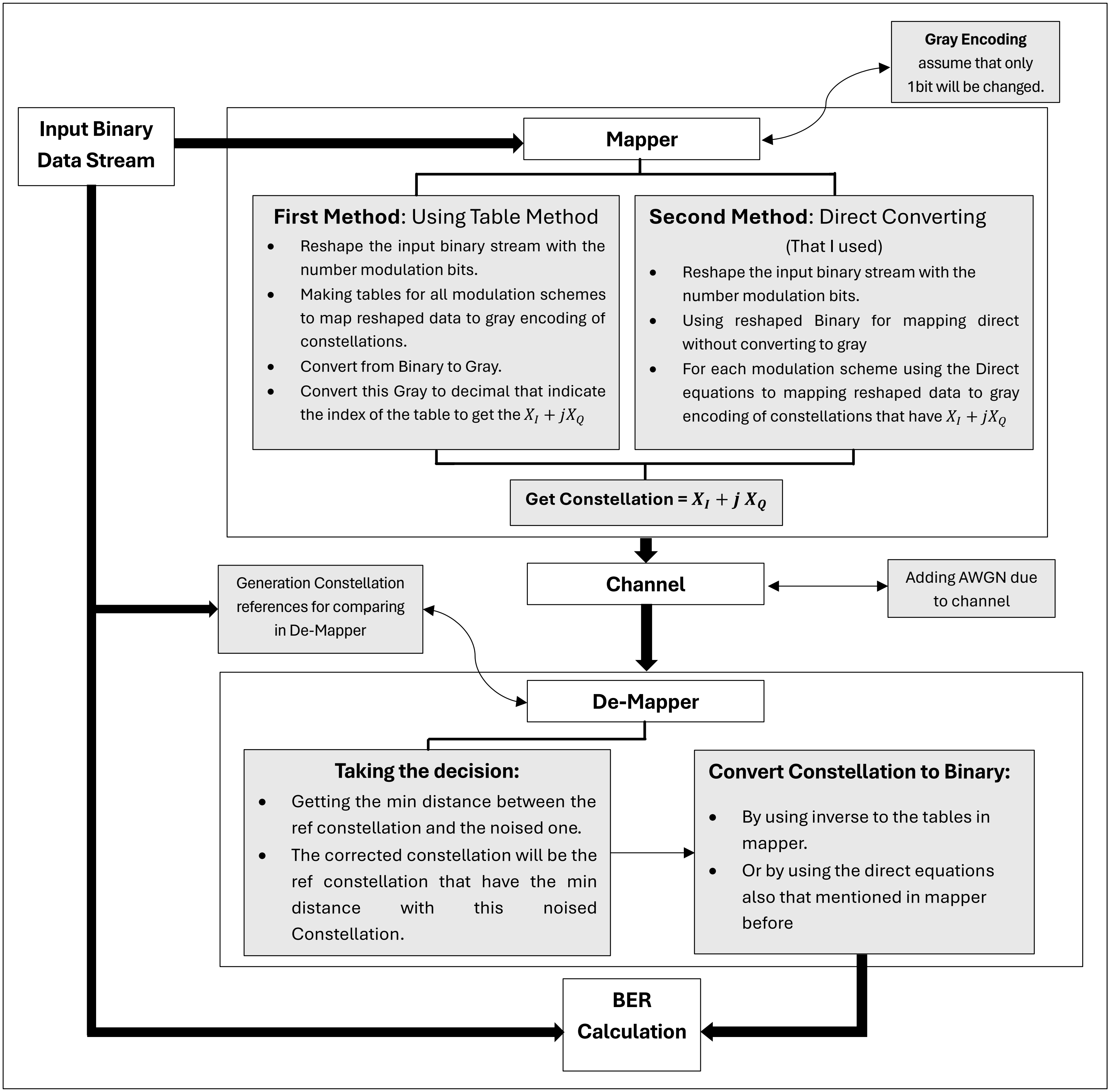
Project 3 communication

Name:	*Sec:*	*B. N*	*ID*
مصطفى إبراهيم محمد ابراهيم	**4**	**19**	**9211158**
يحيى خالد عبد الفتاح محمد	**4**	**40**	**9211362**

Project Flow Chart



Mappers

Firstly, before Mapper:

1- **Generate data bit stream.**
2- **Select the seeking modulation scheme to define num of Modulation bits.**
3- **Reshape the data bit stream by using the num of Modulation bits.**
4- **Passing the reshaped data, scheme and num of modulation bits to Mapper**

Then in Mapper select the ordered Modulation scheme and calling it's function.

Code

```
% Generating Data bits
Data_bits_Generated = randi([0 1],1,NO_of_bits);  % generate the random logic bit-stream
if Schemes == 1
    disp("The current Modulation Scheme is BPSK ") ;
    bits_modulation = 1 ;
 elseif Schemes == 2
    disp("The current Modulation Scheme is QPSK ") ;
    bits_modulation = 2 ;
 elseif Schemes == 3
    disp("The current Modulation Scheme is 8PSK ") ;
    bits_modulation = 3 ;
  elseif Schemes == 4
    disp("The current Modulation Scheme is 16-QAM ") ;
    bits_modulation = 4 ;
  elseif Schemes == 5
    bits_modulation = 1 ;
    disp("The current Modulation Scheme is BFSK ") ;
else % QPSK Binary
    disp("The current Modulation Scheme is QPSK with Binary Encoding ") ;
    bits_modulation = 2 ;
end
% Generate X_inphase and X_Quad vectros
X_inphase_Mapper = zeros (length (Data_bits_Generated)/bits_modulation ,1) ;
X_Quad_Mapper    = zeros (length (Data_bits_Generated)/bits_modulation ,1) ;

% Convert data bits into groups of bits
dataBitsGrouped = reshape(Data_bits_Generated, bits_modulation , [])';
% Calling Mapper Function
[X_inphase_Mapper , X_Quad_Mapper] = Mapper ( dataBitsGrouped  , Schemes );
% Keep Symbol as X_i + j X_Q
Symbols = X_inphase_Mapper' + 1i.* X_Quad_Mapper';
```

Going to Mapper function:

```
% Mapper Block function
function [X_inphase_out , X_Quad_out] = Mapper ( Y_signal  , Schemes_used )
  X_inphase_out = zeros (length (Y_signal),1) ;
  X_Quad_out    = zeros (length (Y_signal),1) ;

  if Schemes_used == 1
     [X_inphase_out , X_Quad_out] = BPSK_Mapper ( Y_signal ) ;
  elseif Schemes_used == 2
     [X_inphase_out , X_Quad_out] = QPSK_Mapper ( Y_signal ) ;
  elseif Schemes_used == 3
     [X_inphase_out , X_Quad_out] = Eight_PSK_Mapper ( Y_signal ) ;
  elseif  Schemes_used==4
     [X_inphase_out , X_Quad_out] = QAM_Mapper ( Y_signal ) ;
  elseif Schemes_used==5
     [X_inphase_out , X_Quad_out] = BFSK_Mapper ( Y_signal ) ;
  else %Schemes_used==6
     [X_inphase_out , X_Quad_out] = QPSK_Binary_Mapper ( Y_signal ) ;
  end
end
```

- ## BPSK Mapping:

It's only 1 bit $'b_1'$, if $b_1 = 0$, $S = -1$ and if $b_1 = 1$, $S = +1$.

So, we can say: $\boldsymbol{X_Q = 0}$ **for any** $\boldsymbol{b_1}$ **and** $\boldsymbol{X_i = 2 \times b_1 - 1}$

Code	Plotting

```
% BPSK Mapping
function [X_inphase , X_Quad] = BPSK_Mapper ( Y_signal_BPSK )
    X_inphase = zeros (length (Y_signal_BPSK),1) ;
    X_Quad    = zeros (length (Y_signal_BPSK),1) ;
    X_inphase = 2*Y_signal_BPSK-1 ;
end
```

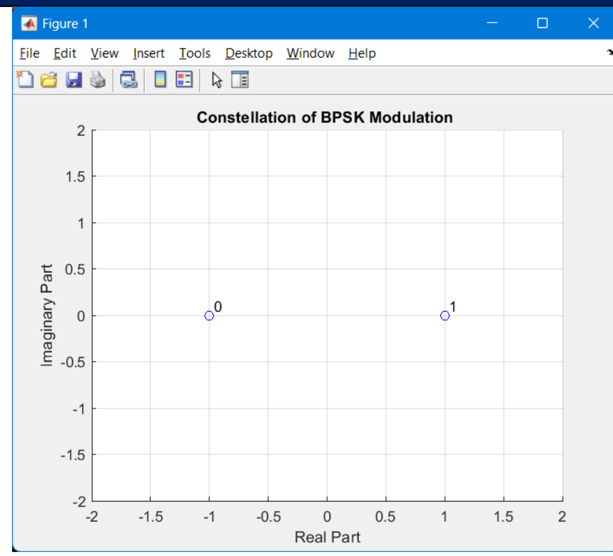


Figure 1: Representation of BPSK's Constellation

- ## QPSK Mapping (Gray encoding):

QPSK used 2 bits $[b_1\ b_2]$ so I need to reshape the input data stream by 2 , I used b_1 *to get* X_i *and* b_2 *to get* X_Q . then from equations:

$$\boldsymbol{X_i = 2 \times b_1 - 1} \text{ and } \boldsymbol{X_Q = 2 \times b_2 - 1}$$

And $S = X_i + jX_Q$, Trying examples $[0\ 0] \rightarrow -1 - j$ & $[0\ 1] \rightarrow -1 + j$ & $[1\ 0] \rightarrow 1 - j$ & $[1\ 1] \rightarrow 1 + j$

Code	Plotting

```
% QPSK Gray representation Mapping
function [X_inphase , X_Quad] = QPSK_Mapper ( Y_signal_QPSK )
    X_inphase = zeros (length (Y_signal_QPSK),1) ;
    X_Quad    = zeros (length (Y_signal_QPSK),1) ;

    X_inphase = 2*Y_signal_QPSK(:,1)-1 ;
    X_Quad    = 2*Y_signal_QPSK(:,2)-1 ;
end
```

Note: From plotting the constellations of QPSK Gray encoding, only chance to flip one bit.

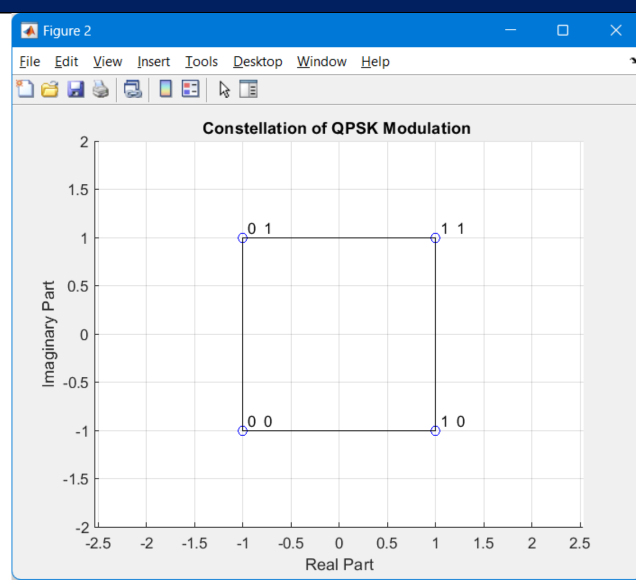


Figure 2: Representation of QPSK's Constellation (Gray encoding)

- ## QPSK Mapping (Binary Encoding):

It didn't have any simple sequence, but the simples one is going to nest if to achieve it.

Code

```
% QPSK Binary Mapping
function [X_inphase , X_Quad] = QPSK_Binary_Mapper( Y_signal_QPSK_binary )
     X_inphase = zeros (length (Y_signal_QPSK_binary),1) ;
     X_Quad    = zeros (length (Y_signal_QPSK_binary),1) ;
 for k = 1 : length (Y_signal_QPSK_binary)
    if Y_signal_QPSK_binary(k,:) == [0 0]
        X_inphase(k) = -1 ;
        X_Quad(k)    = -1 ;
    elseif Y_signal_QPSK_binary(k,:)  == [0 1]
        X_inphase(k) = -1 ;
        X_Quad(k)    =  1 ;

    elseif Y_signal_QPSK_binary(k,:)  == [1 0]
        X_inphase(k) = 1 ;
        X_Quad(k)    = 1 ;
    else
        X_inphase(k) =  1 ;
        X_Quad(k)    = -1 ;
    end
 end
end
```

Note: from plotting there is a chance to flip 2 bits so the BER will be higher.

Plotting

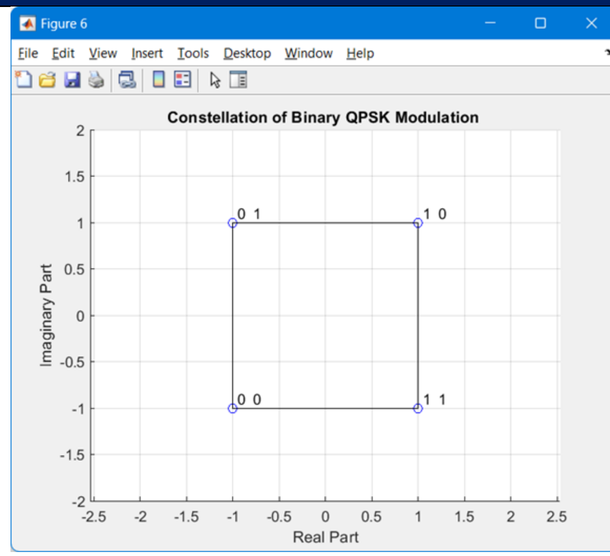


Figure 3: Representation of QPSK's Constellation (Binary encoding)

- ## 8PSK Mapping:

8PSK used 3 bits so the input of binary stream will be reshaped by 3 and 8PSK will treat them as converting the reshape bits (assume they are gray, and we will back them to binary) and convert them to Decimal and called i_{angle} that will be multiple with phase (Phase = pi /4) to get S. Then $\boldsymbol{S(i_{angle}) = e^{j \times i_{angle} \times (\frac{\pi}{4})}}$ as following in code:

Code

```
% 8PSK
function [X_inphase , X_Quad] = Eight_PSK_Mapper ( Y_signal_eight_PSK )
     X_inphase = zeros (length (Y_signal_eight_PSK),1) ;
     X_Quad    = zeros (length (Y_signal_eight_PSK),1) ;
     Z_complex = zeros (length (Y_signal_eight_PSK),1) ;
     % transfer form binary data to the binary of i_angle
     Binary_bits (:,1) = Y_signal_eight_PSK(:,1);
     for i = 2 : 3
         % XOR operation with shifted version of the sequence to get Binary
         Binary_bits(:,i) = xor ( Binary_bits(:,i-1) , Y_signal_eight_PSK(:,i) );
     end
     i_Angle = Binary_bits * (pow2(2:-1:0)');

     for j = 1 : length (Z_complex)
         phase = pi/4 ;
         Z_complex (j) =  1* exp(i_Angle(j) * 1i * phase);
     end
     X_inphase = real (Z_complex);
     X_Quad    = imag (Z_complex);

end
```

Plotting

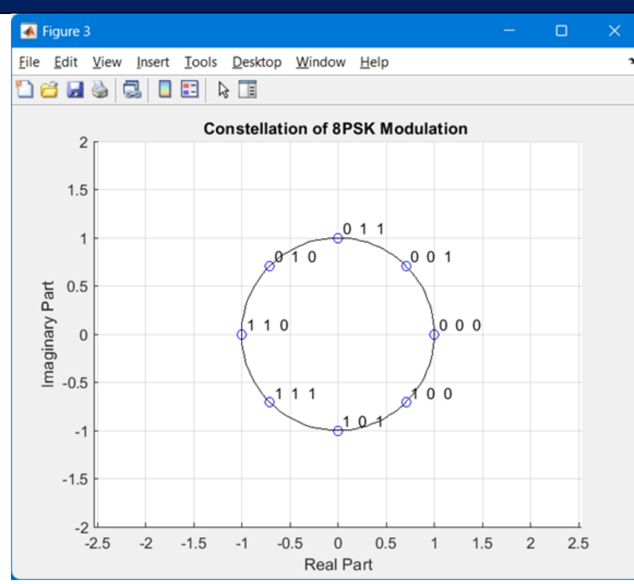


Figure 4: Representation of 8PSK's Constellation.

- ## 16-QAM Mapping:

16-QAM using 4 bits as following $'b_1\,, b_2\,, b_3\,, b_4'$ so it will reshape the input binary stream by 4, we will division the 4 bits into 2 groups of 2 bits. 1st group is $'b_1\,, b_2'$ they will be responsible for X_i and 2nd group is $'b_3, b_4'$ they also will be responsible for X_Q from the following equations. **To determine the sign of** $\boldsymbol{X_i = 2 \times b_1 - 1}$ *and also* $\boldsymbol{X_Q = 2 \times b_3 - 1}$.

And for **the Mag of** $\boldsymbol{X_i}$ ***and*** $\boldsymbol{X_Q}$, $\boldsymbol{|X_i| = 3 - 2 \times b_2}$ *and also* $\boldsymbol{|X_Q| = 3 - 2 \times b_4}$

I. $[0\ X\ 0\ X] \ldots S = (-)|X_i| - j|X_Q|$ and $[0\ 1\ 0\ 1] \ldots S = -1 - j$

II. $[0\ X\ 1\ X] \ldots S = (-)|X_i| + j|X_Q|$ and $[0\ 0\ 1\ 0] \ldots S = -3 + 3j$

III. $[1\ X\ 0\ X] \ldots S = (+)|X_i| - j|X_Q|$ and $[1\ 0\ 0\ 1] \ldots S = 3 - j$

IV. $[1\ X\ 1\ X] \ldots S = (+)|X_i| + j|X_Q|$ and $[1\ 1\ 0\ 0] \ldots S = 1 - 3j$

Code

```
% 16-QAM Mapping
function [X_inphase , X_Quad] = QAM_Mapper ( Y_signal_QAM )
 % [ Sign X_in , Value X_in ,Sign X_Quad , Value X_Quad ]
    X_inphase = zeros (length (Y_signal_QAM),1) ;
    X_Quad    = zeros (length (Y_signal_QAM),1) ;

    Mag_Inphase  = 3 - 2 * Y_signal_QAM(:,2) ;
    Sign_Inphase = 2 * Y_signal_QAM(:,1)-1 ;

    Mag_Quad  = 3 - 2 * Y_signal_QAM(:,4) ;
    Sign_Quad = 2*Y_signal_QAM(:,3)-1 ;

    X_inphase = Mag_Inphase .* Sign_Inphase  ;
    X_Quad =  Mag_Quad .* Sign_Quad ;
end
```

Plotting

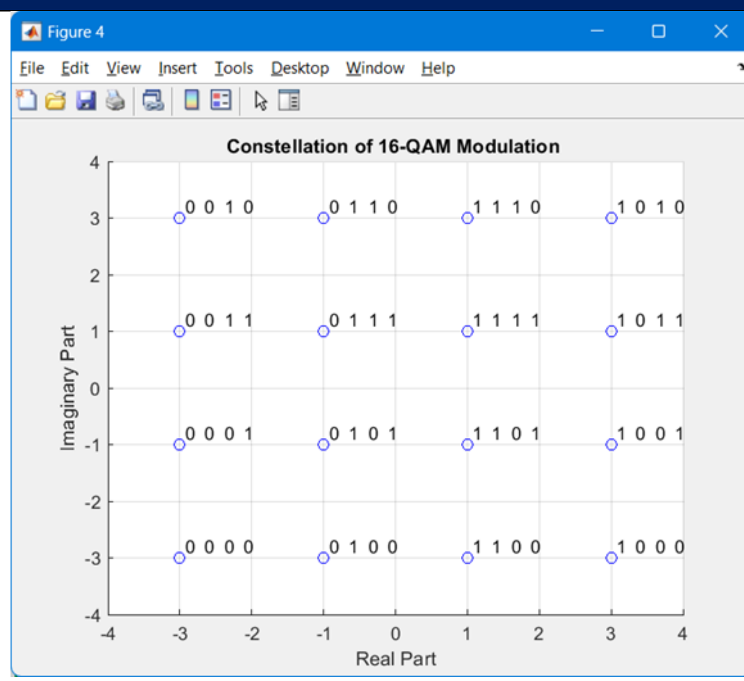


Figure 5: Representation of 16-QAM's Constellation

- ## BFSK Mapping:

In BFSK we have only 1 bit $'b_1'$ if this bit low (logic 0) the output S = 1 and when it high S = j

So, I can say that $\boldsymbol{X_i = complement\ of\ (b_1)\ and\ X_Q = b_1}$

Code

```
% BFSK Mapping
function [X_inphase , X_Quad] = BFSK_Mapper ( Y_signal_BFSK )
    X_inphase = zeros (length (Y_signal_BFSK),1) ;
    X_Quad    = zeros (length (Y_signal_BFSK),1) ;
    % when y_signal_BFSK = 0 ,mapper output = XI + j XQ = 1
    % while y_signal_BFSK = 1 ,  mapper output = XI + j XQ = j
    X_inphase =  1 - Y_signal_BFSK  ; % get the complemente of signal
    X_Quad = Y_signal_BFSK ;
end
```

Plotting

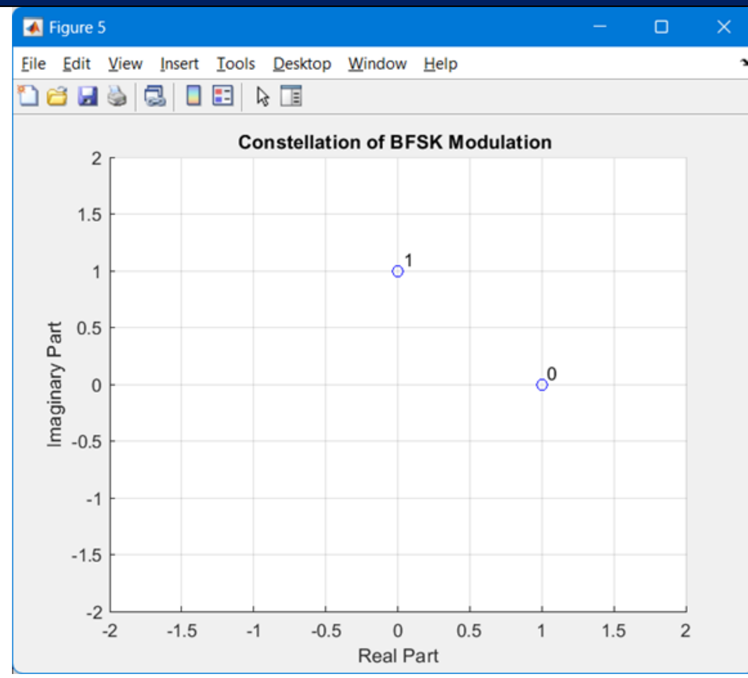


Figure 6: Representation of BFSK's Constellation.

Channel

- **In channel I will generate AWGN and adding it to constellations with different $\frac{E_b}{N_o}$ from -4 to 14 dB in 19 points.**

Code

```
%% Channel
    % calculate E_avg
    E_avg = sum(X_inphase_Mapper(:).*X_inphase_Mapper(:) + X_Quad_Mapper(:).*X_Quad_Mapper(:))/length (Symbols);
    for X= 1 : 19
        Eb = 1 ;
        No(X) = Eb/(10^(Eb_over_NO_dB(X)/10));
        % noise generated
        noise = randn(1 , length(Symbols))+1i .* randn(1 , length(Symbols)) ;
        symbol_with_noise = Symbols + noise * (sqrt((No(X)/2) *(E_avg/bits_modulation))) ;
       if X == 15
            % Plotting the complex numbers as points without space in between
            figure;
            scatter(real(symbol_with_noise), imag(symbol_with_noise), 'filled','SizeData', 1);
            xlabel('Real Part');
            ylabel('Imaginary Part');
            title('constellation after adding noise due to AWGN channel ');
            grid on;
       end
        % call demapper function
        data_stream_demapper = Demapper( symbol_with_noise , constellation_ref , bits_modulation ,Schemes  );
```

De-Mapper

In De-Mapper, this function will take the min distance from constellation ref as pointed in flow chart and then we got the corrected constellation as we can, so we need to get the binary for each case of Modulation schemes. That will be done by using the following equations and nested if. Then get inverse reshape by using the num of modulation bits that used. Then calculate BER.

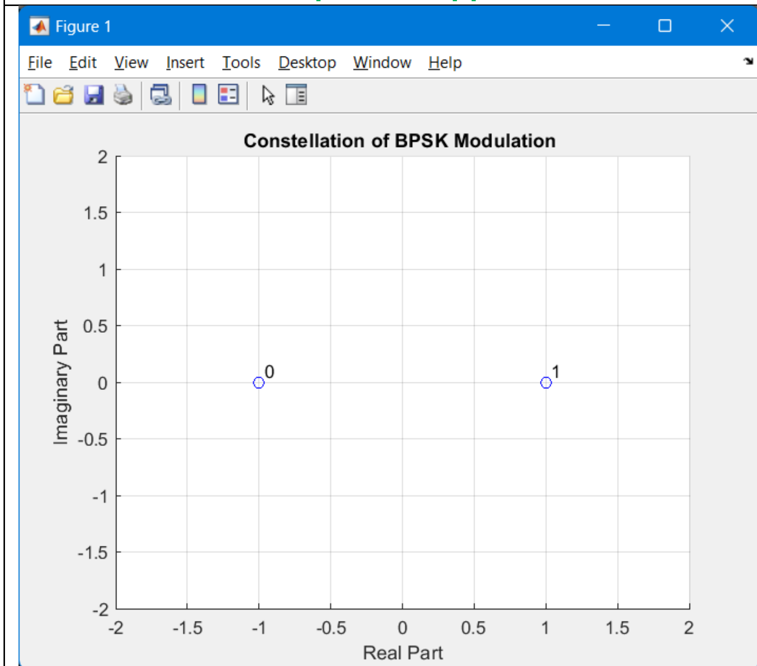
Code

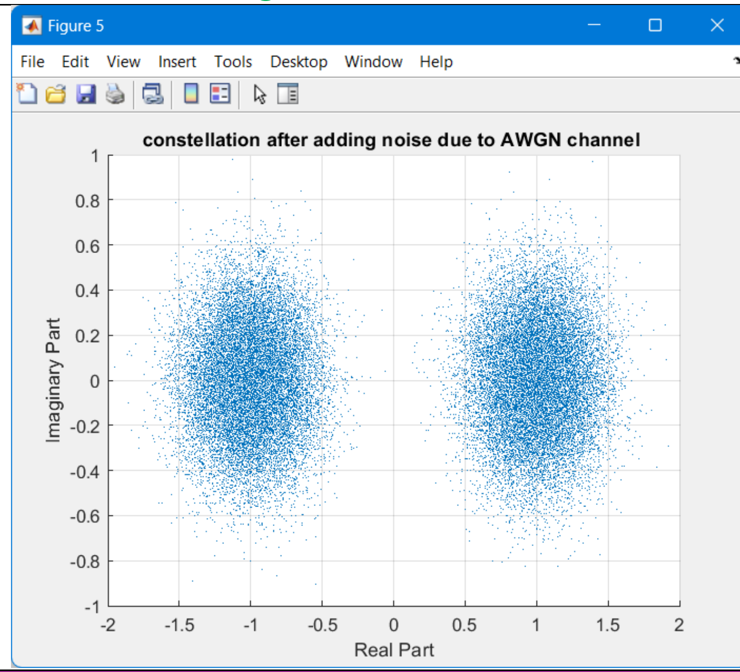
```
% Demapper Block function
function data_Stream_Demapper = Demapper( Symbols_N , ref_symbol , No_Modulation_bits ,Schemes_used)
    XI_Correct = zeros (1,length (Symbols_N)) ;
    XQ_Correct = zeros (1,length (Symbols_N)) ;
    data_Demapper   = zeros (length (Symbols_N),No_Modulation_bits) ;
%     data_Stream_Demapper = zeros (1,length(Symbols_N)*No_Modulation_bits);
    % Get min distance
    for M = 1 : length (Symbols_N)
        [ value , min_Index ] = min (abs(Symbols_N(1,M) - ref_symbol ));
        XI_Correct(1,M) = real(ref_symbol(min_Index,1));
        XQ_Correct(1,M) = imag(ref_symbol(min_Index,1));
        % Convert to get the bits used
        if Schemes_used == 1 % BPSK
            data_Demapper(M) = (XI_Correct(M)+1)/2 ;
         elseif Schemes_used==2
            data_Demapper(M,1) = (XI_Correct(1,M)+1)/2 ;
            data_Demapper(M,2) = (XQ_Correct(1,M)+1)/2 ;
        elseif Schemes_used == 3 % 8PSK
            phase = pi/4 ;
            i_Angle_used = angle(XI_Correct(M)+ 1i.*XQ_Correct(M)) / phase ;
            if i_Angle_used < 0
               i_Angle_used=  i_Angle_used + 8 ;
            end
            dec_to_Binary = rem(floor(i_Angle_used * pow2(-No_Modulation_bits+1:+1:0)), 2);
            data_Demapper(M,:) = Gray_Generator (dec_to_Binary);

        elseif Schemes_used==4  % 16-QAM
            data_Demapper(M,1) = (sign(XI_Correct(M)) + 1) / 2 ;
            data_Demapper(M,2) = (3 - abs(XI_Correct(M)))  / 2 ;
            data_Demapper(M,3) = (sign(XQ_Correct(M)) + 1) / 2 ;
            data_Demapper(M,4) = (3 - abs(XQ_Correct(M)))  / 2 ;

        elseif Schemes_used==5  % BFSK
            if real(Symbols_N(M)) > imag(Symbols_N(M))
                data_Demapper(M)= 0 ;
            else
                data_Demapper(M)= 1 ;
            end
        else   % Schemes_used==6   QPSK with binary encoding
            if [XI_Correct(1,M) XQ_Correct(1,M)] == [1 1]
                data_Demapper(M,:) = [1 0] ;
            elseif [XI_Correct(1,M) XQ_Correct(1,M)] == [1 -1]
                data_Demapper(M,:) = [1 1] ;
            elseif [XI_Correct(1,M) XQ_Correct(1,M)] == [-1 1]
                data_Demapper(M,:) = [0 1] ;
            else
                data_Demapper(M,:) = [0 0] ;
            end
        end
    end
    % Convert reshaped matrix to a single column vector
     data_Stream_Demapper = reshape(data_Demapper',1, []);
end
```

Figures for all Modulation schemes

I. BPSK Modulation

Output of mapper	After adding Noise due to AWGN channel





BER Calculation

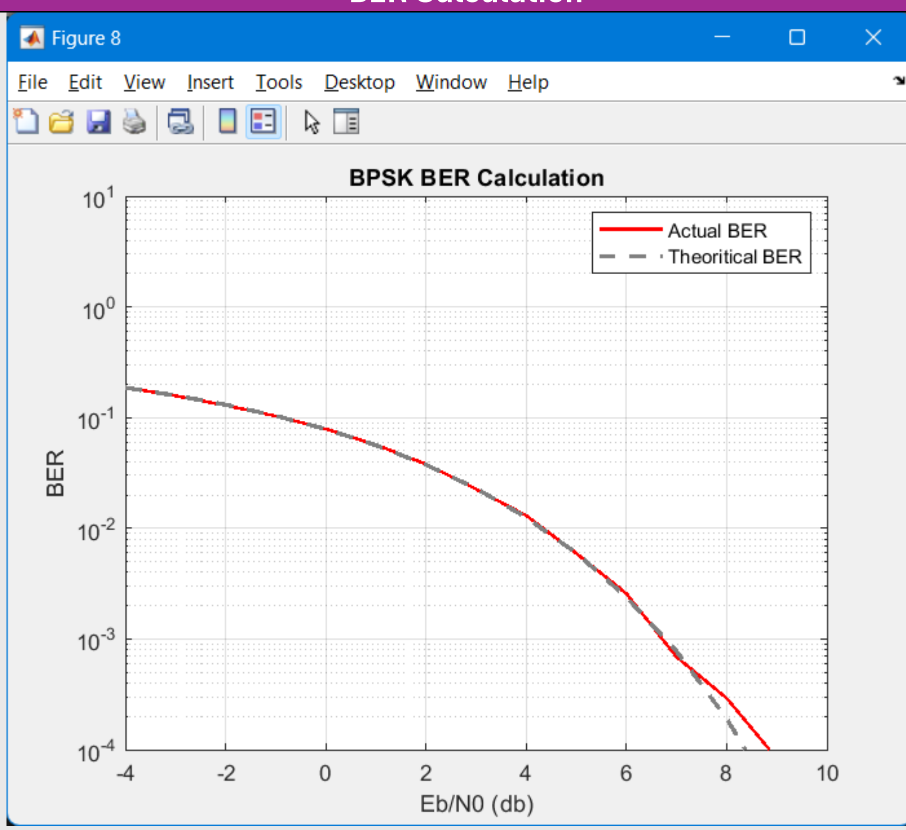
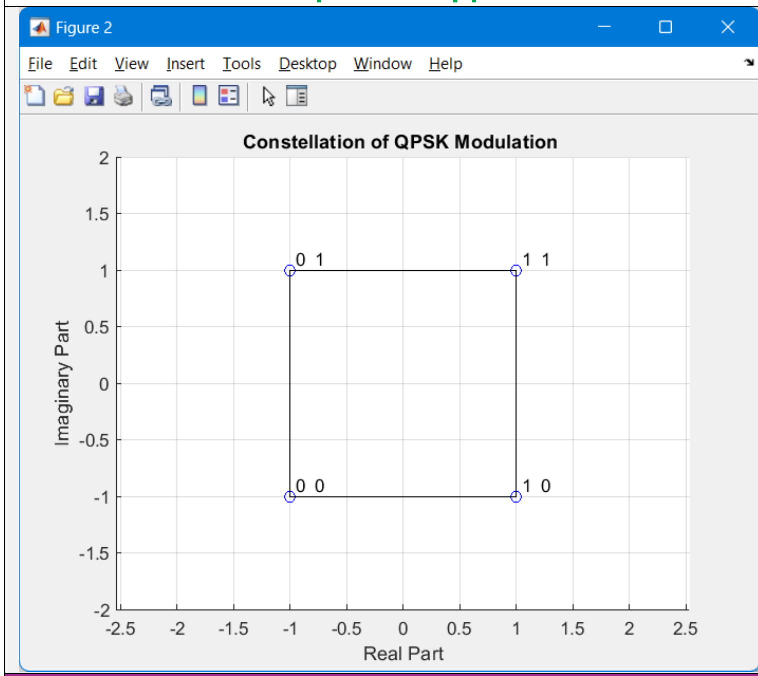
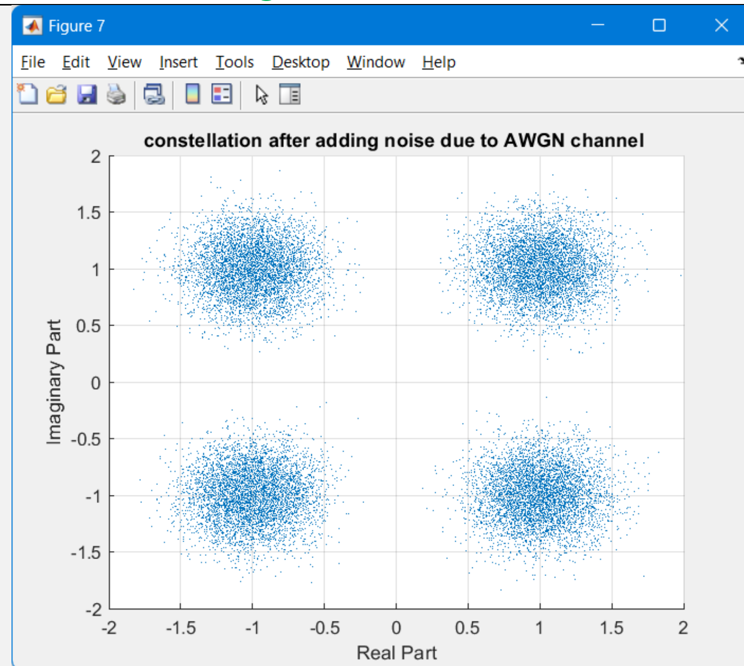


Figure 7:BPSK BER

II. Gray QPSK Modulation

Output of mapper	After adding Noise due to AWGN channel
	

BER Calculation

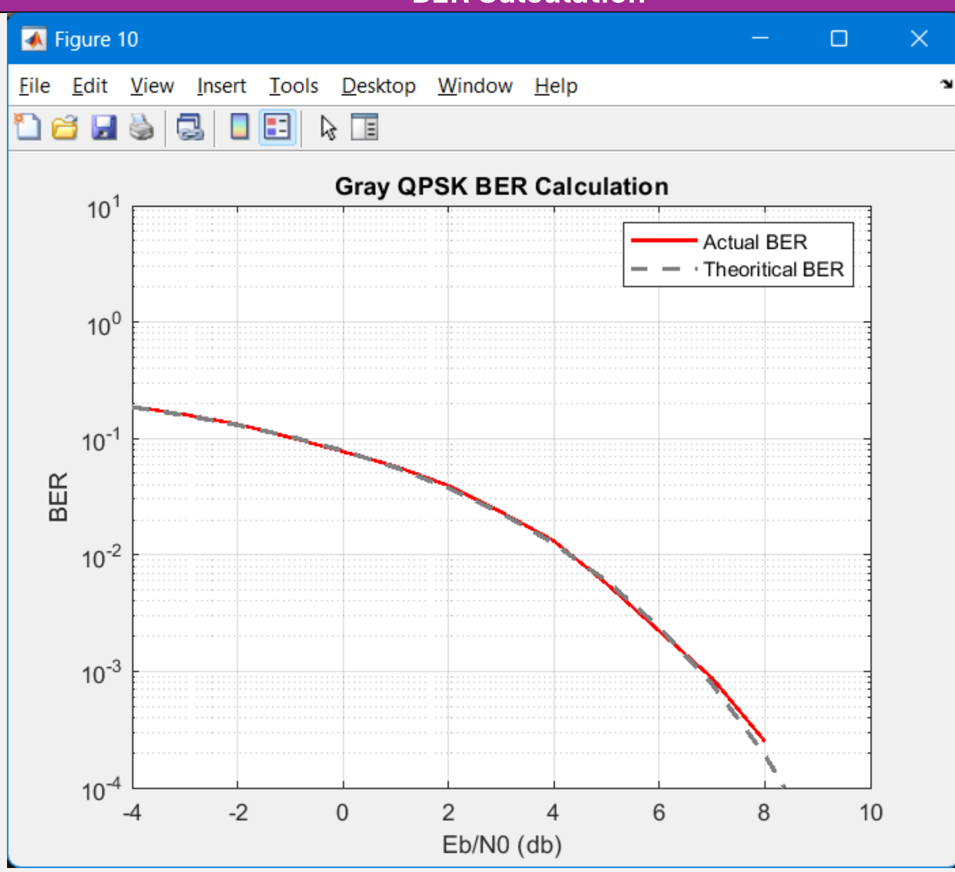
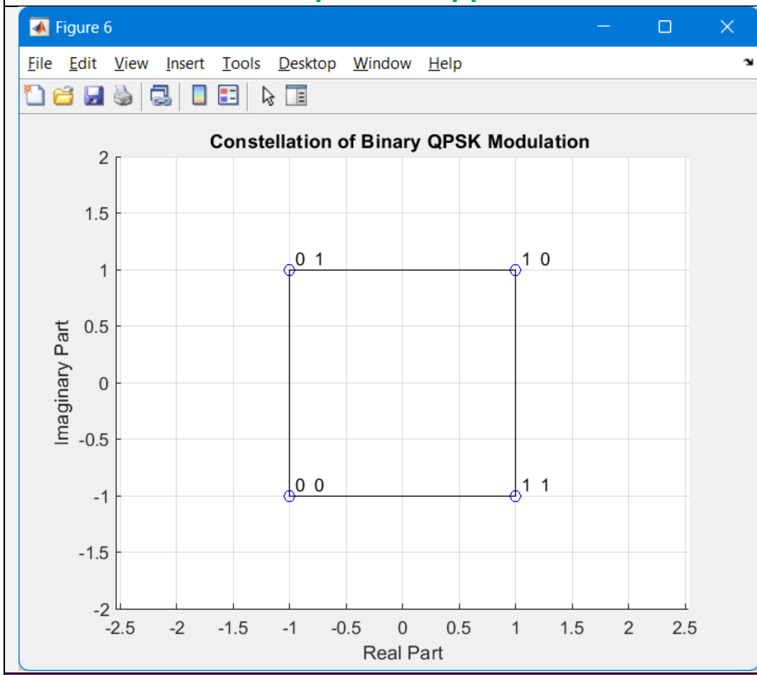
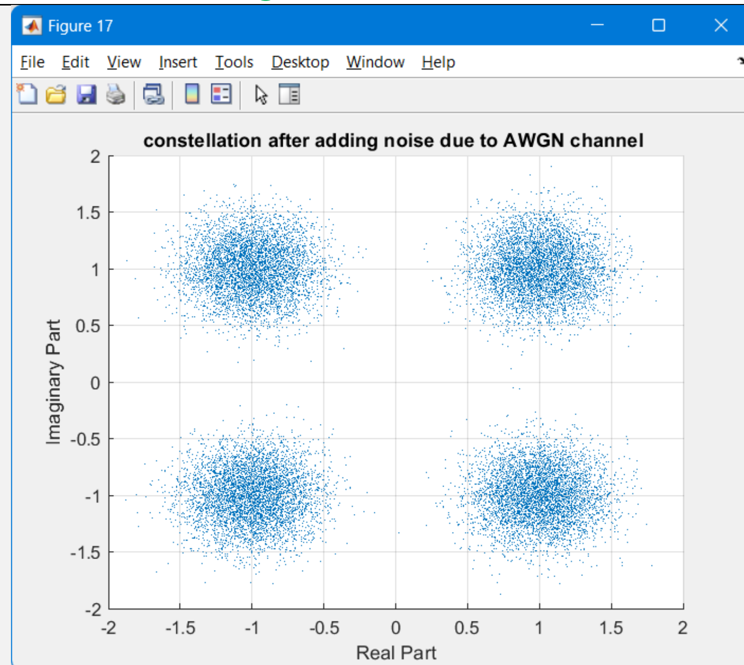


Figure 8:QPSK BER (gray encoding)

III. Binary QPSK Modulation

Output of mapper	After adding Noise due to AWGN channel
	

BER Calculation

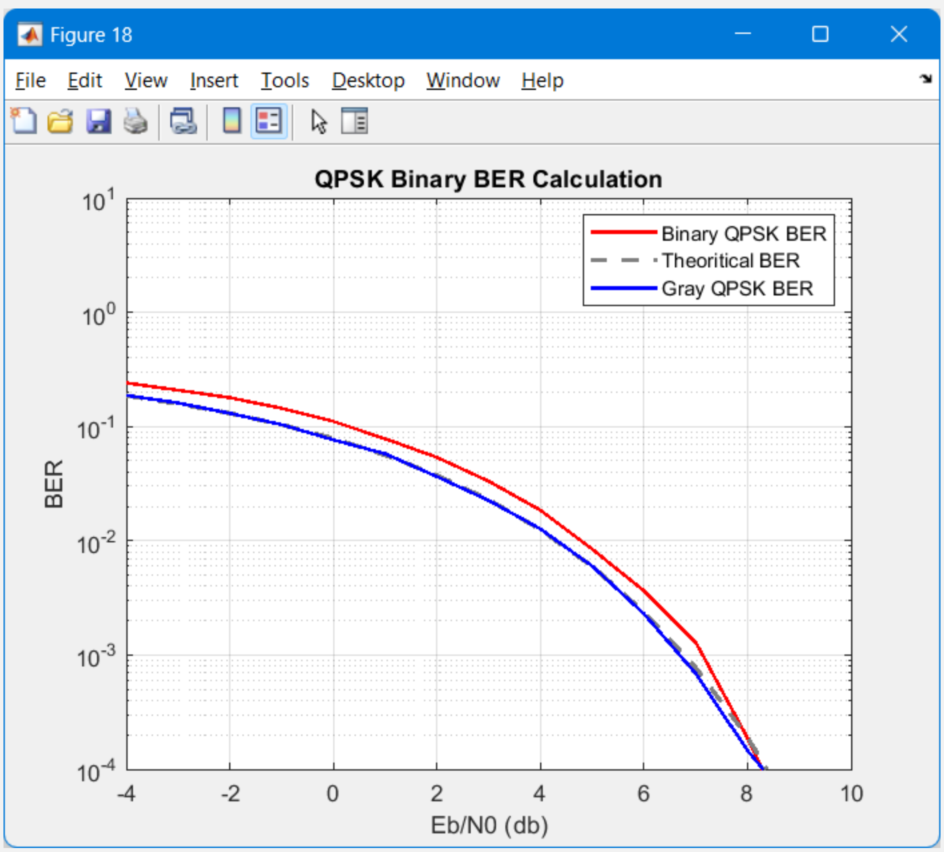
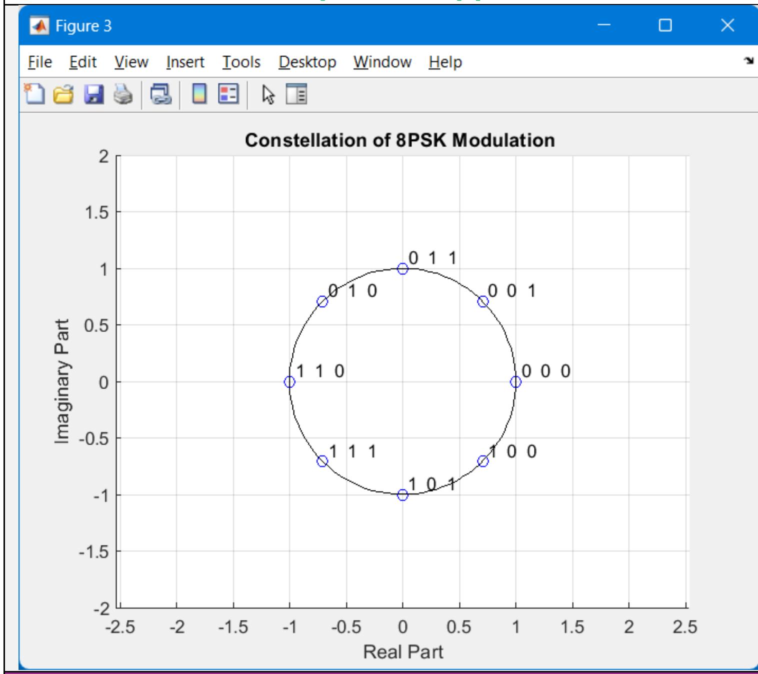
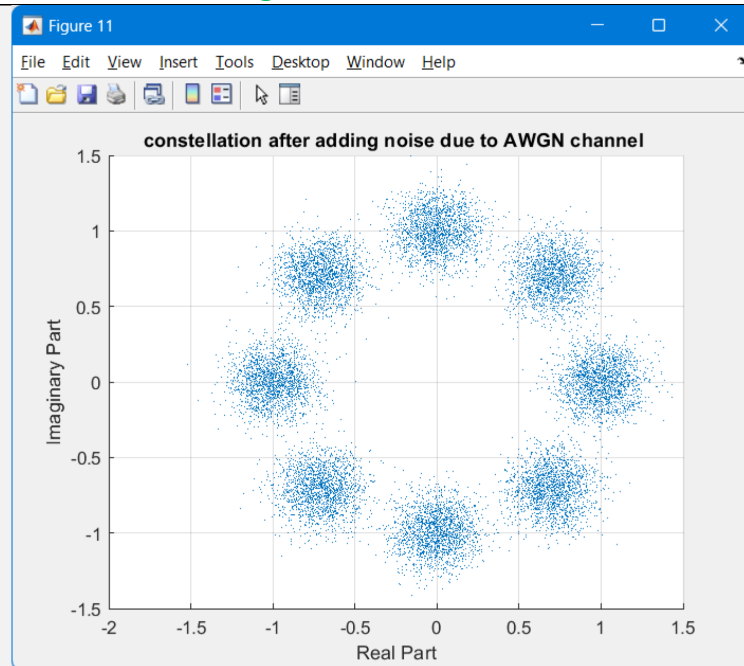


Figure 9: QPSK BER Different encoding

IV. 8PSK Modulation

Output of mapper	After adding Noise due to AWGN channel
	

BER Calculation

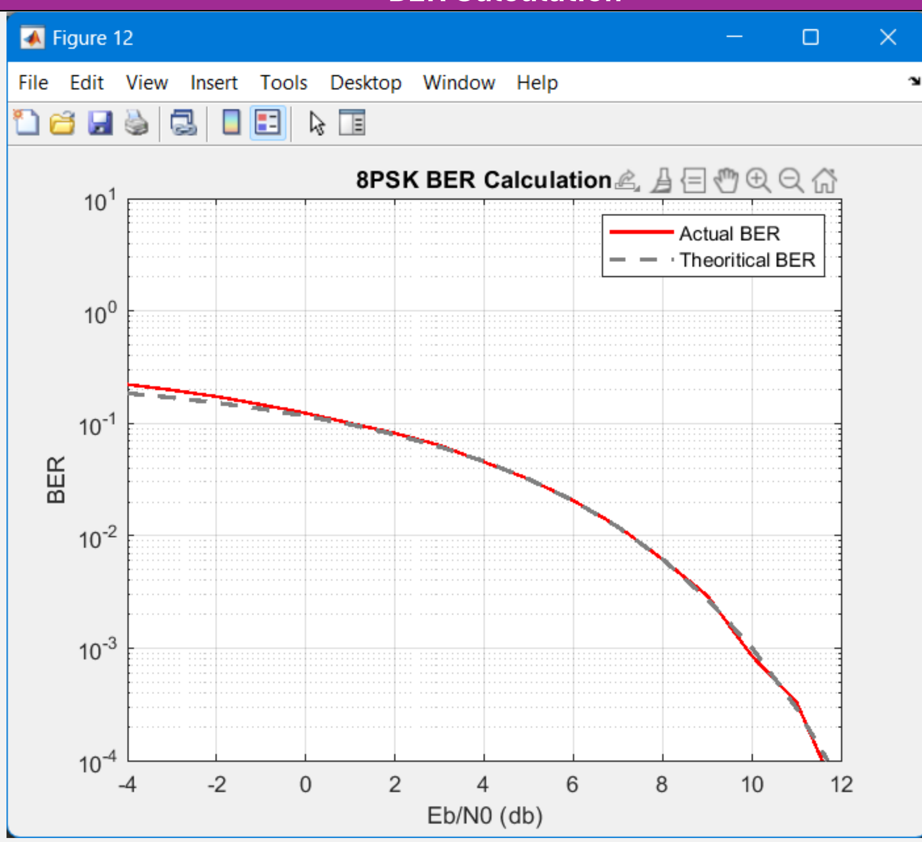
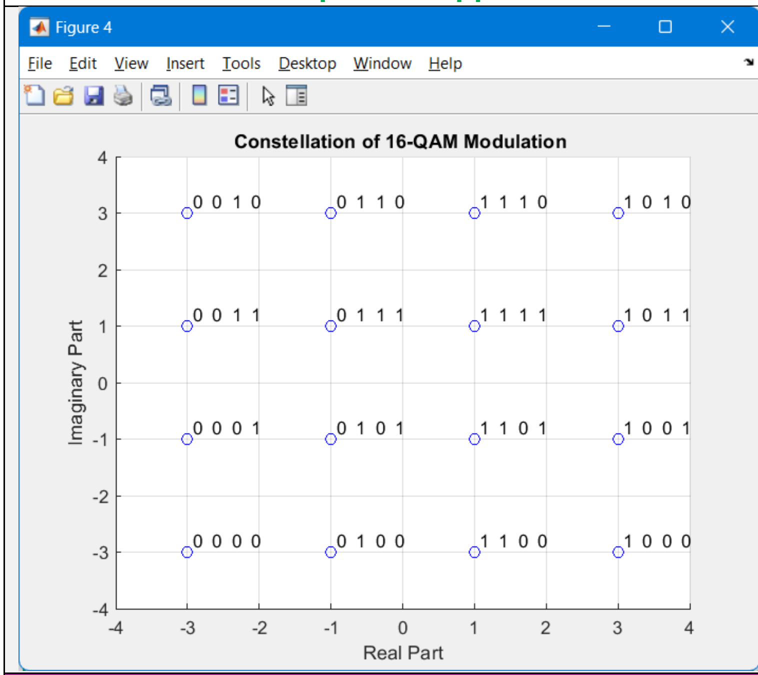
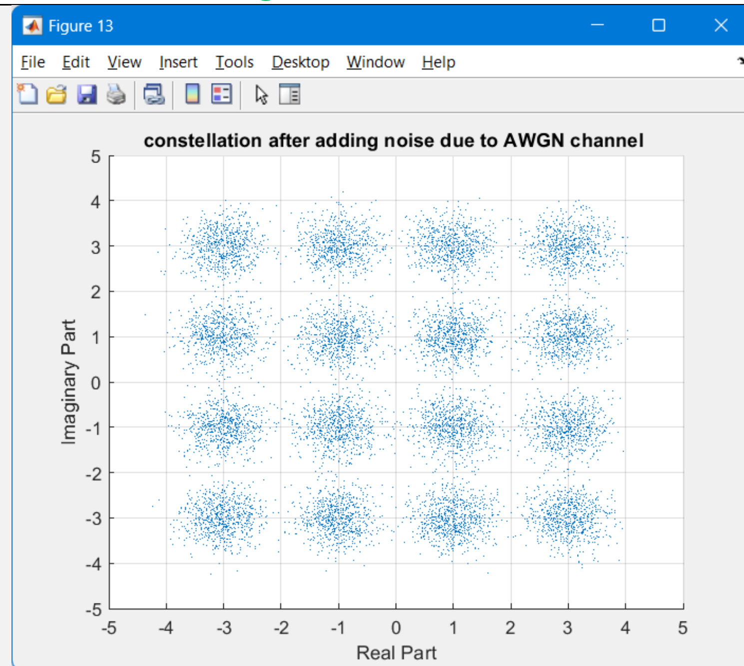


Figure 10: 8PSK BER

V. 16-QAM Modulation

Output of mapper	After adding Noise due to AWGN channel
	

BER Calculation

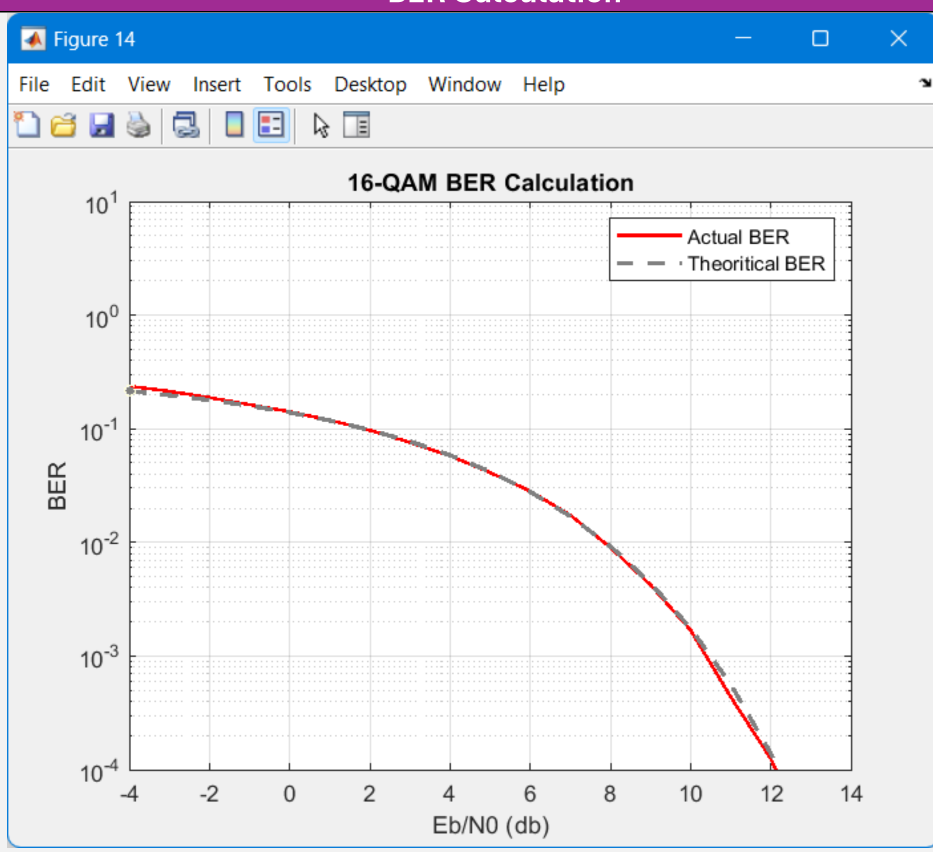
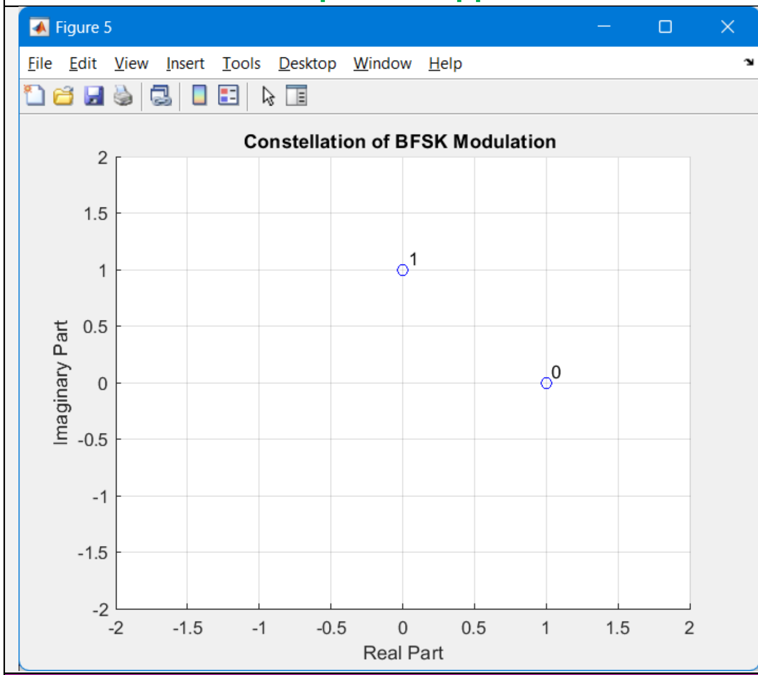
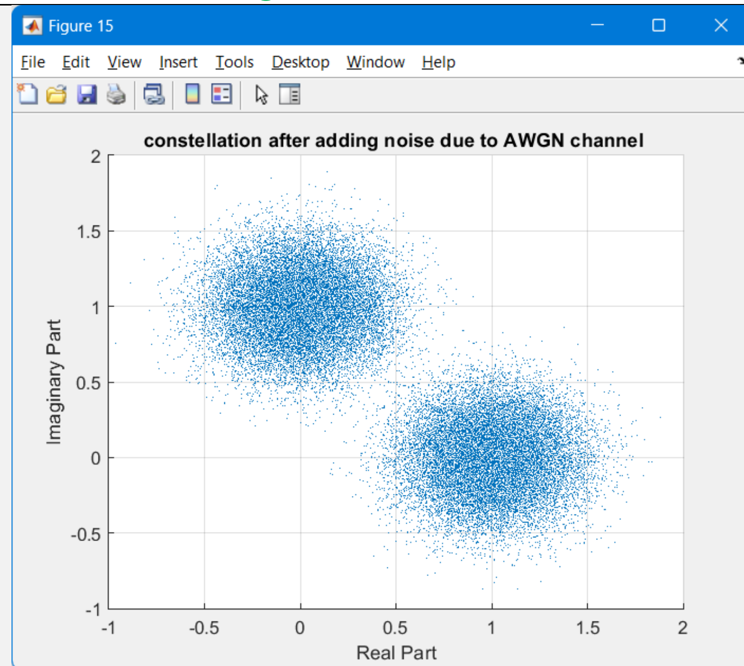


Figure 11: 16-QAM BER

VI. BFSK Modulation (it will appear also in BFSK requirement part)

Output of mapper	After adding Noise due to AWGN channel
	

BER Calculation

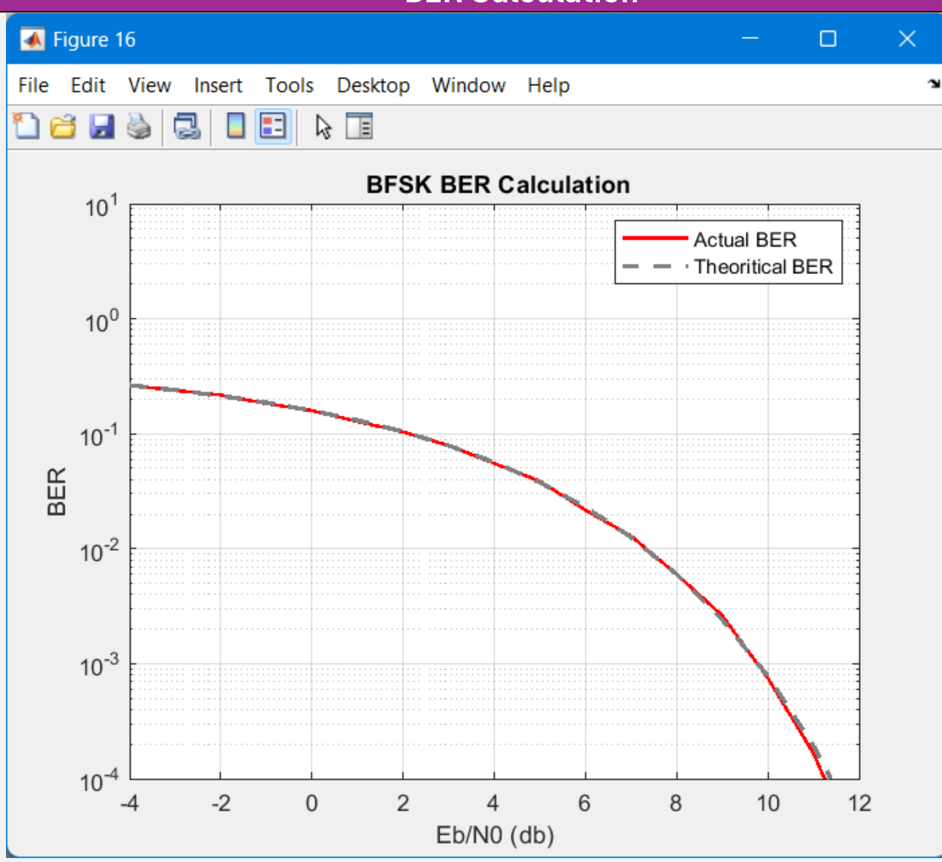


Figure 12:BFSK BER

VII. All modulation

BER Calculation

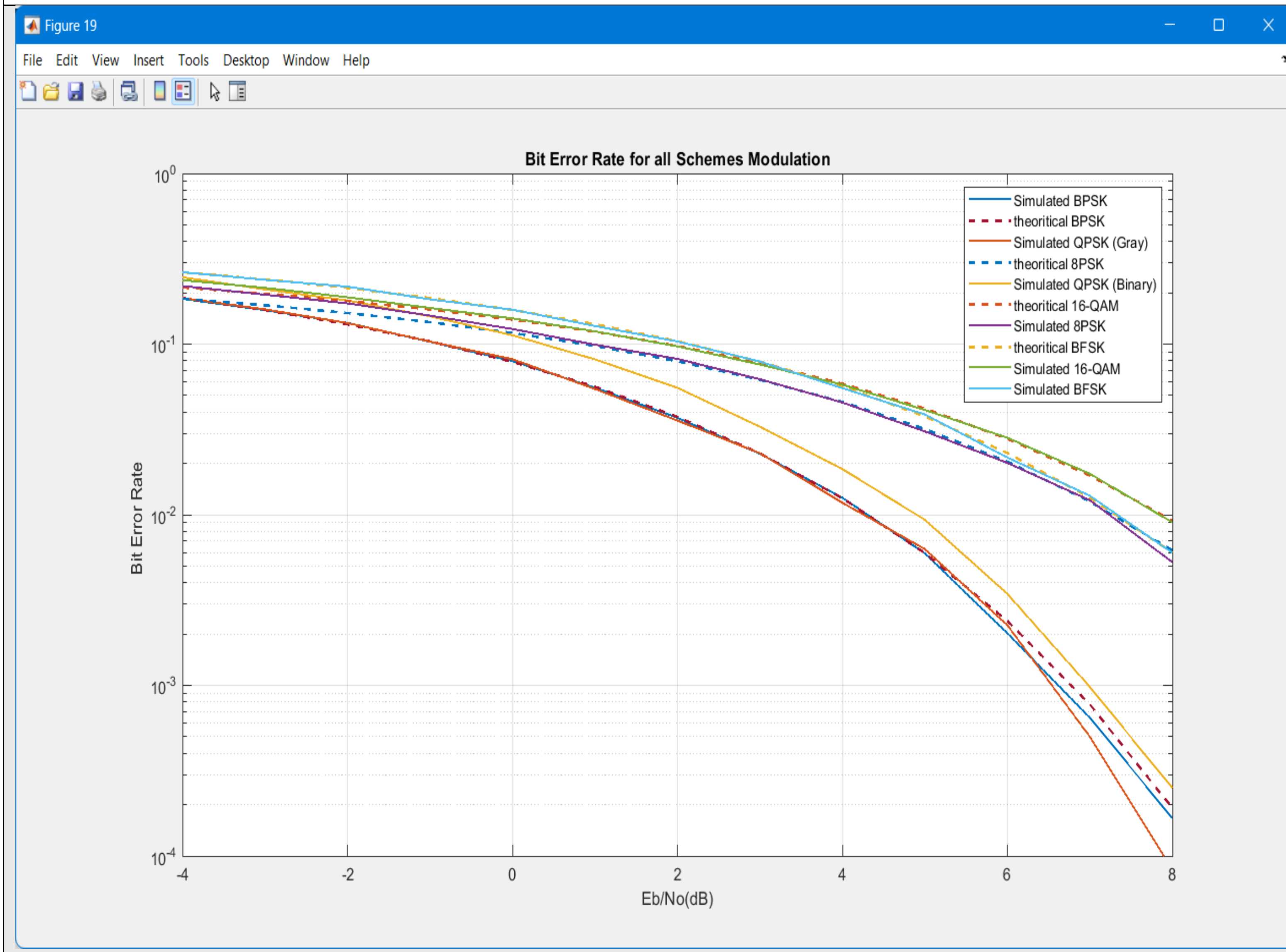


Figure 13:All Modulation BER

Comments

I. QPSK Gray Encoding and BPSK Modulation schemes:

QPSK Gray Encoding and BPSK schemes modulation have the same Bit error rate. Although they have the same symbol rate, QPSK have double bit rate (QPSK have 2 bits in symbol, but BPSK have one bit only). So, using QPSK gray encoding is much better than BPSK, but it needs greater power than BPSK (for sending 2 bits not 1).

II. QPSK Gray Encoding and QPSK Binary Encoding:

Using Gray encoding in Mapper gets less bit error rate than using binary encoding because gray encoding depends on changing in only one bit, so if there is a noise that flips bits as example from '11' to '10' it's better than flipping from '11' to '00'. So, it has smaller bit error rate than Binary.

III. All modulation schemes BER:

- **The BER decay when EB/No increases in all modulation schemes , Also in lower (Eb/No) , the theoretical and simulated BER will have a small difference that happened because we assumed in the theoretical one that only 1bit only can be changed because of Gray encoding and that will not be true in some cases of changing the bits as example from '01' to '10' due to large noise in channel and signal didn't have power larger than noise so the bits will flip and make the simulated will be larger in this case.**

- **The same BER for all modulation schemes can be found but they need different Eb/No, so that will need different higher powers for other modulation schemes.**

- **Because of the non-infinite number of bits, the simulated curves will have distortion, ripples and randomness.**

- **In the Graph of full modulation schemes, Simulated and theoretical BER are matched with each other, and BPSK (or QPSK) has lower bit error rate because they have the same bit energy (Eb = 1) and BPSK has 1 bit per symbol.**

BFSK Modulation Scheme requirements

Consider the BFSK signal given by

$$S_i(t) = \begin{cases} \sqrt{\frac{2E_b}{T_b}} cos(2\pi f_i t) & ,0 < t < T_b \\ 0 & otherwise \end{cases}$$

$and\ T_x\ frequency : f_i = \frac{n_c+i}{T_b}\ ,i\ = 1\ ,2$

1. Bases functions of the signal set:

$$\phi_1(t) = \sqrt{\frac{2}{T_b}}\cos(2\pi f_1 t)\ ,\ f_1 = \frac{n_c+1}{T_b}$$

$$\phi_2(t) = \sqrt{\frac{2}{T_b}}\cos(2\pi f_2 t)\ ,\ f_2 = \frac{n_c+2}{T_b} = f_1 + \frac{1}{T_b}$$

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

$\because \Delta f = f_2 - f_1 = \frac{1}{T_b} \qquad , \qquad @\ f_1 = f_c$

$\therefore f_2 = f1 + \Delta f$

$\therefore S_1(t) = \sqrt{\frac{2E_b}{T_b}}\ cos(2\pi f_1 t) = \sqrt{\frac{2E_b}{T_b}}\ cos(2\pi f_c t)$

$and\ S_2 = \sqrt{\frac{2E_b}{T_b}}\ cos(2\pi f_2\ t)\ = \sqrt{\frac{2E_b}{T_b}}\ cos(2\pi t(f_c + \Delta f))$

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

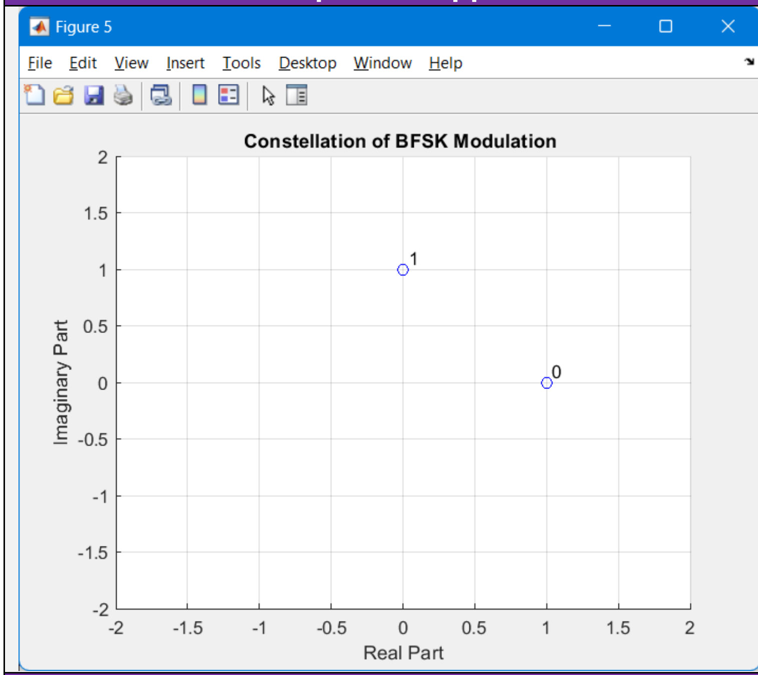
$S_i(t) = real\left(S_{i_{base\ band}}(t)\ \times e^{-j2\pi f_c t}\right)$

$$\therefore S_{1_{base\ band}} = \sqrt{\frac{2E_b}{T_b}}\ cos(2\pi f_c t)$$

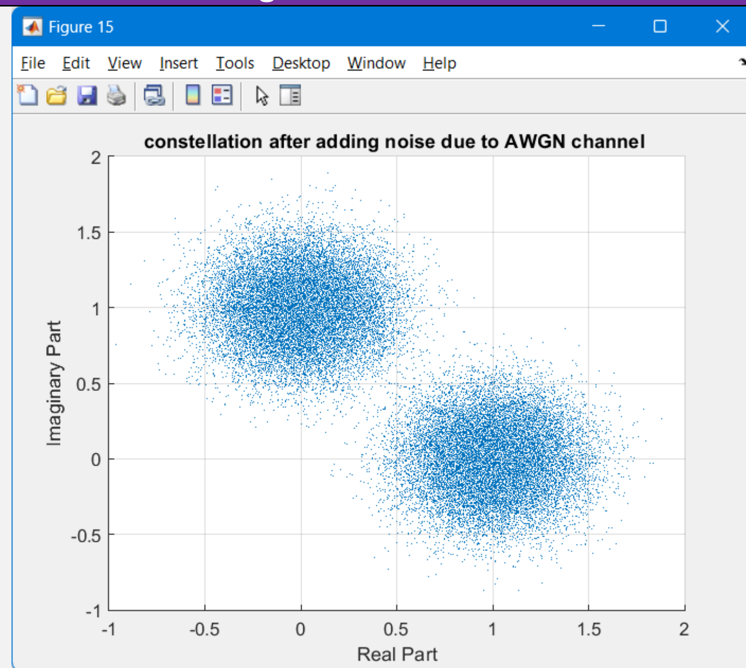
$$\therefore S_{2_{base\ band}} = \sqrt{\frac{2E_b}{T_b}}\ (\ cos(2\pi\Delta f) + j\ sin(2\pi\Delta f)\ )$$

BFSK modulation scheme BER

Output of mapper



After adding Noise due to AWGN channel



BER Calculation

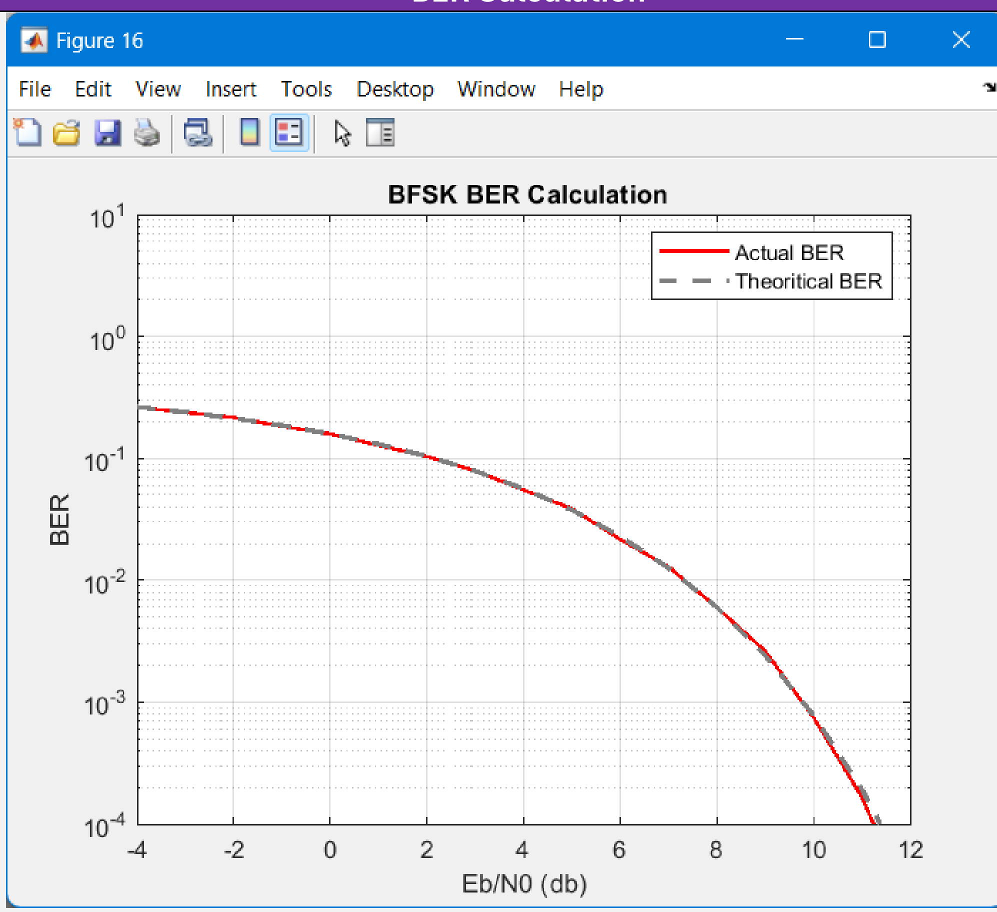


Figure 14 : repeat figure of BFSK BER

The PSD of the signal set using the base band equivalent signal

- Generate random bits and mapped them using $S_{i_{bb}}$, which logic 0 to $S_{1_{BB}}$ and logic 1 to $S_{2_{BB}}$:

```
%% PSD of BFSK
% PSD Parameters
ensemble_size = 500;
num_bits_fsk = 100;
Tb = 50/1000  ; % 50 ms
num_samples_per_bit = 7 ;
sampling_time = Tb/num_samples_per_bit ;
Total_samples = num_samples_per_bit * num_bits_fsk ;
Eb_BFSK = 1 ;
step = (Tb/num_samples_per_bit);
delta_F = 1/Tb ;
% Generate time vector for One bit
t = (0:num_samples_per_bit-1)* step ;
%generating the baseband equivalent signal
S1_BB = sqrt(2*Eb_BFSK/Tb); %equivalent to 0
S2_BB = sqrt(2*Eb_BFSK/Tb)*exp(1i*(2*pi*delta_F*t)); %equivalent to 1
% Generate data matrix for X is No of Realization and y is num of bit +1 for delay
binary_sequence_PSD = randi([0,1], ensemble_size, num_bits_fsk+1);
% Generate BFSK symbol
BFSK_symbols = zeros(ensemble_size,Total_samples+num_samples_per_bit);
for ensemble = 1 : ensemble_size
    for bit = 0 : num_bits_fsk
        if binary_sequence_PSD(ensemble , bit+1) == 1
            BFSK_symbols (ensemble,1+bit* num_samples_per_bit:(bit+1)*num_samples_per_bit)=S2_BB  ;
        else
            BFSK_symbols (ensemble,1+bit* num_samples_per_bit:(bit+1)*num_samples_per_bit)=S1_BB ;
        end
    end
end
```

- Adding delay:

```
% Generating Delay to add it
Td = randi([0,(num_samples_per_bit-1)],ensemble_size,1);
% Define matrix to store the values after adding the delay
Data_BFSK = zeros(ensemble_size, Total_samples);
% Apply the delay to the Binary BFSK Symbol
for i = 1:ensemble_size
    Data_BFSK(i,:)= BFSK_symbols(i, Td(i)+1 : Total_samples + Td(i));
end
```

- Get Statistical auto correlation:

```
% Define auto correlation matrix
initial_Stat_Auto_corr = zeros(size(Data_BFSK));
for i = 1:Total_samples
        for j = 1:Total_samples
            % Select two columns for element-wise multiplication
            DOT_colmuns = conj(Data_BFSK(:, i)) .* Data_BFSK(:, j);
            % Perform element-wise multiplication
            if (j>=i)
                initial_Stat_Auto_corr(:,j-i+1) = initial_Stat_Auto_corr(:,j-i+1)+ DOT_colmuns ;
            end
        end
end
Stat_Auto_corr = zeros ( 1 , Total_samples);
for i = 1 :Total_samples
    Stat_Auto_corr (1 , i ) = sum (initial_Stat_Auto_corr( : , i ) )/(Total_samples*ensemble_size) ;
end
% Concatenate to get the final statistical autocorrelation
Stat_Auto_corr_full = cat (2, conj(fliplr(Stat_Auto_corr(2:end))), Stat_Auto_corr);
```

- Plot the simulated PSD across the theoretical:

```
%Plotting PSD graph
delta = zeros(size(Stat_Auto_corr_full));
fs = 1 /sampling_time ;
M = -Total_samples+1 : Total_samples - 1;
f = M * fs * (Tb)/ (size(M,2))  ;  %Sampling frequency
delta(600) = 1/(4*Tb);  % Set delta function at desired frequency
delta(800) = 1/(4*Tb);
for H = 1 : length (f)
    delta(H) =abs( delta(H) + (4 * cos(pi * f(H))^2)/((pi)^2 *(4*(f(H))^2 -1)^2));
end
PSD = abs(fftshift(fft(Stat_Auto_corr_full)))/(4*Total_samples); %normalized for 2Eb
figure('Name','PSD for BFSK Scheme');
plot(f , delta , 'r');
hold on ;
plot(f , PSD ,'b' );
legend('theoritical PSD ' ,'Simulated PSD  ' );
hold off;
title("PSD vs Frequency");
xlabel("Normalized frequency , F*T_b ");
ylabel("Normalized PSD ,S_B(F)/2E_b ");
ylim([0,0.8]);
xlim([-2,2]);
grid on;
```

The Plot of PSD For BFSK Modulation Scheme

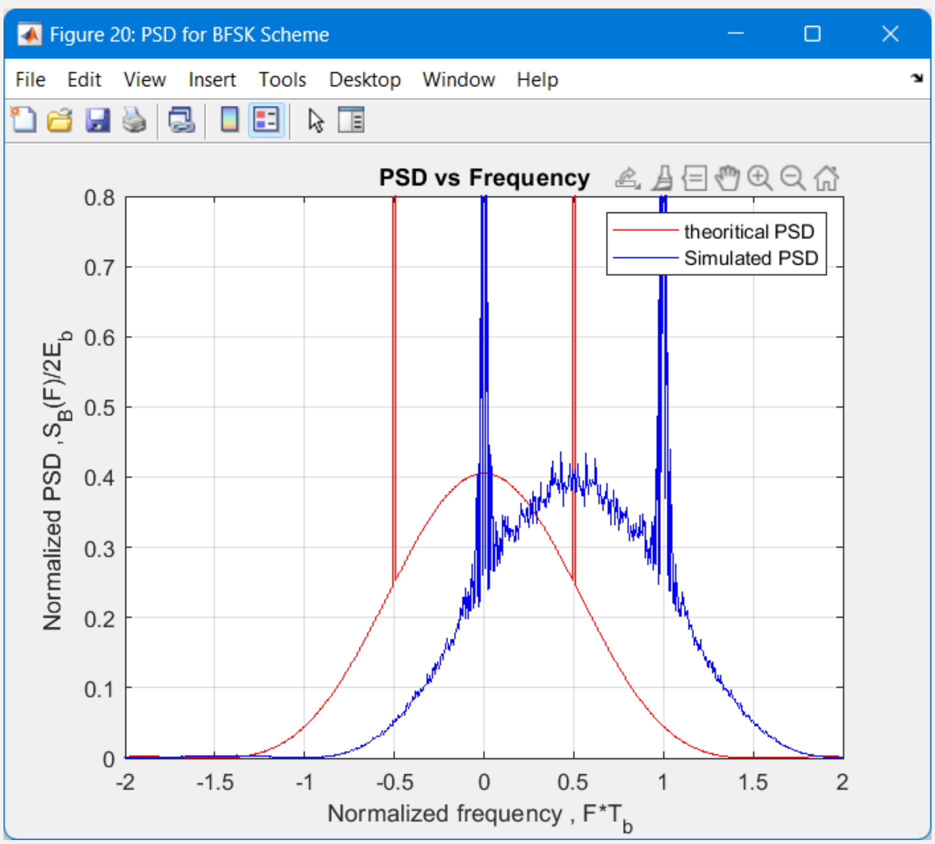


Figure 15 : BFSK PSD

Comments on The Figure of BPSK PSD

1. **simulated PSD took $f_1 = f_C$ and $f_2 = \Delta f = \frac{1}{T_b}$.**
2. **theoretical PSD took $f_1 = f_c - \frac{1}{2T_b}$ and $f_2 = f_c + \frac{1}{2T_b}$**
3. **the simulated PSD graph will have 2 Deltas $\delta(f) \begin{cases} @\, f = 0 \\ @\, f = \frac{1}{T_b}\ HZ \end{cases}$, because this simulated PSD was in Base band so f_1 will bring the 1st delta at 0 and f_2 will bring another delta at Δf HZ which equal $\frac{1}{T_b}$ HZ .**
4. **The Theoretical PSD = $\frac{E_b}{2T_b}\left[\delta\left(f - \frac{1}{2T_b}\right) + \delta\left(f + \frac{1}{2T_b}\right)\right] + \frac{8E_b \cos^2(\pi T_b f)}{\pi^2\left(4T_b^2 f^2 - 1\right)^2}$**

So, it will have 2 Deltas $\delta(f) \begin{cases} @\, f = -\frac{1}{2T_b}\ HZ \\ @\, f = +\frac{1}{2T_b}\ HZ \end{cases}$

5. **Y – Axis normalized twice, 1st by division the length of PSD and 2nd by division 2Eb to get S(f)/2Eb.**

6. **X– Axis normalized also by multiplication Tb and fs to get fTb.**

Code Project 3

```
clc ;
clear ;
close all;
% parameters
QAM_symbol= 16 ;
No_Symbols = 1000 ;
NO_of_bits = QAM_symbol * No_Symbols * 3 ;
% noise
Noise_points = 19 ;
%Generate BER vectros
BPSK_BER_Actual  = zeros(Noise_points,1);
QPSK_BER_Actual  = zeros(Noise_points,1);
QPSK_BER_Actual_binary  = zeros(Noise_points,1);
Eight_BER_Actual = zeros(Noise_points,1);
QAM_BER_Actual   = zeros(Noise_points,1);
BFSK_BER_Actual = zeros(Noise_points,1);
BER_Actual = zeros(Noise_points,1);

BPSK_BER_theoritical = zeros(Noise_points,1);
QPSK_BER_theoritical = zeros(Noise_points,1);
Eight_PSK_BER_theoritical = zeros(Noise_points,1);
QAM_BER_theoritical = zeros(Noise_points,1);
BFSK_BER_theoritical = zeros(Noise_points,1);

No = zeros(1,Noise_points);
Eb_over_NO_dB= -4 : +1 : 14 ;
% test modulation and plot Constellation
Test_Modulation_Schemes();
% Run-all
for Schemes = 1 : 6
    % Generating Data bits
    Data_bits_Generated = randi([0 1],1,NO_of_bits);  % generate the random logic bit-stream
    if Schemes == 1
        disp("The current Modulation Scheme is BPSK ") ;
        bits_modulation = 1 ;
     elseif Schemes == 2
        disp("The current Modulation Scheme is QPSK ") ;
        bits_modulation = 2 ;
     elseif Schemes == 3
        disp("The current Modulation Scheme is 8PSK ") ;
        bits_modulation = 3 ;
      elseif Schemes == 4
        disp("The current Modulation Scheme is 16-QAM ") ;
        bits_modulation = 4 ;
      elseif Schemes == 5
        bits_modulation = 1 ;
        disp("The current Modulation Scheme is BFSK ") ;
    else % QPSK Binary
        disp("The current Modulation Scheme is QPSK with Binary Encoding ") ;
        bits_modulation = 2 ;
    end
    % Generate X_inphase and X_Quad vectros
    X_inphase_Mapper = zeros (length (Data_bits_Generated)/bits_modulation ,1) ;
    X_Quad_Mapper    = zeros (length (Data_bits_Generated)/bits_modulation ,1) ;

    % Convert data bits into groups of bits
    dataBitsGrouped = reshape(Data_bits_Generated, bits_modulation , [])';
   % Calling Mapper Function
    [X_inphase_Mapper , X_Quad_Mapper] = Mapper ( dataBitsGrouped  , Schemes );
    % Keep Symbol as X_i + j X_Q
    Symbols = X_inphase_Mapper' + 1i.* X_Quad_Mapper';
```

```
    % Create constellation references
    data_ref = 0 : pow2(bits_modulation)-1 ;
    % Calculate the binary representation
    data_test_binary = rem(floor(data_ref(:) * pow2(-bits_modulation+1:+1:0)), 2);
    [XI_ref , XQ_ref] = Mapper ( data_test_binary  , bits_modulation );
    constellation_ref = XI_ref + 1i.* XQ_ref;

%% Channel
    % calculate E_avg
    E_avg = sum(X_inphase_Mapper(:).*X_inphase_Mapper(:) + X_Quad_Mapper(:).*X_Quad_Mapper(:))/length
(Symbols);
    for X= 1 : 19
        Eb = 1 ;
        No(X) = Eb/(10^(Eb_over_NO_dB(X)/10));
        % noise generated
        noise = randn(1 , length(Symbols))+1i .* randn(1 , length(Symbols)) ;
        symbol_with_noise = Symbols + noise * (sqrt((No(X)/2) *(E_avg/bits_modulation))) ;
       if X == 15
            % Plotting the complex numbers as points without space in between
            figure;
            scatter(real(symbol_with_noise), imag(symbol_with_noise), 'filled','SizeData', 1);
            xlabel('Real Part');
            ylabel('Imaginary Part');
            title('constellation after adding noise due to AWGN channel ');
            grid on;
       end
        % call demapper function
        data_stream_demapper = Demapper( symbol_with_noise , constellation_ref ,
bits_modulation ,Schemes  );
        % check error
        check_errors = (data_stream_demapper ~= Data_bits_Generated);
        BER_Actual(X) = length(check_errors(check_errors==1))/ length(data_stream_demapper) ;
        if Schemes == 1
            BPSK_BER_theoritical(X)= 0.5 *erfc (sqrt(1/No(X)));
            BPSK_BER_Actual(X) = BER_Actual(X);
        elseif Schemes == 2
            QPSK_BER_theoritical(X)= 0.5 *erfc (sqrt(1/No(X)));
            QPSK_BER_Actual(X) = BER_Actual(X);
        elseif Schemes == 3
            Eight_PSK_BER_theoritical(X)= (1/3) * erfc(sqrt(3*(1/No(X)))*sin(pi/8));
            Eight_BER_Actual(X) = BER_Actual(X);
        elseif Schemes == 4
            QAM_BER_theoritical(X)= (3/8)* erfc(sqrt(((1/No(X))/2.5)));
            QAM_BER_Actual(X) = BER_Actual(X);
        elseif Schemes == 5
            BFSK_BER_theoritical(X)= (0.5)* erfc(sqrt(0.5/No(X)));
            BFSK_BER_Actual(X) = BER_Actual(X);
        else % Schemes == 6
            QPSK_BER_theoritical(X)= 0.5 *erfc (sqrt(1/No(X)));
            QPSK_BER_Actual_binary(X) = BER_Actual(X);
        end
    end

    figure ;
    semilogy (Eb_over_NO_dB,BER_Actual,'r','linewidth',1.5);
    hold on;
   if Schemes == 1
        semilogy(Eb_over_NO_dB,BPSK_BER_theoritical, '--','color',[0.5, 0.5, 0.5],'linewidth',2);
        title("BPSK BER Calculation");
        legend("Actual BER","Theoritical BER");
    elseif Schemes == 2
        semilogy(Eb_over_NO_dB,QPSK_BER_theoritical, '--','color',[0.5, 0.5, 0.5],'linewidth',2);
         title("Gray QPSK BER Calculation");
```

```
         legend("Actual BER","Theoritical BER");
     elseif Schemes == 3
        semilogy(Eb_over_NO_dB,Eight_PSK_BER_theoritical, '--','color',[0.5, 0.5, 0.5],'linewidth',2);
         title("8PSK BER Calculation");
         legend("Actual BER","Theoritical BER");
    elseif  Schemes == 4
        semilogy(Eb_over_NO_dB,QAM_BER_theoritical, '--','color',[0.5, 0.5, 0.5],'linewidth',2);
         title("16-QAM BER Calculation");
        legend("Actual BER","Theoritical BER");
    elseif  Schemes==5
       semilogy(Eb_over_NO_dB,BFSK_BER_theoritical, '--','color',[0.5, 0.5, 0.5],'linewidth',2);
       title(" BFSK BER Calculation");
       legend("Actual BER","Theoritical BER");
    else
        semilogy(Eb_over_NO_dB,QPSK_BER_theoritical, '--','color',[0.5, 0.5, 0.5],'linewidth',2);
         title("Binary and Gray QPSK  BER Calculation");
         hold on ;
         semilogy(Eb_over_NO_dB,QPSK_BER_Actual,'b','linewidth',1.5);
         title("QPSK Binary BER Calculation");
         legend("Binary QPSK BER","Theoritical BER " ,"Gray QPSK BER" );
    end
    grid  on ;
    ylim([10^-4,10^1]);
    xlabel("Eb/N0 (db)");
    ylabel("BER");
    hold off;
end
% plot all graph
all_Simulated_BER = [BPSK_BER_Actual , QPSK_BER_Actual ...
                     QPSK_BER_Actual_binary , Eight_BER_Actual ,...
                     QAM_BER_Actual, BFSK_BER_Actual];
all_Theoritical_BER = [BPSK_BER_theoritical,Eight_PSK_BER_theoritical ,...
                         QAM_BER_theoritical ,BFSK_BER_theoritical ];
figure;
colorOrder = get(gca, 'ColorOrder');
for C_S = 1 : 6
    semilogy(Eb_over_NO_dB,all_Simulated_BER(:,C_S)','Color', colorOrder(mod(C_S-1, size(colorOrder, 1)) +
1, :), 'LineWidth', 1.2);
    hold on;
    if C_S <= 4
        semilogy(Eb_over_NO_dB,all_Theoritical_BER(:,C_S)','--','Color', colorOrder(mod(C_S+5,
size(colorOrder, 1)) + 1, :), 'LineWidth', 1.52);
        hold on;
    end
end
legend('Simulated BPSK  '       , 'theoritical BPSK ',...
       'Simulated QPSK (Gray)  ', 'theoritical 8PSK ',...
       'Simulated QPSK (Binary)', 'theoritical 16-QAM ',...
       'Simulated 8PSK '        , 'theoritical BFSK ',...
       'Simulated 16-QAM '      , 'Simulated BFSK ');
grid on
xlabel('Eb/No(dB)');
ylabel('Bit Error Rate ');
title('Bit Error Rate for all Schemes Modulation');
ylim([1e-4,1]);
xlim([-4 , 8]);

%% PSD of BFSK
% PSD Parameters
ensemble_size = 500;
num_bits_fsk = 100;
Tb = 50/1000  ; % 50 ms
num_samples_per_bit = 7 ;
```

```matlab
sampling_time = Tb/num_samples_per_bit ;
Total_samples = num_samples_per_bit * num_bits_fsk ;
Eb_BFSK = 1 ;
step = (Tb/num_samples_per_bit);
delta_F = 1/(Tb) ;
% Generate time vector for One bit
t = (0:num_samples_per_bit-1)* step ;
%generating the baseband equivalent signal
S1_BB = sqrt(2*Eb_BFSK/Tb); %equivalent to 0
S2_BB = sqrt(2*Eb_BFSK/Tb)*exp(1i*(2*pi*delta_F*t)); %equivalent to 1
% Generate data matrix for X is No of Realization and y is num of bit +1 for delay
binary_sequence_PSD = randi([0,1], ensemble_size, num_bits_fsk+1);
% Generate BFSK symbol
BFSK_symbols = zeros(ensemble_size,Total_samples+num_samples_per_bit);
for ensemble = 1 : ensemble_size
    for bit = 0 : num_bits_fsk
        if binary_sequence_PSD(ensemble , bit+1) == 1
            BFSK_symbols (ensemble,1+bit* num_samples_per_bit:(bit+1)*num_samples_per_bit)=S2_BB  ;
        else
            BFSK_symbols (ensemble,1+bit* num_samples_per_bit:(bit+1)*num_samples_per_bit)=S1_BB ;
        end
    end
end
% Generating Delay to add it
Td = randi([0,(num_samples_per_bit-1)],ensemble_size,1);
% Define matrix to store the values after adding the delay
Data_BFSK = zeros(ensemble_size, Total_samples);
% Apply the delay to the Binary BFSK Symbol
for i = 1:ensemble_size
    Data_BFSK(i,:)= BFSK_symbols(i, Td(i)+1 : Total_samples + Td(i));
end
% Define auto correlation matrix
initial_Stat_Auto_corr = zeros(size(Data_BFSK));
for i = 1:Total_samples
        for j = 1:Total_samples
            % Select two columns for element-wise multiplication
            DOT_colmuns = conj(Data_BFSK(:, i)) .* Data_BFSK(:, j);
            % Perform element-wise multiplication
            if (j>=i)
                initial_Stat_Auto_corr(:,j-i+1) = initial_Stat_Auto_corr(:,j-i+1)+ DOT_colmuns ;
            end
        end
end
Stat_Auto_corr = zeros ( 1 , Total_samples);
for i = 1 :Total_samples
    Stat_Auto_corr (1 , i ) = sum (initial_Stat_Auto_corr( : , i ) )/(Total_samples*ensemble_size) ;
end
% Concatenate to get the final statistical autocorrelation
Stat_Auto_corr_full = cat (2, conj(fliplr(Stat_Auto_corr(2:end))), Stat_Auto_corr);

%Plotting PSD graph
delta = zeros(size(Stat_Auto_corr_full));
fs = 1 /sampling_time ;
M = -Total_samples+1 : Total_samples - 1;
f = M * fs * (Tb)/ (size(M,2))  ;  %Sampling frequency
delta(600) = 1/(4*Tb);  % Set delta function at desired frequency
delta(800) = 1/(4*Tb);
for H = 1 : length (f)
    delta(H) =abs( delta(H) + (4 * cos(pi * f(H))^2)/((pi)^2 *(4*(f(H))^2 -1)^2));
end
PSD = abs(fftshift(fft(Stat_Auto_corr_full)))/(4*Total_samples); %normalized for 2Eb
figure('Name','PSD for BFSK Scheme');
plot(f , delta , 'r');
```

```
hold on ;
plot(f , PSD ,'b' );
legend('theoritical PSD ' ,'Simulated PSD  ' );
hold off;
title("PSD vs Frequency");
xlabel("Normalized frequency , F*T_b ");
ylabel("Normalized PSD ,S_B(F)/2E_b ");
ylim([0,0.8]);
xlim([-2,2]);
grid on;

%% Useful Functions
% Mapper Block function
function [X_inphase_out , X_Quad_out] = Mapper ( Y_signal  , Schemes_used )
  X_inphase_out = zeros (length (Y_signal),1) ;
  X_Quad_out    = zeros (length (Y_signal),1) ;

  if Schemes_used == 1
     [X_inphase_out , X_Quad_out] = BPSK_Mapper ( Y_signal ) ;
  elseif Schemes_used == 2
     [X_inphase_out , X_Quad_out] = QPSK_Mapper ( Y_signal ) ;
  elseif Schemes_used == 3
     [X_inphase_out , X_Quad_out] = Eight_PSK_Mapper ( Y_signal ) ;
  elseif  Schemes_used==4
     [X_inphase_out , X_Quad_out] = QAM_Mapper ( Y_signal ) ;
  elseif Schemes_used==5
     [X_inphase_out , X_Quad_out] = BFSK_Mapper ( Y_signal ) ;
  else %Schemes_used==6
     [X_inphase_out , X_Quad_out] = QPSK_Binary_Mapper ( Y_signal ) ;
  end
end

% Demapper Block function
function data_Stream_Demapper = Demapper( Symbols_N , ref_symbol , No_Modulation_bits ,Schemes_used)
    XI_Correct = zeros (1,length (Symbols_N)) ;
    XQ_Correct = zeros (1,length (Symbols_N)) ;
    data_Demapper   = zeros (length (Symbols_N),No_Modulation_bits) ;
%     data_Stream_Demapper = zeros (1,length(Symbols_N)*No_Modulation_bits);
    % Get min distance
    for M = 1 : length (Symbols_N)
        [ value , min_Index ] = min (abs(Symbols_N(1,M) - ref_symbol ));
        XI_Correct(1,M) = real(ref_symbol(min_Index,1));
        XQ_Correct(1,M) = imag(ref_symbol(min_Index,1));
        % Convert to get the bits used
        if Schemes_used == 1 % BPSK
            data_Demapper(M) = (XI_Correct(M)+1)/2 ;
         elseif Schemes_used==2
            data_Demapper(M,1) = (XI_Correct(1,M)+1)/2 ;
            data_Demapper(M,2) = (XQ_Correct(1,M)+1)/2 ;
        elseif Schemes_used == 3 % 8PSK
            phase = pi/4 ;
            i_Angle_used = angle(XI_Correct(M)+ 1i.*XQ_Correct(M)) / phase ;
            if i_Angle_used < 0
               i_Angle_used=  i_Angle_used + 8 ;
            end
            dec_to_Binary = rem(floor(i_Angle_used * pow2(-No_Modulation_bits+1:+1:0)), 2);
            data_Demapper(M,:) = Gray_Generator (dec_to_Binary);

        elseif Schemes_used==4  % 16-QAM
            data_Demapper(M,1) = (sign(XI_Correct(M)) + 1) / 2 ;
            data_Demapper(M,2) = (3 - abs(XI_Correct(M)))  / 2 ;
            data_Demapper(M,3) = (sign(XQ_Correct(M)) + 1) / 2 ;
            data_Demapper(M,4) = (3 - abs(XQ_Correct(M)))  / 2 ;
```

```
        elseif Schemes_used==5  % BFSK
            if real(Symbols_N(M)) > imag(Symbols_N(M))
                data_Demapper(M)= 0 ;
            else
                data_Demapper(M)= 1 ;
            end
        else   % Schemes_used==6   QPSK with binary encoding
            if [XI_Correct(1,M) XQ_Correct(1,M)] == [1 1]
                data_Demapper(M,:) = [1 0] ;
            elseif [XI_Correct(1,M) XQ_Correct(1,M)] == [1 -1]
                data_Demapper(M,:) = [1 1] ;
            elseif [XI_Correct(1,M) XQ_Correct(1,M)] == [-1 1]
                data_Demapper(M,:) = [0 1] ;
            else
                data_Demapper(M,:) = [0 0] ;
            end
        end
    end
    % Convert reshaped matrix to a single column vector
     data_Stream_Demapper = reshape(data_Demapper',1, []);
end

% BPSK Mapping
function [X_inphase , X_Quad] = BPSK_Mapper ( Y_signal_BPSK )
    X_inphase = zeros (length (Y_signal_BPSK),1) ;
    X_Quad    = zeros (length (Y_signal_BPSK),1) ;
    X_inphase = 2*Y_signal_BPSK-1 ;
end

% QPSK Gray representation Mapping
function [X_inphase , X_Quad] = QPSK_Mapper ( Y_signal_QPSK )
    X_inphase = zeros (length (Y_signal_QPSK),1) ;
    X_Quad    = zeros (length (Y_signal_QPSK),1) ;

    X_inphase = 2*Y_signal_QPSK(:,1)-1 ;
    X_Quad    = 2*Y_signal_QPSK(:,2)-1 ;
end

% 8PSK
function [X_inphase , X_Quad] = Eight_PSK_Mapper ( Y_signal_eight_PSK )
     X_inphase = zeros (length (Y_signal_eight_PSK),1) ;
     X_Quad    = zeros (length (Y_signal_eight_PSK),1) ;
     Z_complex = zeros (length (Y_signal_eight_PSK),1) ;
     % transfer form binary data to the binary of i_angle
     Binary_bits (:,1) = Y_signal_eight_PSK(:,1);
     for i = 2 : 3
         % XOR operation with shifted version of the sequence to get Binary
         Binary_bits(:,i) = xor ( Binary_bits(:,i-1) , Y_signal_eight_PSK(:,i) );
     end
     i_Angle = Binary_bits * (pow2(2:-1:0)');

     for j = 1 : length (Z_complex)
         phase = pi/4 ;
         Z_complex (j) =  1* exp(i_Angle(j) * 1i * phase);
     end
     X_inphase = real (Z_complex);
     X_Quad    = imag (Z_complex);

end

% 16-QAM Mapping
function [X_inphase , X_Quad] = QAM_Mapper ( Y_signal_QAM )
```

```
 % [ Sign X_in , Value X_in ,Sign X_Quad , Value X_Quad ]
     X_inphase = zeros (length (Y_signal_QAM),1) ;
     X_Quad    = zeros (length (Y_signal_QAM),1) ;

     Mag_Inphase  = 3 - 2 * Y_signal_QAM(:,2) ;
     Sign_Inphase = 2 * Y_signal_QAM(:,1)-1 ;

     Mag_Quad  = 3 - 2 * Y_signal_QAM(:,4) ;
     Sign_Quad = 2*Y_signal_QAM(:,3)-1 ;

     X_inphase = Mag_Inphase .* Sign_Inphase  ;
     X_Quad =  Mag_Quad .* Sign_Quad ;
end

% BFSK Mapping
function [X_inphase , X_Quad] = BFSK_Mapper ( Y_signal_BFSK )
     X_inphase = zeros (length (Y_signal_BFSK),1) ;
     X_Quad    = zeros (length (Y_signal_BFSK),1) ;
     % when y_signal_BFSK = 0 ,mapper output = XI + j XQ = 1
     % while y_signal_BFSK = 1 ,  mapper output = XI + j XQ = j
     X_inphase =  1 - Y_signal_BFSK  ; % get the complemente of signal
     X_Quad = Y_signal_BFSK ;
end

% QPSK Binary Mapping
function [X_inphase , X_Quad] = QPSK_Binary_Mapper( Y_signal_QPSK_binary )
     X_inphase = zeros (length (Y_signal_QPSK_binary),1) ;
     X_Quad    = zeros (length (Y_signal_QPSK_binary),1) ;
 for k = 1 : length (Y_signal_QPSK_binary)
    if Y_signal_QPSK_binary(k,:) == [0 0]
        X_inphase(k) = -1 ;
        X_Quad(k)    = -1 ;
    elseif Y_signal_QPSK_binary(k,:)  == [0 1]
        X_inphase(k) = -1 ;
        X_Quad(k)    =  1 ;

    elseif Y_signal_QPSK_binary(k,:)  == [1 0]
        X_inphase(k) = 1 ;
        X_Quad(k)    = 1 ;
    else
        X_inphase(k) =  1 ;
        X_Quad(k)    = -1 ;
    end
 end
end

% Gray function Generator
function Gray_bits = Gray_Generator ( Y_signal_Binary )
     Gray_bits = zeros (size (Y_signal_Binary)) ;
     % Perform shift
     shiftedArray = [zeros(size (Y_signal_Binary,1),1), Y_signal_Binary(:,1:size (Y_signal_Binary,2) -1)];
     Gray_bits  = xor (Y_signal_Binary , shiftedArray) ;
end

% test Modulations
function Test_Modulation_Schemes()
disp("Test Modulation Schemes is running... ") ;
 for k = 1 : 6
    if k==1 % BPSK
      num_bits = 1;
    elseif k ==2 %Gray QPSK
      num_bits = 2;
    elseif k==3 % 8PSK
```

```
      num_bits = 3;
    elseif k==4 % 16-QAM
      num_bits = 4;
    elseif k==5 % BFSK
      num_bits = 1;
    else % K == 6 Binary QPSK
      num_bits = 2;
    end
    data_test = 0 : pow2(num_bits)-1 ;
    % Calculate the binary representation
    data_test_binary = rem(floor(data_test(:) * pow2(-num_bits+1:+1:0)), 2);
    % rem () take reminder of divsion
    % if num_bits = 4
    % data_test(:) convert from Row to column vectors
    % 15 * (2^-3 , 2^-2 , 2^-1 , 2^0 ) = ( 1 ,3 ,7 , 15 )
    % rem (( 1 ,3 ,7 , 15 ) / 2) = ( 1 ,1 ,1 ,1 )
    [X_inphase_Mapper , X_Quad_Mapper] = Mapper ( data_test_binary  , k );
    Symbols_constellation = X_inphase_Mapper + 1i.* X_Quad_Mapper;
    % Plot the complex number
    figure;
    hold on;
    for j = 1:length(Symbols_constellation)
        % Plot complex number
        plot(real(Symbols_constellation(j)), imag(Symbols_constellation(j)),'bo' );
        % Add text annotation for magnitude
        text(real(Symbols_constellation(j)), imag(Symbols_constellation(j)),...
        [' ' num2str(data_test_binary(j,1:num_bits))], 'VerticalAlignment',...
        'bottom' , 'FontSize', 10);
    end
    hold on;
    xlabel('Real Part');
    ylabel('Imaginary Part');
    ylim([-2,2]);
    xlim([-2,2]);
    grid on;
    if k == 1
        title('Constellation of BPSK Modulation ');
    elseif k==2
        title('Constellation of Gray QPSK Modulation ');
        % Define square vertices
        x = [1  1 -1 -1  1 ];
        y = [1 -1 -1  1  1 ];
        % Plot square
        plot(x, y, 'k', 'LineWidth', 0.5);
        axis equal;
    elseif k == 3
        title('Constellation of 8PSK Modulation ');
        % Define angles
        theta = linspace(0, 2*pi, 100); % 100 points around the circle
        % Define radius
        r = 1;
        % Calculate circle coordinates
        x = r * cos(theta);
        y = r * sin(theta);
        % Plot circle
        plot(x, y,'k','LineWidth', 0.5);
        axis equal;
    elseif k == 4
        title('Constellation of 16-QAM Modulation ');
        ylim([-4,4]);
        xlim([-4,4]);
    elseif k == 5
        title('Constellation of BFSK Modulation ');
```

```
    else
       title('Constellation of Binary QPSK Modulation ');
        % Define square vertices
        x = [1  1 -1 -1  1 ];
        y = [1 -1 -1  1  1 ];
        % Plot square
        plot(x, y, 'k', 'LineWidth', 0.5);
        axis equal;
    end
    hold off;
 end
 disp("Testing finshed... ") ;
end
```