Faculty of Enigeering

Cairo University

Digital Communications Project 3

Modulation Techniques

Presented for **ELC 3070** MATLAB Project

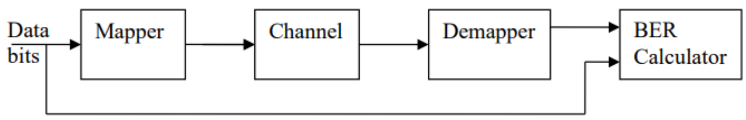
Presented to:

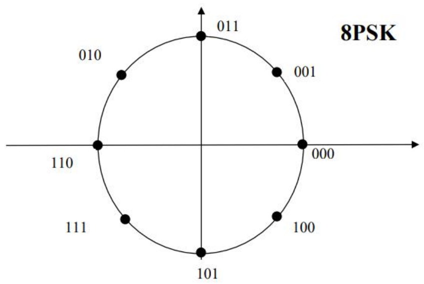
Dr. Mohamed Khairy

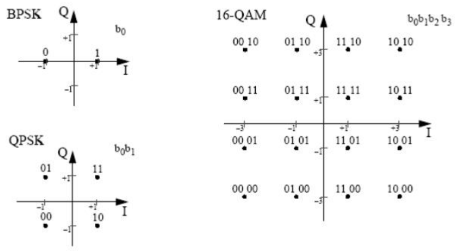
T.A: Mohamed Khaled

(3rd Year Electronics and Electrical Communication Engineers)	
مجدي أحمد عباس عبد الحميد الابرق	Sec: 3 / I.D: 9210899 / BN: 36
أحمد عادل يونس سيد	Sec: 1 / I.D: 9213073 / BN: 16

Role of each member:
Each one of us created his own code, and in one meeting we came together on the best version by merging the 2 codes and wrote the documentation







BPSK

Binary Phase Shift Keying

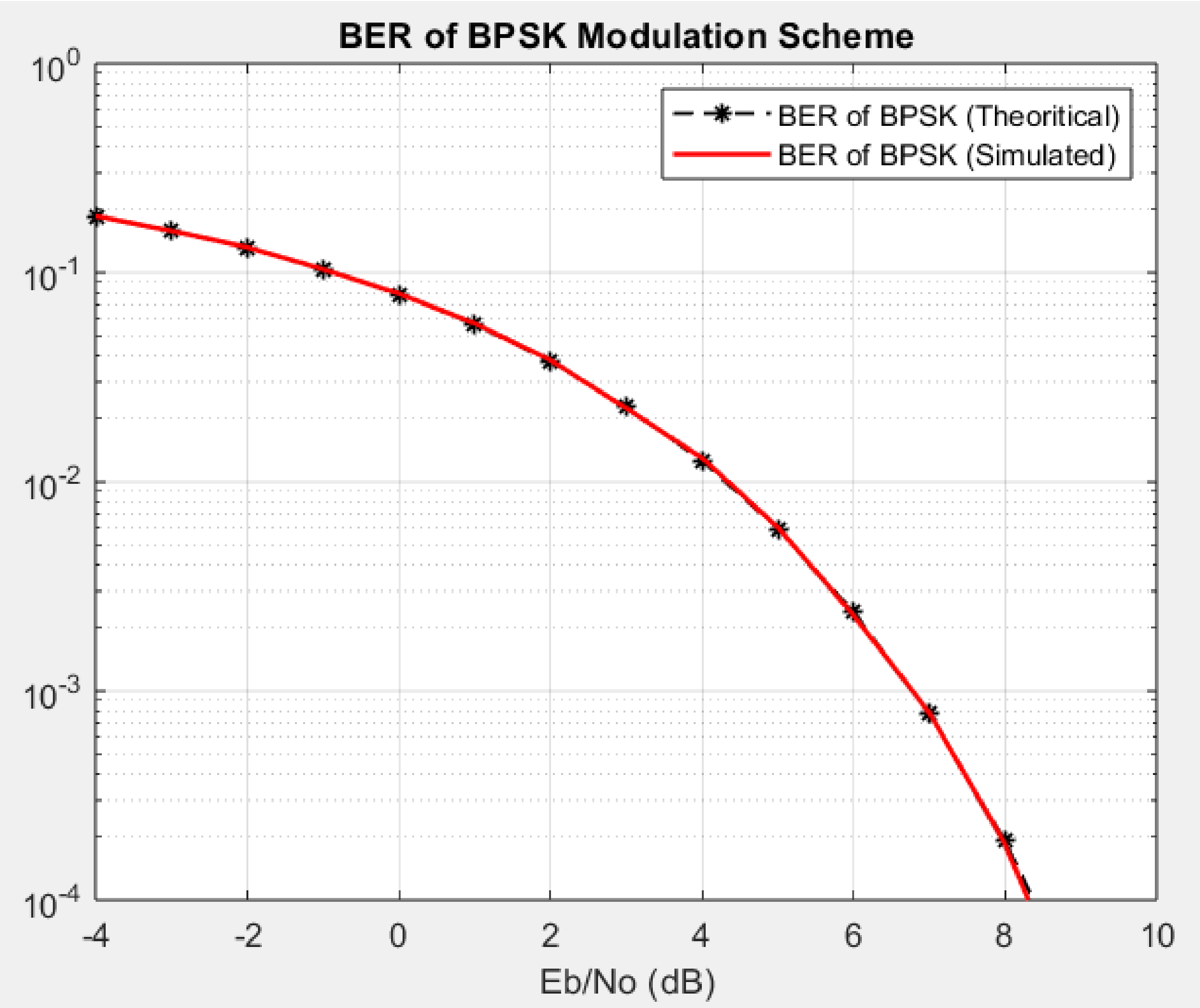


Figure 1: BER vs SNR for BPSK

Comments

As shown in Fig. (1) the simulated value of the BER is very close to the theoretical value which is

$$BER_{theoritical} = \frac{1}{2}\, erfc\left(\sqrt{\frac{E_b}{N_o}}\right)$$

QPSK

Quadrature Phase Shift Keying

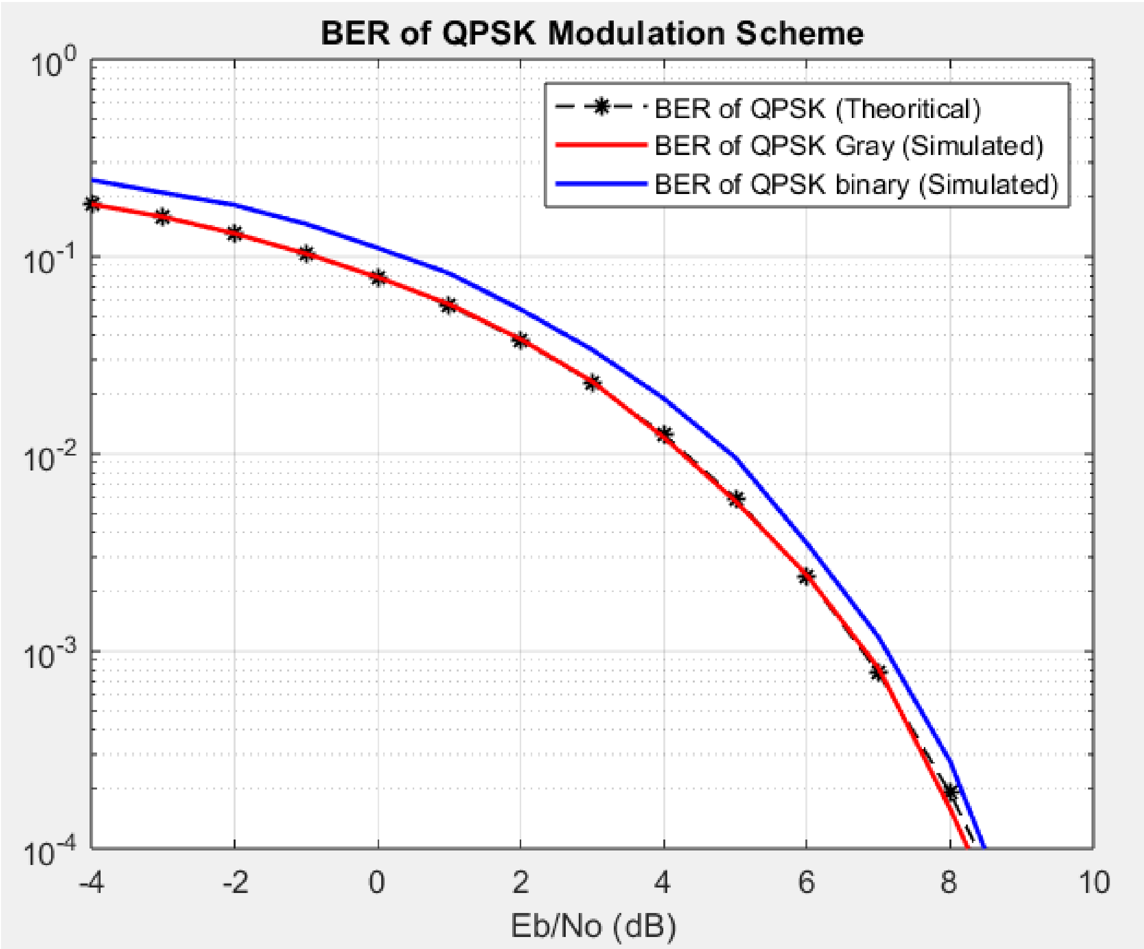


Figure 2: BER vs SNR for QPSK

Comments

The theoretical BER of QPSK is the same as BER of the BPSK which is

$$BER_{theoritical} = \frac{1}{2}\, erfc\left(\sqrt{\frac{E_b}{N_o}}\right)$$

As we notice from Fig. (2), in binary coded QPSK, the bit error rate tends to be higher compared to Gray coding. This outcome aligns with expectations since neighboring symbols in binary coding can differ by more than one bit, unlike gray coding where adjacent symbols vary by only a single bit. Consequently, Gray coding offers superior performance with lower bit error rates.

8 - PSK

8 - Phase Shift Keying

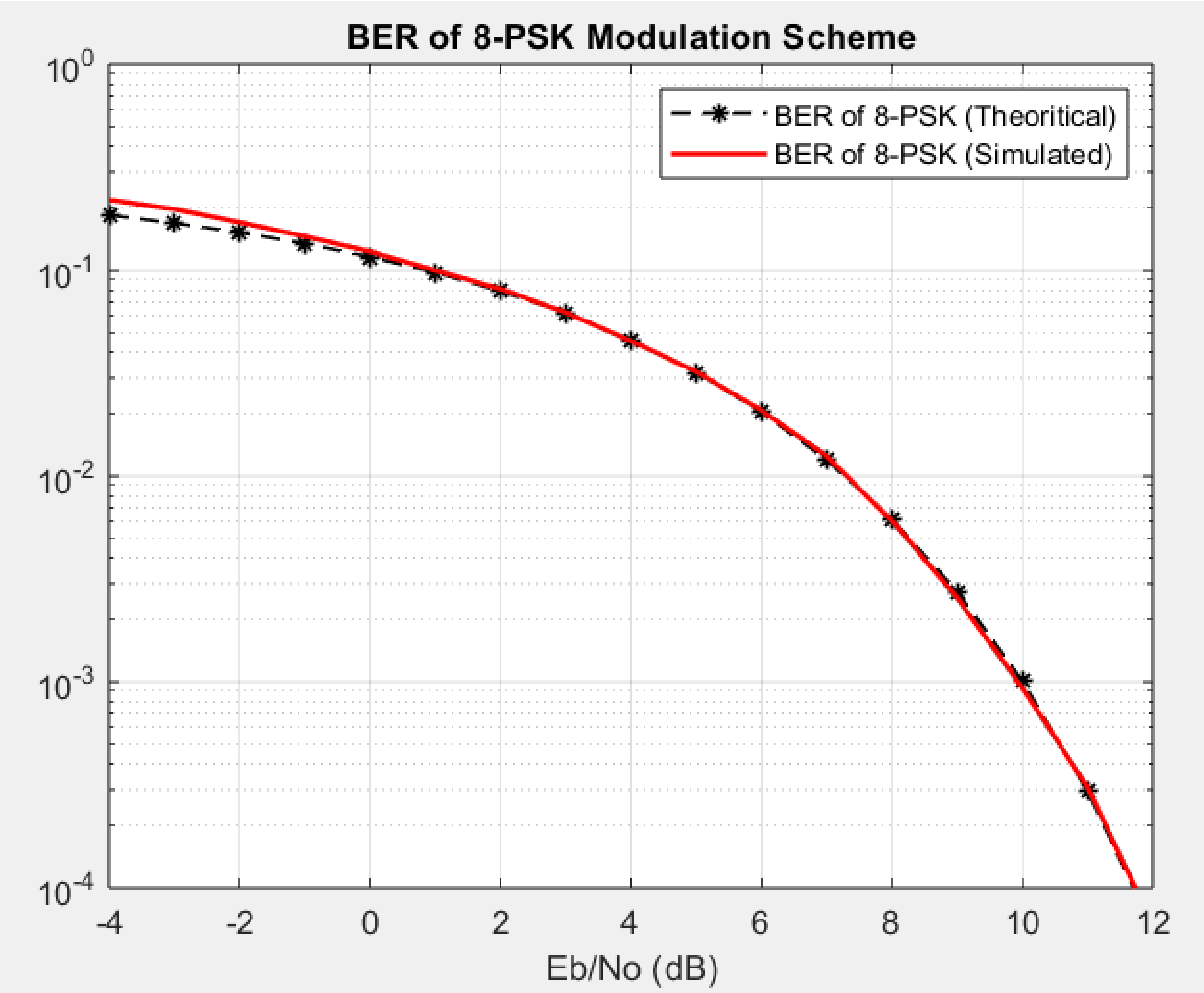


Figure 3: BER vs SNR for 8 - PSK.

Comments

From Fig. (3) we notice that 8-PSK has a higher BER compared to the BPSK & QPSK. Therefore, we can conclude that in General phase shift keying as the number of symbols increase (M) → the BER will also increase.

16 - QAM

16 - Quadrature Amplitude Modulation

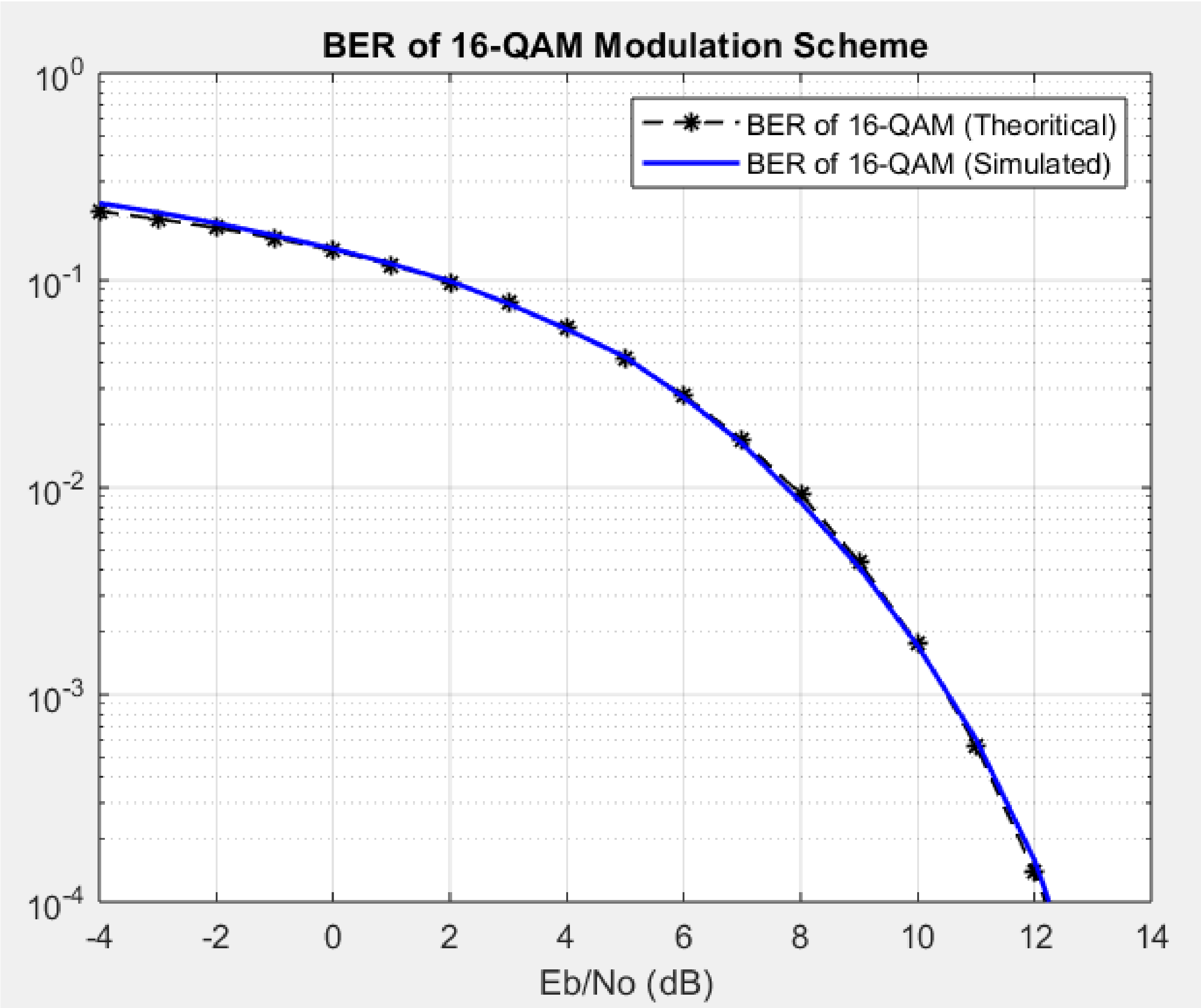


Figure 4: BER vs SNR for 16 - QAM.

BFSK

Binary Frequency Shift Keying

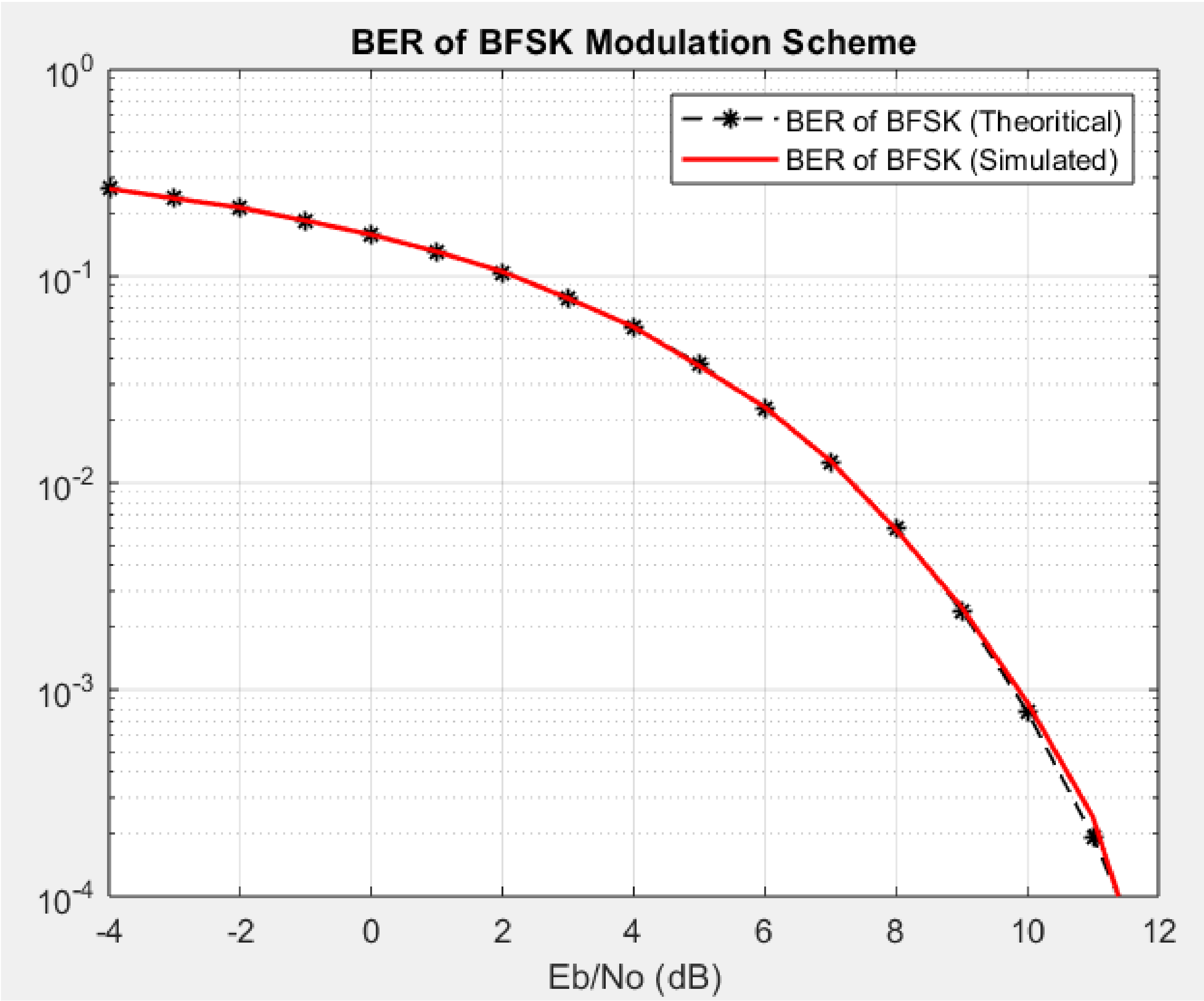


Figure 5: BER vs SNR for BFSK

Comments

The Basis Function of the BFSK is:

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

$$\emptyset_2(t) = \sqrt{\frac{2E_b}{T_b}}\cos(2\pi f_2 t) \quad \text{where } f_i = \frac{n_c+i}{T_b} \qquad (i = 1,2) \qquad (0 \le t \le T_b)$$

Note: n_c represent the (number of cycles -1) of the first symbol & $\Delta f = \frac{n}{2T_b}$

Where Δf represent the separation frequency, in most cases we deal with its minimum value $\left(\frac{1}{2T_b}\right)$

The Baseband Equivalent Signals of the BFSK:

$$s_{BB} = S_I + j \times S_Q$$

Where the S_I represent the in-phase component & S_Q represent the quadrature component

$$S_I = \sqrt{\frac{2E_b}{T_b}}\cos\left(\frac{\pi t}{T_b}\right) \qquad , \qquad S_Q = \mp\sqrt{\frac{2E_b}{T_b}}\sin\left(\frac{\pi t}{T_b}\right)$$

Therefore, in the base band we can represent the first symbol (which refer to zero) by

$S_{1_{BB}} = \sqrt{\frac{2E_b}{T_b}}$ and represent the second symbol (which refer to one) by

$$S_{2_{BB}} = \sqrt{\frac{2E_b}{T_b}}\left(\cos\left(\frac{2\pi t}{T_b}\right) + j\sin\left(\frac{2\pi t}{T_b}\right)\right)$$

PSD - BFSK

Power Spectral Density For Binary Phase Shift Keying

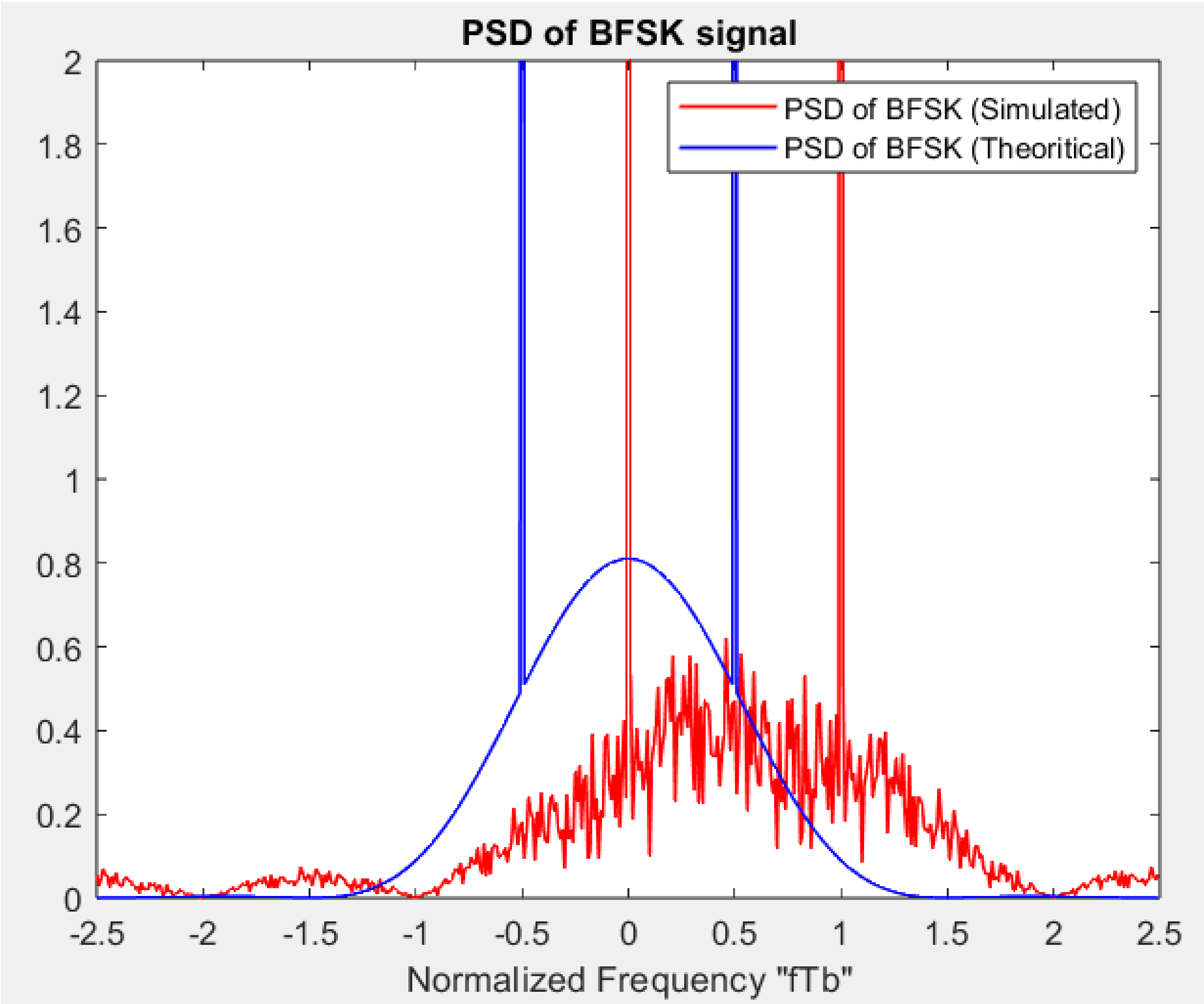


Figure 6: Power Spectral Density for BFSK

Comments

we calculate the simulated PSD from generating certain ensemble with random bits then mapping these bits to $(S_{1_{BB}} \& S_{2_{BB}})$. After that we calculate the PSD and compare this result with the Theoretical PSD

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

All in One

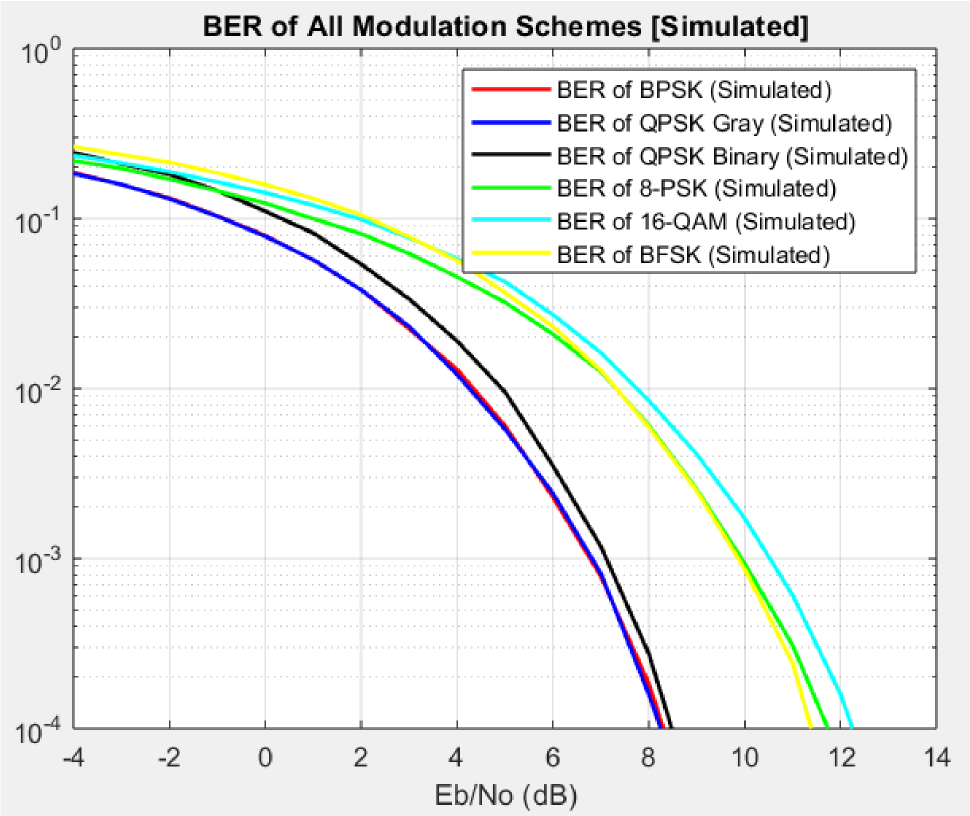


Figure 7: All Simulated BER vs SNR (For All Modulation Schemes)

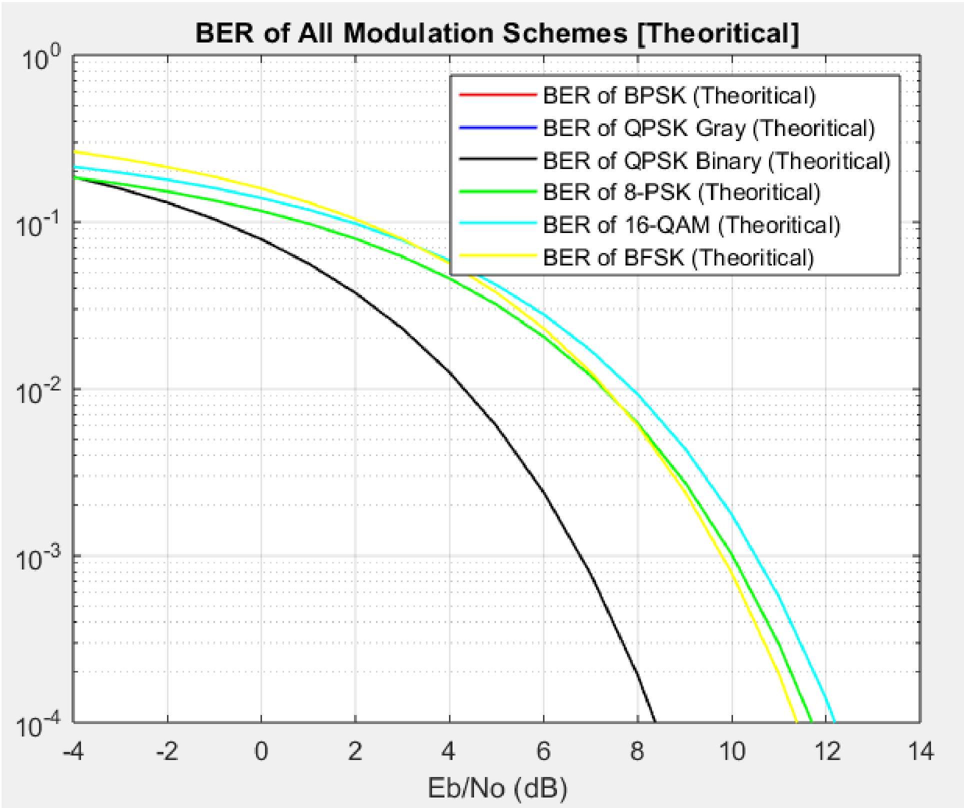


Figure 8: All Theoretical BER vs SNR (For All Modulation Schemes)

Full MATLAB Code

```
clc
clear
close all;

%% ---------- Generating the Random Data Sequence -------------- %%

% we can choose any value of number of bits under conditions that it can be divided into number of
bits in each symbol that is 1(for BPSK & BFSK) & 2(QPSK) & 3(8 - PSK) & 4(16 - QAM)
Number_of_Bits = 120000;
Random_Data = randi([0 1] , Number_of_Bits , 1);  % generate all the random bits
Eb = 1;    % Eb represent the bit eneregy
SNR_range_db = (-4: 15);
% where SNR_range = Eb/No that exactly represent (1/2) * SNR in DB values
SNR_range_linear = zeros(length(SNR_range_db),1);

%% --- General Mapper stage for BPSK & QPSK & 8 - PSK & 16 - QAM & BFSK ----- %%

% ------------ Declare binary phase shift keying (BPSK) ------------ %
BPSK = sqrt(Eb) * (2 * Random_Data - 1);

% -------Declare Quadrature phase shift keying (QPSK) in gray coding ---------%
QPSK_gray = zeros(Number_of_Bits/2,1);
for i = 1: 2: Number_of_Bits
  if(Random_Data(i) == 0 && Random_Data(i + 1) == 0 )
    QPSK_gray((i + 1)/2) = sqrt(Eb) * complex(-1, -1);
  elseif(Random_Data(i) == 0 && Random_Data(i + 1) == 1)
    QPSK_gray((i + 1)/2) = sqrt(Eb) * complex(-1,1);
  elseif(Random_Data(i) == 1 && Random_Data(i + 1) == 0)
    QPSK_gray((i + 1)/2) = sqrt(Eb) * complex(1, -1);
  elseif(Random_Data(i) == 1 && Random_Data(i + 1) == 1)
    QPSK_gray((i + 1)/2) = sqrt(Eb) * complex(1,1);
  end
end

% ----- Declare Quadrature phase shift keying (QPSK) in binary coding ---------%
QPSK_binary = zeros(Number_of_Bits/2,1);
for i = 1: 2: Number_of_Bits
  if(Random_Data(i) == 0 && Random_Data(i + 1) == 0 )
    QPSK_binary((i + 1)/2) = sqrt(Eb) * complex(-1, -1);
  elseif(Random_Data(i) == 0 && Random_Data(i + 1) == 1)
    QPSK_binary((i + 1)/2) = sqrt(Eb) * complex(-1,1);
  elseif(Random_Data(i) == 1 && Random_Data(i + 1) == 0)
    QPSK_binary((i + 1)/2) = sqrt(Eb) * complex(1,1);
  elseif(Random_Data(i) == 1 && Random_Data(i + 1) == 1)
    QPSK_binary((i + 1)/2) = sqrt(Eb) * complex(1, -1);
  end
end
```

```
% ------------- Declare 8 - PSK in Gray coding --------------%
MPSK = zeros(Number_of_Bits/3,1);
for i = 1:3:Number_of_Bits
  if(Random_Data(i) == 0 && Random_Data(i + 1) == 0 && Random_Data(i + 2) == 0)
    MPSK((i + 2)/3) = sqrt(3 * Eb) * complex(1,0);
% Here we multiply by factor sqrt(Es) and Es = 8 * Eb
  elseif(Random_Data(i) == 0 && Random_Data(i + 1) == 0 && Random_Data(i + 2) == 1)
    MPSK((i + 2)/3) = sqrt(1.5 * Eb) * complex(1,1);
% where we multiply by factor sqrt(Es/2) as this symbol exist at phase 45
  elseif(Random_Data(i) == 0 && Random_Data(i + 1) == 1 && Random_Data(i + 2) == 0)
    MPSK((i + 2)/3) = sqrt(1.5 * Eb) * complex(-1,1);
  elseif(Random_Data(i) == 0 && Random_Data(i + 1) == 1 && Random_Data(i + 2) == 1)
    MPSK((i + 2)/3) = sqrt(3 * Eb) * complex(0,1);
  elseif(Random_Data(i) == 1 && Random_Data(i + 1) == 0 && Random_Data(i + 2) == 0)
    MPSK((i + 2)/3) = sqrt(1.5 * Eb) * complex(1, -1);
  elseif(Random_Data(i) == 1 && Random_Data(i + 1) == 0 && Random_Data(i + 2) == 1)
    MPSK((i + 2)/3) = sqrt(3 * Eb) * complex(0, -1);
  elseif(Random_Data(i) == 1 && Random_Data(i + 1) == 1 && Random_Data(i + 2) == 0)
    MPSK((i + 2)/3) = sqrt(3 * Eb) * complex(-1,0);
  elseif(Random_Data(i) == 1 && Random_Data(i + 1) == 1 && Random_Data(i + 2) == 1)
    MPSK((i + 2)/3) = sqrt(1.5 * Eb) * complex(-1, -1);
  end
end
% -------------Declare 16 - QAM in Gray coding --------------- %
Eo = 1;    % Eo represent symbol energy
MQAM = zeros(Number_of_Bits/4,1);
real_MQAM = zeros(Number_of_Bits, 1);
img_MQAM = zeros(Number_of_Bits, 1);
for i = 1:4:Number_of_Bits
  if(Random_Data(i) == 0 && Random_Data(i + 1) == 0)
% First two bits control the real part of the MQAM signal
    real_MQAM((i + 3)/4) = -3;
  elseif(Random_Data(i) == 0 && Random_Data(i + 1) == 1)
    real_MQAM((i + 3)/4) = -1;
  elseif(Random_Data(i) == 1 && Random_Data(i + 1) == 1)
    real_MQAM((i + 3)/4) = 1;
  elseif(Random_Data(i) == 1 && Random_Data(i + 1) == 0)
    real_MQAM((i + 3)/4) = 3;
  end
  if(Random_Data(i + 2) == 0 && Random_Data(i + 3) == 0)
% Second two bits control the imaginary part of the MQAM signal
    img_MQAM((i + 3)/4) = -3;
  elseif(Random_Data(i + 2) == 0 && Random_Data(i + 3) == 1)
    img_MQAM((i + 3)/4) = -1;
  elseif(Random_Data(i + 2) == 1 && Random_Data(i + 3) == 1)
    img_MQAM((i + 3)/4) = 1;
  elseif(Random_Data(i + 2) == 1 && Random_Data(i + 3) == 0)
    img_MQAM((i + 3)/4) = 3;
  end
MQAM((i + 3)/4) = sqrt(Eo) * complex(real_MQAM((i + 3)/4), img_MQAM((i + 3)/4));
end
```

```
% − − − − − Declare Binary frequency shift Keying (BFSK) − − − − − − − − − − − − − − − − −%
BFSK = zeros(length(Random_Data),1);
for i = 1 : length(BFSK)
  if(Random_Data(i) == 0)
     BFSK(i) = sqrt(Eb) ∗ complex(1,0);
  else
     BFSK(i) = sqrt(Eb) ∗ complex(0,1);
  end
end
%% − − − − − −BPSK [Channel stage & DeMapper stage & BER calculation] − − − − − − − − −%%
 % − − − − − − − − − − − − BPSK Channel stage [Adding the Noise term] − − − − − − %
BPSK_recieved = zeros(length(BPSK),1);
BER_BPSK_Theoritical = zeros(length(SNR_range_db),1);
BER_BPSK_Simulated = zeros(length(SNR_range_db),1);
for i = 1 : length(SNR_range_db)
  SNR_range_linear = 10^(SNR_range_db(i)/10);
  No_BPSK = Eb / SNR_range_linear;
% No_BPSK signal specified to the noise generated for BPSK signal
  noise = sqrt(No_BPSK/2).∗ randn(length(BPSK),1);
%sqrt((No/2) ∗ Eb) refer to standard deviation − −−> sqrt(variance)
  BPSK_recieved = BPSK + noise;
% − − − − − − − − − − − − − − BPSK DeMapper stage − − − − − − − − − − − − − − − − − − − −%
BPSK_demapped = zeros(length(BPSK),1);
for j = 1 : length(BPSK)      % here I use the concept of the decision region as the threshold = 0
  if(BPSK_recieved(j) > 0)
    BPSK_demapped(j) = 1;
  else
    BPSK_demapped(j) = 0;
  end
end
% − − − − − − − − − − − − BER calculation for the BPSK Scheme − − − − − − − − − − %
error_bits_BPSK = 0;
  for j = 1 : length(Random_Data)
    if(BPSK_demapped(j) ~ = Random_Data(j))
      error_bits_BPSK = error_bits_BPSK + 1;
    end
  end
BER_BPSK_Theoritical(i, 1) = 0.5 ∗ erfc(sqrt(Eb/No_BPSK));
BER_BPSK_Simulated(i, 1) = error_bits_BPSK/Number_of_Bits;
end

%% − − − QPSK(Gray coding)[Channel stage & DeMapper stage & BER calculation] − − − %%

% − − − − − − QPSK Gray channel stage [Adding the noise term] − − − − − − − − − − − − − − −%
QPSK_gray_recieved = zeros(length(QPSK_gray),1);
BER_QPSK_gray_Theoritical = zeros(length(SNR_range_db),1);
BER_QPSK_gray_Simulated = zeros(length(SNR_range_db),1);
```

```
for i = 1 : length(SNR_range_db)
  SNR_range_linear = 10^(SNR_range_db(i)/10);
  No_QPSK_gray = Eb/SNR_range_linear;
% since the baseband signal has two components(in phase, quadrature)
                                    therefore the noise will also has two components
  real_noise = sqrt(No_QPSK_gray/2).* randn(length(QPSK_gray),1);
  img_noise = sqrt(No_QPSK_gray/2) .* randn(length(QPSK_gray),1);
  QPSK_gray_recieved = QPSK_gray + complex(real_noise, img_noise);
% --------QPSK (Gray) DeMapper stage ---------------------%
  QPSK_gray_demapped = zeros(length(QPSK_gray),1);
  for j = 1: length(QPSK_gray)    % loop specified for the gray coding
    if(real(QPSK_gray_recieved(j)) > 0 && imag(QPSK_gray_recieved(j)) > 0)
      QPSK_gray_demapped(j) = 3;
    elseif(real(QPSK_gray_recieved(j)) > 0 && imag(QPSK_gray_recieved(j)) < 0)
      QPSK_gray_demapped(j) = 2;
    elseif(real(QPSK_gray_recieved(j)) < 0 && imag(QPSK_gray_recieved(j)) > 0)
      QPSK_gray_demapped(j) = 1;
    elseif(real(QPSK_gray_recieved(j)) < 0 && imag(QPSK_gray_recieved(j)) < 0)
      QPSK_gray_demapped(j) = 0;
    end
  end
QPSK_gray_demapped = de2bi(QPSK_gray_demapped, 2, 'left - msb');
% de2bi() function convert the decimal values into equivalent binary values
% ---------- BER calculation for QPSK (Gray coding) ------------- %
error_bits_QPSK_gray = 0;
inc_var = 1; % this variable specified for incrementing the index of Random_Data variable in the loop
  for j = 1 : length(Random_Data)/2
    for k = 1 : 2
      if(QPSK_gray_demapped(j, k) ~= Random_Data(inc_var))
        error_bits_QPSK_gray = error_bits_QPSK_gray + 1;
      end
      inc_var = inc_var + 1;
    end
  end
BER_QPSK_gray_Theoritical(i, 1) = 0.5 * erfc(sqrt(Eb/No_QPSK_gray));
BER_QPSK_gray_Simulated(i, 1) = error_bits_QPSK_gray/Number_of_Bits;
end

%% --- QPSK(Binary coding)[Channel stage & DeMapper stage & BER calculation] ----%%

% ---- QPSK Binary channel stage [Adding the noise term] -------------%
QPSK_binary_recieved = zeros(length(QPSK_binary),1);
BER_QPSK_binary_Theoritical = zeros(length(SNR_range_db),1);
BER_QPSK_binary_Simulated = zeros(length(SNR_range_db),1);
for i = 1 : length(SNR_range_db)
  SNR_range_linear = 10^(SNR_range_db(i)/10);
  No_QPSK_binary = Eb/SNR_range_linear;
  real_noise = sqrt(No_QPSK_binary/2).* randn(length(QPSK_binary),1);
  img_noise = sqrt(No_QPSK_binary/2) .* randn(length(QPSK_binary),1);
  QPSK_binary_recieved = QPSK_binary + complex(real_noise, img_noise);
```

```
% ---------- QPSK (Binary) DeMapper stage -----------------%
  QPSK_binary_demapped = zeros(length(QPSK_binary),1);
  for j = 1:length(QPSK_binary)    % loop specified for the binary coding
    if(real(QPSK_binary_recieved(j)) > 0 && imag(QPSK_binary_recieved(j)) > 0)
      QPSK_binary_demapped(j) = 2;
    elseif(real(QPSK_binary_recieved(j)) > 0 && imag(QPSK_binary_recieved(j)) < 0)
      QPSK_binary_demapped(j) = 3;
    elseif(real(QPSK_binary_recieved(j)) < 0 && imag(QPSK_binary_recieved(j)) > 0)
      QPSK_binary_demapped(j) = 1;
    else
      QPSK_binary_demapped(j) = 0;
    end
  end
QPSK_binary_demapped = de2bi(QPSK_binary_demapped,2,'left-msb');
% ---------- BER calculation for QPSK (Binary coding) ------------ %
error_bits_QPSK_binary = 0;
  inc_var = 1;
  for j = 1 : length(Random_Data)/2
    for k = 1 : 2
      if(QPSK_binary_demapped(j,k) ~= Random_Data(inc_var))
        error_bits_QPSK_binary = error_bits_QPSK_binary + 1;
      end
      inc_var = inc_var + 1;
    end
  end
BER_QPSK_binary_Theoritical(i,1) = 0.5 * erfc(sqrt(Eb/No_QPSK_binary));
BER_QPSK_binary_Simulated(i,1) = error_bits_QPSK_binary/Number_of_Bits;
End
%% --8-PSK [Channel stage & DeMapper stage & BER calculation] ---------%%
% -------8-PSK channel stage [Adding the noise term] --------------- %
MPSK_recieved = zeros(length(MPSK),1);
BER_MPSK_Theoritical = zeros(length(SNR_range_db),1);
BER_MPSK_Simulated = zeros(length(SNR_range_db),1);
for i = 1 : length(SNR_range_db)
  SNR_range_linear = 10^(SNR_range_db(i)/10);
  No_MPSK = Eb/SNR_range_linear;
% since the baseband signal has two components(in phase, quadrature)therefore the noise will also has two components
  real_noise = sqrt(No_MPSK/2).* randn(length(MPSK),1);
% we divide by 6 (2 * 3) as the Es = 3 * Eb
  img_noise = sqrt(No_MPSK/2) .* randn(length(MPSK),1);
  MPSK_recieved = MPSK + complex(real_noise,img_noise);
% -------- DeMapper for 8-PSK Scheme ------------------------%
MPSK_demapped = zeros(length(MPSK),1);
MPSK_table = sqrt(3 * Eb) * [ complex(1,0); (1/sqrt(2)) * complex(1,1); (1/sqrt(2))
             * complex(-1,1); complex(0,1); (1/sqrt(2))
             * complex(1,-1); complex(0,-1); complex(-1,0); (1/sqrt(2)) * complex(-1,-1) ];
    for j = 1 : length(MPSK)
      [~,Min_index] = min(abs(MPSK_recieved(j) - MPSK_table));
      MPSK_demapped(j) = Min_index - 1;
    end
MPSK_demapped = de2bi(MPSK_demapped,3,'left-msb');
```

```
% − − − − − − − − − − −BER calculation for 8 − PSK scheme − − − − − − − − − − − − − − − %
error_bits_MPSK = 0;
  inc_var = 1;
  for j = 1 : length(Random_Data)/3
    for k = 1 : 3
      if(MPSK_demapped(j, k) ~ = Random_Data(inc_var))
        error_bits_MPSK = error_bits_MPSK + 1;
      end
      inc_var = inc_var + 1;
    end
  end
BER_MPSK_Theoritical(i, 1) = (1/3) ∗ erfc(sqrt( (3 ∗ Eb)/No_MPSK ) ∗ sind(180/8) );
BER_MPSK_Simulated(i, 1) = error_bits_MPSK/Number_of_Bits;
end
%% − − − − − 16 − QAM [Channel stage & DeMapper stage & BER calculation] − − − − − %%
% − − − − − − − − 16 − QAM channel stage [Adding the noise term] − − − − − − − − − − − − −%
MQAM_recieved = zeros(length(MQAM),1);
BER_MQAM_Theoritical = zeros(length(SNR_range_db),1);
BER_MQAM_Simulated = zeros(length(SNR_range_db),1);
for i = 1 : length(SNR_range_db)
  SNR_range_linear = 10^(SNR_range_db(i)/10);
  No_MQAM = Eb/SNR_range_linear;
% since the baseband signal has two components(in phase, quadrature) therefore the noise will also has two components
  real_noise = sqrt((No_MQAM/2) ∗ 2.5).∗ randn(length(MQAM),1);
% As in QAM the Eb = 2.5 Eo
  img_noise = sqrt((No_MQAM/2) ∗ 2.5) .∗ randn(length(MQAM),1);
  MQAM_recieved = MQAM + complex(real_noise, img_noise);
% − − − − − − − − − − DeMapper for 16 − QAM Scheme − − − − − − − − − − − − − − %
MQAM_demapped = zeros(length(MQAM),1);
% In this table we will store all the 16 sybmols for the QAM
 MQAM_table = sqrt(Eo) ∗ [complex(−3, −3); complex(−3, −1); complex(−3,3); complex(−3,1); complex(−1, −3);
 complex(−1, −1); complex(−1,3); complex(−1,1); complex(3, −3); complex(3, −1); complex(3,3); complex(3,1);
                       complex(1, −3); complex(1, −1); complex(1,3); complex(1,1)];
for j = 1 : length(MQAM)
  [~ , Min_index] = min(abs(MQAM_recieved(j) − MQAM_table));
  MQAM_demapped(j) = Min_index − 1;
end
MQAM_demapped = de2bi(MQAM_demapped, 4, 'left − msb');
% − − − − − − − − BER calculation for 16 − QAM scheme − − − − − − − − − − − − − − − − − −%
 error_bits_MQAM = 0;
  inc_var = 1;
  for j = 1 : length(Random_Data)/4
    for k = 1 : 4
      if(MQAM_demapped(j, k) ~ = Random_Data(inc_var))
        error_bits_MQAM = error_bits_MQAM + 1;
      end
      inc_var = inc_var + 1;
    end
  end
BER_MQAM_Theoritical(i, 1) = (1.5/4) ∗ erfc( sqrt(Eb/(2.5 ∗ No_MQAM)) );
BER_MQAM_Simulated(i, 1) = error_bits_MQAM/Number_of_Bits;
end
```

```
%% ---- BFSK [Channel stage & DeMapper stage & BER calculation] ---------- %%
% -------- BFSK channel stage [Adding the noise term] -----------%
BFSK_recieved = zeros(length(BFSK),1);
BER_BFSK_Theoritical = zeros(length(SNR_range_db),1);
BER_BFSK_Simulated = zeros(length(SNR_range_db),1);
for i = 1 : length(SNR_range_db)
  SNR_range_linear = 10^(SNR_range_db(i)/10);
  No_BFSK = Eb/SNR_range_linear;
% since the baseband signal has two components(in phase, quadrature) therefore the noise will also has two components
  real_noise = sqrt(No_BFSK/2).* randn(length(BFSK),1);
  img_noise = sqrt(No_BFSK/2) .* randn(length(BFSK),1);
  BFSK_recieved = BFSK + complex(real_noise, img_noise);
% ------------- BFSK DeMapper stage ---------------------%
BFSK_demapped = zeros(length(BFSK),1);
BFSK_table = sqrt(Eb) * [complex(1,0); complex(0,1)];
for j = 1 : length(BFSK)
  [~ , Min_index] = min(abs(BFSK_recieved(j) - BFSK_table));
  BFSK_demapped(j) = Min_index - 1;
end
% ------------ BER calculation for BFSK scheme ---------------- %
 error_bits_BFSK = 0;
  for j = 1 : length(Random_Data)
    if(BFSK_demapped(j) ~= Random_Data(j))
      error_bits_BFSK = error_bits_BFSK + 1;
    end
  end
BER_BFSK_Theoritical(i, 1) = 0.5 * erfc(sqrt(Eb/(2 * No_BFSK)));
BER_BFSK_Simulated(i, 1) = error_bits_BFSK/Number_of_Bits;
end
%% ------- Plotting the BER vs EB/No for different Schemes ------------ %%
% ---------- plotting the BER for BPSK ---------------%
figure;
semilogy(SNR_range_db , BER_BPSK_Theoritical, 'k *-- ', 'linewidth', 1);
hold on;
semilogy(SNR_range_db , BER_BPSK_Simulated , 'r', 'linewidth', 1.5);
hold off;
title('BER of BPSK Modulation Scheme');
xlabel('Eb/No (dB)');
ylim([10^(-4) 10^(0)]);
grid on;
legend('BER of BPSK (Theoritical)', 'BER of BPSK (Simulated)', 'Location', 'NorthEast')
% ------- plotting the BER for QPSK (Gray & Binary) --------------%
figure;
semilogy(SNR_range_db , BER_QPSK_gray_Theoritical, 'k *-- ', 'linewidth', 1);
hold on;
semilogy(SNR_range_db , BER_QPSK_gray_Simulated , 'r', 'linewidth', 1.5);
hold on;
semilogy(SNR_range_db , BER_QPSK_binary_Simulated , 'b', 'linewidth', 1.5);
hold off;
title('BER of QPSK Modulation Scheme');
xlabel('Eb/No (dB)');
```

```
ylim([10^(−4) 10^(0)]);
grid on;
legend('BER of QPSK (Theoritical)','BER of QPSK Gray (Simulated)','BER of QPSK binary (Simulated)','Location','NorthEast')
% −−−−−−−−−− plotting the BER for 8 − PSK −−−−−−−−−−−−−−−%
figure;
semilogy(SNR_range_db , BER_MPSK_Theoritical, 'k ∗ −− ', 'linewidth', 1);
hold on;
semilogy(SNR_range_db , BER_MPSK_Simulated , 'r', 'linewidth', 1.5);
hold off;
title('BER of 8 − PSK Modulation Scheme');
xlabel('Eb/No (dB)');
ylim([10^(−4) 10^(0)]);
grid on;
legend('BER of 8 − PSK (Theoritical)', 'BER of 8 − PSK (Simulated)', 'Location', 'NorthEast')
% −−−−−−−−−−−−− plotting the BER for 16 − QAM −−−−−−−−−−−−−−−%
figure;
semilogy(SNR_range_db , BER_MQAM_Theoritical, 'k ∗ −− ', 'linewidth', 1);
hold on;
semilogy(SNR_range_db , BER_MQAM_Simulated , 'b', 'linewidth', 1.5);
hold off;
title('BER of 16 − QAM Modulation Scheme');
xlabel('Eb/No (dB)');
ylim([10^(−4) 10^(0)]);
grid on;
legend('BER of 16 − QAM (Theoritical)', 'BER of 16 − QAM (Simulated)', 'Location', 'NorthEast')
% −−−−−−−−−−−−− plotting the BER for BFSK −−−−−−−−−−−−−−−−−−−− %
figure;
semilogy(SNR_range_db , BER_BFSK_Theoritical, 'k ∗ −− ', 'linewidth', 1);
hold on;
semilogy(SNR_range_db , BER_BFSK_Simulated , 'r', 'linewidth', 1.5);
hold off;
title('BER of BFSK Modulation Scheme');
xlabel('Eb/No (dB)');
ylim([10^(−4) 10^(0)]);
grid on;
legend('BER of BFSK (Theoritical)', 'BER of BFSK (Simulated)', 'Location', 'NorthEast')
% −−−−−−General Plot for All Modulation Schemes [simulated] −−−−−−−−−−− %
figure;
semilogy(SNR_range_db , BER_BPSK_Simulated, 'r', 'linewidth', 1.5);
hold on;
semilogy(SNR_range_db , BER_QPSK_gray_Simulated , 'b', 'linewidth', 1.5);
hold on;
semilogy(SNR_range_db , BER_QPSK_binary_Simulated , 'k', 'linewidth', 1.5);
hold on;
semilogy(SNR_range_db , BER_MPSK_Simulated , 'g', 'linewidth', 1.5);
hold on;
semilogy(SNR_range_db , BER_MQAM_Simulated , 'c', 'linewidth', 1.5);
hold on;
semilogy(SNR_range_db , BER_BFSK_Simulated , 'y', 'linewidth', 1.5);
hold off;
title('BER of All Modulation Schemes [Simulated]');
```

```
xlabel('Eb/No (dB)');
ylim([10^(-4) 10^(0)]);
grid on;
legend('BER of BPSK (Simulated)','BER of QPSK Gray (Simulated)', 'BER of QPSK Binary (Simulated)'
,' BER of 8 - PSK (Simulated)',' BER of 16 - QAM (Simulated)','BER of BFSK (Simulated)','Location','NorthEast')
% ------- General Plot for All Modulation Schemes [Theoritical] ----------%
figure;
semilogy(SNR_range_db , BER_BPSK_Theoritical, 'r', 'linewidth', 1);
hold on;
semilogy(SNR_range_db , BER_QPSK_gray_Theoritical , 'b', 'linewidth', 1);
hold on;
semilogy(SNR_range_db , BER_QPSK_binary_Theoritical , 'k', 'linewidth', 1);
hold on;
semilogy(SNR_range_db , BER_MPSK_Theoritical , 'g', 'linewidth', 1);
hold on;
semilogy(SNR_range_db , BER_MQAM_Theoritical , 'c', 'linewidth', 1);
hold on;
semilogy(SNR_range_db , BER_BFSK_Theoritical , 'y', 'linewidth', 1);
hold off;
title('BER of All Modulation Schemes [Theoritical]');
xlabel('Eb/No (dB)');
ylim([10^(-4) 10^(0)]);
grid on;
legend('BER of BPSK (Theoritical)','BER of QPSK Gray (Theoritical)','BER of QPSK Binary (Theoritical)',
'BER of 8 - PSK (Theoritical)','BER of 16 - QAM (Theoritical)','BER of BFSK (Theoritical)','Location','NorthEast')
%% --- Calculating the power spectral density (PSD) of BFSK Scheme --------%%
% ------Generate random data (Ensemble) ---------------%
Number_of_bits = 100;
Number_of_realizations = 1000;
Number_of_samples = 5;
ensemble_data = randi([0, 1], Number_of_realizations, Number_of_bits );
ensemble_data_rep = repelem(ensemble_data, 1, Number_of_samples);
% repeat each element in the ensemble
Fs = 100;                  % Sampling frequency (Hz)
Ts = 1/Fs;                 % Sampling Period
Tb = Ts * Number_of_samples;          % Bit duration period
t = 0: Ts: Tb - 0.01;
TX_data = zeros(Number_of_realizations, Number_of_bits * Number_of_samples);
% ---- Mapping the ensemble data into BFSK baseband symbols ---------- %
S1 = sqrt((2 * Eb)/Tb);    % this represent the first BFSK symbol that represent 0
S2 = sqrt((2 * Eb)/Tb) * complex(cos((2 * pi/Tb) * t), sin((2 * pi/Tb) * t));
 % this represent the second BFSK symbol that represent 1
count = 1;
for i = 1 : size(TX_data, 1)
  for j = 1 : size(TX_data, 2)
    if(ensemble_data_rep(i, j) == 0)
      TX_data(i, j) = S1;
    else
      if(count > Number_of_samples)
        count = 1;
      end
      TX_data(i, j) = S2(count);
```

```
      count = count + 1;
    end
  end
end
% − − − − − − − −  Calculate the statistical Autocorrelation Function  − − − − − −%
stat_ACF  =  zeros(1, Number_of_bits ∗ Number_of_samples);  % Initialize array for autocorrelation
N  =  length(stat_ACF);
for tau  =  (−N/2 + 1) :  N/2
  stat_ACF(tau  +  N/2) =  sum( conj(TX_data(: , N/2)) .∗  TX_data(: , N/2  +  tau) )/ Number_of_realizations;
end
% − − − − − − − Calculate the PSD of the BFSK (Simulated & Theoritical)  − − − − − −%
PSD_BFSK_simulated  =  abs(fftshift(fft(stat_ACF)));
freq_range  =  (−N/2: N/2 − 1)  ∗  (Fs/N);             % frequency range before Normalization
freq_range_norm  =  (−N/2: N/2 − 1)  ∗  (Fs/N)  ∗ Tb;       % frequency range after Normalization

delta1  =  abs(freq_range  −  0.5 / Tb)  <  0.01;
% This is an approximate method to produce impulse signal
delta2  =  abs(freq_range  +  0.5 / Tb)  <  0.01;
PSD_Theoritical  =  (2 ∗ Eb/Tb)  ∗  (delta1  +  delta2)  +
    (8 ∗ Eb ∗ cos(pi  ∗  Tb  ∗  freq_range). ^2) ./ (pi^2  ∗  (4  ∗  Tb^2  ∗  freq_range. ^2  −  1). ^2);
figure;
plot(freq_range_norm , ( PSD_BFSK_simulated/Fs )/(2 ∗ Eb) , 'r', 'linewidth', 1);
% we divide by 2Eb for normalization
hold on;
plot(freq_range_norm , PSD_Theoritical, 'b', 'linewidth', 1)
title('PSD of BFSK signal');
xlabel('Normalized Frequency "fTb"');
legend ('PSD of BFSK (Simulated)', 'PSD of BFSK (Theoritical)')
ylim ([0 2])
xlim([−2.5 2.5])
```

The END Thank You