157
Mahmoud Abdel-Halim Ahmed Eissa	4	9211092

Contents

List of figures

Section 1 : Single Carrier System

1.1 : The mapper

-First, we generated a vector of 120,000 bits which represents the input data bits.
-Then we started by creating a mapper for BPSK by multiplying each element by 2 and subtracting 1, where 1 is mapped to 1 and 0 is mapped to -1 as shown in Figure 1.
-For QPSK, we first converted the 1*120,000 vector into a matrix of size 60,000*2. Then, we converted each binary number in the matrix into a decimal number. After that, we created a transformation vector with a gray constellation, and the decimal number acted as an index to this vector. The output is shown in Figure 1.
-To create 8PSK, like the previous step, we converted the vector into a 40,000*3 matrix, then converted the binary numbers into decimal numbers, and mapped them into the transformation vector as illustrated in Figure 1.
-Finally, we implemented 16QAM by converting the data vector into 30,000*4 then using the same steps as before, and the output is displayed in Figure 1.

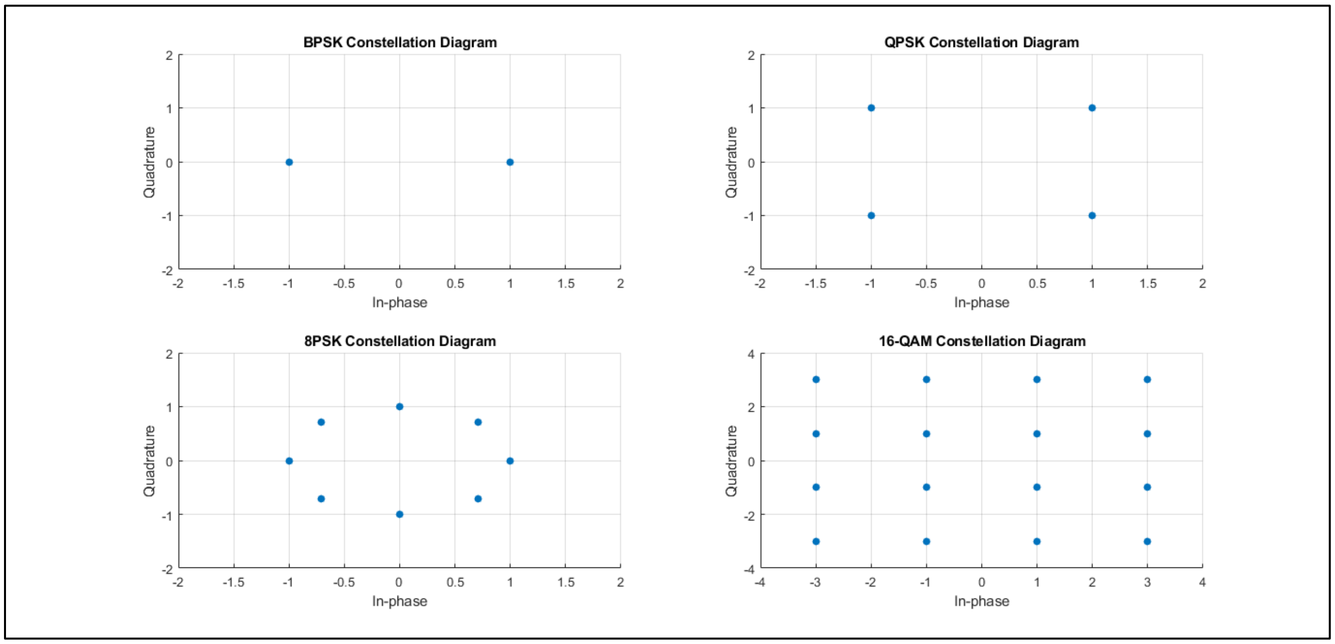


Figure 1: BPSK, QPSK, 8PSK, and 16QAM constellations.

1.2 : The channel

We added noise to the transmitted bits (AWGN) as shown in figure 2 by using the randn() command which is multiplied by $\sqrt{\frac{N0}{2} * \frac{Eavg}{nb}}$ first we calculated the average energy per symbol by computing E where $E = real(i)^2 + imaj(i)^2$ then we calculated the $Eavg = \frac{E}{number\ of\ symbols}$ then we calculated the average energy per bit $Eb = \frac{Eavg}{nb}$ and the values are show in figure 3.

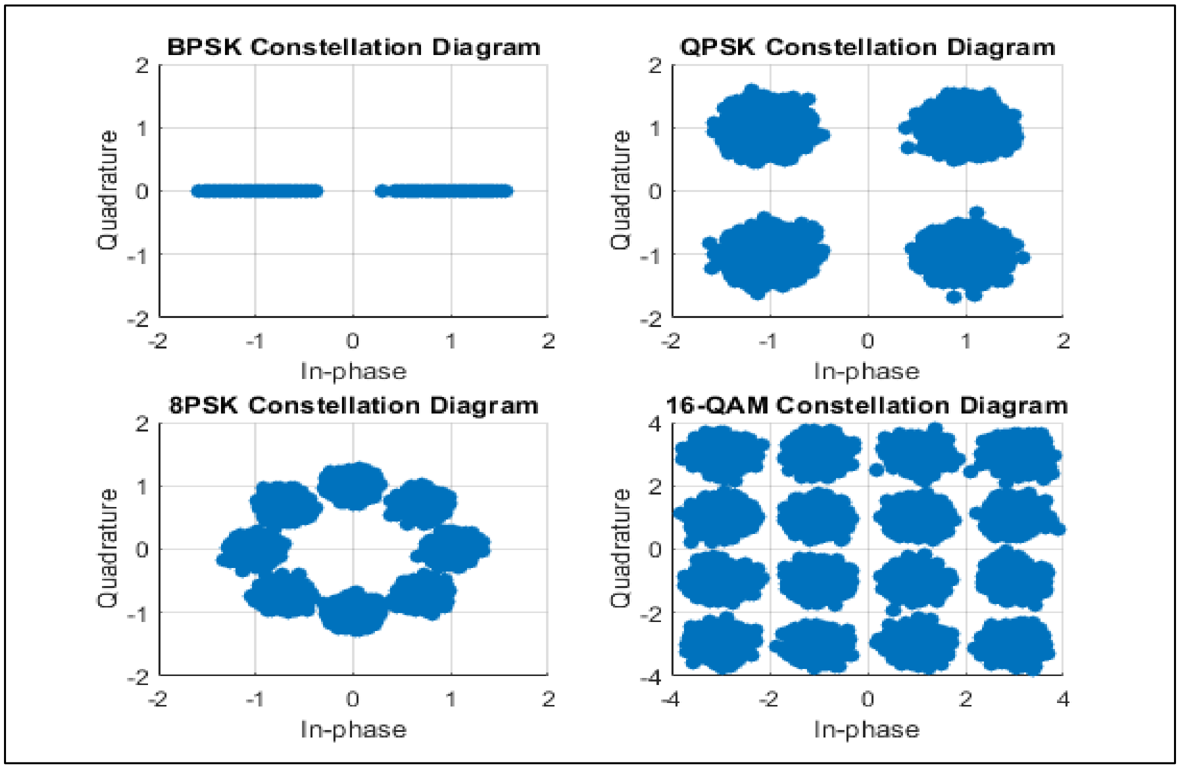
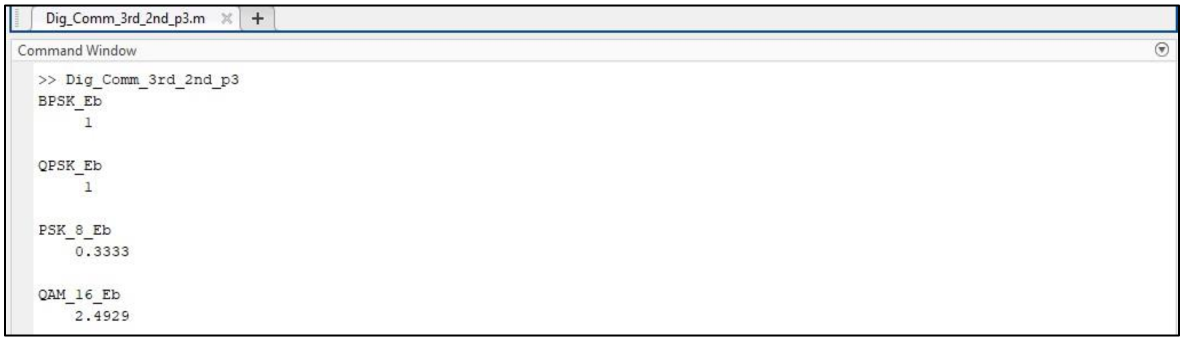


Figure 2: after adding AWGN.



```
>> Dig_Comm_3rd_2nd_p3
BPSK_Eb
     1

QPSK_Eb
     1

PSK_8_Eb
    0.3333

QAM_16_Eb
    2.4929
```

Figure 3: The Eb for each modulation schemes.

1.3 : The Demapper

We created a Demapper for BPSK and QPSK using the boundary region method. And for the 8PSK and 16QAM we computed the difference between the received symbol and each point in the constellation, and we determined the transmitted bit based on the smallest difference. We then calculated the BER by comparing the received signal with the actual transmitted signal and accumulating the number of bits that do not match the corresponding transmitted bit then divided it over the number of transmitted bits.

1.4 : Tasks

1.4.1: BER .vs Eb/No for all modulation schemes.

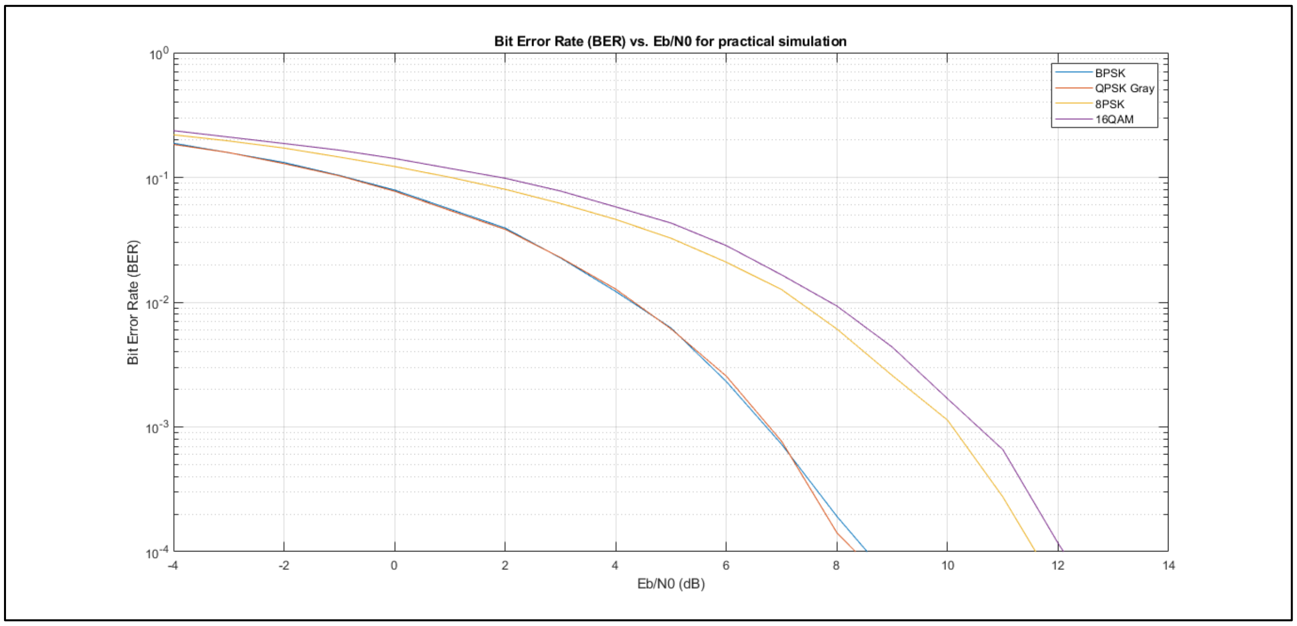


Figure 4: BER .vs Eb/N0 for all modulation schemes.

<u>Comment</u>

-We observe that the BER of the BPSK and QPSK-Gray are nearly the same because the using of QPSK-Gray make that if there is a symbol error and it enters an adjacent region there will be just an error in just 1 bit, and the probability of the symbol to enter to a distant region is very small.

-And as shown in figure 4 the BER of 8PSK is bigger than the BPSK and QPSK because the 8PSK has 3 bits per symbol which make it more sensitive to the error, and each symbol has a boundary region smaller than the QPSK and BPSK.

-Finally, the 16QAM is the biggest BER because each symbol is represented by 4 bits which make it more sensitive to the noise, but it has the best energy efficiency.

1.4.2 : The BER .vs Eb/No for all modulation schemes theoretical and practical in the same graph.

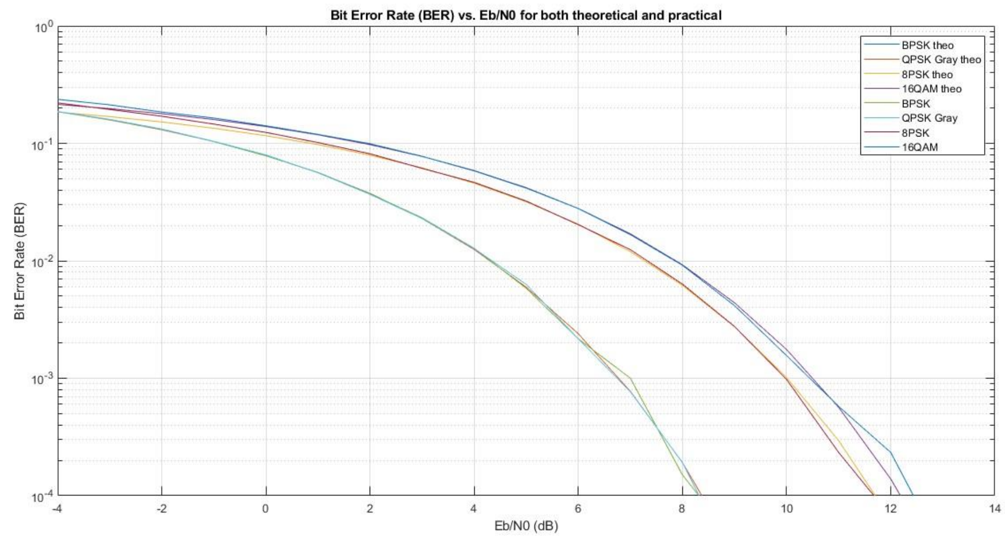


Figure 5: BER .vs Eb/N0 for all modulation schemes theoretical and simulated.

Comment

-The close match between the theoretical and practical as shown in figure 5 is due to the large number of bits used, and the more bits we used the more identicality we will find.

1.4.3 : The BER vs Eb/N0 for each modulation schemes alone

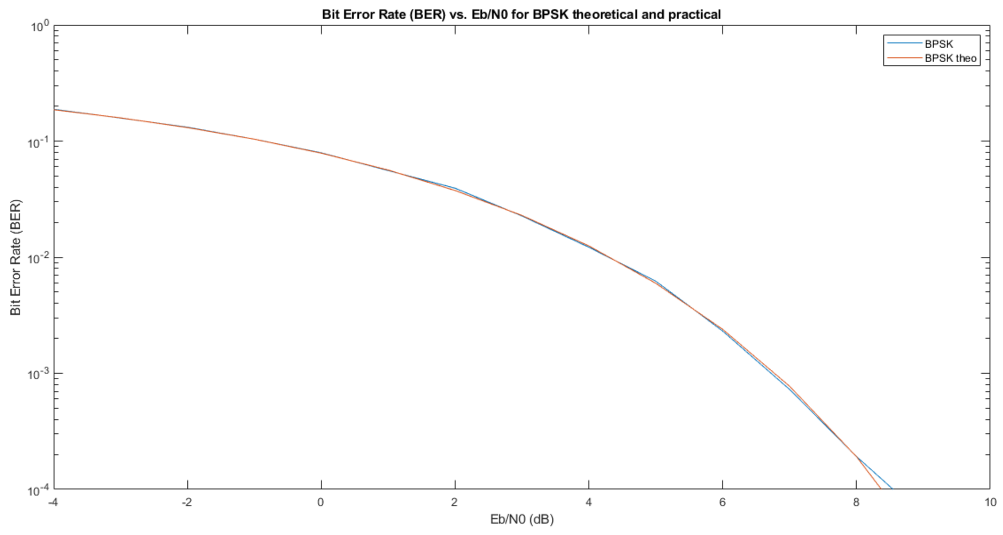


Figure 6: The BER .vs Eb/N0 for BPSK theoretical and practical.

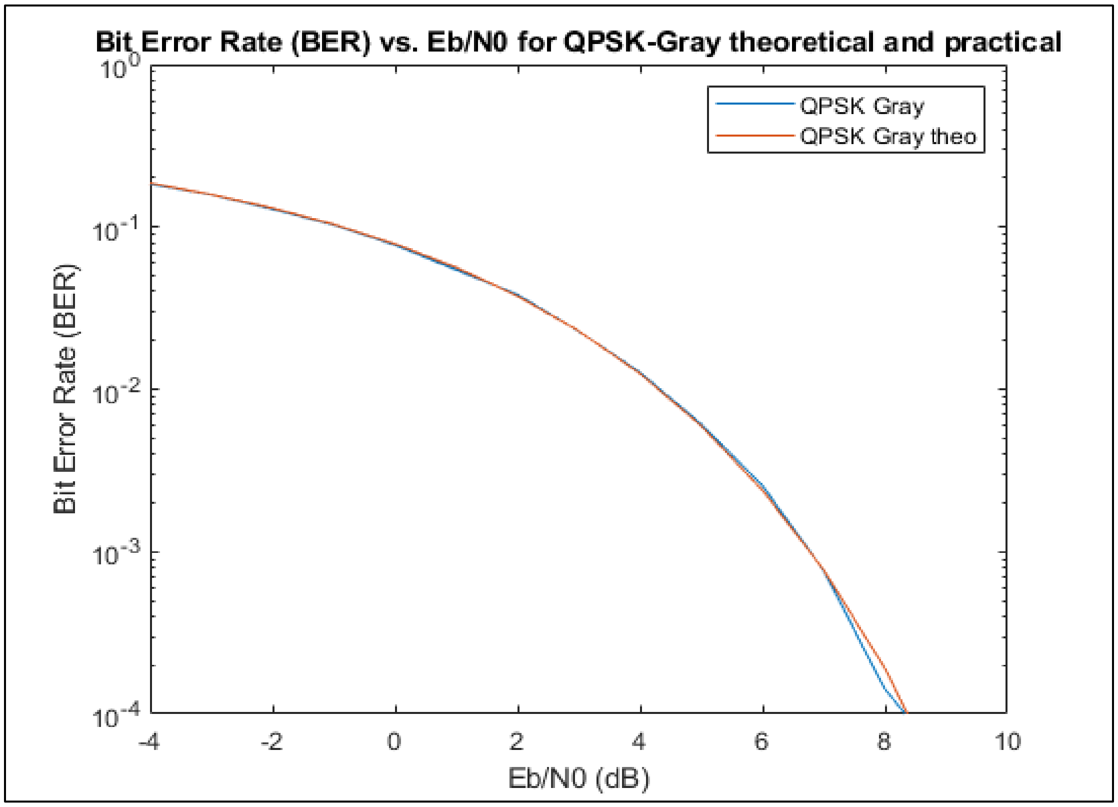


Figure 7:The BER .vs Eb/N0 for QPSK-Gray theoretical and practical.

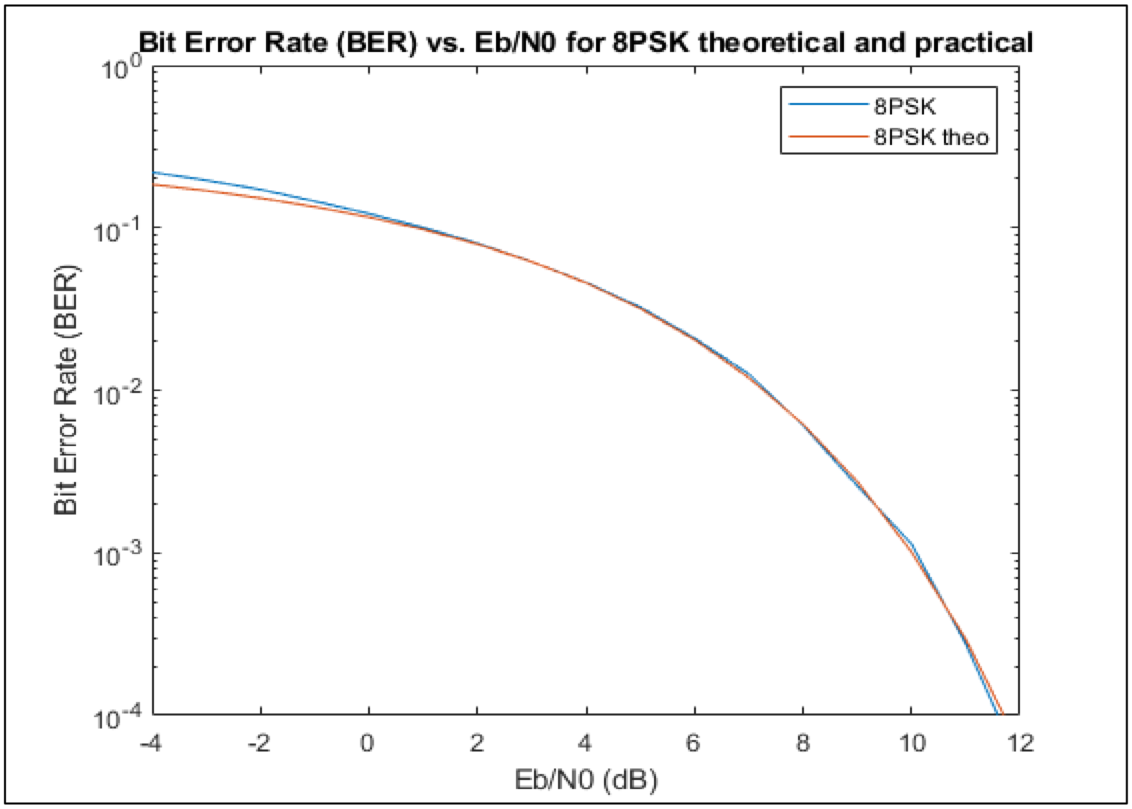


Figure 8: The BER .vs Eb/N0 for 8PSK theoretical and practical.

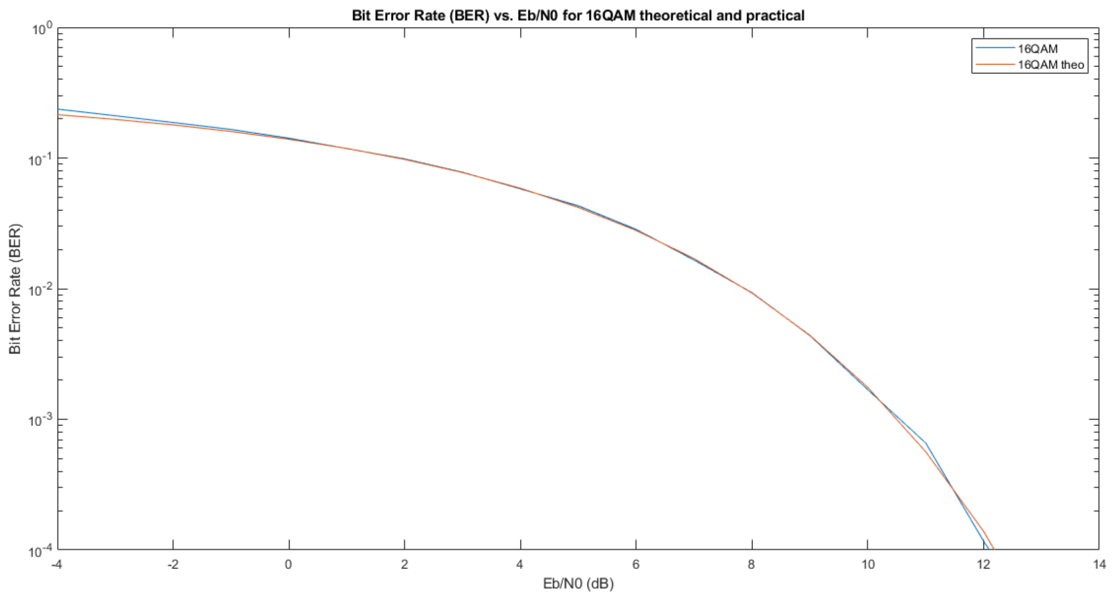


Figure 9 : The BER .vs Eb/N0 for 16QAM theoretical and practical.

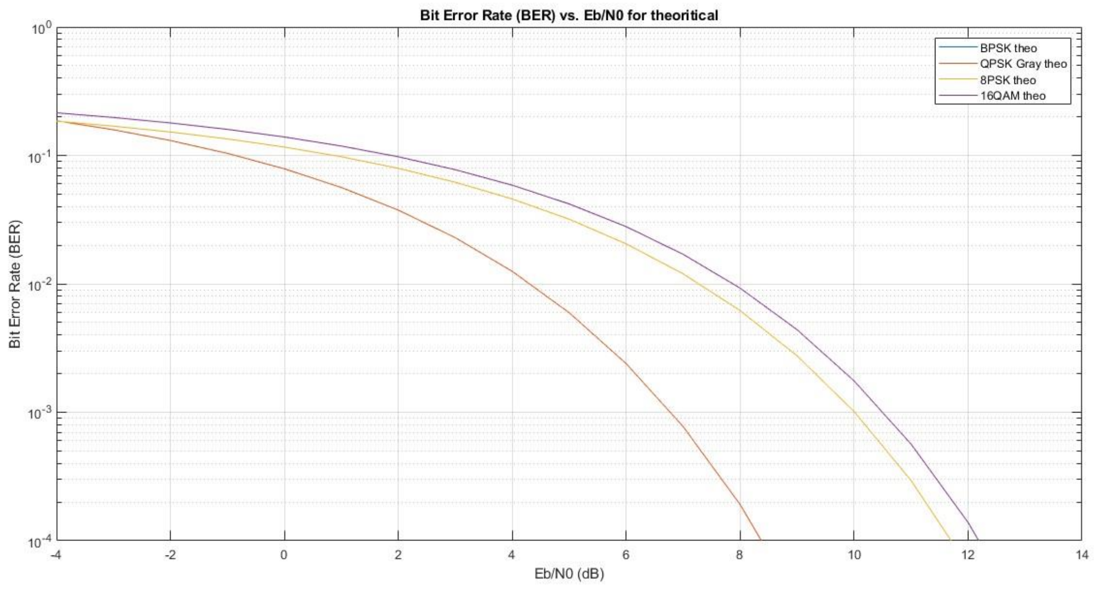


Figure 10 : All theoretical in one graph.

1.4.4 : QPSK-Gray and QPSK-not_Gray

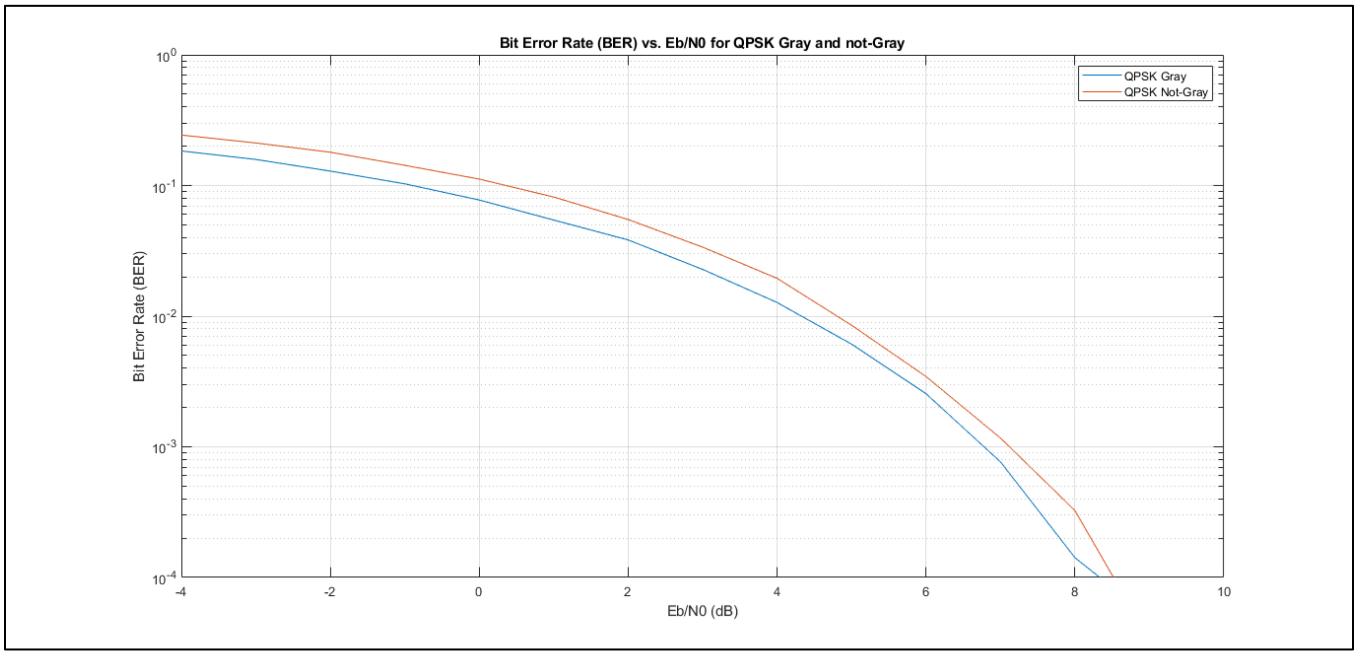


Figure 11: BER .vs Eb/N0 for QPSK Gray and Not Gray.

<u>Comment</u>

-As expected, we noticed that the BER of QPSK-not Gray is bigger than the QPSK-Gray because a symbol error in QPSK-Non-Gray results in a two-bit error, whereas in QPSK-Gray, it leads to only a single-bit error.

1.5 BFSK

The basis function of B-FSK modulation is:

$$\phi_{1(t)} = \sqrt{\frac{2E_b}{T_b}} cos(2\pi f_c t) \qquad \phi_{2(t)} = \sqrt{\frac{2E_b}{T_b}} cos(2\pi (f_c + \Delta f)t)$$

The baseband equivalent signal:

Assuming $f_c = f_1$ we get:

$$s_{1BB} = \sqrt{\frac{2E_b}{T_b}} \qquad s_{2BB} = \sqrt{\frac{2E_b}{T_b}} \left(cos(2\pi\Delta f t) + jsin(2\pi\Delta f t)\right)$$

Bit Error Rate:

To calculate the BER, we map 0 to 1 and 1 to i since each basis function represents a specific frequency and both are orthogonal to each other.

To recover the data after the channel we find the angle the point makes with the x-axis. If it is between 45 degrees and 225 degrees, then the original symbol represents 1 and vice versa, and we get the theoretical BER from $P_e = \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{2N_o}}\right)$

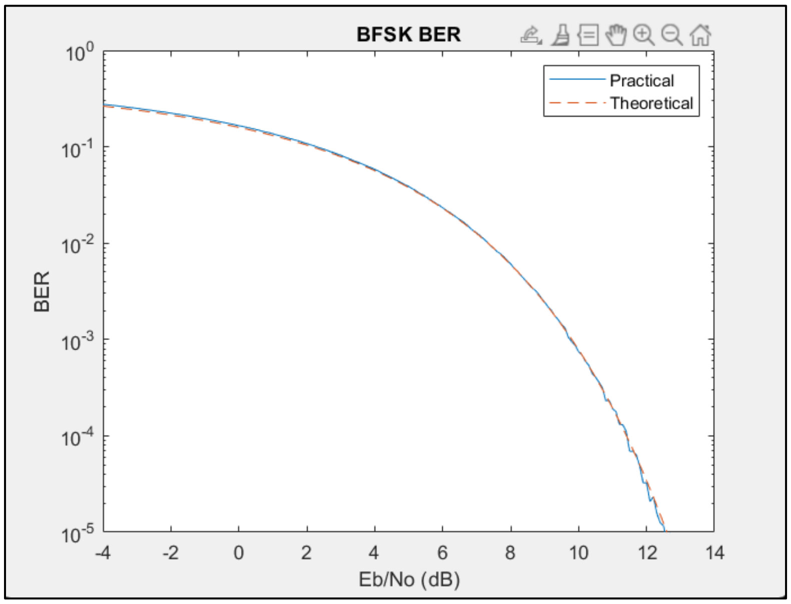


Figure 12 : BFSK BER.

Comment:

We see that the simulated and theoretical bit error are very close at different values of SNR.

Autocorrelation:

To calculate the autocorrelation function, we generate an ensemble of 10k realization and map the bits to the baseband equivalent (0 to s_{1BB} and 1 to s_{2BB}) and take 7 samples from each bit. Then we add random delay to each realization

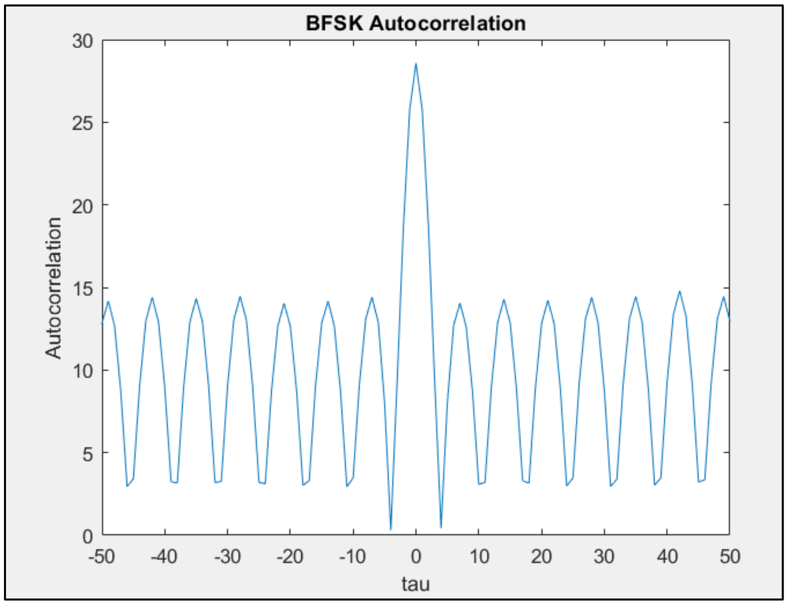


Figure 13 : Autocorrelation of BFSK.

PSD:

To calculate PSD, we get the Fourier transform of the autocorrelation function then we calculate the theoretical from the equation:

$$S_{BB}(f) = \frac{2E_b}{T_b}\left(\delta\left(f - \frac{1}{T_b}\right) + \delta\left(f + \frac{1}{T_b}\right) + \frac{8E_b cos^2(\pi f T_b)}{\pi^2(4T_b^2 f^2 - 1)^2}\right)$$

We shifted the theoretical PSD due to difference in the mapping between the theoretical and the simulated PSD.

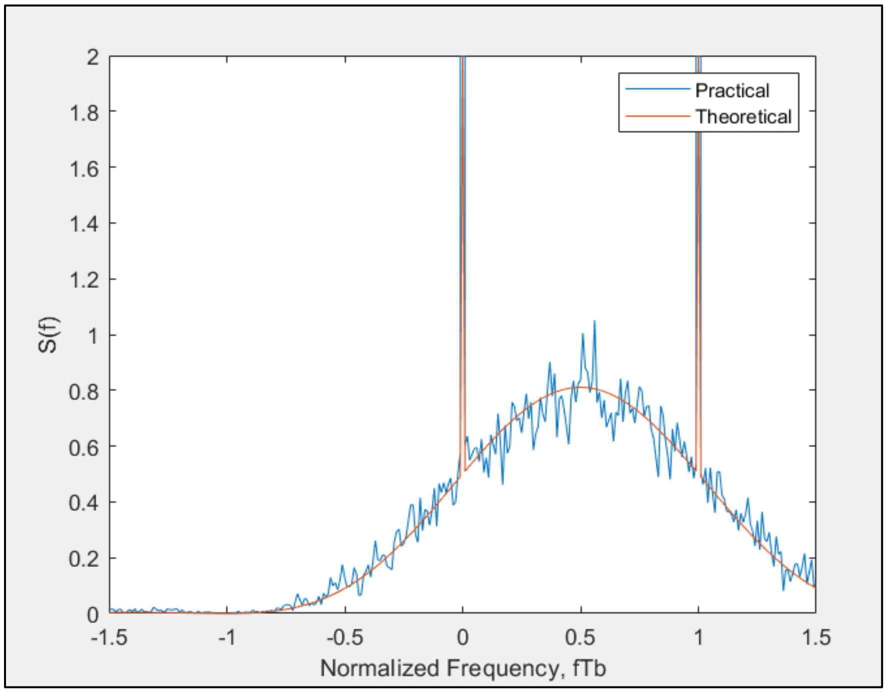


Figure 14 : PSD of BFSK

Comment:

We notice from the above figure that $\Delta f = \frac{1}{T_b}$, the $BW = \Delta f,$ and the practical PSD is very close to the theoretical.

MATLAB Code:

```
%---------------------Task 1.4------------------------------
number_of_bits = 12000;
input_data = randi([0 1], 1, number_of_bits);
%---------------------------------------------------------
BPSK_Mod = input_data * 2 - 1;
BPSK_Mod_squared = BPSK_Mod .^ 2;
BPSK_Eave = mean(BPSK_Mod_squared);
BPSK_Eb = BPSK_Eave /1;
disp('BPSK_Eb');
disp(BPSK_Eb);
%---------------------------------------------------------
QPSK_data = reshape(input_data, 2, number_of_bits/2)';
QPSK_decimal_data = bin2dec(num2str(QPSK_data));
QPSK_Constellation = [-1-1j -1+1j 1-1j 1+1j];
QPSK_Mod = QPSK_Constellation(QPSK_decimal_data+1);
QPSK_E = zeros(1,number_of_bits/2);
for i = 1:number_of_bits/2
    QPSK_E(1,i) = real(QPSK_Mod(i))^2 + imag(QPSK_Mod(i))^2;
end
QPSK_Eave = mean(QPSK_E);
QPSK_Eb = QPSK_Eave/2;
disp('QPSK_Eb');
disp(QPSK_Eb);
%---------------------------------------------------------
QPSK_Constellation_notGray = [-1-1j 1-1j 1+1j -1+1j];
QPSK_Mod_notGray = QPSK_Constellation_notGray(QPSK_decimal_data+1);
QPSK_NotGray_E = zeros(1,number_of_bits/2);
for i = 1:number_of_bits/2
    QPSK_NotGray_E(1,i) = real(QPSK_Mod_notGray(i))^2 +
imag(QPSK_Mod_notGray(i))^2;
end
QPSK_NotGray_Eave = mean(QPSK_NotGray_E);
QPSK_NotGray_Eb = QPSK_NotGray_Eave/2;
%---------------------------------------------------------
num = 1/sqrt(2);
PSK_8_data = reshape(input_data,3,number_of_bits/3)';
PSK_8_decimal_data = bin2dec(num2str(PSK_8_data));
PSK_8_Constellation = [1+0j num+num*1j -num+num*1j 0+1j num-num*1j 0-1j -1+0j -
num-num*1j];
PSK_8_Mod = PSK_8_Constellation(PSK_8_decimal_data+1);
PSK_8_E = zeros(1,number_of_bits/3);
for i = 1:number_of_bits/3
    PSK_8_E(1,i) = real(PSK_8_Mod(i))^2 + imag(PSK_8_Mod(i))^2;
end
PSK_8_Eave = mean(PSK_8_E);
PSK_8_Eb = PSK_8_Eave/3;
disp('PSK_8_Eb');
disp(PSK_8_Eb);
%---------------------------------------------------------
QAM_16_data = reshape(input_data, 4, number_of_bits/4)';
QAM_16_decimal_data = bin2dec(num2str(QAM_16_data));
```

```matlab
QAM_16_Constellation = [-3-3j -3-1j -3+3j -3+1j -1-3j -1-1j -1+3j -1+1j 3-3j 3-
1j 3+3j 3+1j 1-3j 1-1j 1+3j 1+1j];
QAM_16_Mod = QAM_16_Constellation(QAM_16_decimal_data+1);
QAM_16_E = zeros(1,number_of_bits/4);
for i = 1:number_of_bits/4
    QAM_16_E(1,i) = real(QAM_16_Mod(i))^2 + imag(QAM_16_Mod(i))^2;
end
QAM_16_Eave = mean(QAM_16_E);
QAM_16_Eb = QAM_16_Eave/4;
disp('QAM_16_Eb');
disp(QAM_16_Eb);
%-------------------------------------------------------
figure;
subplot(2, 2, 1);
scatter(real(BPSK_Mod), imag(BPSK_Mod), 'filled');
title('BPSK Constellation Diagram');
xlabel('In-phase');
ylabel('Quadrature');
axis([-2 2 -2 2]);
grid on;

subplot(2, 2, 2);
scatter(real(QPSK_Mod), imag(QPSK_Mod), 'filled');
title('QPSK Constellation Diagram');
xlabel('In-phase');
ylabel('Quadrature');
axis([-2 2 -2 2]);
grid on;

subplot(2, 2, 3);
scatter(real(PSK_8_Mod), imag(PSK_8_Mod), 'filled');
title('8PSK Constellation Diagram');
xlabel('In-phase');
ylabel('Quadrature');
axis([-2 2 -2 2]);
grid on;

subplot(2, 2, 4);
scatter(real(QAM_16_Mod), imag(QAM_16_Mod), 'filled');
title('16-QAM Constellation Diagram');
xlabel('In-phase');
ylabel('Quadrature');
axis([-4 4 -4 4]);
grid on;
%--------------------------------------------------------
SNR_range = -4:1:14;
BER_1 = zeros(size(SNR_range));
BER_2 = zeros(size(SNR_range));
BER_3 = zeros(size(SNR_range));
BER_4 = zeros(size(SNR_range));
BER_5 = zeros(size(SNR_range));
BER_1_theo = zeros(size(SNR_range));
BER_2_theo = zeros(size(SNR_range));
BER_3_theo = zeros(size(SNR_range));
BER_4_theo = zeros(size(SNR_range));
```

```
BER_5_theo = zeros(size(SNR_range));
for c = 1:length(SNR_range)
    % Convert Eb/N0 from dB to linear scale
    SNR = 10^(SNR_range(c)/10);
    % Calculate noise power (N0) using Eb/N0 and the signal energy (Eb)
    N0 = 1 / SNR;
    %-----------------------------------------------------
    BPSK_Mod_noisy        = BPSK_Mod         + sqrt(BPSK_Eb*N0/2)        *
randn(1,number_of_bits);
    QPSK_Mod_noisy        = QPSK_Mod         + sqrt(QPSK_Eb*N0/2)        *
(randn(1,number_of_bits/2)+1j*randn(1,number_of_bits/2));
    QPSK_Mod_notGray_noisy= QPSK_Mod_notGray + sqrt(QPSK_NotGray_Eb*N0/2)*
(randn(1,number_of_bits/2)+1j*randn(1,number_of_bits/2));
    PSK_8_Mod_noisy       = PSK_8_Mod        + sqrt(PSK_8_Eb*N0/2)       *
(randn(1,number_of_bits/3)+1j*randn(1,number_of_bits/3));
    QAM_16_Mod_noisy      = QAM_16_Mod       + sqrt(QAM_16_Eb*N0/2)      *
(randn(1,number_of_bits/4)+1j*randn(1,number_of_bits/4));
    %-----------------------------------------------------
    BPSK_Demapp = zeros(1,number_of_bits);
    for i = 1:number_of_bits
         if(BPSK_Mod_noisy(i)>0)
        BPSK_Demapp(1,i) = 1;
         else
        BPSK_Demapp(1,i) = 0;
         end
    end
%---------------------------------------------------------
    QPSK_Demapp = zeros(1,number_of_bits);
    k = 1;
    for i = 1:number_of_bits/2
        if(real(QPSK_Mod_noisy(i))>0 && imag(QPSK_Mod_noisy(i))>0)
            QPSK_Demapp(1,k) = 1;
            QPSK_Demapp(1,k+1)=1;
        elseif(real(QPSK_Mod_noisy(i))>0 && imag(QPSK_Mod_noisy(i))<0)
            QPSK_Demapp(1,k) = 1;
            QPSK_Demapp(1,k+1)=0;
        elseif(real(QPSK_Mod_noisy(i))<0 && imag(QPSK_Mod_noisy(i))<0)
            QPSK_Demapp(1,k) =0;
            QPSK_Demapp(1,k+1)=0;
        else
            QPSK_Demapp(1,k) = 0;
            QPSK_Demapp(1,k+1)=1;
        end
        k = k + 2;
    end
    %-------------------------------------------------------
    QPSK_NotGray_Demapp = zeros(1,number_of_bits);
    k = 1;
    for i = 1:number_of_bits/2
        if(real(QPSK_Mod_notGray_noisy(i))>0 &&
imag(QPSK_Mod_notGray_noisy(i))>0)
            QPSK_NotGray_Demapp(1,k) = 1;
            QPSK_NotGray_Demapp(1,k+1)=0;
        elseif(real(QPSK_Mod_notGray_noisy(i))>0 &&
imag(QPSK_Mod_notGray_noisy(i))<0)
```

```matlab
            QPSK_NotGray_Demapp(1,k) = 0;
            QPSK_NotGray_Demapp(1,k+1)=1;
        elseif(real(QPSK_Mod_notGray_noisy(i))<0 &&
imag(QPSK_Mod_notGray_noisy(i))<0)
            QPSK_NotGray_Demapp(1,k) =0;
            QPSK_NotGray_Demapp(1,k+1)=0;
        else
            QPSK_NotGray_Demapp(1,k) = 1;
            QPSK_NotGray_Demapp(1,k+1)=1;
        end
        k = k + 2;
    end
    %--------------------------------------------------------
    conv1 = [0 0 0; 0 0 1 ; 0 1 0 ; 0 1 1 ; 1 0 0 ; 1 0 1 ; 1 1 0 ; 1 1 1];
    demp1 = zeros(number_of_bits/3,3);
    for i = 1:number_of_bits/3
        [~, index] = min(abs(PSK_8_Mod_noisy(i) - PSK_8_Constellation));
        demp1(i,:) = conv1(index,:);
    end
    PSK_8_Demapp = reshape(demp1.', 1, number_of_bits);
    %--------------------------------------------------------
    conv2 = [0 0 0 0;0 0 0 1 ;0 0 1 0 ;0 0 1 1 ;0 1 0 0 ;0 1 0 1 ;0 1 1 0 ;0 1 1
1;
             1 0 0 0;1 0 0 1;1 0 1 0;1 0 1 1;1 1 0 0;1 1 0 1;1 1 1 0;1 1 1 1];
    demp2 = zeros(number_of_bits/4,4);
    for i = 1:number_of_bits/4
        [~, index] = min(abs(QAM_16_Mod_noisy(i) - QAM_16_Constellation));
        demp2(i,:) = conv2(index,:);
    end
    QAM_16_Demapp = reshape(demp2.', 1, number_of_bits);
    %--------------------------------------------------------
    Error_bits_1    = sum(BPSK_Demapp ~= input_data);
    BER_1(c)        = Error_bits_1 / number_of_bits;
    BER_1_theo(c)   = 0.5 * erfc(sqrt(BPSK_Eb/N0));

    Error_bits_2    = sum(QPSK_Demapp ~= input_data);
    BER_2(c)        = Error_bits_2 / number_of_bits;
    BER_2_theo(c)   = 0.5 * erfc(sqrt(QPSK_Eb/N0));

    Error_bits_3    = sum(QPSK_NotGray_Demapp ~= input_data);
    BER_3(c)        = Error_bits_3 / number_of_bits;

    Error_bits_4    = sum(PSK_8_Demapp ~= input_data);
    BER_4(c)        = Error_bits_4 / number_of_bits;
    BER_4_theo(c)   = (1/3)*erfc(sqrt(3/N0)*sin(pi/8));

    Error_bits_5    = sum(QAM_16_Demapp ~= input_data);
    BER_5(c)        = Error_bits_5 / number_of_bits;
    BER_5_theo(c)   = (1.5/4) * erfc(sqrt(1/(2.5*N0)));
end
%--------------------------------------------------------
figure;
subplot(2, 2, 1);
scatter(real(BPSK_Mod_noisy), imag(BPSK_Mod_noisy), 'filled');
title('BPSK Constellation Diagram');
```

```matlab
xlabel('In-phase');
ylabel('Quadrature');
axis([-2 2 -2 2]);
grid on;

subplot(2, 2, 2);
scatter(real(QPSK_Mod_noisy), imag(QPSK_Mod_noisy), 'filled');
title('QPSK Constellation Diagram');
xlabel('In-phase');
ylabel('Quadrature');
axis([-2 2 -2 2]);
grid on;

subplot(2, 2, 3);
scatter(real(PSK_8_Mod_noisy), imag(PSK_8_Mod_noisy), 'filled');
title('8PSK Constellation Diagram');
xlabel('In-phase');
ylabel('Quadrature');
axis([-2 2 -2 2]);
grid on;

subplot(2, 2, 4);
scatter(real(QAM_16_Mod_noisy), imag(QAM_16_Mod_noisy), 'filled');
title('16-QAM Constellation Diagram');
xlabel('In-phase');
ylabel('Quadrature');
axis([-4 4 -4 4]);
grid on;
%------------------------------------------------------
%------------------------------------------------------
figure;
semilogy(SNR_range, BER_1);
hold on;
semilogy(SNR_range, BER_1_theo);
title('Bit Error Rate (BER) vs. Eb/N0 for BPSK theoretical and practical');
xlabel('Eb/N0 (dB)');
ylabel('Bit Error Rate (BER)');
legend('BPSK', 'BPSK theo');
ylim([1e-4, 1]);

figure;
semilogy(SNR_range, BER_2);
hold on;
semilogy(SNR_range, BER_2_theo);
title('Bit Error Rate (BER) vs. Eb/N0 for QPSK-Gray theoretical and practical');
xlabel('Eb/N0 (dB)');
ylabel('Bit Error Rate (BER)');
legend('QPSK Gray', 'QPSK Gray theo');
ylim([1e-4, 1]);

figure;
semilogy(SNR_range, BER_4);
hold on;
semilogy(SNR_range, BER_4_theo);
title('Bit Error Rate (BER) vs. Eb/N0 for 8PSK theoretical and practical');
```

```
xlabel('Eb/N0 (dB)');
ylabel('Bit Error Rate (BER)');
legend('8PSK', '8PSK theo');
ylim([1e-4, 1]);
figure;
semilogy(SNR_range, BER_5);
hold on;
semilogy(SNR_range, BER_5_theo);
title('Bit Error Rate (BER) vs. Eb/N0 for 16QAM theoretical and practical');
xlabel('Eb/N0 (dB)');
ylabel('Bit Error Rate (BER)');
legend('16QAM', '16QAM theo');
ylim([1e-4, 1]);
%-------------------------------------------------------
figure;
semilogy(SNR_range, BER_1);
hold on;
semilogy(SNR_range, BER_2);
semilogy(SNR_range, BER_4);
semilogy(SNR_range, BER_5);
title('Bit Error Rate (BER) vs. Eb/N0 for practical simulation');
xlabel('Eb/N0 (dB)');
ylabel('Bit Error Rate (BER)');
grid on;
legend('BPSK', 'QPSK Gray' ,'8PSK', '16QAM');
ylim([1e-4, 1]);
%-------------------------------------------------------
figure;
semilogy(SNR_range, BER_1_theo);
hold on;
semilogy(SNR_range, BER_2_theo);
semilogy(SNR_range, BER_4_theo);
semilogy(SNR_range, BER_5_theo);
title('Bit Error Rate (BER) vs. Eb/N0 for theoretical');
xlabel('Eb/N0 (dB)');
ylabel('Bit Error Rate (BER)');
grid on;
legend('BPSK theo', 'QPSK Gray theo','8PSK theo','16QAM theo');
ylim([1e-4, 1]);
%-------------------------------------------------------
figure;
semilogy(SNR_range, BER_1_theo);
hold on;
semilogy(SNR_range, BER_2_theo);
semilogy(SNR_range, BER_4_theo);
semilogy(SNR_range, BER_5_theo);
semilogy(SNR_range, BER_1);
semilogy(SNR_range, BER_2);
semilogy(SNR_range, BER_4);
semilogy(SNR_range, BER_5);
title('Bit Error Rate (BER) vs. Eb/N0 for both theoretical and practical');
xlabel('Eb/N0 (dB)');
ylabel('Bit Error Rate (BER)');
grid on;
```

```
legend('BPSK theo', 'QPSK Gray theo','8PSK theo','16QAM theo','BPSK', 'QPSK
Gray' ,'8PSK', '16QAM');
ylim([1e-4, 1]);
%-----------------------------------------------------
figure;
semilogy(SNR_range, BER_2);
hold on;
semilogy(SNR_range, BER_3);
title('Bit Error Rate (BER) vs. Eb/N0 for QPSK Gray and not-Gray');
xlabel('Eb/N0 (dB)');
ylabel('Bit Error Rate (BER)');
grid on;
legend('QPSK Gray','QPSK Not-Gray');
ylim([1e-4, 1]);
%-----------------------------------------------------
%-----------------------------------------------------
%-------------------------BFSK------------------------
%-----------------------------------------------------
%-----------------------------------------------------

%%BFSK
N=1200000; M =2;Eb=1;
snr_dB = -4:0.1:14;
binary_data = randi([0 1], 1, N);
BFSK_mapped = zeros(1, N);
BFSK_N_errors = zeros(size(snr_dB));
BFSK_theoretical = zeros(size(snr_dB));
%% BFSK Mapper
% mapping 0 -> 1, 1 -> i
for j = 1:N
    if(binary_data(j) == 0)
        BFSK_mapped(j) = 1;
    else
        BFSK_mapped(j) = 1i;
    end
end
%% Channel
% adding gaussian noise with zero mean and No/2 variance
for j=1:length(snr_dB)
    No = Eb/(10^(snr_dB(j)/10));
    var = sqrt(No/2);
    noise = (randn(size(BFSK_mapped))+1i* randn(size(BFSK_mapped))) .*
sqrt(No/2) ; % real and img gaussian noise
    channel_output = BFSK_mapped + noise;

%% BFSK Demapper
    BFSK_demapped = zeros(1, N);
    for k = 1:N
        p = angle(channel_output(k));
        if(p>pi/4 && p<5*pi/4) % 1 -> angle with x axis between 45 and 225
            BFSK_demapped(k) = 1;
        else
            BFSK_demapped(k) = 0;
        end
    end
```

```
%% BFSK BER
    % practical
    for k = 1:N
        if(binary_data(k) ~= BFSK_demapped(k))
            BFSK_N_errors(j) = BFSK_N_errors(j)+1;
        end
    end
     BFSK_N_errors(j) = BFSK_N_errors(j)/N;
    % theoretical
    BFSK_theoretical(j) = 0.5*erfc(sqrt(Eb/(2*No)));
end
%% Plotting
figure('Name', 'BFSK BER');
semilogy(snr_dB, BFSK_N_errors);
hold on;
semilogy(snr_dB, BFSK_theoretical, "--");
hold off;
title('BFSK BER');
legend('Practical', 'Theoretical')
xlabel('Eb/No (dB)');
ylabel('BER')
xlim([-4 14])
ylim([10^-5 1])

%% Part 1.5
% declaring parametrs
N = 100;
N_realization = 10000;
data = randi([0 1], N_realization, N+1);
sampled_data = repelem(data,1, 7);
Tb = 0.07; % each sample take 0.01 second
t = 0:0.01:0.07;
Fs = 100;
tx_out = zeros(N_realization, 707);
tx_with_delay = zeros(N_realization, 700);
% mapping to BB signals
for i =  1:N_realization
    for j = 1:7:707
        if(sampled_data(i,j) == 0)
            tx_out(i,j:j+6) = sqrt(2*Eb/Tb);
        else
            for k = 1:7
                tx_out(i,j+k-1) = sqrt(2*Eb/Tb)*(cos(2*pi*t(k)/Tb) +
1i*sin(2*pi*t(k)/Tb));
            end
        end
    end
end

% random delay
for i = 1:N_realization
    r = randi([0 6]);
    tx_with_delay(i,:) = tx_out(i,r+1:700+r);
end
```

```
% Autocorrelation
BFSK_autocorr = zeros(1,700);
for j = -349:350
    i = j+350;
    p = conj(tx_with_delay(:, 350)) .* tx_with_delay(:, i);
    BFSK_autocorr(i) = sum(p)/length(p);
end
Rx_BFSK = BFSK_autocorr;

figure('Name', 'Autocorrelation');
plot([-699:699] -350, abs(fliplr([Rx_BFSK Rx_BFSK(2:end)])));
xlabel("tau"); ylabel("Autocorrelation");
xlim([-50 50]);
title("BFSK Autocorrelation");

% PSD
BFSK_PSD = fftshift(fft(Rx_BFSK));
f = (-350:349)/700 *Fs;
% PSD_theoritical = 2*Eb/Tb *(dirac(f - 1) + dirac(f)) +
8*Eb*cos(pi*(f).^2/(pi^2*(4*Tb^2*f.^2-1)));
PSD_theoritical = (8*cos(pi*Tb*f).^2)./(pi^2*(4*Tb^2*f.^2-1).^2);

idx = PSD_theoritical == Inf; % find the location of Inf
PSD_theoritical(idx) = 2; % change Inf to a finite value for plotting
figure('Name', 'PSD');
 plot(f*Tb, abs(BFSK_PSD)/100);
 hold on;
 plot(f*Tb+0.5, abs(PSD_theoritical));
 hold off;
xlim([-1.5 1.5])
ylim([0 2]);
legend('Practical', 'Theoretical')
xlabel('Normalized Frequency, fTb'); ylabel('S(f)');
```