Faculty of Engineering – Cairo University

Electronics And Electrical Communication Department

Third year-Mainstream

Digital Communications Project #3

Submitted by:

ID	Sec	BN	Name
9210352	2	7	حسام صابر محمود سيد
9211418	4	47	يوسف عصام ابو بكر محمد

Table of Contents

Role of Each Member:

- Both Hossam and Yousef worked separately on the coding part, looking for different methods in terms of technical variety, optimization, and code readability to implement the various functions and logic in attempt to find the best implementations in terms of the mentioned specifications. The technical writing of the report was then divided into the sections outlined by the Table of Contents which were distributed between the two team members.
- *Hossam:* Sections (II, III, IV, VI, and XI)
- *Yousef:* Sections (V, VII, VIII, IX, and X)

Table of Figures

I) Introduction

Modulation is one of the basic and most important processes performed in Communication Systems. It is the process of injecting the intended information “which is the message to be transmitted” inside a carrier generated with a certain uplink frequency. Previously, we studied modulation in the Analog domain and outlined different modulation techniques such as Amplitude Modulation (AM), Phase Modulation (PM) and Frequency Modulation (FM). In the Digital domain, our concerns are shifted towards analyzing the performance of our communication systems by studying the probability of error (BER). We are going to apply the basic concepts studied in Digital Communication Systems related to Signal Space Analysis to study different modulation techniques and their performances.

II) Single Carrier System Overview

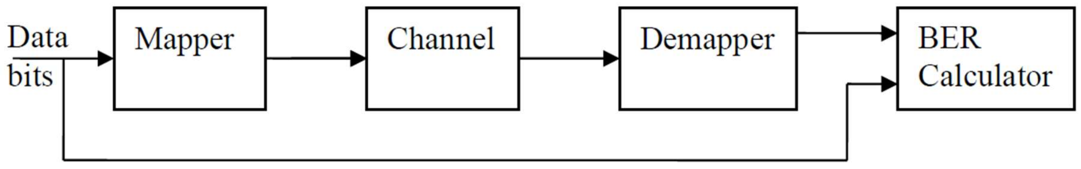

Figure 1: Single Carrier Communication System

A basic communication system that performs digital modulation, mainly consists of three components which are derived from the signal space analysis to perform different modulation techniques.

These basic blocks are the Mapper, the Channel, and the Demapper.

The Mapper receives the raw binary data and converts each group of bits into corresponding symbols according to the signal space constellation of the used modulation technique.

A typical Channel has multiple effects on the transmitted message. We are concerned with a single effect thus far, which is the superposition of random Gaussian Noise over the transmitted signal which is the main source of error in data transmission in our study.

The Demapper receives the transmitted signal after noise superposition, and performs the decision to map the received symbols to a certain signal space constellation point depending on the agreed upon modulation scheme. The estimated symbols are then demapped to binary data.

After that, we are going to compare the received binary data with the original raw transmitted data to obtain the number of bits which are received incorrectly to estimate the BER in this modulation scheme under the given SNR and encoding conditions.

In the next sections, we are going to explain how the different modulation techniques are implemented and show how each of these blocks are implemented till the BER is simulated for the different modulation schemes.

III) Modulation Schemes Implementation

There are four Modulation schemes which are BPSK, QPSK, 8PSK and 16QAM

- BPSK

This scheme has only one basis function

$$\phi_1 = \sqrt{\frac{2}{T}}\cos(2\pi f_c t), \qquad 0 < t < T_B$$

and represent each bit by one of two symbols.

$$S_1 = \sqrt{E_B} * \phi_1, \qquad S_2 = -\sqrt{E_B} * \phi_1$$

- QPSK

This scheme has two basis function

$$\phi_1 = \sqrt{\frac{2}{T}}\cos(2\pi f_c t),\ \phi_2 = \sqrt{\frac{2}{T}}\sin(2\pi f_c t), \qquad 0 < t < T$$

and represent each two bits by one of four symbols.

$$S_i = \sqrt{E} * \cos\left((2i-1) * \frac{\pi}{4}\right)\phi_1 - \sqrt{E} * \sin\left((2i-1) * \frac{\pi}{4}\right)\phi_2, \qquad i = 1{:}\,4$$

- 8PSK

This scheme has two basis function

$$\phi_1 = \sqrt{\frac{2}{T}}\cos(2\pi f_c t),\ \phi_2 = \sqrt{\frac{2}{T}}\sin(2\pi f_c t), \qquad 0 < t < T$$

and represent each three bits by one of eight symbols.

$$S_i = \sqrt{E} * \cos\left((i-1) * \frac{\pi}{4}\right)\phi_1 - \sqrt{E} * \sin\left((i-1) * \frac{\pi}{4}\right)\phi_2, \qquad i = 1{:}\,8$$

- 16-QAM

This scheme has two basis function

$$\phi_1 = \sqrt{\frac{2}{T}}\cos(2\pi f_c t),\ \phi_2 = \sqrt{\frac{2}{T}}\sin(2\pi f_c t)\ \ 0 < t < T$$

and represent each four bits by one of sixteen symbols.

$$S_i = \sqrt{E} * a_i * \phi_1 - \sqrt{E} * \mathrm{b_i} * \phi_2, \qquad a_i, b_i = \pm 1, \pm 3, i = 1{:}\,16$$

IV) Mapper Logic Implementation

Mapper is used to map bits to symbols based on modulation scheme where setup is in base band equivalent, also mapping is done using grey code to minimize BER.

Each group of bits are converted to string to be converted to equivalent decimal then mapped to suitable symbol in constellation table where symbols in constellation table are arranged based on grey representation.

```
function transmittedMessage = mapper(rawMessage, modulationScheme)
    N_Bits = length(rawMessage);
    bitsPerSymbol = log2(length(modulationScheme));

    transmittedMessage = zeros(1, N_Bits / bitsPerSymbol);
    for j = 1:N_Bits/bitsPerSymbol
        startBit = bitsPerSymbol * (j - 1) + 1;
        endBit = bitsPerSymbol * j;

        % An efficient binary to decimal conversion
        index = bin2dec(string(char(rawMessage(startBit:endBit) + '0'))) + 1;
        transmittedMessage(j) = modulationScheme(index);
    end
end
```

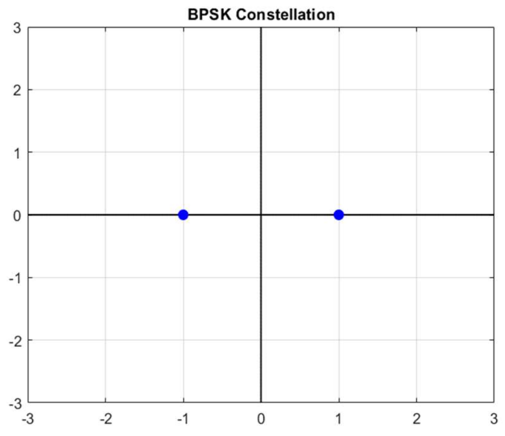

Figure 2: A polar plot of the BPSK constellation table

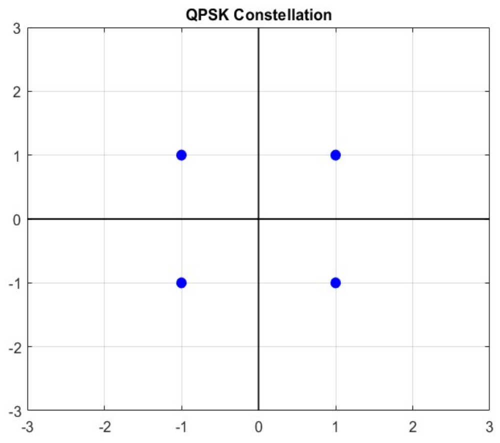

Figure 3: A polar plot of the QPSK constellation table

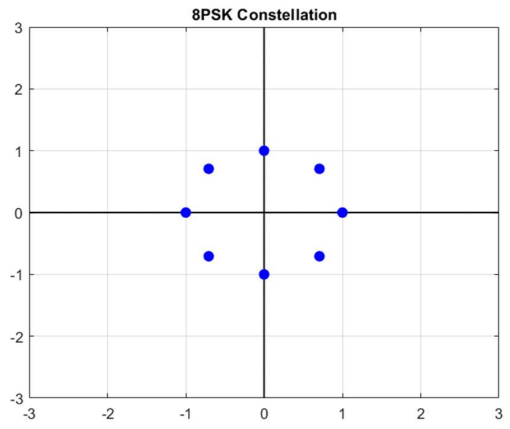

Figure 4: A polar plot of the 8PSK constellation table

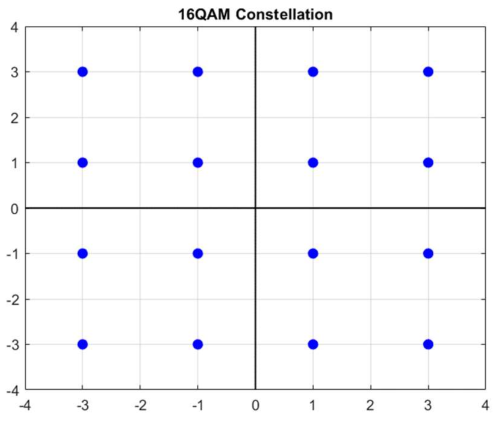

Figure 5: A polar plot of the 16-QAM constellation table

- Mapping Tables

Bits	Decimal	Symbol
BPSK		
0	0	1
1	1	-1
QPSK (Grey)		
00	0	$-1-i$
01	1	$-1+i$
10	2	$1-i$
11	3	$1+i$
QPSK (Not Grey)		
00	0	$-1-i$
01	1	$-1+i$
10	2	$1+i$
11	3	$1-i$
8PSK		
000	0	1
001	1	$\frac{1}{\sqrt{2}}+\frac{i}{\sqrt{2}}$
010	2	$-\frac{1}{\sqrt{2}}+\frac{i}{\sqrt{2}}$
011	3	i
100	4	$\frac{1}{\sqrt{2}}-\frac{i}{\sqrt{2}}$
101	5	$-1i$
110	6	-1
111	7	$-\frac{1}{\sqrt{2}}-\frac{i}{\sqrt{2}}$
16QAM		
0000	0	$-3-3j$
0001	1	$-3-1j$
0010	2	$-3+3j$
0011	3	$-3+1j$
0100	4	$-1-3j$
0101	5	$-1-1j$
0110	6	$-1+3j$
0111	7	$-1+1j$
1000	8	$3-3j$
1001	9	$3-1j$
1010	10	$3+3j$
1011	11	$3+1j$
1100	12	$1-3j$
1101	13	$1-1j$
1110	14	$1+3j$
1111	15	$1+1j$

Modulation Tables are implemented in our code using arrays as follows:

```
M_BPSK_Table = [-1, 1];
M_QPSK_Grey_Table = [-1-1i, -1+1i, 1-1i, 1+1i];
M_8PSK_Table = [
    1, 1/sqrt(2)+1i/sqrt(2), -1/sqrt(2)+1i/sqrt(2), 1j,...
    1/sqrt(2)-1i/sqrt(2), -1i, -1, -1/sqrt(2)-1i/sqrt(2)
];
M_16QAM_Table = [
    -3-3j, -3-1j, -3+3j, -3+1j,...
    -1-3j, -1-1j, -1+3j, -1+1j,...
    3-3j, 3-1j, 3+3j, 3+1j,...
    1-3j, 1-1j, 1+3j, 1+1j
];
```

This method of implementation of the mapper functionality and the modulation schemes makes the logic of the entire script unified. If we needed to add another modulation scheme “as we will see later”, it will just be simply implemented by adding a modulation scheme table `M_Scheme_Table` as an array in the script and using the same function for mapping the binary data into symbols.

V) Demapper Logic Implementation

The Demapper block provides us with a decision on the received signal set symbols and based on this decision, the decided symbols are retranslated to binary data.

The decision is made based on the maximum likelihood rule which reduces to the following form: The received symbol $\boldsymbol{x}$ is decided as $\boldsymbol{s_i}$ if $l(m_i) = |\boldsymbol{x} - \boldsymbol{s_i}|^2$ is minimum.

This means that a received symbol is decided by calculating the Euclidean distance between the received signal point and all other signal set points and the signal set point with the minimum Euclidean distance is the decided symbol.

The maximum likelihood rule is a unified method that can be used in the decision regardless of the implemented modulation scheme. For this reason, the Demapper Block can be implemented as a unified function which we are going to call `demapper` that is not specific to the type of modulation used. As mentioned previously, the type of modulation is separately implemented using modulation scheme tables which are used by the `demapper` function to demap the received messages. The `demapper` function is provided with the received message and the modulation scheme table of the intended modulation method. It obtains the number of bits per symbol to which the received symbols will be demapped from the size of the modulation scheme table "which is the size of the signal space of the modulation technique".

The function calculates the distance between each of the elements of the modulation scheme table and the each symbol in the received message, and gets the index of the symbol in the table with minimum distance. This index is then converted to binary to convert the symbol to binary data using our implementation of the decimal to binary converter called `myDecimalToBinaryVector` which is optimized to perform decimal to binary conversion.

The following code snippets show our implementation of the `demapper` function:

```
function estimatedMessage = demapper(receivedMessage, modulationScheme)
    bitsPerSymbol = log2(length(modulationScheme));
    N_Bits = bitsPerSymbol * length(receivedMessage);

    estimatedMessage = zeros(N_Bits / bitsPerSymbol, bitsPerSymbol);
    for k = 1:N_Bits/bitsPerSymbol
        % Apply maximum likelihood rule
        % Find the symbol with minimum Euclidean distance
        [~, index] = min(abs(receivedMessage(k) - modulationScheme));
        estimatedBits = myDecimalToBinaryVector(index - 1, bitsPerSymbol);
        estimatedMessage(k,:) = estimatedBits;
    end

    estimatedMessage = reshape(estimatedMessage', [1, N_Bits]);
end
```

```
function binaryVector = myDecimalToBinaryVector(decimal, numBits)
    % An efficient and optimized decimal to binary conversion
    binaryVector = dec2bin(decimal) - '0';
    binaryVector = [zeros(1, numBits - length(binaryVector)) binaryVector];
end
```

Our `demapper` function was tested with the `mapper` function "in absence of noise" to validate that the `demapper` function behaves correctly by demapping the symbols correctly to the original bit stream and was found out to work as expected.

VI) Channel Noise Addition

Channel is modeled by AWGN $\sim \mathrm{N}\left(0, \frac{N_o}{2}\right)$ added on each bit but symbols are used so noise per bit should be converted to noise per symbol

$$N_S = \left(\frac{E_b * n_b}{N_o * E_{avg}}\right)^{-1}$$

Where $\frac{E_b}{N_o}$, n_b and E_{avg} are SNR given, number of bits in each scheme and average energy of transmitted symbols respectively, where this noise is added to both real and imaginary parts.

Noise is generated using `randn` function

$$\mu + \sigma * randn()$$

N_s is adjusted depending on the simulated value for $\frac{E_b}{N_0}$. We are going to simulate noise for $\frac{E_b}{N_0}$ lying in the range of -4:12 dB with step 1 dB.

```
EBoverN0_dB = -4:1:12;
EBoverN0_linear = 10 .^ (EBoverN0_dB / 10);
```

The channel noise N_s is adjusted by depending on the current simulated value of $\frac{E_b}{N_0}$ and the average energy of the constellation. The noise is then added by creating a complex number vector whose real and imaginary parts are both normally distributed $\sim N\left(0, \frac{N_s}{2}\right)$ and adding this vector to the mapped message.

```
bitsPerSymbol = log2(length(modulationScheme));
Eavg = mean(abs(modulationScheme) .^ 2);      % Get average energy of constellation
Ns = Eavg ./ (bitsPerSymbol * EBoverN0_linear);

NI = sqrt(Ns(m) / 2) * randn(1, N_Bits / bitsPerSymbol);
NQ = sqrt(Ns(m) / 2) * randn(1, N_Bits / bitsPerSymbol);
noise = NI + 1i * NQ;
receivedMessage = transmittedMessage + noise;                  % Add Channel Noise
```

We then plot the signal point of the received message over the signal space constellation for different constellation techniques to visualize the effect of different values of the SNR $\frac{E_b}{N_0}$ on the decision:

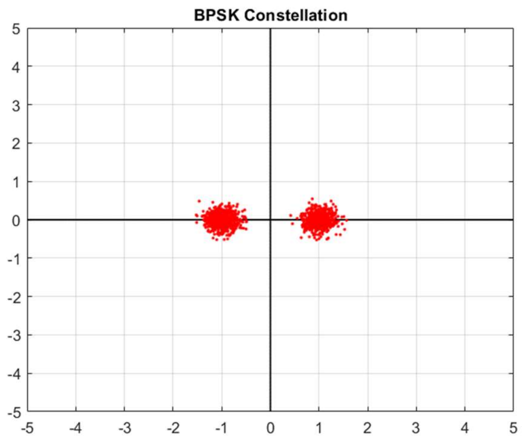

Figure 6: Noise Addition with SNR = 12 dB on BPSK

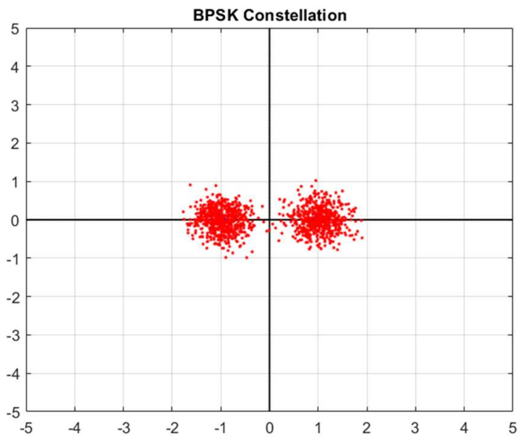

Figure 7: Noise Addition with SNR = 7 dB on BPSK

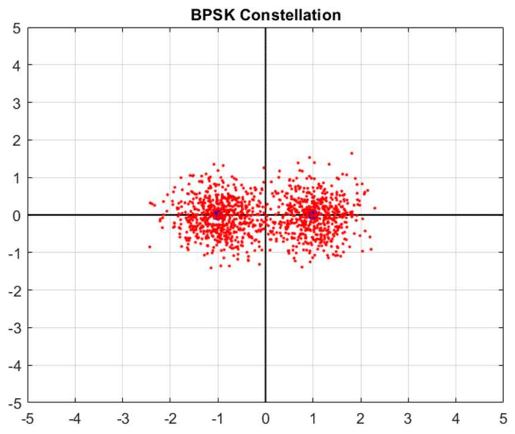

Figure 8: Noise Addition with SNR = 3 dB on BPSK

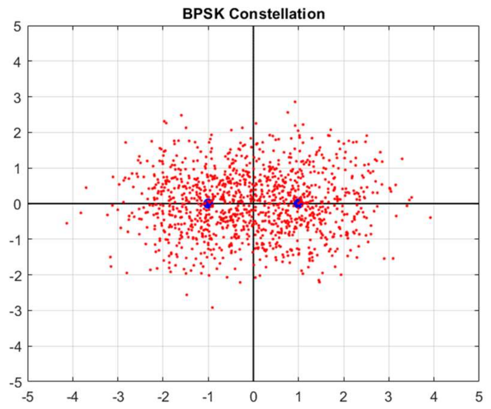

Figure 9: Noise Addition with SNR = -2 dB on BPSK

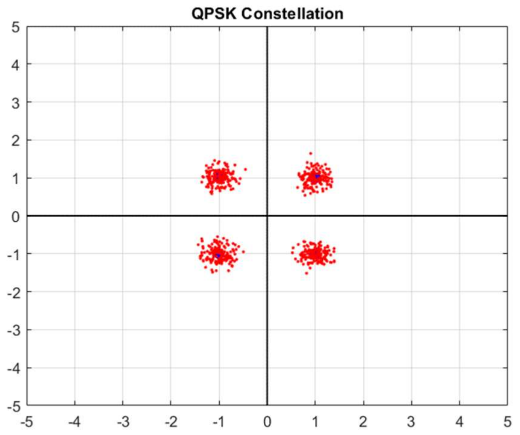

Figure 10: Noise Addition with SNR = 12 dB on QPSK

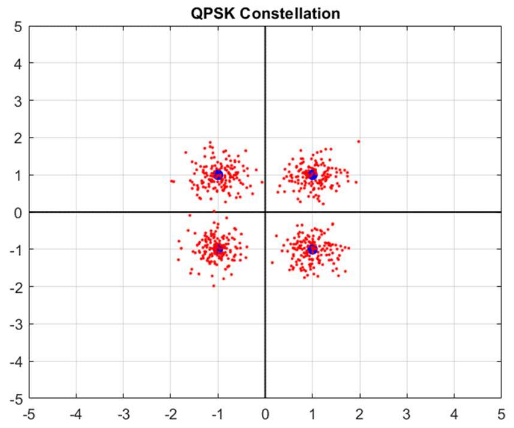

Figure 11: Noise Addition with SNR = 7 dB on QPSK

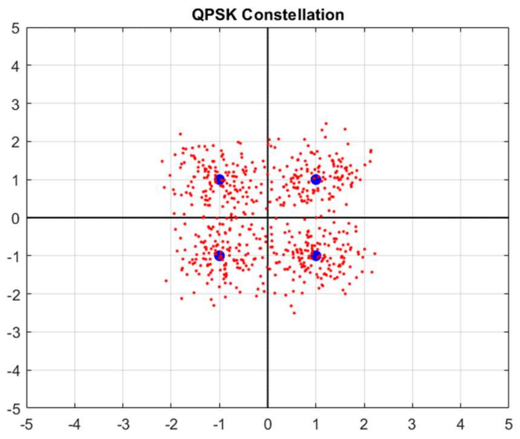

Figure 12: Noise Addition with SNR = 3 dB on QPSK

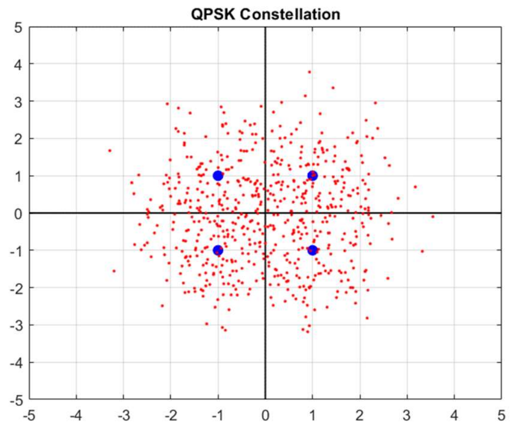

Figure 13: Noise Addition with SNR = -2 dB on QPSK

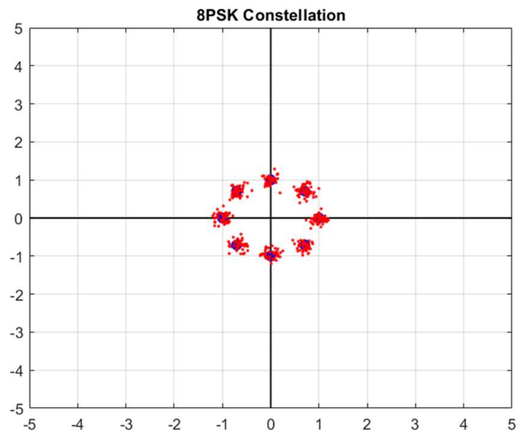

Figure 14: Noise Addition with SNR = 12 dB on 8PSK

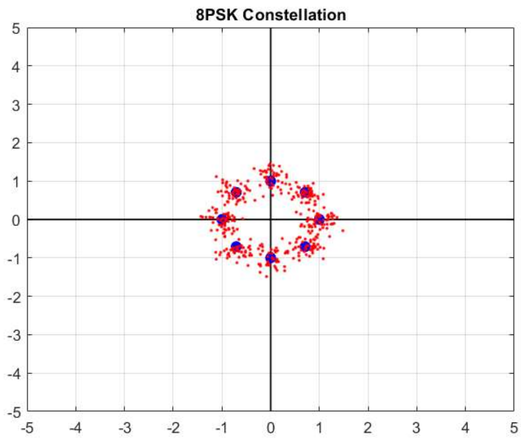

Figure 15: Noise Addition with SNR = 7 dB on 8PSK

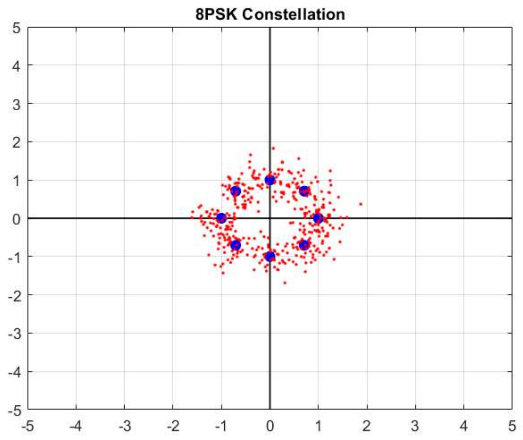

Figure 16: Noise Addition with SNR = 3 dB on 8PSK

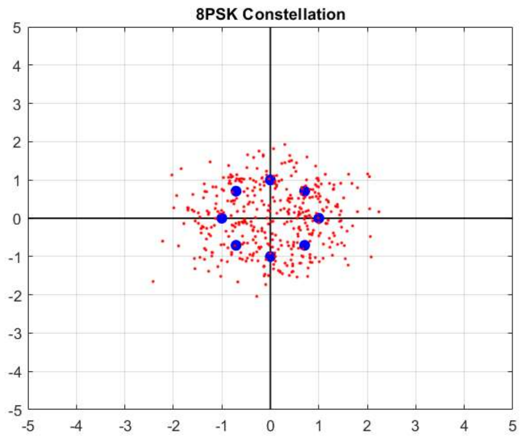

Figure 17: Noise Addition with SNR = -2 dB on 8PSK

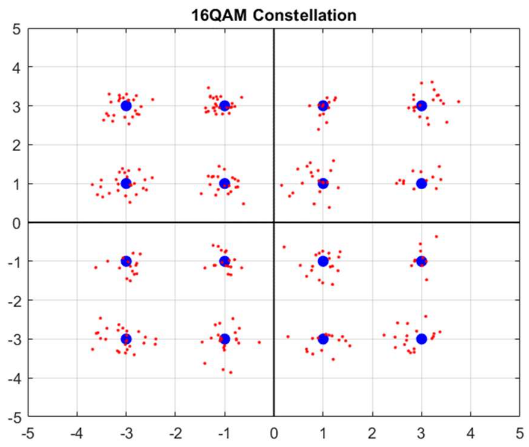

Figure 18: Noise Addition with SNR = 12 dB on 16-QAM

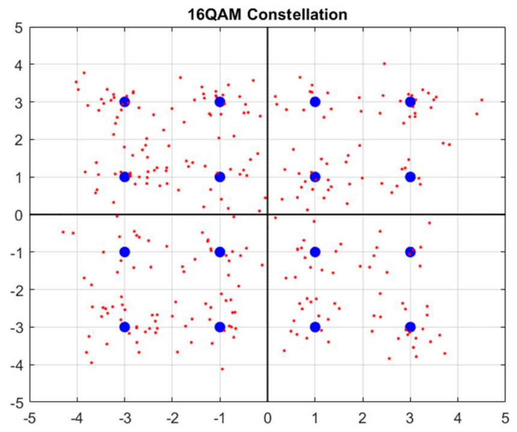

Figure 19: Noise Addition with SNR = 7 dB on 16-QAM

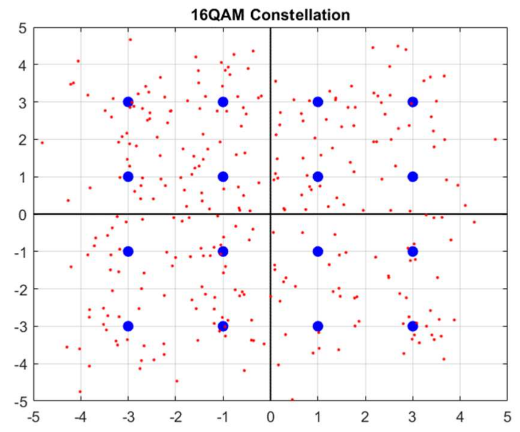

Figure 20: Noise Addition with SNR = 3 dB on 16-QAM

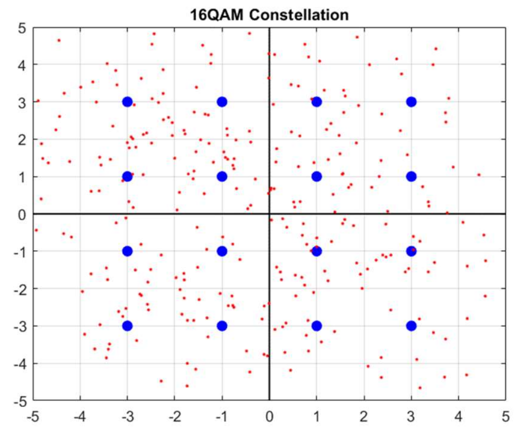

Figure 21: Noise Addition with SNR = -2 dB on 16-QAM

It is obvious that noise affects the location of sent symbols on constellation and from the plots we can estimate how good the BER for each scheme based on how good the symbols are well separated where BPSK<QPSK<8PSK16QAM<BFSK.

VII) BER simulation for BPSK, QPSK, 8PSK and 16-QAM

Next, we are going to simulate the BER of the four modulation schemes discussed in the previous sections which are BPSK, QPSK, 8PSK and 16-QAM.

Before this, we are going to outline the theoretical BER of the four schemes.

- BPSK:

For BPSK, the theoretical BER = $\frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_0}}\right)$

This can be obtained by performing integration on the decision boundaries which reduces to the closed form written in terms of the complementary error function and the SNR $\frac{E_b}{N_0}$.

- QPSK:

For QPSK, the theoretical BER = $\frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_0}}\right)$

Since the QPSK is Grey-Encoded, the integration on the decision boundaries can make use of the independence obtained from the Grey Encoding, which makes the integration reduce to the same form obtained in the BFSK.

- 8PSK:

In general, performing the integration on the decision boundaries for general MPSK modulation schemes is not an easy task and usually can not be written in a closed form. So either this integration can be evaluated using MATLAB or in better method, we are going to use the tight union bounds. For a general MPSK scheme, a tight union bound can be obtained in the form: $\frac{1}{\log_2 M} erfc\left(\sin\left(\frac{\pi}{M}\right)\sqrt{\log_2 M \times \frac{E_b}{N_0}}\right)$.

For 8PSK, the theoretical BER can be approximated as $\approx \frac{1}{3} erfc\left(\sin\left(\frac{\pi}{8}\right)\sqrt{\frac{3E_b}{N_0}}\right)$

- 16-QAM:

For 16-QAM, an exact closed form expression for the BER is not easy to obtain. However, in a similar manner done for the general MPSK schemes, an approximation for the theoretical BER for the 16-QAM can be obtained = $\frac{3}{8} erfc\left(\sqrt{\frac{E_b}{2.5N_0}}\right)$

BER simulation for the four schemes are done in the same manner with the same logic and the same functions for mapping, channel noise addition and demapping.

Therefore the four modulation schemes tables are combined in a cell array called `modulationSchemesTables` in which the script is set up for running in a loop that performs the same mapping and demapping steps till the BER is obtained.

Also the expressions outlined in this section for the theoretical BER are combined in another cell array corresponding to the modulation schemes in the `modulationSchemesTable` cell array. Another cell array for the actual BER is initialized which is filled inside the loop.

```
modulationSchemes = {
    M_BPSK_Table, M_QPSK_Grey_Table,...
    M_8PSK_Table, M_16QAM_Table
};
BER_theoritical = {
    0.5*erfc(sqrt(EBoverN0_linear)),...
    0.5*erfc(sqrt(EBoverN0_linear)),...
    erfc(sin(pi/8) * sqrt(3 * EBoverN0_linear)) / 3,...
    0.375*erfc(sqrt(0.4*EBoverN0_linear))
};
BER_actual = cell(1, length(modulationSchemes));
```

The loop basically performs the three steps that are explained in detail in the previous sections which are the mapping, channel noise addition and the demapping. For every value for $\frac{E_b}{N_0}$ from (-4:12) dB, the BER is calculated for the final received message after decision.

```
% The BER simulation for all schemes goes in a loop
for i = 1:length(modulationSchemes)
    modulationScheme = modulationSchemes{i};         % Choose Scheme
    M_Name = modulationSchemesNames{i};              % Get scheme name
    bitsPerSymbol = log2(length(modulationScheme)); % Obtain number of bits per symbol
    Eavg = mean(abs(modulationScheme) .^ 2);         % Obtain Average Energy of Constellation
    N0 = Eavg ./ (bitsPerSymbol * EBoverN0_linear); % Adjust Noise according to SNR and Eavg

    % Generate Baseband Equivalent Signal from Raw Data Using Modulation Scheme
    transmittedMessage = mapper(rawMessage, modulationScheme);

    % Initialize BER vector
    BER_actual{i} = zeros(1, length(EBoverN0_dB));

    for m = 1:length(EBoverN0_dB)
        % Create Channel Noise
        NI = sqrt(N0(m) / 2) * randn(1, N_Bits / bitsPerSymbol);
        NQ = sqrt(N0(m) / 2) * randn(1, N_Bits / bitsPerSymbol);
        noise = NI + 1i * NQ;
        receivedMessage = transmittedMessage + noise;               % Add Channel Noise

        % Decision Block
        estimatedMessage = demapper(receivedMessage, modulationScheme);
        BER_actual{i}(m) = length(find(rawMessage ~= estimatedMessage));
    end
    BER_actual{i} = BER_actual{i} / N_Bits;

end
```

For each modulation scheme, the actual BER is plotted with the theoretical BER overlaid on the same plot as shown in (Figures 22:25). The simulated BER for the four modulation schemes are plotted overlaid on the same plot as shown in (Figure 26).

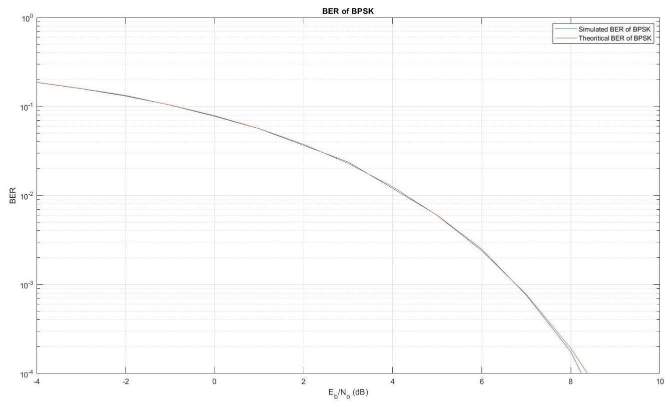

Figure 22: Simulated vs Theoretical BER for BPSK

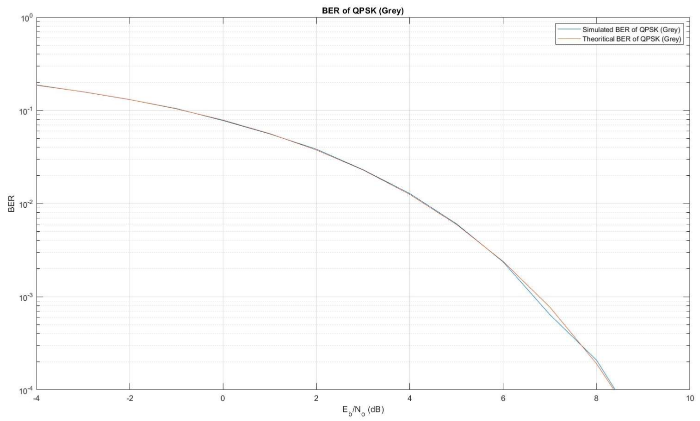

Figure 23: Simulated vs Theoretical BER for QPSK

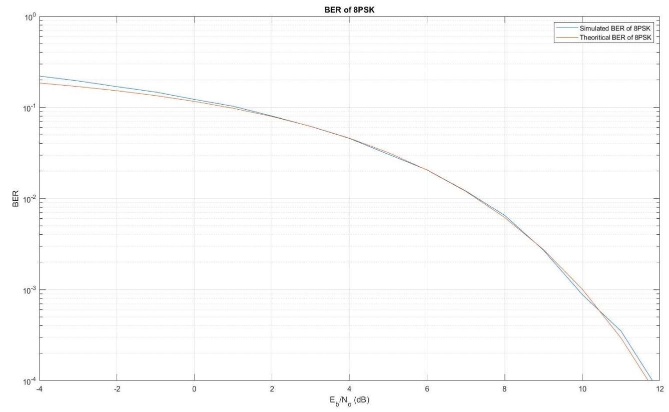

Figure 24: Simulated vs Theoretical BER for 8PSK

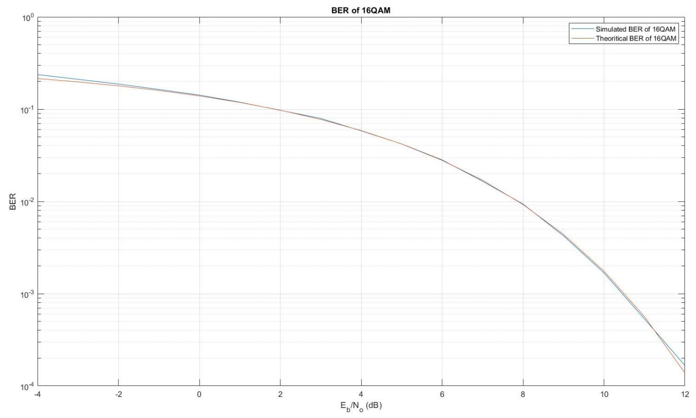

Figure 25: Simulated vs Theoretical BER for 16-QAM

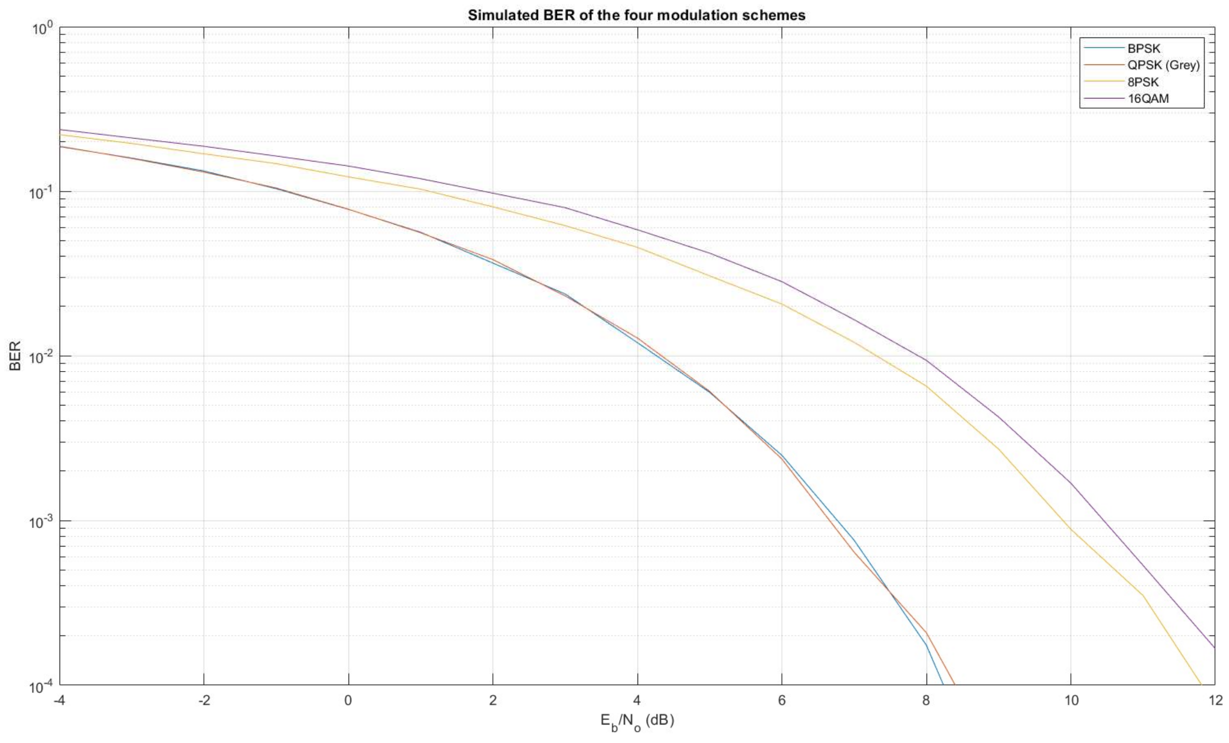

Figure 26: Simulated BER for BPSK, QPSK, 8PSK and 16-QAM

- The first thing worth noting from the plots in (Figures 22:25), is that the simulated BER for the four modulation schemes almost matches the theoritical BER.

- Another observation worth noting, which can also be noted in the theoritical BER expressions mentioned previously, is that the BPSK and the QPSK both yield the same BER which can also be shown clearly in the simulated BER in (Figure 26) where the simulated BER of BPSK and QPSK almost coincide with each other. This is because BPSK only uses one basis function in its signal space compared to QPSK where two basis functions are used which enables us to use the BW more efficiently by increasing the size of the signal set and maintaining a low BER that does not degrade the performance. So QPSK provides us with an efficient usage of the bandwidth (Double the data rate of the BPSK with the same BW or the same data rate with half the BW of the BPSK), with the same BER. For this reason, BPSK is not usually favored when using more than one basis function is an available option since QPSK provides us with the same performance.

- For a general MPSK scheme, increasing M “increasing the number of points of the signal set” increases the symbol rate for the same BW “or decreases the BW for the same symbol rate”. The cost of this, is a degraded performance by increasing the BER for the same $\frac{E_b}{N_0}$. To achieve a certain level of BER, 8PSK requires larger $\frac{E_b}{N_0}$ than that required for QPSK. This can be clearly seen in (Figure 26) when comparing the simulated BER of QPSK and 8PSK.
- For this reason, 16PSK is an impractical modulation scheme since the BER is much more degraded when compared 8PSK and QPSK. For M > 8, PSK is normally not favored. For this reason, 16-QAM is preferred when a 16 element signal set is required.

- 16-QAM provides us with four times the symbol rate for the same BW when compared to BPSK but the cost of this, is a degradation in the BER as can be seen when compared to BPSK, QPSK and 8PSK in (Figure 26).

VIII) BER simulation for Grey Encoded QPSK and Non-Grey Encoded QPSK

The bits are encoded in the signal space points of the constellation in each one of the discussed modulation schemes using Grey Encoding since it minimizes the BER for the same SER. The SER "which represents the probability of error in symbol decision" depends primarily on the Signal to Noise Ratio (SNR) "$\frac{E_s}{N_s}$". On the other hand, the BER "which represents the probability of bit error" depends on both the SNR and how the bits are encoded in the signal space set. Choosing Grey Encoding as the encoding scheme for the bits in the constellation makes the each signal point different from its closest neighboring signal points in only one bit, which minimizes the BER as much as possible for the same SER.

In this section, we are going to observe this difference by simulating the QPSK modulation scheme with two different encoding schemes, one with Grey Encoding and the other with a different encoding scheme as shown in (Figure 27).

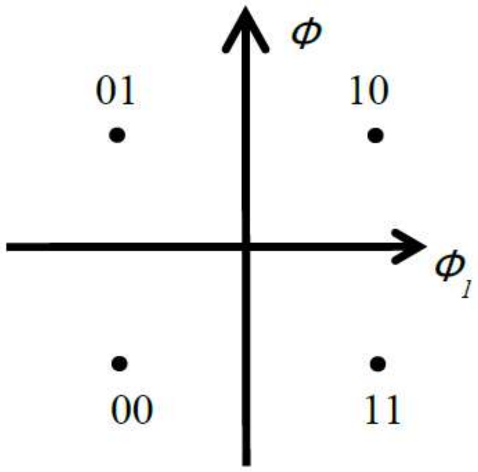

Figure 27: Non-Grey Encoded QPSK Constellation

Previously, the Grey Encoded QPSK scheme was implemented and simulated. In this section, we are going to implement the Non-Grey Encoded QPSK considered in (Figure 27) by simply adding a new modulation scheme table as mentioned in (Section III).

Bits	Decimal	Symbol
	QPSK (Non-Grey Encoded)	
00	0	$-1-1j$
01	1	$-1+1j$
10	2	$1+1j$
11	3	$1-1j$

```
M_QPSK_Not_Grey_Table = [-1-1i, -1+1i, 1+1i, 1-1i];
```

We then add this table to the `modulationSchemesTable` cell array

```
modulationSchemes = {
    M_BPSK_Table, M_QPSK_Grey_Table, M_8PSK_Table,...
    M_16QAM_Table, M_QPSK_Not_Grey_Table
};
```

Adding this table makes the main loop simulate the BER of this modulation scheme. We then plot the simulated BER for both the Grey Encoded and the Non-Grey Encoded QPSK schemes as shown in (Figure 28).

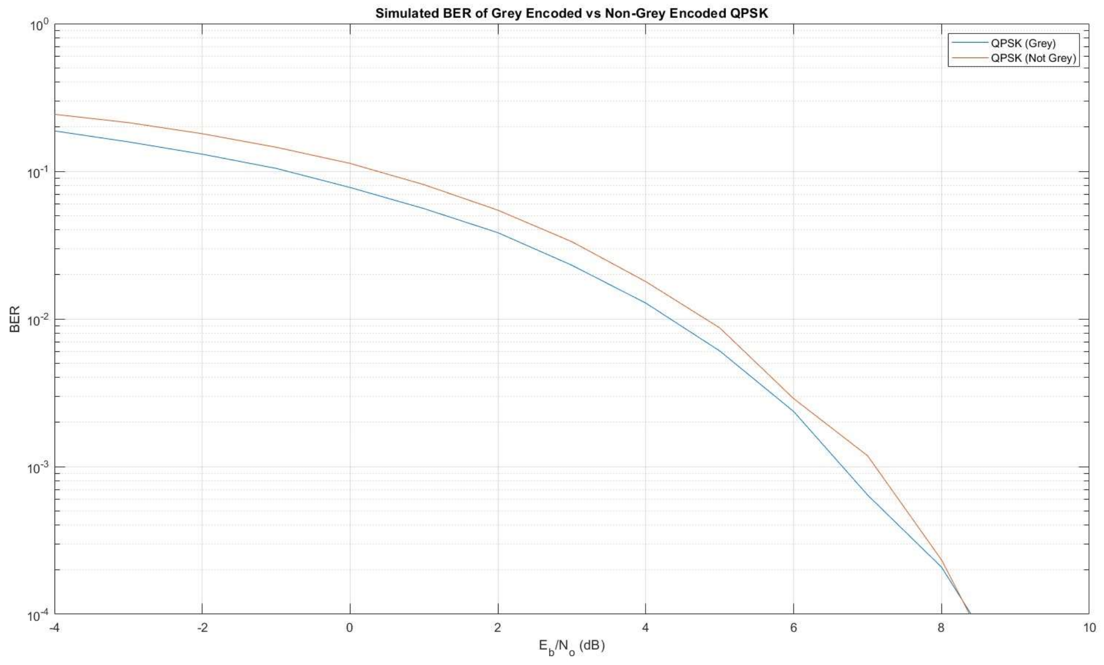

Figure 28: Simulated BER for Grey Encoded vs Non-Grey Encoded QPSK

As can be seen in (Figure 28), the two modulation schemes are done in the same $\frac{E_b}{N_0}$ range which means both must have the same SER. However, Grey Encoded QPSK achieves less BER for the same range compared to the Non-Grey Encoded QPSK. The reason for this, is Grey Encoding makes the encoded bits of the signal points differ from its closest neighboring signal points "which are the signal points to which error might occur" in only one bit, which minimizes the probability of error in bits as much as possible for the same error in symbol. For this reason, Grey Encoding is almost always preferred in modulations schemes.

IX) BFSK (Basis Functions and Baseband Equivalent Signal)

For a BFSK signal, where 0 is transmitted as $s_1(t)$ and 1 is transmitted as $s_2(t)$ where:

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

where $f_i = \frac{n_c+i}{T_b}$ and $i = 1,2$

It is important to note that our choice of f_1 *and* f_2 result in $s_1(t)$ and $s_2(t)$ being orthogonal signals. Thus, the basis functions are simply obtained by normalizing in $s_1(t)$ and $s_2(t)$ to obtain the basis functions $\phi_1(t)$ and $\phi_2(t)$ as follows:

$$\phi_i(t) = \begin{cases} \sqrt{\frac{2}{T_b}} \cos(2\pi f_i t), & 0 \le t \le T_b \\ 0 & \text{otherwise} \end{cases}$$

This provides us with a new modulation scheme whose signal space constellation can be illustrated in (Figure 29).

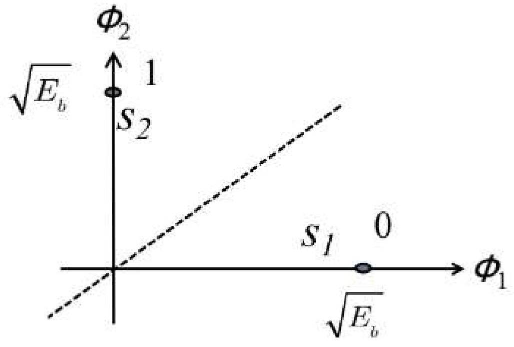

Figure 29: Signal Space Constellation of the BFSK modulation scheme

This will be useful in implementing the modulation scheme since as mentioned before, implementing a new modulation scheme is simply done by adding a new modulation scheme table as follows:

Bits	Decimal	Symbol
	BFSK	
0	0	1
1	1	$1j$

```
M_BFSK_Table = [1, 1j];
```

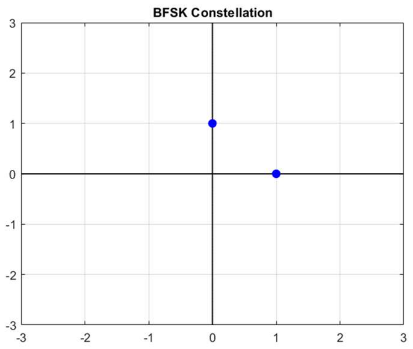

Figure 30: A polar plot of the BFSK constellation table

We then add this table to the `modulationSchemesTable` cell array to simulate the BER in the next section:

```
modulationSchemes = {
    M_BPSK_Table, M_QPSK_Grey_Table, M_8PSK_Table,...
    M_16QAM_Table, M_BFSK_Table, M_QPSK_Not_Grey_Table
};
```

Next, we are going to obtain the baseband equivalent signals. To obtain the baseband equivalent signals, we need to put our signals in the form:

$$x(t) = x_I(t)\cos(2\pi f_c t) - x_Q(t)\sin(2\pi f_c t)$$

And hence, the baseband equivalent signal can be written as: $x(t) = x_I(t) + jx_Q(t)$

Normally in other modulation techniques, there is usually one baseband equivalent representation for our signals since the carrier frequency is explicit. However, the essence of the FSK modulation techniques makes use of multiple frequencies. Therefore, we have freedom in our choice of what exactly the carrier frequency can be. We can choose any frequency in the vicinity of our modulation frequencies $f_1\ and\ f_2$ to be the carrier frequency.

We are going to choose the carrier frequency f_c to be f_1.

$$\therefore s_1(t) = \sqrt{\frac{2E_b}{T_b}}\cos(2\pi f_c t),$$

$$s_2(t) = \sqrt{\frac{2E_b}{T_b}}\cos\left(2\pi(f_c + \frac{1}{T_b})t\right) = \sqrt{\frac{2E_b}{T_b}}\cos\left(\frac{2\pi t}{T_b}\right)\cos(2\pi f_c t) - \sqrt{\frac{2E_b}{T_b}}\sin\left(\frac{2\pi t}{T_b}\right)\sin(2\pi f_c t)$$

Therefore, the baseband equivalent signals are:

$$s_{1BB}(t) = \sqrt{\frac{2E_b}{T_b}}$$

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

Our choice of the baseband equivalent signals are going to be of great importance in later sections when we are going to deal with the PSD of the BFSK modulation scheme.

X) BFSK (BER simulation)

Before simulating the BER of the BFSK we need to outline the theoretical expression for the BER of the BFSK modulation scheme.

The theoretical BER of BFSK can be considered using the same expression of the BPSK but instead of $\sqrt{\frac{E_b}{N_0}}$ we use $\sqrt{\frac{E_b}{2N_0}}$ since the distance between the two signal points is $\sqrt{2E_b}$ instead of $2\sqrt{E_b}$ in the case of BPSK.

Thus, the theoretical BER of BFSK = $\frac{1}{2}erfc\left(\sqrt{\frac{E_b}{2N_0}}\right)$

We add this expression in the cell array dedicated to the theoretical BER of the modulation schemes:

```
BER_theoritical = {
    0.5*erfc(sqrt(EBoverN0_linear)),...
    0.5*erfc(sqrt(EBoverN0_linear)),...
    erfc(sin(pi/8) * sqrt(3 * EBoverN0_linear)) / 3,...
    0.375*erfc(sqrt(0.4*EBoverN0_linear)),...
    0.5*erfc(sqrt(0.5*EBoverN0_linear))
};
```

By doing this, and by adding the modulation scheme table of BFSK to the `modulationSchemesTable` cell array, the BER of BFSK can be simulated and compared with the theoretical BER of BFSK using the same loop in which the other four modulation techniques are simulated in (Sections VII and VIII). We perform mapping using the mapper function then add channel noise in the same manner mentioned in (Section VI). We then plot the signal points of the received message over the signal space constellation to visualize the effect of different values of the SNR $\frac{E_b}{N_0}$ on the decision:

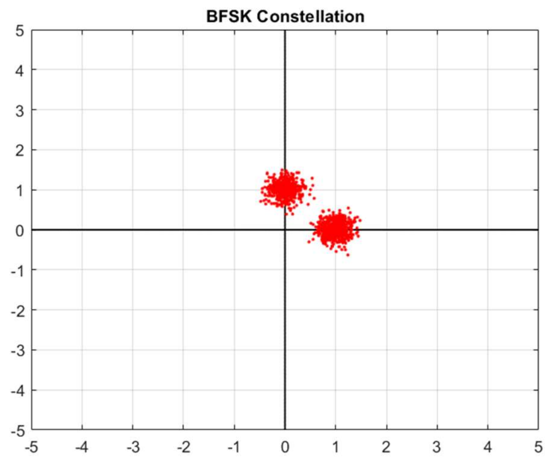

Figure 31: Noise Addition with SNR = 12 dB on BFSK

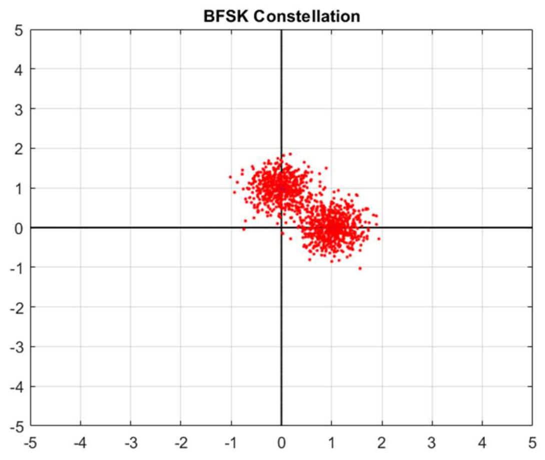

Figure 32: Noise Addition with SNR = 7 dB on BFSK

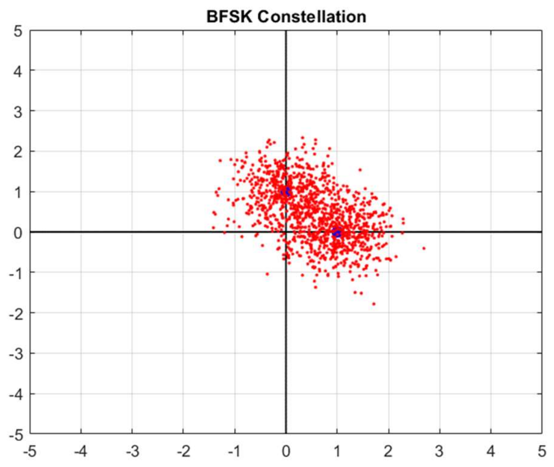

Figure 33: Noise Addition with SNR = 3 dB on BFSK

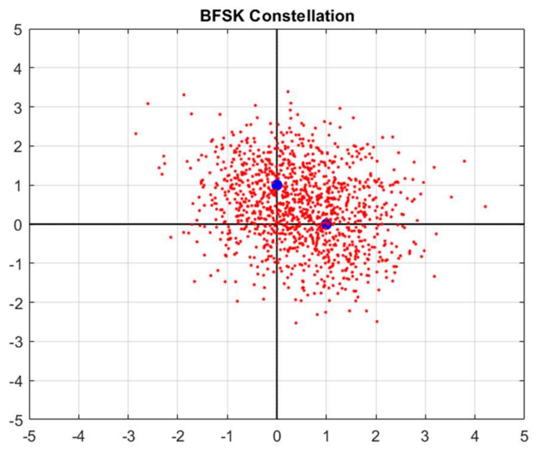

Figure 34: Noise Addition with SNR = -2 dB on BFSK

The theoretical and simulated BER of BFSK are plotted overlaid on the same plot as shown:

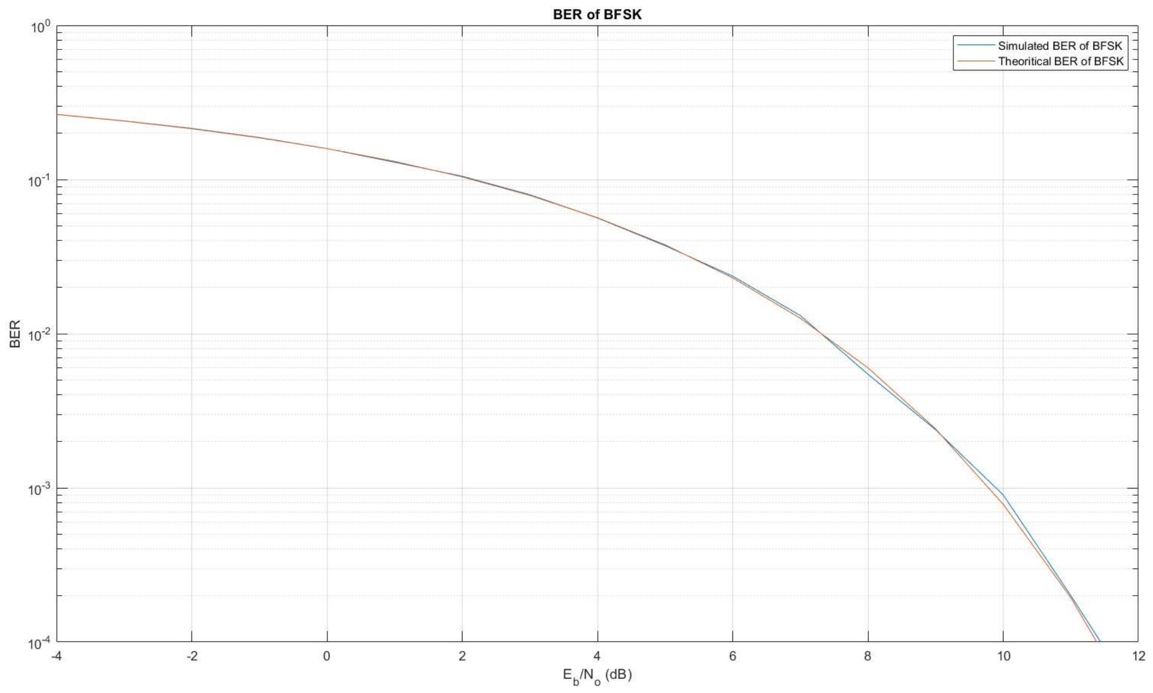

Figure 35: Simulated vs Theoretical BER for BFSK

XI) BFSK (PSD simulation)

To simulate PSD, it is required to generate ensemble of random bit stream where $T_b = 1sec$ and 100 bits and 500 realizations are used with 7 samples for each bit resulting in $F_s = 7Hz$ with random initial start, where, as mentioned before $f_c\ assumed\ to\ be\ f_1$ achieving BB equivalent signals

$$s_{1BB}(t) = \sqrt{\frac{2E_b}{T_b}}$$

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

where 0 is mapped to S_{1B} and 1 is mapped to S_{2BB} then initial start added using circular shift to add randomness.

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

The only difference compared to Project 1 is that values are complex so during calculation of autocorrelation, complex conjugate should be used.

```
Fc = 1/Tb;
F1 = 1/Tb;
F2 = 2/Tb;
t = 0:1/N_Samples:(1-1/N_Samples);
S1 = sqrt(2*Eb/Tb)*cos(2*pi*(F1-Fc)*t)+1i*sqrt(2*Eb/Tb)*sin(2*pi*(F1-Fc)*t);
S2 = sqrt(2*Eb/Tb)*cos(2*pi*(F2-Fc)*t)+1i*sqrt(2*Eb/Tb)*sin(2*pi*(F2-Fc)*t);

for i=1:1:N_realizations
    for j=1:N_Samples:N_Bits*N_Samples

        if(initial_ensemble(i,j)==0)
            initial_ensemble(i,j:j+N_Samples-1) = S1;
        else
            initial_ensemble(i,j:j+N_Samples-1) = S2;
        end
    end
end
for i=1:500
    PLR_ensemble(i,:) = circshift(initial_ensemble(i,:),initial_start(i));
end

for t=1:(N_Bits*N_Samples)
    PLR_S_cor(t) = PLR_S_cor(t)+sum(conj(PLR_ensemble(:,N_Bits*N_Samples/2))...
        .*PLR_ensemble(:,t))/N_realizations;
end
```

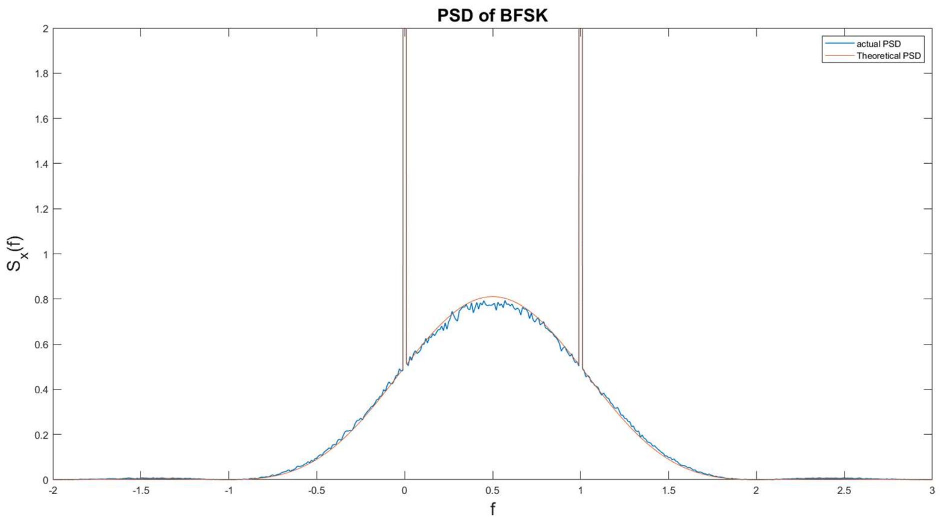

Figure 36: PSD simulation and Theoretical Curves Overlaid

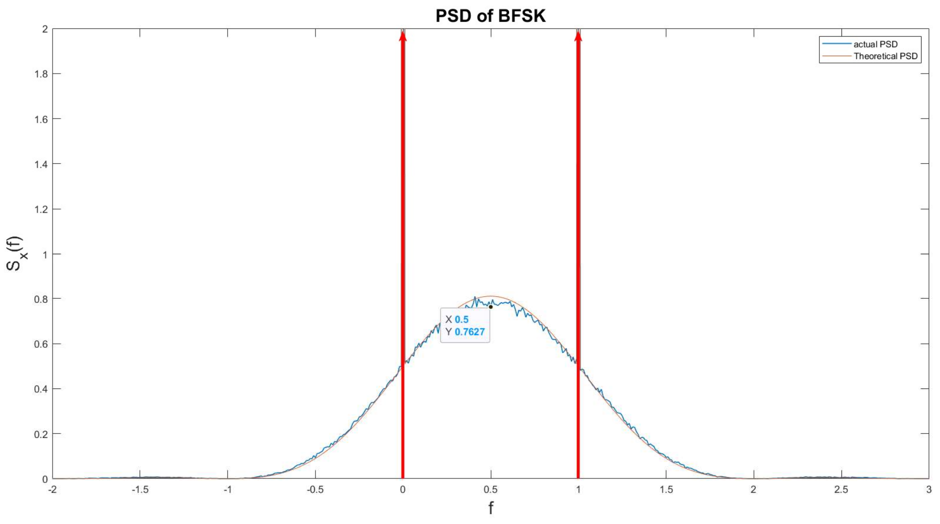

Figure 37: PSD simulation and Theoretical Curves Overlaid with important points annotated

XII) Appendix

Our code consists of five parts:

- The main script `main.m` which performs the basic steps of the single carrier communication system till simulating the BER and plotting the results obtained in this literature.
- The unified `mapper` function which performs the mapping functionality.
- The unified `demapper` function which performs the demapping functionality.
- Our implementation of the optimized decimal to binary converter used in the demapping process `myDecimalToBinaryVector`.
- The PSD simulation script `BFSK_PSD.m` for the BFSK modulation scheme which utilizes the baseband equivalent signal obtained in (Section X).
- The `main.m` script runs BER simulation on a transmitted message of length 120000 bits on six modulation schemes (BPSK, QPSK “Grey”, QPSK “Not Grey”, 8PSK, 16-QAM, and BFSK). The entire script takes 54.768 seconds to run.
- The `BFSK_PSD.m` script runs PSD simulations on an Ensemble of 500 realizations each of length 100 bits to perform statistical autocorrelation. The script is run for 1000 iterations to produce a smoothened and averaged version of the PSD. The entire script takes 74.385 seconds to run.

`main.m`

```
close all;
clear;
N_Bits = 120000;

% Modulation Schemes Tables
M_BPSK_Table = [-1, 1];
M_QPSK_Grey_Table = [-1-1i, -1+1i, 1-1i, 1+1i];
M_8PSK_Table = [
    1, 1/sqrt(2)+1i/sqrt(2), -1/sqrt(2)+1i/sqrt(2), 1j,...
    1/sqrt(2)-1i/sqrt(2), -1i, -1, -1/sqrt(2)-1i/sqrt(2)
];
M_16QAM_Table = [
    -3-3j, -3-1j, -3+3j, -3+1j,...
    -1-3j, -1-1j, -1+3j, -1+1j,...
    3-3j, 3-1j, 3+3j, 3+1j,...
    1-3j, 1-1j, 1+3j, 1+1j
];
M_BFSK_Table = [1, 1j];
M_QPSK_Not_Grey_Table = [-1-1i, -1+1i, 1+1i, 1-1i];

% Random Bit Stream
rawMessage = randi([0 1], [1 N_Bits]);

% Define SNR range
EBoverN0_dB = -4:1:12;
EBoverN0_linear = 10 .^ (EBoverN0_dB / 10);

% Cell Array carrying tables for modulation schemes
modulationSchemes = {
    M_BPSK_Table, M_QPSK_Grey_Table, M_8PSK_Table,...
    M_16QAM_Table, M_BFSK_Table, M_QPSK_Not_Grey_Table
};
modulationSchemesNames = {'BPSK', 'QPSK (Grey)', '8PSK', '16QAM', 'BFSK', 'QPSK (Not Grey)'};
BER_theoritical = {
    0.5*erfc(sqrt(EBoverN0_linear)),...
    0.5*erfc(sqrt(EBoverN0_linear)),...
    erfc(sin(pi/8) * sqrt(3 * EBoverN0_linear)) / 3,...
    0.375*erfc(sqrt(0.4*EBoverN0_linear)),...
    0.5*erfc(sqrt(0.5*EBoverN0_linear)),...
    0.5*erfc(sqrt(EBoverN0_linear))
};
BER_actual = cell(1, length(modulationSchemes));

% The BER simulation for all schemes goes in a loop
for i = 1:length(modulationSchemes)
    modulationScheme = modulationSchemes{i};        % Choose Scheme
    M_Name = modulationSchemesNames{i};             % Get scheme name
    bitsPerSymbol = log2(length(modulationScheme)); % Obtain number of bits per symbol
    Eavg = mean(abs(modulationScheme) .^ 2);        % Obtain Average Energy of Constellation
    N0 = Eavg ./ (bitsPerSymbol * EBoverN0_linear); % Adjust Noise according to SNR and Eavg

    % Generate Baseband Equivalent Signal from Raw Data Using Modulation Scheme
    transmittedMessage = mapper(rawMessage, modulationScheme);

    % Initialize BER vector
    BER_actual{i} = zeros(1, length(EBoverN0_dB));

    for m = 1:length(EBoverN0_dB)
        % Create Channel Noise
        NI = sqrt(N0(m) / 2) * randn(1, N_Bits / bitsPerSymbol);
        NQ = sqrt(N0(m) / 2) * randn(1, N_Bits / bitsPerSymbol);
        noise = NI + 1i * NQ;
        receivedMessage = transmittedMessage + noise;               % Add Channel Noise

        % Decision Block
        estimatedMessage = demapper(receivedMessage, modulationScheme);
        BER_actual{i}(m) = length(find(rawMessage ~= estimatedMessage));
    end
    BER_actual{i} = BER_actual{i} / N_Bits;

    % Plot Theoritical vs Simulated Overlaid
    figure;
    semilogy(EBoverN0_dB, BER_actual{i});
    hold on;
    semilogy(EBoverN0_dB, BER_theoritical{i});
    grid on;
    xlabel('E_b/N_o (dB)'); ylabel('BER');
    title(['BER of ' M_Name]);
    legend({['Simulated BER of ' M_Name], ['Theoritical BER of ' M_Name]});
    ylim([10e-5 1]);
end

% Plot BPSK, QPSK, 8PSK, and 16-QAM overlaid
figure;
for i = 1:4
    semilogy(EBoverN0_dB, BER_actual{i});
    hold on;
end
xlabel('E_b/N_o (dB)'); ylabel('BER');
title('Simulated BER of the four modulation schemes');
ylim([10e-5 1]);
grid on; hold off;
legend(modulationSchemesNames);

% Plot Grey vs Non-Grey Encoded QPSK
figure;
semilogy(EBoverN0_dB, BER_actual{2});
hold on;
semilogy(EBoverN0_dB, BER_actual{6});
xlabel('E_b/N_o (dB)'); ylabel('BER');
title('Simulated BER of Grey Encoded vs Non-Grey Encoded QPSK');
ylim([10e-5 1]);
grid on; hold off;
legend({modulationSchemesNames{2}, modulationSchemesNames{6}});
```

mapper.m

```
function transmittedMessage = mapper(rawMessage, modulationScheme)
    N_Bits = length(rawMessage);
    bitsPerSymbol = log2(length(modulationScheme));

    transmittedMessage = zeros(1, N_Bits / bitsPerSymbol);
    for j = 1:N_Bits/bitsPerSymbol
        startBit = bitsPerSymbol * (j - 1) + 1;
        endBit = bitsPerSymbol * j;

        % An efficient binary to decimal conversion
        index = bin2dec(string(char(rawMessage(startBit:endBit) + '0'))) + 1;
        transmittedMessage(j) = modulationScheme(index);
    end
end
```

demapper.m

```
function estimatedMessage = demapper(receivedMessage, modulationScheme)
    bitsPerSymbol = log2(length(modulationScheme));
    N_Bits = bitsPerSymbol * length(receivedMessage);

    estimatedMessage = zeros(N_Bits / bitsPerSymbol, bitsPerSymbol);
    for k = 1:N_Bits/bitsPerSymbol
        % Apply maximum likelihood rule
        % Find the symbol with minimum Euclidean distance
        [~, index] = min(abs(receivedMessage(k) - modulationScheme));
        estimatedBits = myDecimalToBinaryVector(index - 1, bitsPerSymbol);
        estimatedMessage(k,:) = estimatedBits;
    end

    estimatedMessage = reshape(estimatedMessage', [1, N_Bits]);
end
```

myDecimalToBinaryVector.m

```
function binaryVector = myDecimalToBinaryVector(decimal, numBits)
    % An efficient and optimized decimal to binary conversion
    binaryVector = dec2bin(decimal) - '0';
    binaryVector = [zeros(1, numBits - length(binaryVector)) binaryVector];
end
```

BFSK_PSD.m

```
Eb = 1;
Tb = 1;
N_Bits = 100;
N_Samples = 7;
N_realizations = 500;
Fc = 1/Tb;
F1 = 1/Tb;
F2 = 2/Tb;
iteration = 1000;
PLR_S_cor = zeros(1,N_Bits*N_Samples);
t = 0:1/N_Samples:(1-1/N_Samples);
S1 = sqrt(2*Eb/Tb)*cos(2*pi*(F1-Fc)*t)+1i*sqrt(2*Eb/Tb)*sin(2*pi*(F1-Fc)*t);
S2 = sqrt(2*Eb/Tb)*cos(2*pi*(F2-Fc)*t)+1i*sqrt(2*Eb/Tb)*sin(2*pi*(F2-Fc)*t);
for k=1:1:iteration
initial_start = randi([0,N_Samples-1],N_realizations,1);
initial_ensemble = repelem(randi([0,1],N_realizations,N_Bits),1,N_Samples);
PLR_ensemble = zeros(N_realizations,N_Bits*N_Samples);
for i=1:1:N_realizations
    for j=1:N_Samples:N_Bits*N_Samples
```

```
            if(initial_ensemble(i,j)==0)
            initial_ensemble(i,j:j+N_Samples-1) = S1;
        else
                 initial_ensemble(i,j:j+N_Samples-1) = S2;
        end
    end
end
for i=1:500
    PLR_ensemble(i,:) = circshift(initial_ensemble(i,:),initial_start(i));
end
for t=1:(N_Bits*N_Samples)
    PLR_S_cor(t) = PLR_S_cor(t)+sum(conj(PLR_ensemble(:,N_Bits*N_Samples/2))...
        .*PLR_ensemble(:,t))/N_realizations;
end
end
Fs = N_Samples;
n = N_Bits*N_Samples;
f=(-n/2:n/2-1)*Fs/n;
figure;
plot(f,abs(fftshift(fft(PLR_S_cor)))/(Fs*iteration),'linewidth',1);
hold on;
plotTheoriticalBFSK(f, Eb, Tb);
title('PSD of BFSK','fontsize',18);
annotation('arrow',[0.44 0.44],[0.11 0.92],'color','r','linewidth',3);
annotation('arrow',[0.595 0.595],[0.11 0.92],'color','r','linewidth',3);
xlabel('f','fontsize',18);
ylabel('S_x(f)','fontsize',18);
legend('actual PSD','Theoretical PSD');
xlim([-2 3]);
ylim([0 2]);

function plotTheoriticalBFSK(f, Eb, Tb)
    phi = 8*Eb*cos(pi*Tb*(f-0.5/Tb)).^2 ./ (pi^2 * (4*Tb^2*(f-0.5/Tb).^2 - 1).^2);
    phi(find(phi == Inf)) = 0;
    PSD = phi + diracDelta(length(f), find(f == 0)) + diracDelta(length(f), find(f
== 1));
    plot(f, PSD);

    function delta = diracDelta(vectorSize, deltaPosition, deltaValue)
        if (~exist('deltaValue', 'var'))
            deltaValue = 100;
        end
        delta = [zeros(1, deltaPosition - 1) deltaValue zeros(1, vectorSize -
deltaPosition)];
    end
end
```