Cairo University
Faculty of Engineering

Department of Computer Engineering

ELC 3070 – Spring 2024

Communications 2

Project #3

Submitted to

Dr. Mohamed Khairy

Dr. Nafie

Eng. Mohamed Khaled

Submitted by

Team: 12

Name	Section	BN
عمرو عبد المتجلي احمد متولي	3	16
علياء عصام الدين نجيب محمد	3	6

Role of each member:

Name	Role
عمرو عبد المتجلي احمد متولي	• Made the code & some of the report
علياء عصام الدين نجيب محمد	• Made the code and some of the report

Contents

Figures:

1. Tasks:

- ## 1.1. Theoretical vs practical BER graph of the BPSK:

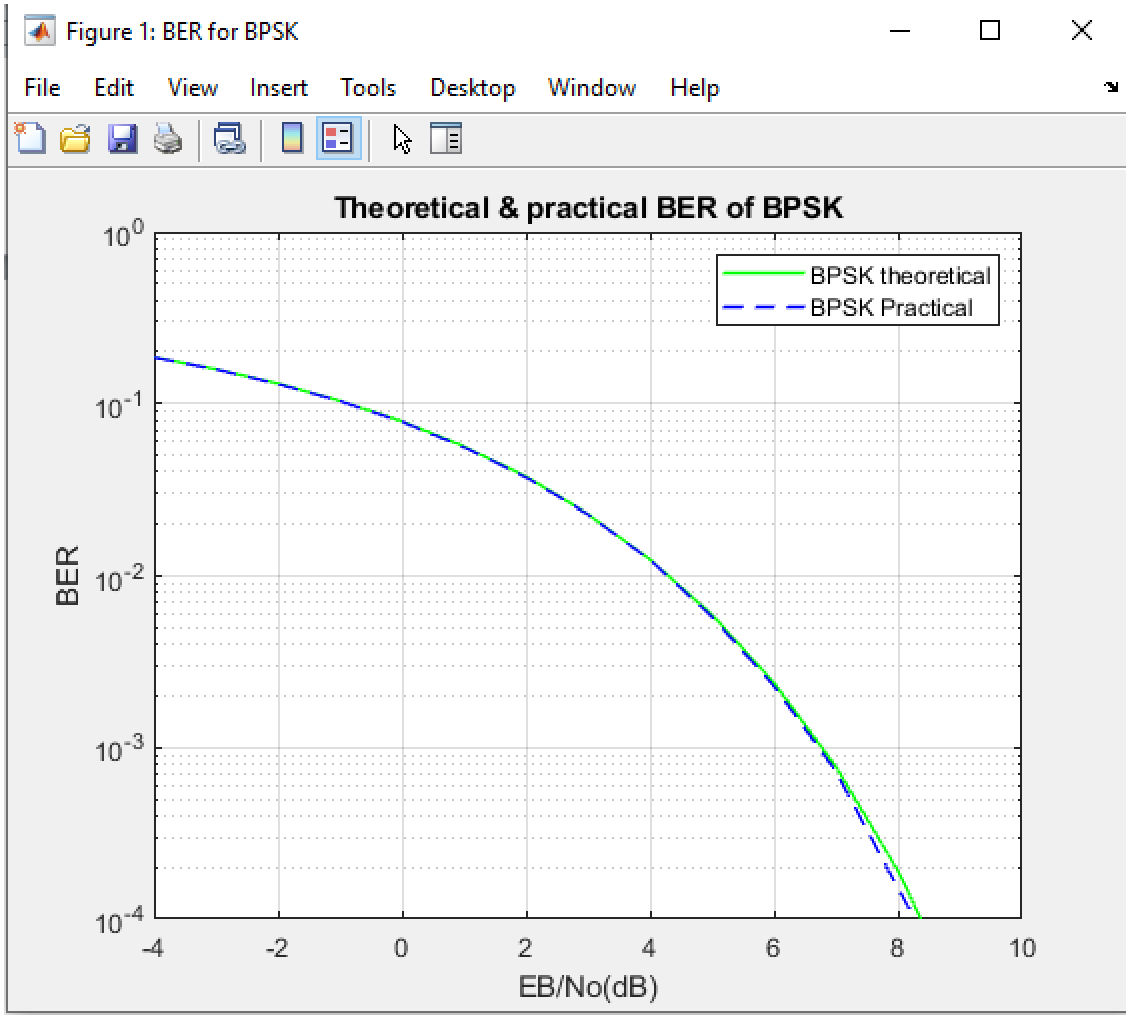

Figure 1: Theoretical and practical BER vs EB/No for BPSK

Comment: As we can see they are almost the same and they become more similar when the number of binary data used increases, we used number of bits equals 48*100000.

- **1.2. Theoretical vs practical BER graph of the QPSK Gray-Encoded:**

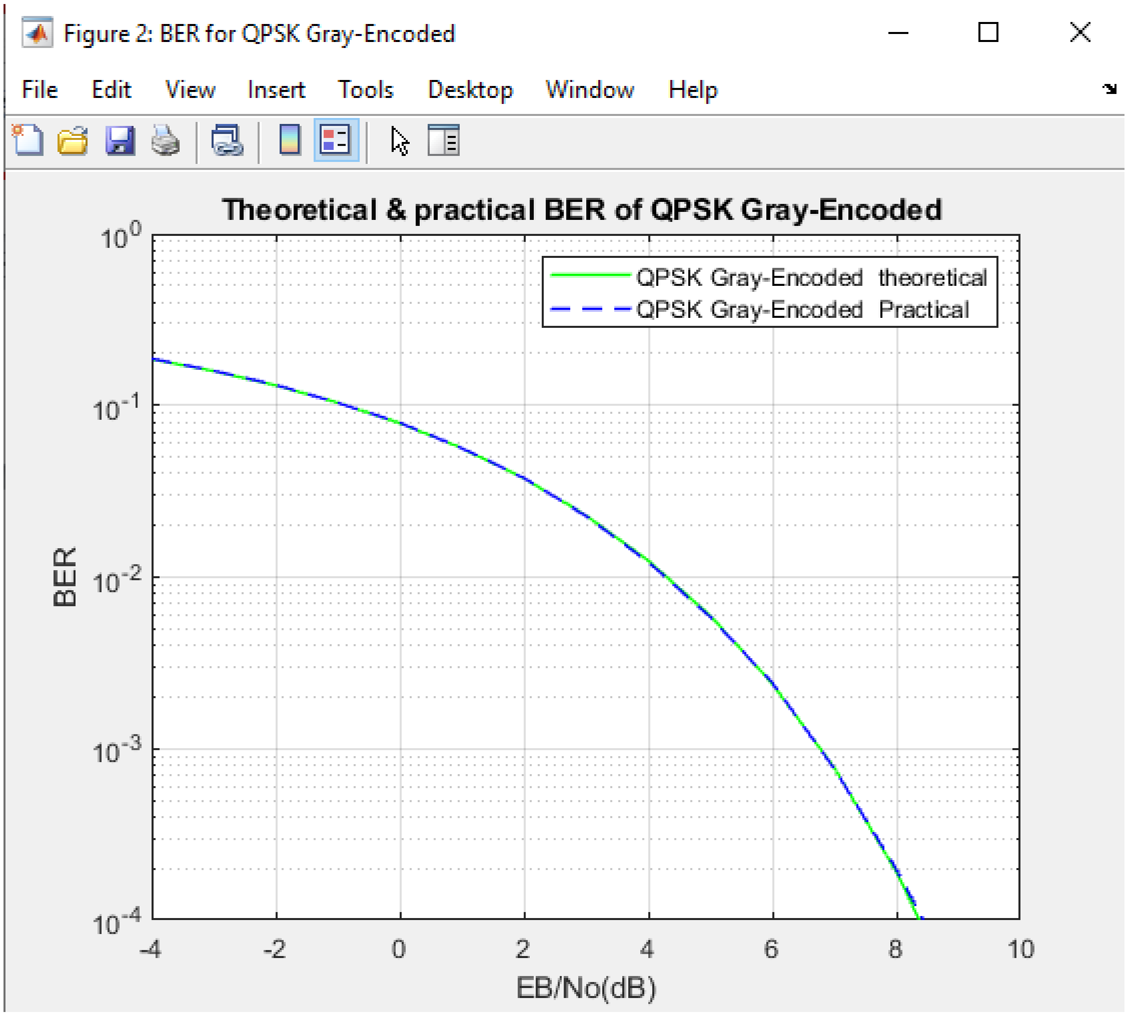

Figure 2: Theoretical and practical BER vs EB/No for QPSK Grey-Encoded

Comment: As we can see they are almost the same.

- **1.3. Theoretical vs practical BER graph of the QPSK Not-Grey-Encoded:**

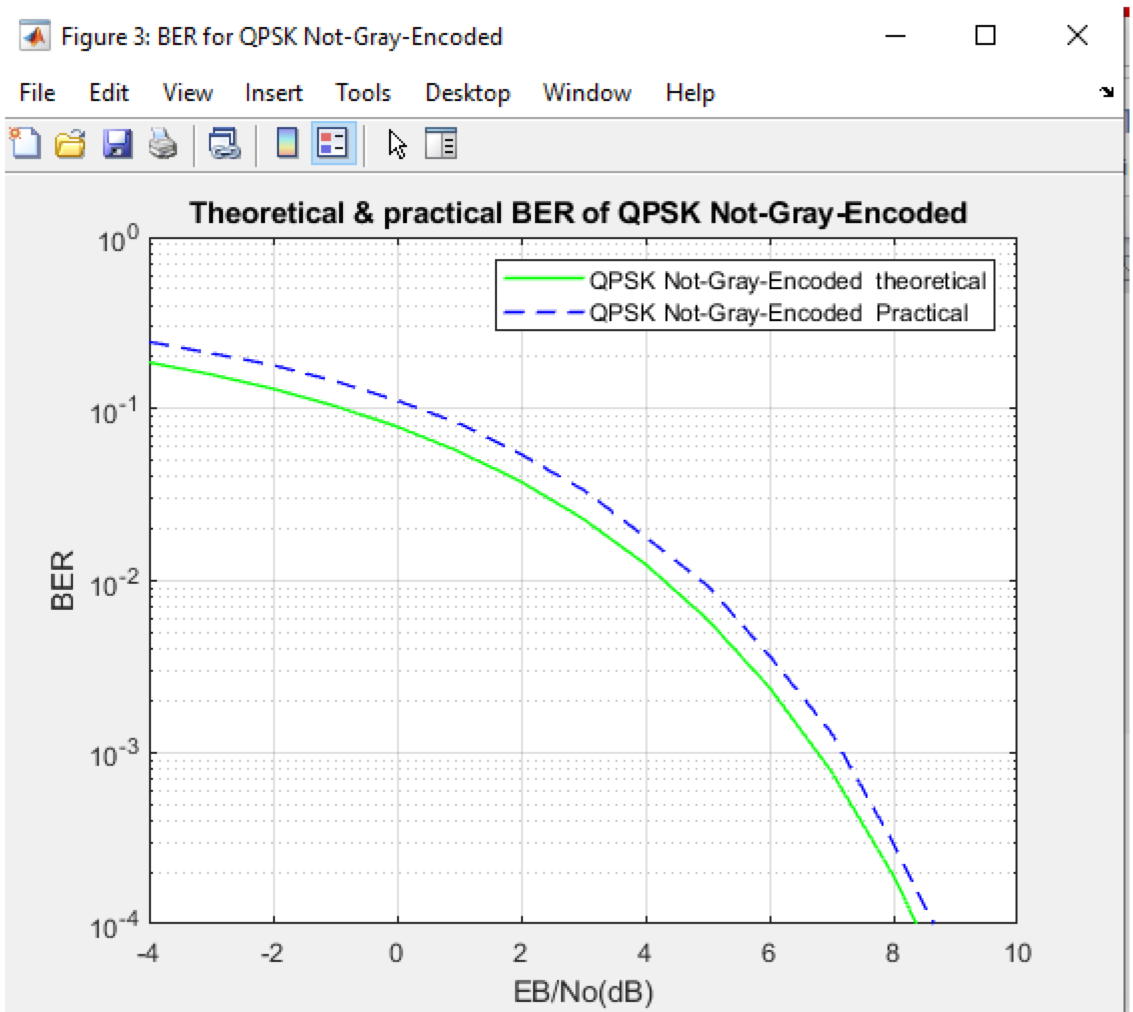

Figure 3: Theoretical and practical BER vs EB/No for QPSK Not-Grey-Encoded

Comment: As we can see the practical BER of QPSK Not-Grey-Encoded is above the theoretical with a slightly larger value indicating there is more error as there will be two bits changes in some cases.

- ## 1.4. Theoretical vs practical BER graph of the 8PSK:

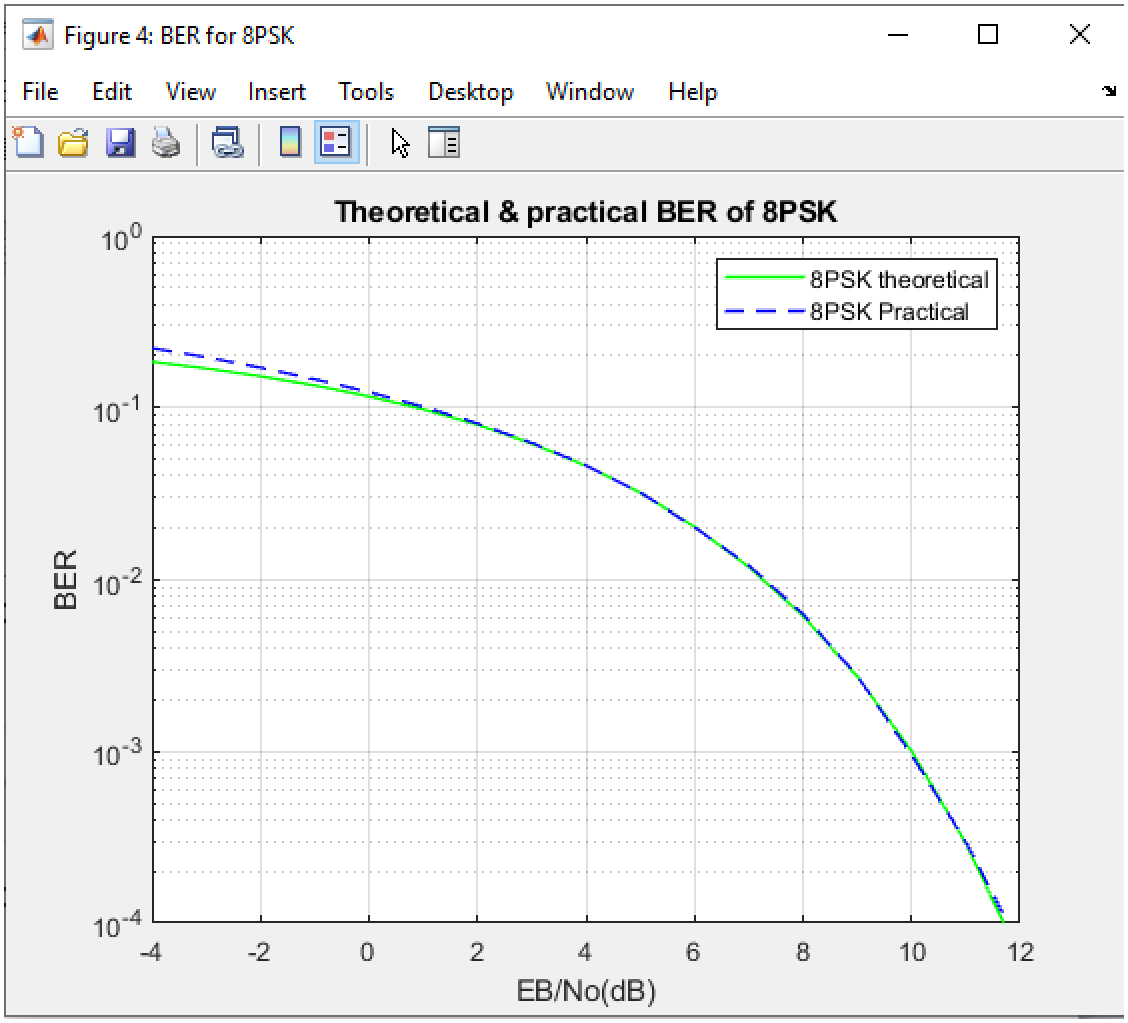

Figure 4: Theoretical and practical BER vs EB/No for 8PSK

Comment: As we can see they are almost the same and they become more similar when the number of binary data used increases.

- **1.5. Theoretical vs practical BER graph of the 16QAM:**

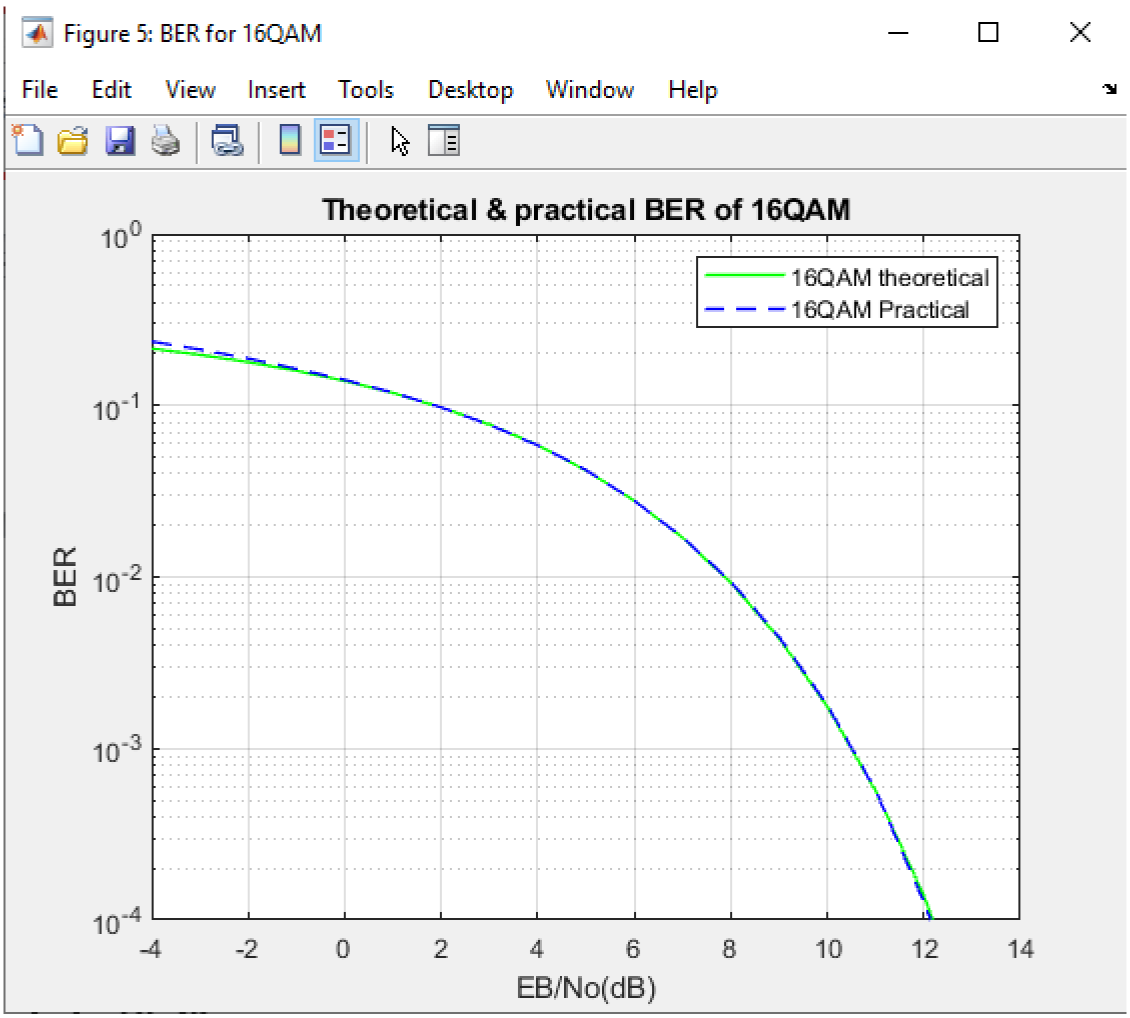

Figure 5: Theoretical and practical BER vs EB/No for 16QAM

Comment: As we can see they are almost the same.

- **1.6. Theoretical and practical BER graph for all the four modulation schemes:**

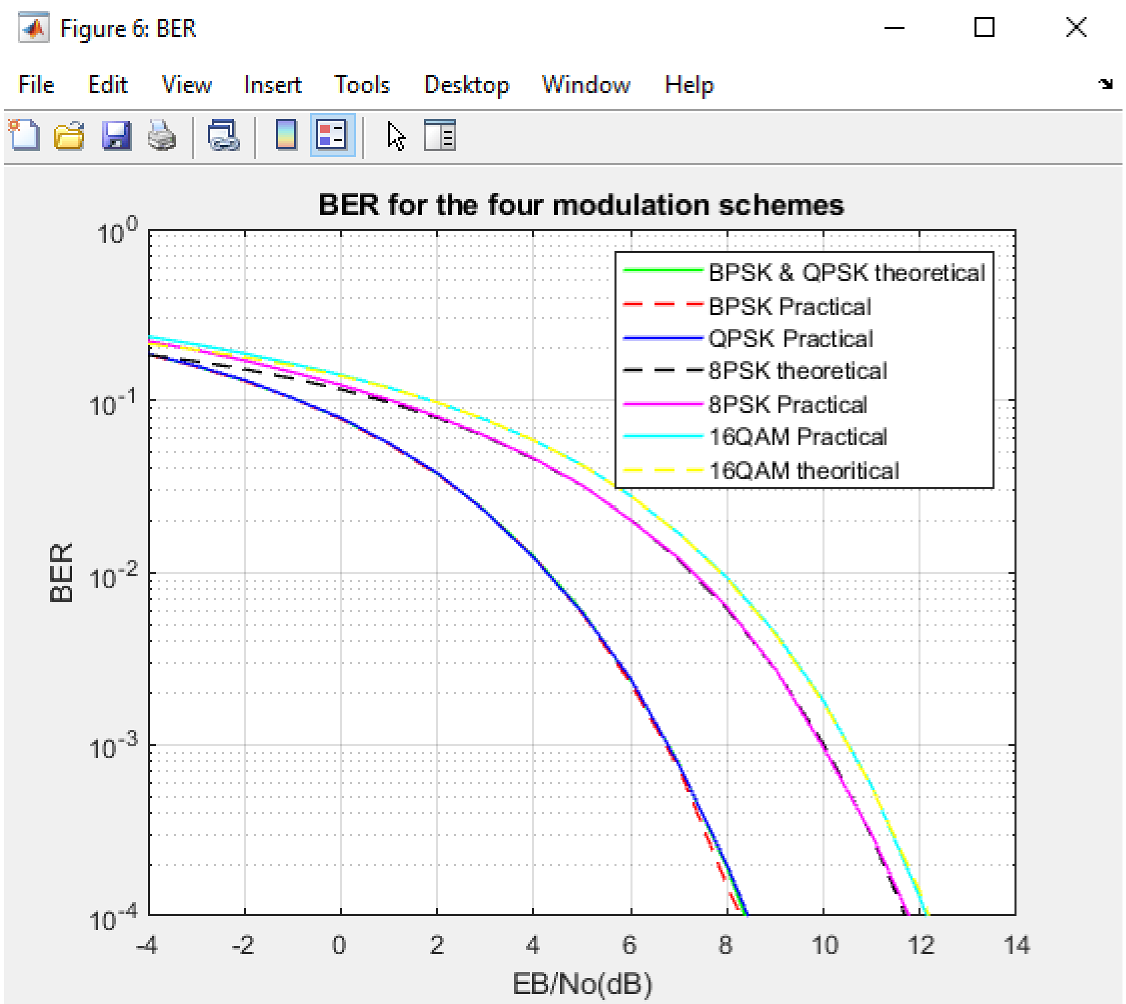

Figure 6: Theoretical and practical BER vs EB/No for all four modulation schemes

Comment:

1) The power of the signal in case of 16QAM must be higher than that of BPSK by 8.5dB to have the same BER for both cases.
2) As we can see the BER of the BPSK and QPSK are the same and also their practical BER are also the same as there will be a change for one bit at a time from one symbol to another one beside it.

3) The BER of the BPSK and QPSK is smaller than that of 16QAM and 8PSK for the same energy per bit so they need more energy to have the same BER.
4) In case of 16QAM, we used more bits than the previous types of modulation and also used grey coding for the used bits resulting in 16 symbols being used, so this increases the BER as the region for each symbols decreases significantly.

- **1.7. Theoretical and practical BER graph for QPSK Gray-Encoded and Not-Gray-Encoded:**

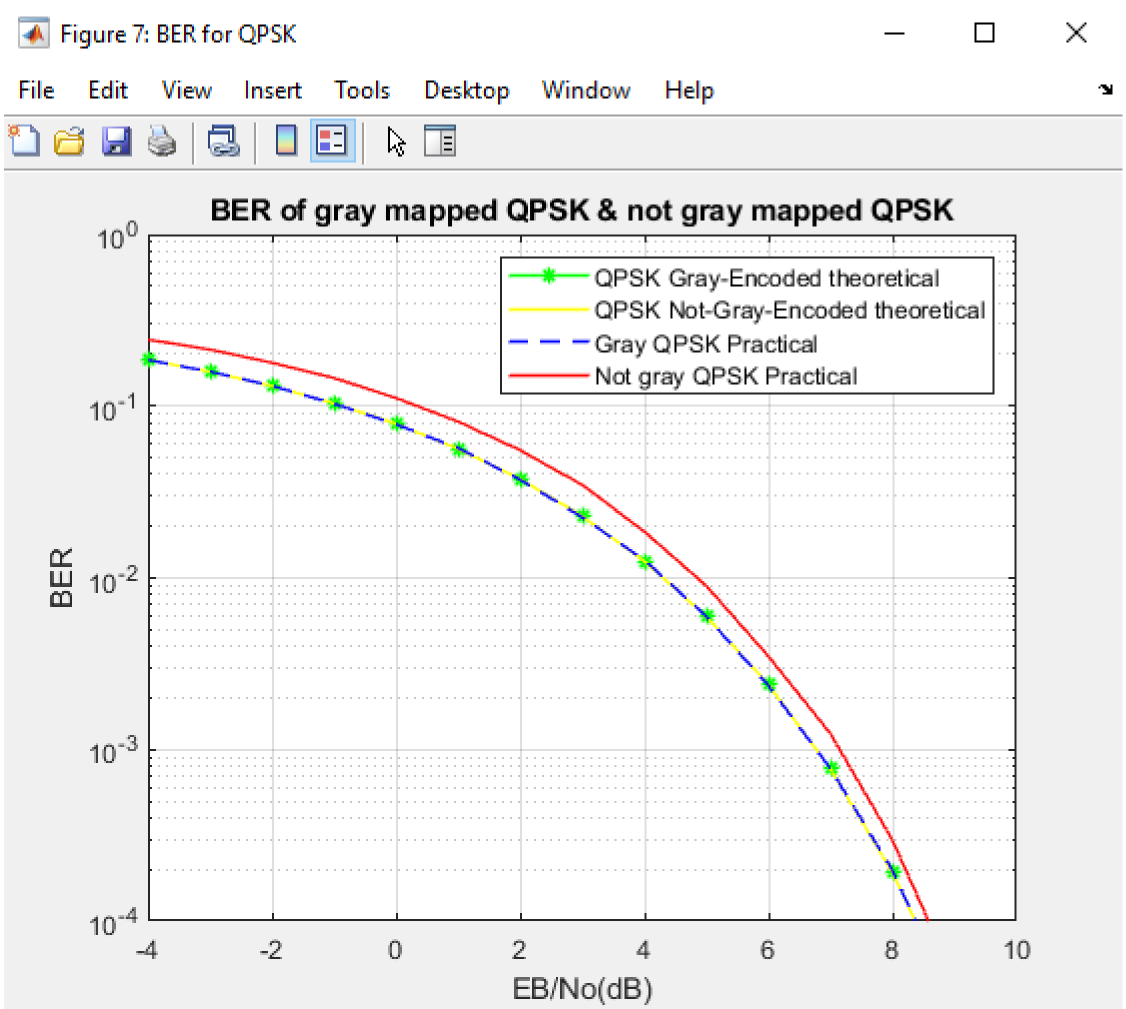

Figure 7: Theoretical and practical BER vs EB/No for QPSK Gray-Encoded and Not-Grey-Encoded

Comment:

As we can see the BER of QPSK Not-Grey-Encoded is above the QPSK Grey-Encoded with a slightly larger value indicating there is more error as in QPSK Not-Grey-Encoded there will be two bits changes in some cases when we go from one symbol to the other symbol beside it, while in the QPSK Grey-Encoded there will be only one bit changes at a time when we go from one symbol to the other symbol beside it, so we concluded that QBSK Gray-Encoding decreases the BER.

2. <u>BFSK:</u>

- **2.1. This is the basis functions of the signal set:**

$$\emptyset_i(t) = \sqrt{\frac{2}{T_b}} * \cos(2\pi f_i t)$$

$$where\ T_X\ frequency\ is\ f_i = \frac{n_c + i}{T_b} \quad , i = 1,2$$

This is the 2 basis functions of the signal set:

$$\emptyset_1(t) = \sqrt{\frac{2}{T_b}} * \cos(2\pi f_1 t) \quad ,$$

$$\emptyset_2(t) = \sqrt{\frac{2}{T_b}} * \cos(2\pi f_2 t)$$

- **2.2. An expression for the baseband equivalent signals for this set, indicating the carrier frequency used:**

$$S(t) = \sqrt{\frac{2E_b}{T_b}} * \cos(2\pi f_1 t) \quad or \quad \sqrt{\frac{2E_b}{T_b}} * \cos(2\pi f_2 t)$$

The carrier frequency used is $\boldsymbol{f_c = f_1}$

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

$$S_2(t) = \sqrt{\frac{2E_b}{T_b}} * \cos(2\pi (f_c + \Delta f) t)$$

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

The two baseband equivalent signals used for this set:

$$S_{1BB}(t) = \sqrt{\frac{2E_b}{T_b}} \ , \ If\ the\ bit\ transmitted\ is\ 0$$

$$S_{2BB}(t) = \sqrt{\frac{2E_b}{T_b}} \ (\cos(2\pi \Delta f t) + j * \sin(2\pi \Delta f t)) \ ,$$

$$If\ the\ bit\ transmitted\ is\ 1$$

- **2.3. Theoretical vs practical BER graph of the BFSK:**

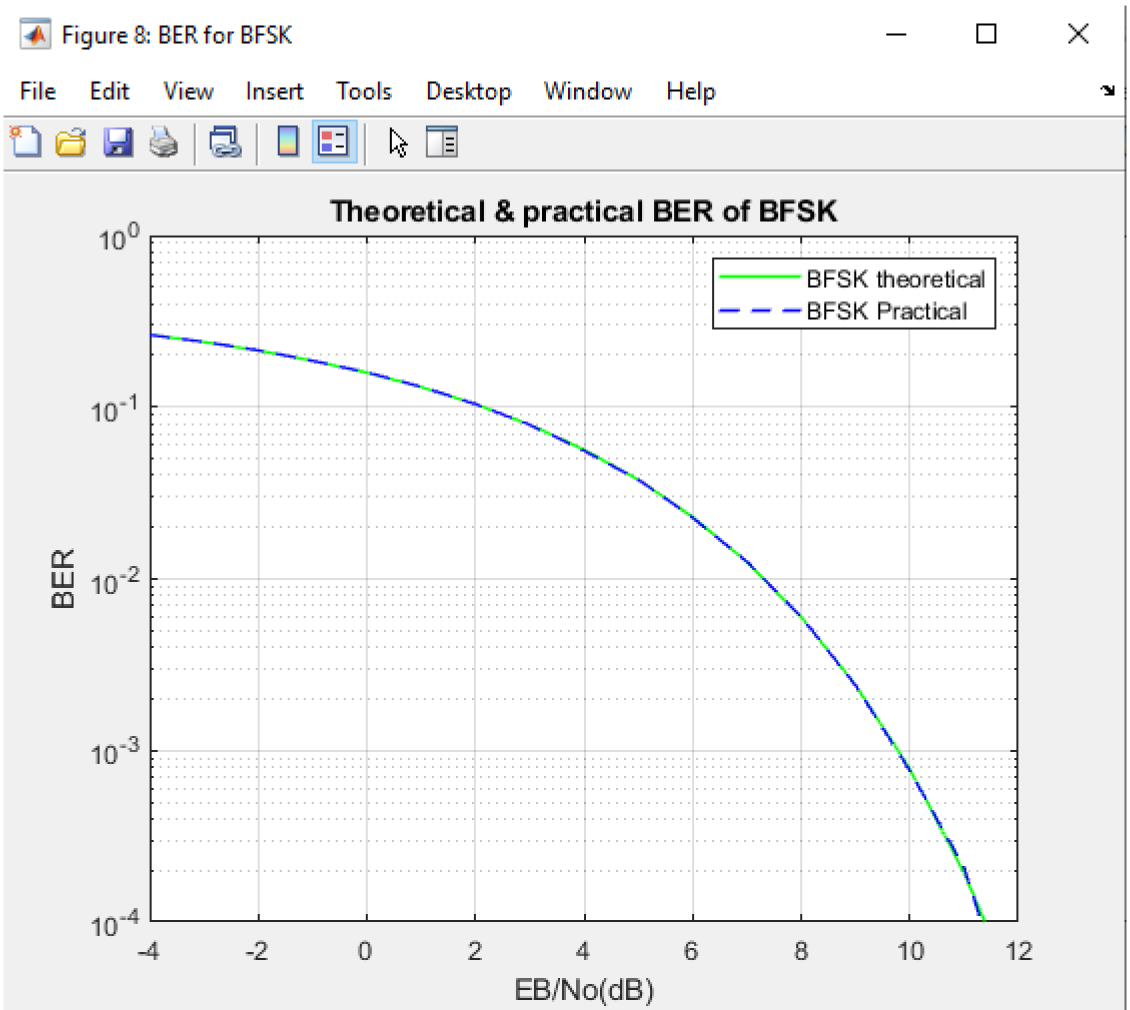

Figure 8: Theoretical and practical BER vs EB/No for BFSK

Comment: As we can see they are almost the same same and they become more similar when the number of binary data used increases, also the BER of BFSK is good.

- **2.4. PSD graph of BFSK using the baseband equivalent signal:**

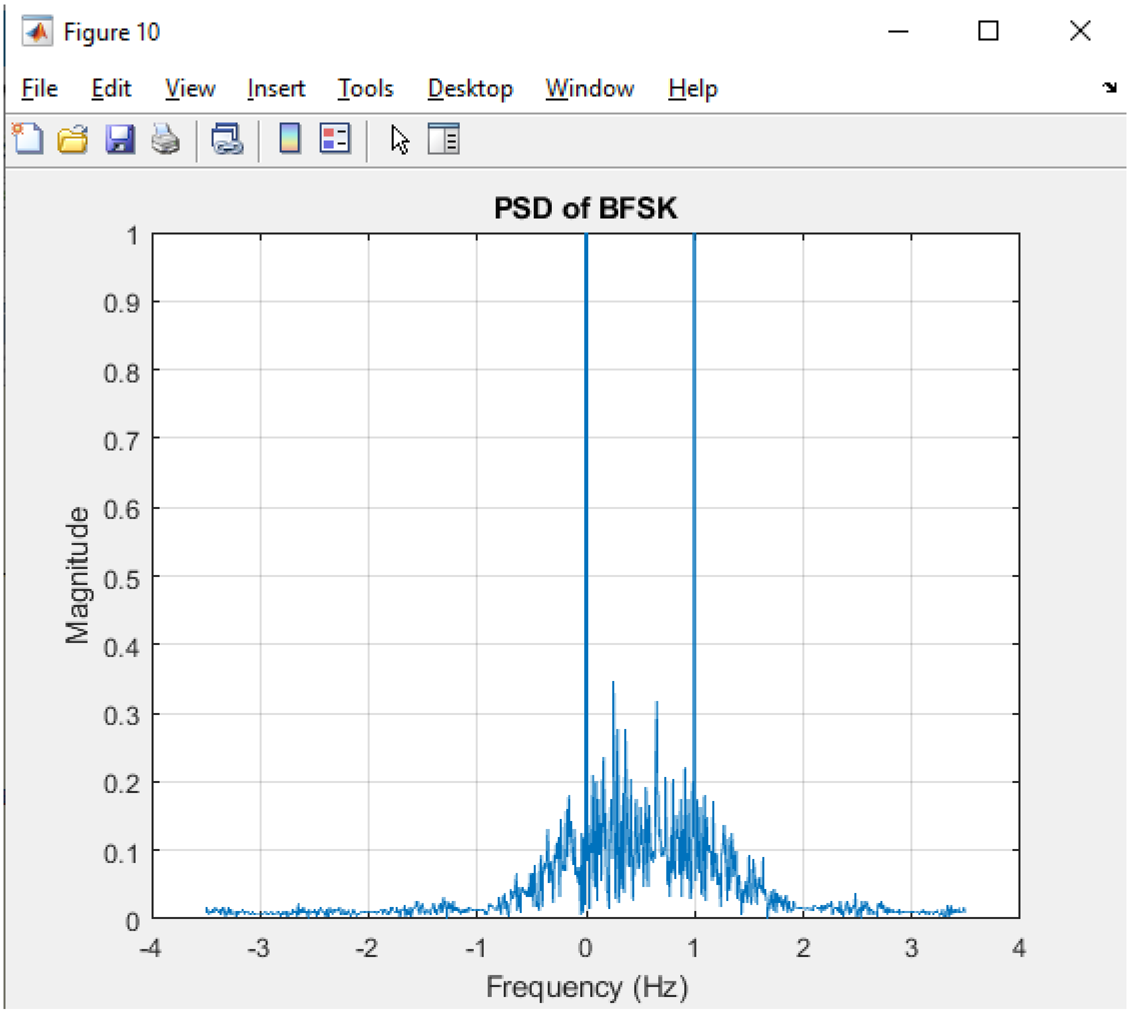

Figure 9: PSD of BFSK

3. <u>The MATLAB Code:</u>

```
clc;
close all;
clear all;
Number_of_bits = 48*10000;
SNR_dB= -4:1:14;
% Generate a random bits
    Original_bits = randi([0, 1], 1, Number_of_bits);

```

```
%Calculating No considering Eb(bit energy) = 1 for all modultaion schemes
No = zeros(size(SNR_dB));
for x = 1:length(SNR_dB)
     No(x) = 10^(-SNR_dB(x)/10);
end

%% BPSK

% mapper
BPSK_mapped = (2*Original_bits)-1;

% Chanel

BPSK_AWGN = (randn(1,Number_of_bits))./sqrt(2);
BPSK_BER = zeros(size(SNR_dB));

for x = 1:length(SNR_dB)

     recieved_BPSK = BPSK_mapped + sqrt(No(x))*BPSK_AWGN;

   % demapper

     BPSK_demapped = zeros(1,Number_of_bits);

for k = 1:Number_of_bits
    if (recieved_BPSK(k) < 0)
        BPSK_demapped(k) = 0;
    else
        BPSK_demapped(k) = 1;
    end

    % BPSK practical BER

    if (BPSK_demapped(k)~=Original_bits(k))
            BPSK_BER(x)=BPSK_BER(x)+1;
    end
end
end
%Therotical BER of BPSK
  BPSK_BER = BPSK_BER./Number_of_bits;
  BPSK_theoretical_BER = 0.5*erfc(sqrt(10.^(SNR_dB/10)));

%plotting BER of BPSK therotical vs practical
figure('Name' , 'BER for BPSK');
semilogy(SNR_dB , BPSK_theoretical_BER,'g','linewidth',1);
hold on;
semilogy(SNR_dB , BPSK_BER,'b --','linewidth',1);
hold off;
```

```
title('Theoretical & practical BER of BPSK');
xlabel('EB/No(dB)');  ylabel('BER');  grid on;
legend('BPSK theoretical','BPSK Practical','Location','NorthEast')
ylim([10^-4, 10^0]);
grid on;

%% QPSK & 8PSK

% Convert the numbers to strings
str_bits = arrayfun(@num2str, Original_bits, 'UniformOutput', false);

% Concatenate adjacent strings
QPSK_binary_bits_str = strcat(str_bits(1:2:end), str_bits(2:2:end));
M8PSK_binary_bits_str = strcat(str_bits(1:3:end), str_bits(2:3:end), str_bits(3:3:end));

% Convert the binary number to its decimal equivalent
QPSK_bits_in_decimal = bin2dec(QPSK_binary_bits_str);
QPSK_bits_in_binary = dec2bin(QPSK_bits_in_decimal);
M8PSK_bits_in_decimal = bin2dec(M8PSK_binary_bits_str);
M8PSK_bits_in_binary = dec2bin(M8PSK_bits_in_decimal);

% Shift the decimal number to the right
QPSK_shifted_bits_in_decimal = bitshift(QPSK_bits_in_decimal, -1); % Shift one position to the right
M8PSK_shifted_bits_in_decimal = bitshift(M8PSK_bits_in_decimal, -1); % Shift one position to the right

% Perform XOR operation
QPSK_decimal_gray_encoded = bitxor(QPSK_bits_in_decimal, QPSK_shifted_bits_in_decimal);
M8PSK_decimal_gray_encoded = bitxor(M8PSK_bits_in_decimal, M8PSK_shifted_bits_in_decimal);

% Converting to binary
QPSK_binary_gray_encoded = dec2bin(QPSK_decimal_gray_encoded);
M8PSK_binary_gray_encoded = dec2bin(M8PSK_decimal_gray_encoded);

% mapping gray QPSK

QPSK_gray_mapped = zeros(1,Number_of_bits/2);
QPSK_gray_energy_summation = 0;
for k = 1:Number_of_bits/2
    if(QPSK_binary_gray_encoded(k,:) == '00')
        QPSK_gray_mapped(k) = -1-1i;
    elseif(QPSK_binary_gray_encoded(k,:) == '01')
        QPSK_gray_mapped(k) = -1+1i;
```

```
        elseif(QPSK_binary_gray_encoded(k,:) == '11')
            QPSK_gray_mapped(k) = 1+1i;
        elseif(QPSK_binary_gray_encoded(k,:) == '10')
            QPSK_gray_mapped(k) = 1-1i;
        end
        QPSK_gray_energy_summation = QPSK_gray_energy_summation + abs(QPSK_gray_mapped(k))^2;
    end
    Eavg_QPSK_gray = QPSK_gray_energy_summation/(Number_of_bits/2);

    % mapping not gray QPSK

    QPSK_not_gray_mapped = zeros(1,Number_of_bits/2);
    QPSK_not_gray_energy_summation = 0;
    for k = 1:Number_of_bits/2
        if(QPSK_bits_in_binary(k,:) == '00')
            QPSK_not_gray_mapped(k) = -1-1i;
        elseif(QPSK_bits_in_binary(k,:) == '01')
            QPSK_not_gray_mapped(k) = -1+1i;
        elseif(QPSK_bits_in_binary(k,:) == '11')
            QPSK_not_gray_mapped(k) = 1-1i;
        elseif(QPSK_bits_in_binary(k,:) == '10')
            QPSK_not_gray_mapped(k) = 1+1i;
        end
        QPSK_not_gray_energy_summation = QPSK_not_gray_energy_summation + abs(QPSK_not_gray_mapped(k))^2;
    end
    Eavg_QPSK_not_gray = QPSK_not_gray_energy_summation/(Number_of_bits/2);

    % mapping 8PSK

    M8PSK_mapped = zeros(1,Number_of_bits/3);
    M8PSK_energy_summation = 0;
    for k = 1:(Number_of_bits/3)
        if(M8PSK_binary_gray_encoded(k,:) == '000')
            M8PSK_mapped(k) = 1;
        elseif(M8PSK_binary_gray_encoded(k,:) == '001')
            M8PSK_mapped(k) = 0.7071+0.7071i;
        elseif(M8PSK_binary_gray_encoded(k,:) == '011')
            M8PSK_mapped(k) = 1i;
        elseif(M8PSK_binary_gray_encoded(k,:) == '111')
            M8PSK_mapped(k) = -0.7071-0.7071i;
        elseif(M8PSK_binary_gray_encoded(k,:) == '101')
            M8PSK_mapped(k) = -1i;
        elseif(M8PSK_binary_gray_encoded(k,:) == '100')
            M8PSK_mapped(k) = 0.7071-0.7071i;
        elseif(M8PSK_binary_gray_encoded(k,:) == '110')
            M8PSK_mapped(k) = -1;
        elseif(M8PSK_binary_gray_encoded(k,:) == '010')
```

```
            M8PSK_mapped(k) = -0.7071+0.7071i;
        end
        M8PSK_energy_summation = M8PSK_energy_summation + abs(M8PSK_mapped(k))^2;
    end
    Eavg_M8PSK = M8PSK_energy_summation/(Number_of_bits/3);

    % Initializeing BER vectors
    QPSK_gray_BER = zeros(size(SNR_dB));
    QPSK_not_gray_BER = zeros(size(SNR_dB));
    QAM16_BER = zeros(size(SNR_dB));
    M8PSK_BER = zeros(size(SNR_dB));

    for i = 1:length(SNR_dB)

        % QPSK gray channel
        VARI_QPSK_gray = sqrt((No(i)*Eavg_QPSK_gray)/(2*2));
        real_noise_QPSK_gray = VARI_QPSK_gray.* randn(1 , Number_of_bits/2);
        img_noise_QPSK_gray = VARI_QPSK_gray .* randn(1 , Number_of_bits/2);
        recieved_QPSK_Gray = QPSK_gray_mapped + (real_noise_QPSK_gray+(1i*img_noise_QPSK_gray));

        % QPSK gray demapper
        QPSK_gray_demapped = zeros(1,Number_of_bits/2);

        for k=1:Number_of_bits/2

            im = imag(recieved_QPSK_Gray(k));
            re = real(recieved_QPSK_Gray(k));
            if((re<=0) && (im<=0))
                    QPSK_gray_demapped(k) = 0;
            elseif((re<=0) && (im>=0))
                    QPSK_gray_demapped(k) = 1;
            elseif((re>=0) && (im<=0))
                    QPSK_gray_demapped(k) = 2;
            elseif((re>=0) && (im>=0))
                    QPSK_gray_demapped(k) = 3;
            end
        end

       QPSK_gray_demapped = dec2bin(QPSK_gray_demapped);

       % gray QPSK BER

        for k=1:Number_of_bits/2
            if ~((QPSK_binary_gray_encoded(k,1)==QPSK_gray_demapped(k,1)))
                QPSK_gray_BER(i) = QPSK_gray_BER(i) + 1;
```

```
        end
        if ~((QPSK_binary_gray_encoded(k,2)==QPSK_gray_demapped(k,2)))
            QPSK_gray_BER(i) = QPSK_gray_BER(i) + 1;
        end
    end
    % QPSK not gray channel
    VARI_QPSK_not_gray = sqrt((No(i)*Eavg_QPSK_not_gray)/(2*2));
    real_noise_QPSK_not_gray = VARI_QPSK_not_gray.* randn(1 , Number_of_bits/2);
    img_noise_QPSK_not_gray = VARI_QPSK_not_gray .* randn(1 , Number_of_bits/2);
    recieved_QPSK_not_gray = QPSK_not_gray_mapped + (real_noise_QPSK_not_gray +(1i*img_noise_QPSK_not_gray));
    % QPSK_not_gray demapper
    QPSK_not_gray_demapped = zeros(1,Number_of_bits/2);
    for k=1:Number_of_bits/2
        im = imag(recieved_QPSK_not_gray(k));
        re = real(recieved_QPSK_not_gray(k));
        if (re<0)
            if(im<0)
                QPSK_not_gray_demapped(k) = 0;
            else
                QPSK_not_gray_demapped(k) = 1;
            end
        else
            if(im<0)
                QPSK_not_gray_demapped(k) = 3;
            else
                QPSK_not_gray_demapped(k) = 2;
            end
        end
    end
    QPSK_not_gray_demapped = dec2bin(QPSK_not_gray_demapped);
    % not gray QPSK BER
    for k=1:Number_of_bits/2
        if ~((QPSK_bits_in_binary(k,1)==QPSK_not_gray_demapped(k,1)))
            QPSK_not_gray_BER(i) = QPSK_not_gray_BER(i) + 1;
        end
        if ~((QPSK_bits_in_binary(k,2)==QPSK_not_gray_demapped(k,2)))
            QPSK_not_gray_BER(i) = QPSK_not_gray_BER(i) + 1;
        end
```

```
        end

        % 8PSK channel

        VARI_M8PSK = sqrt((No(i)*Eavg_M8PSK)/(2*3));
        real_noise_M8PSK = VARI_M8PSK.* randn(1 , Number_of_bits/3);
        img_noise_M8PSK = VARI_M8PSK .* randn(1 , Number_of_bits/3);
        recieved_M8PSK_Gray = M8PSK_mapped + (real_noise_M8PSK + (1i*img_noise_M8PSK));

          % M8PSK demapper

        M8PSK_demapped = zeros(1,Number_of_bits/3);

        for k=1:Number_of_bits/3

            ang= angle(recieved_M8PSK_Gray(k));
            if (ang>2.7489)
                M8PSK_demapped(k) = 6;
            elseif (ang>1.9635)
                M8PSK_demapped(k) = 2;
            elseif (ang>1.1781)
                M8PSK_demapped(k) = 3;
            elseif (ang>0.3927)
                M8PSK_demapped(k) = 1;
            elseif (ang>-0.3927)
                M8PSK_demapped(k) = 0;
            elseif (ang>-1.1781)
                M8PSK_demapped(k) = 4;
            elseif (ang>-1.9635)
                M8PSK_demapped(k) = 5;
            elseif (ang>-2.7489)
                M8PSK_demapped(k) = 7;
            else
                M8PSK_demapped(k) = 6;
            end
        end
       M8PSK_demapped = dec2bin(M8PSK_demapped);

          % 8PSK practical BER

        for k=1:Number_of_bits/3
            if ~((M8PSK_binary_gray_encoded(k,1)==M8PSK_demapped(k,1)))
                M8PSK_BER(i) = M8PSK_BER(i) + 1;
            end
            if ~((M8PSK_binary_gray_encoded(k,2)==M8PSK_demapped(k,2)))
                M8PSK_BER(i) = M8PSK_BER(i) + 1;
```

```
            end
            if ~((M8PSK_binary_gray_encoded(k,3)==M8PSK_demapped(k,3)))
                M8PSK_BER(i) = M8PSK_BER(i) + 1;
            end
        end

    end
    %Therotical BER of QPSK Gray-Encoded
     QPSK_gray_BER = QPSK_gray_BER./Number_of_bits;
     QPSK_gray_theoretical_BER = 0.5*erfc(sqrt(10.^(SNR_dB/10)));
    %plotting BER of QPSK Gray-Encoded therotical vs practical
    figure('Name' , 'BER for QPSK Gray-Encoded ');
    semilogy(SNR_dB , QPSK_gray_theoretical_BER,'g','linewidth',1);
    hold on;
    semilogy(SNR_dB , QPSK_gray_BER,'b --','linewidth',1);
    hold off;
    title('Theoretical & practical BER of QPSK Gray-Encoded ');
    xlabel('EB/No(dB)');  ylabel('BER');  grid on;
    legend('QPSK Gray-Encoded  theoretical','QPSK Gray-Encoded Practical','Location','NorthEast')
    ylim([10^-4, 10^0]);
    grid on;



    %Therotical BER of QPSK Not-Gray-Encoded
     QPSK_not_gray_BER = QPSK_not_gray_BER./Number_of_bits;
     QPSK_not_gray_theoretical_BER = 0.5*erfc(sqrt(10.^(SNR_dB/10)));
    %plotting BER of QPSK Not-Gray-Encoded therotical vs practical
    figure('Name' , 'BER for QPSK Not-Gray-Encoded ');
    semilogy(SNR_dB , QPSK_not_gray_theoretical_BER,'g','linewidth',1);
    hold on;
    semilogy(SNR_dB , QPSK_not_gray_BER,'b --','linewidth',1);
    hold off;
    title('Theoretical & practical BER of QPSK Not-Gray-Encoded ');
    xlabel('EB/No(dB)');  ylabel('BER');  grid on;
    legend('QPSK Not-Gray-Encoded  theoretical','QPSK Not-Gray-Encoded  Practical','Location','NorthEast')
    ylim([10^-4, 10^0]);
    grid on;




    %Therotical BER of 8PSK
     M8PSK_BER = M8PSK_BER./Number_of_bits;
     M8PSK_theoretical_BER = (1/3)*erfc(sqrt(3*10.^(SNR_dB/10))*sin(22/(8*7)));
```

```matlab
%plotting BER of 8PSK therotical vs practical
figure('Name' , 'BER for 8PSK');
semilogy(SNR_dB , M8PSK_theoretical_BER,'g','linewidth',1);
hold on;
semilogy(SNR_dB , M8PSK_BER,'b --','linewidth',1);
hold off;
title('Theoretical & practical BER of 8PSK');
xlabel('EB/No(dB)');  ylabel('BER');  grid on;
legend('8PSK theoretical','8PSK Practical','Location','NorthEast')
ylim([10^-4, 10^0]);
grid on;

%% 16QAM

% Mapper

QAM16_binary_bits_str = strcat(str_bits(1:4:end), str_bits(2:4:end), str_bits(3:4:end), str_bits(4:4:end));
QAM16_bits_in_decimal = bin2dec(QAM16_binary_bits_str);
QAM16_shifted_bits_in_decimal = bitshift(QAM16_bits_in_decimal, -1); % Shift one position to the right
QAM16_decimal_gray_encoded = bitxor(QAM16_bits_in_decimal, QAM16_shifted_bits_in_decimal);
QAM16_binary_gray_encoded = dec2bin(QAM16_decimal_gray_encoded);

QAM16_mapped = zeros(1,Number_of_bits/4);
QAM16_energy_summation = 0;

for k = 1:Number_of_bits/4

    if(QAM16_binary_gray_encoded(k,:) == '0000')
        QAM16_mapped(k) = -3-3i;
    elseif(QAM16_binary_gray_encoded(k,:) == '0001')
        QAM16_mapped(k) = -3-1i;
    elseif(QAM16_binary_gray_encoded(k,:) == '0010')
        QAM16_mapped(k) = -3+3i;
    elseif(QAM16_binary_gray_encoded(k,:) == '0011')
        QAM16_mapped(k) = -3+1i;
    elseif(QAM16_binary_gray_encoded(k,:) == '0100')
        QAM16_mapped(k) = -1-3i;
    elseif(QAM16_binary_gray_encoded(k,:) == '0101')
        QAM16_mapped(k) = -1-1i;
    elseif(QAM16_binary_gray_encoded(k,:) == '0110')
        QAM16_mapped(k) = -1+3i;
    elseif(QAM16_binary_gray_encoded(k,:) == '0111')
        QAM16_mapped(k) = -1+1i;
    elseif(QAM16_binary_gray_encoded(k,:) == '1000')
        QAM16_mapped(k) = 3-3i;
    elseif(QAM16_binary_gray_encoded(k,:) == '1001')
```

```
            QAM16_mapped(k) = 3-1i;
        elseif(QAM16_binary_gray_encoded(k,:) == '1010')
            QAM16_mapped(k) = 3+3i;
        elseif(QAM16_binary_gray_encoded(k,:) == '1011')
            QAM16_mapped(k) = 3+1i;
        elseif(QAM16_binary_gray_encoded(k,:) == '1100')
            QAM16_mapped(k) = 1-3i;
        elseif(QAM16_binary_gray_encoded(k,:) == '1101')
            QAM16_mapped(k) = 1-1i;
        elseif(QAM16_binary_gray_encoded(k,:) == '1110')
            QAM16_mapped(k) = 1+3i;
        elseif(QAM16_binary_gray_encoded(k,:) == '1111')
            QAM16_mapped(k) = 1+1i;
        end
        QAM16_energy_summation = QAM16_energy_summation + abs(QAM16_mapped(k))^2;
    end
    QAM16_energy_avg = QAM16_energy_summation/(Number_of_bits/4);

    Eavg_16QAM = QAM16_energy_avg;

    for i = 1:length(SNR_dB)

        % QAM16 Channel

        VARI_16QAM = sqrt((No(i)/2)*(Eavg_16QAM/4));
        real_noise_16QAM = VARI_16QAM .* randn(1 , Number_of_bits/4);
        img_noise_16QAM = VARI_16QAM .* randn(1 , Number_of_bits/4);
        received_16QAM = QAM16_mapped + (real_noise_16QAM + (1i*img_noise_16QAM));

        %%% QAM16 demapper

        QAM16_demapped = zeros(1,Number_of_bits/4);

        for k = 1:Number_of_bits/4

            im = imag(received_16QAM(k));
            re = real(received_16QAM(k));
            if(re<=-2)
                if(im<=-2)
                    QAM16_demapped(k) = 0;
                elseif (im<=0)
                    QAM16_demapped(k) = 1;
                elseif (im<=2)
                    QAM16_demapped(k) = 3;
                else
                    QAM16_demapped(k) = 2;
                end
```

```
            elseif(re<=0)
                if(im<=-2)
                    QAM16_demapped(k) = 4;
                elseif (im<=0)
                    QAM16_demapped(k) = 5;
                elseif (im<=2)
                    QAM16_demapped(k) = 7;
                else
                    QAM16_demapped(k) = 6;
                end
            elseif (re<=2)
                if(im<=-2)
                    QAM16_demapped(k) = 12;
                elseif (im<=0)
                    QAM16_demapped(k) = 13;
                elseif (im<=2)
                    QAM16_demapped(k) = 15;
                else
                    QAM16_demapped(k) = 14;
                end
            else
                if(im<=-2)
                    QAM16_demapped(k) = 8;
                elseif (im<=0)
                    QAM16_demapped(k) = 9;
                elseif (im<=2)
                    QAM16_demapped(k) = 11;
                else
                    QAM16_demapped(k) = 10;
                end
            end
        end
       QAM16_demapped = dec2bin(QAM16_demapped);

       % QAM16 practical BER

        for k=1:Number_of_bits/4

            if ~((QAM16_binary_gray_encoded(k,1)==QAM16_demapped(k,1)))
                QAM16_BER(i) = QAM16_BER(i) + 1;
            end
            if ~((QAM16_binary_gray_encoded(k,2)==QAM16_demapped(k,2)))
                QAM16_BER(i) = QAM16_BER(i) + 1;
            end
            if ~((QAM16_binary_gray_encoded(k,3)==QAM16_demapped(k,3)))
                QAM16_BER(i) = QAM16_BER(i) + 1;
            end
```

```
            if ~((QAM16_binary_gray_encoded(k,4)==QAM16_demapped(k,4)))
                QAM16_BER(i) = QAM16_BER(i) + 1;
            end
        end
    end

      %Therotical BER of 16QAM
      QAM16_BER = QAM16_BER./Number_of_bits;
      QAM16_theoretical_BER = (3/8)*erfc(sqrt(10.^(SNR_dB/10))/sqrt(2.5));
      %plotting BER of 16QAM therotical vs practical
      figure('Name' , 'BER for 16QAM');
      semilogy(SNR_dB , QAM16_theoretical_BER,'g','linewidth',1);
      hold on;
      semilogy(SNR_dB , QAM16_BER,'b --','linewidth',1);
      hold off;
      title('Theoretical & practical BER of 16QAM');
      xlabel('EB/No(dB)');  ylabel('BER');  grid on;
      legend('16QAM theoretical','16QAM Practical','Location','NorthEast')
      ylim([10^-4, 10^0]);
      grid on;



    %%
    % ploting
    figure('Name' , 'BER');
    semilogy(SNR_dB , BPSK_theoretical_BER,'g','linewidth',1);
    hold on;
    semilogy(SNR_dB , BPSK_BER,'r --','linewidth',1);
    hold on;
    semilogy(SNR_dB , QPSK_gray_BER,'b','linewidth',1);
    hold on;
    semilogy(SNR_dB , M8PSK_theoretical_BER,'k --','linewidth',1);
    hold on;
    semilogy(SNR_dB , M8PSK_BER,'m','linewidth',1);
    hold on;
    semilogy(SNR_dB , QAM16_BER,'c','linewidth',1);
    hold on;
    semilogy(SNR_dB , QAM16_theoretical_BER,'y --','linewidth',1);
    hold off;
    title('BER for the four modulation schemes');
    xlabel('EB/No(dB)');  ylabel('BER');  grid on;
    legend('BPSK & QPSK theoretical','BPSK Practical','QPSK Practical','8PSK theoretical','8PSK Practical','16QAM Practical','16QAM theoritical','Location','NorthEast')
    ylim([10^-4, 10^0]);
    grid on;
```

```
    figure('Name' , 'BER for QPSK');
    semilogy(SNR_dB , QPSK_gray_theoretical_BER,'g-*','linewidth',1);
    hold on;
    semilogy(SNR_dB , QPSK_not_gray_theoretical_BER,'y','linewidth',1);
    hold on;
    semilogy(SNR_dB , QPSK_gray_BER,'b --','linewidth',1);
    hold on;
    semilogy(SNR_dB , QPSK_not_gray_BER,'r','linewidth',1);
    hold off;
    title('BER of gray mapped QPSK & not gray mapped QPSK');
    xlabel('EB/No(dB)');  ylabel('BER');  grid on;
    legend('QPSK Gray-Encoded theoretical','QPSK Not-Gray-Encoded theoretical','Gray QPSK Practical','Not gray QPSK Practical','Location','NorthEast')
    ylim([10^-4, 10^0]);
    grid on;

    %% BFSK

    % mapping

    BFSK_mapped = zeros(1,Number_of_bits);

    for k=1:Number_of_bits

        if(Original_bits(k) == 0)
            BFSK_mapped(k) = 1;
        else
            BFSK_mapped(k) = 1i;
        end
    end

    BFSK_BER = zeros(size(SNR_dB));

    for i = 1:length(SNR_dB)

        % BFSK channel

        VARI_BFSK = sqrt(No(i)/2);
        real_noise_BFSK = VARI_BFSK .* randn(1 , Number_of_bits);
        img_noise_BFSK = VARI_BFSK .* randn(1 , Number_of_bits);
        received_BFSK = BFSK_mapped + (real_noise_BFSK + (1i*img_noise_BFSK));


        % BFSK demapper
        BFSK_demapped = zeros(1,Number_of_bits);
```

```

        for k=1:Number_of_bits

            im = imag(received_BFSK(k));
            re = real(received_BFSK(k));
            if(re>im)
                BFSK_demapped(1,k) = 0;
            else
                BFSK_demapped(1,k) = 1;
            end
        end

       % BFSK practical BER

        for k=1:Number_of_bits

            if ~(BFSK_demapped(1,k)==Original_bits(1,k))
                BFSK_BER(i) = BFSK_BER(i) + 1;
            end
        end
    end
    %Therotical BER of BFSK
    BFSK_BER = BFSK_BER./Number_of_bits;
    BFSK_theoretical_BER = 0.5*erfc(sqrt(10.^((SNR_dB/10))/2));

    %plotting BER of BFSK therotical vs practical

    figure('Name' , 'BER for BFSK');
    semilogy(SNR_dB , BFSK_theoretical_BER,'g','linewidth',1);
    hold on;
    semilogy(SNR_dB , BFSK_BER,'b --','linewidth',1);
    hold off;
    title('Theoretical & practical BER of BFSK');
    xlabel('EB/No(dB)');  ylabel('BER');  grid on;
    legend('BFSK theoretical','BFSK Practical','Location','NorthEast')
    ylim([10^-4, 10^0]);
    grid on;


    %%
    Number_of_waveforms = 500;
    Number_of_bits = 100;

    % Generating an ensemble consists of 500 realization , each 101 bits
    Data = randi([0 1], Number_of_waveforms, Number_of_bits+1) ;

    %Activating the DAC every 10ms
```

```
    Transmitter_ensemble = zeros( Number_of_waveforms , (Number_of_bits+1)*7 );

    bit_duration = 0.07;
    signal_mag = sqrt( 2*1/bit_duration ) ;                %
    time = 0.01 : 0.01 : 0.07;

    S_Baseband = signal_mag * cos(2*pi*(1/bit_duration) * time) + 1i * signal_mag * sin( 2*pi*(1/bit_duration) * time) ;

    %construct the ensemble
    lag = 1;
    for k = 1 : Number_of_bits+1
        for i = 1 : Number_of_bits+1
            if (Data(k , i) ~= 0)
                Transmitter_ensemble(k , lag) = S_Baseband(1); Transmitter_ensemble(k , lag+1) = S_Baseband(2); Transmitter_ensemble(k , lag+2) = S_Baseband(3); Transmitter_ensemble(k , lag+3) = S_Baseband(4);
                Transmitter_ensemble(k , lag+4) = S_Baseband(5); Transmitter_ensemble(k , lag+5) =S_Baseband(6) ; Transmitter_ensemble(k , lag+6) = S_Baseband(7);

            elseif (Data(k , i) == 0)
                Transmitter_ensemble(k , lag) = signal_mag; Transmitter_ensemble(k , lag+1) = signal_mag; Transmitter_ensemble(k , lag+2) = signal_mag; Transmitter_ensemble(k , lag+3) = signal_mag;
                Transmitter_ensemble(k , lag+4) = signal_mag; Transmitter_ensemble(k , lag+5) = signal_mag; Transmitter_ensemble(k , lag+6) = signal_mag;
            end
            lag = lag + 7;
        end
        lag = 1;
    end

    %delay
    delayed_for_BFSK_ensemble = randi ([0 6], Number_of_waveforms, 1) ; %delay array for polar_nrz
    BFSK = zeros (Number_of_waveforms, Number_of_bits*7) ; % initialization

    % adding the delay
    for i = 1:Number_of_waveforms
        BFSK(i, :) = Transmitter_ensemble(i, delayed_for_BFSK_ensemble(i)+1 : (Number_of_bits*7 + delayed_for_BFSK_ensemble(i)));
    end

    % calculating the autocorrelation of the ensemble
```

```
 BFSK_avg_auto = zeros( 1 ,Number_of_bits*7); % initialization of the autocorrelation matrix for polar_rz

 for lag = 0 : (Number_of_bits*7)-1
     autocorrelation = 0;

     for i = 1 : Number_of_waveforms
         autoCorrelation = conj(BFSK(i , 1))*BFSK(i, lag + 1);
         autocorrelation = autocorrelation + autoCorrelation;

     end
     BFSK_avg_auto(lag+1) = autocorrelation/500;
 end


PSD = abs(fftshift(fft(BFSK_avg_auto)));

fs = 100;
frequency_axis = -fs/2 : fs/700 : fs/2 - fs/700;

figure;
plot(frequency_axis*bit_duration, PSD/(fs*2));
title("PSD of BFSK");
xlabel("Frequency (Hz)");
ylabel("Magnitude");
ylim([0,1]);
grid on;
```