Performance of Hamming Code over AWGN Channel

Honesh Roopchand
Student ID: 40071769

and

Mamaleshwar Kovi Gowri Kumar
Student ID: 40126008

ELEC 6141Wireless Communications
Department of Electrical and Computer Engineering
Concordia University
Montreal, Canada

***Abstract*— Wireless communication systems employ forward error correction codes to achieve reliable transmission. This project presents the Bit Error Rate (BER) performance evaluation of the Hamming code for transmission over an AWGN channel and provides comparison of the performance against its un-coded counterpart transmission. The coded scheme provides an average gain of 1.25 dB in E_b/N_0 over its un-coded scheme in AWGN channel with a code rate of 11/15 and QPSK modulation for a BER of order 10^{-6}.**

Keywords — Hamming code, syndrome decoding, AGWN, QPSK

I. INTRODUCTION

Hamming codes are the first class of linear block codes devised for forward error correction [1]. Hamming codes have a minimum distance of 3 and therefore are capable of correcting any single error over the span of the code block length. Hamming codes are perfect codes and can be decoded easily using a syndrome table-lookup scheme. Their variations, mainly their shortened versions, have been widely used for error control in digital communication and data storage systems over the years owing to their high rates and decoding simplicity.

In this project, a point-to-point wireless communication system with a transmission data rate, R_b = 1 Mbps is considered. A (15,11) Hamming code is employed as the forward error correction code at the transmitter's encoder whose generator matrix G is given below. QPSK mapping is used to modulate the data bits into signal for transmission over a AWGN channel. At the receiver, syndrome table-lookup decoding scheme is employed for decoding the received codeword.

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

This report is structured as follows. Section II gives the transmitter and receiver system models. Section III presents the simulation results and analysis. Section IV presents the conclusions.

II. TRANSMITTER AND RECEIVER SYSTEMS

A. Encoding and transmission

The block diagram of the transmitter is shown in Figure 1. The data packets generated are first sent to the encoder with rate, $r = k/n = 11/15$ and the associated code words, v that is generated from the (15,11) Hamming code are fed to the QPSK mapper

which converts each pair of bits in the code word to a symbol. Following which, the signal is transmitted in an AWGN communication channel.

Like any other linear block codes, a message vector, u of size $1\times k$ can be encoded using the generator matrix, G of size $k\times n$ to obtain a codeword, v of length, n as $v_{(1\times n)} = u_{(1\times k)} \cdot G_{(k\times n)}$
The generator matrix in the systematic form is represented as $G_{(k\times n)} = \left[P_{(k\times m)} \middle| I_{(k\times k)} \right]$, where, P is the parity-check submatrix, I is an identity submatrix of given sizes, and $m = n - k$. The parity-check matrix, H corresponding to this aforementioned generator matrix is therefore represented as $H_{(m\times n)} = \left[I_{(m\times m)} \middle| P^T{}_{(m\times k)} \right]$,where P^T is the transpose of the parity-check submatrix, P.

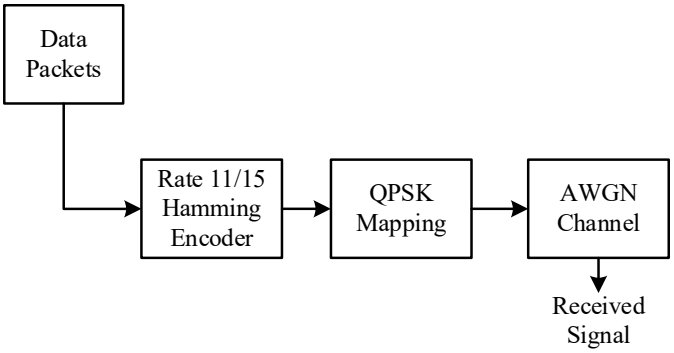

Figure 1: Transmitter system

In systematic form, the codeword, v for transmission can be written as $v_{(1\times n)} = \left[p_{(1\times m)} \middle| u_{(1\times k)} \right]$, where, p is the parity portion with m parity bits. All operations are carried over the Galois field, GF(2^1).

B. Decoding and reception

At the receiver side, the reverse processes are performed as depicted in Figure 2. The received signal is passed to QPSK de-mapper to demodulate the signal into the codeword, $\hat{v}$ (of n bits). The syndrome, S of the received codeword is then computed in order to check for validity of the de-mapped word and detect errors, if any, in same word. The syndrome, S is calculated as follows:

$$S = \hat{v} \cdot H^T$$

where, H is the parity check matrix (of size $m\times n$) of the corresponding generator matrix, G.

If the syndrome, S of size $1\times m$ results to an all-zero vector, the decoder assumes that the codeword is a valid one amongst the possible code words for the generator matrix employed and hence, proceeds for computation of the bit error rate.

If $S \neq 0$, the received code word, $\hat{v}$ contains erroneous bit(s) and it is required to be corrected. Since $G \cdot H^T = 0$, $\hat{v} = v + e$ and $v \cdot H^T = 0$, then,

$$S = \hat{v} \cdot H^T = (v + e) \cdot H^T$$

$$S = e \cdot H^T$$

where, e is the single-bit error pattern of the code.

It is clear that the syndrome, S depends on the error pattern, e (vector) only. Thus, when $S \neq 0$, the decoder looks up in the syndrome table for the matching syndrome with S and adds its corresponding single-bit error vector with the demodulated word, $\hat{v}$ in an attempt to correct a single bit in error. Thereafter, the bit error rate of the decoded word is computed. In case, no match of syndrome with S is found the syndrome table, the decoder simply accepts the demodulated word, $\hat{v}$ as the codeword and move on to calculate the BER. It is interesting to note that this Hamming code is capable of correcting only one bit in error in the codeword, and while correcting, it can also introduce more errors in the attempt to correct a single bit.

Moreover, any codeword before or after decoding may be altered by the noise or by the decoder during error correction respectively such that the codeword is transformed into another valid codeword of corresponding to the generator matrix (or Hamming code). In such scenario, the BER will be underestimated (giving low BER) despite of getting a valid but not same transmitted codeword.

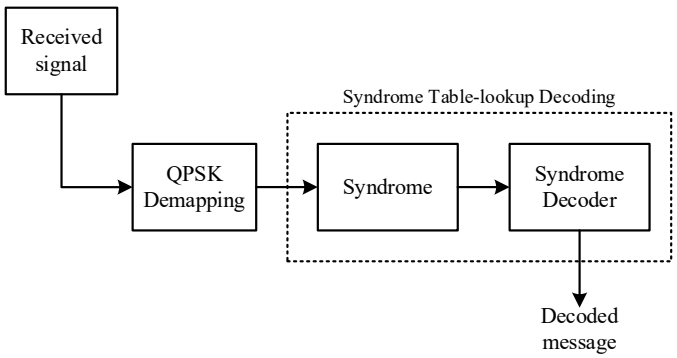

Figure 2: Receiver system

III. SIMULATION RESULTS AND ANALYSIS

The performances of the following schemes were analyzed and compared.

Scheme 1(a): Simulated without coding
The un-coded QPSK system simulated over AWGN channel, without the Hamming Encoder and Syndrome Decoder at the transmitter and receiver respectively. The transmitter and receiver frameworks are described in section II.

Scheme 1(b): Theoretical without coding
It is simply the theoretical version of Scheme 1(a), that is without encoding and decoding for forward error correction.

Scheme 2(a): Simulated with Hamming code
The Hamming coded QPSK system simulated over AWGN channel. The transmitter and receiver frameworks are given in Figure 1 and Figure 2 respectively.

Scheme 2(b): Theoretical with Hamming code
It is the theoretical version of Scheme 2(a).

In all simulations, the same values of the parameters as defined in the Section II for the system models of this project is employed, wherever applicable. Additional parameters for the simulations are as follows:

- Information rate, R_b at the input of the transmitter: 1 Mbps
- Code rate = 11/15

The BER, P_b for Scheme 1(b): Theoretical without coding is computed as follows:

$$P_b = Q\left(\sqrt{\frac{2E_b}{N_0}}\right)$$

An approximated BER, P_b (after decoding) for Scheme 2(b): Theoretical with coding is obtained as follows [2], where, t is the number of errors that the code can correct. In this case of Hamming code, t = 1, and p_{coded} is the probability of one bit in error after encoding.

$$p_{coded} = Q\left(\sqrt{\frac{2E_b}{N_0}\times\frac{k}{n}}\right)$$

$$P_b \approx \sum_{i=t+1}^{n} \frac{j}{n}\binom{n}{j}(p_{coded})^j(1-p_{coded})^{n-j}$$

Figure 3 shows the graph of BER against E_b/N_0 for the Schemes 1(a) and 1(b). It is observed that the un-coded scheme 1(a) provides identical BER performance as its theoretical version (scheme 1(b)) which confirms the expected result from the simulation.

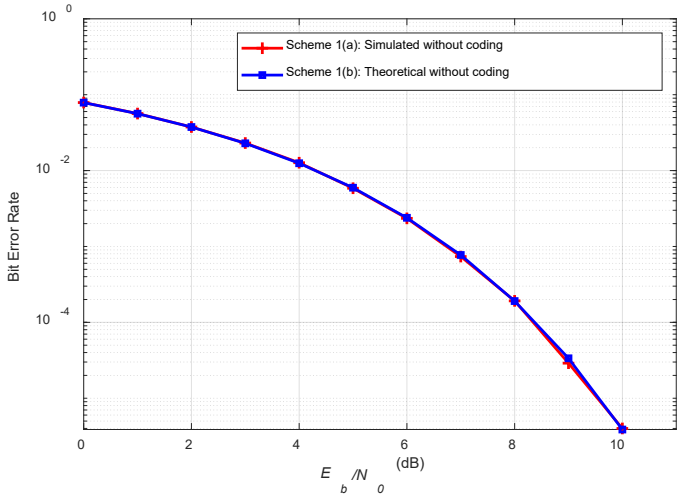

Figure 3: Error performance of system without coding (Schemes 1(a) and 1(b))

The graph of BER against E_b/N_0 for Schemes 2(a) and 2(b) is presented in Figure 4. In line with theoretical expectation, the BER performance for both schemes (simulated and theoretical) is supposed to be same. However, in this case it is seen that the simulated coded system provides inferior performance compared to the theoretical one. This is due to the fact that since the formula used to compute the theoretical BER does not consider the worst case scenarios where the decoder can add additional t or more errors while trying to correct the single error (t=1). Hence, the formula provides an overestimate performance in BER with respect to the actual performance of the decoder.

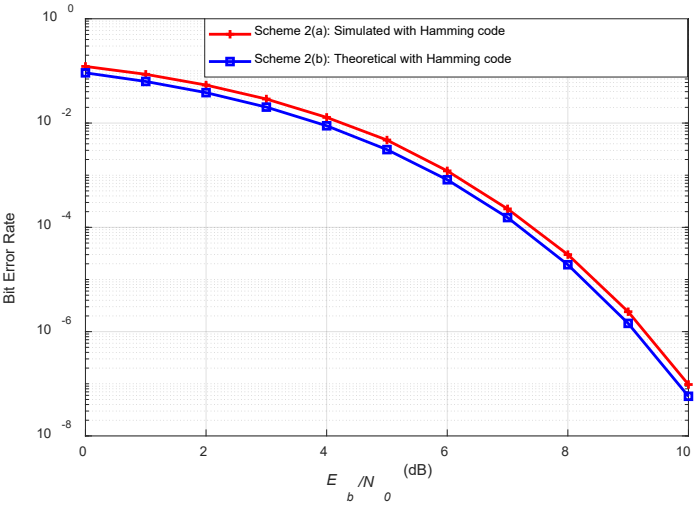

Figure 4: Error performance of system with Hamming code (Schemes 2(a) and 2(b))

The performances of Schemes 1(b) and 2(a) are presented in Figure 5 for comparative analysis. It is

observed that, as per theoretical expectation, the coded system provides superior performance over its un-coded counterpart system for the same value of E_b/N_0 when operating at higher SNR (bit energy to noise power ratio). It is also noted that a gain of approximately 1.25 dB on average is obtained for a BER of order 10^{-6} when the single-bit error correcting code (Hamming code) is applied to the system. However, for low values of the ratio of bit energy to noise power (E_b/N_0 < 4 dB), the un-coded system outperforms the coded one for BER $>10^{-2}$. This is because with code rate of 11/15, the energy per bit is shared between the parity bit for each data bit. Therefore, at low bit energy, the decoder is unable to excessively perform so as to compensate for the reduction of energy per bit after encoding with respect to the same significantly high value of noise power induced without encoding. It is to be noted from a theoretical point of view the performance of the system (Scheme 1) for a BER > 10^{-2} is important because it is revealing a new characteristic of the system whereby it is seen that an average gain of 1.0 dB can be obtained in this BER region even without any channel coding. However, from a practical point of view, the performance of the system for BER > 10^{-2} is not really relevant.

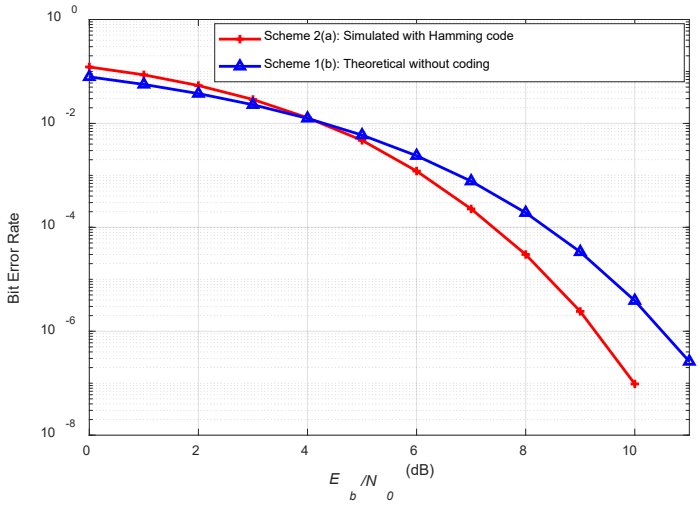

Figure 5: Bit error performance of system with and without coding (Schemes 2(a) and 1(b))

IV. CONCLUSION

This project presented the simulation of a wireless transmission system with Hamming code operating in an AWGN channel. Simulations were performed with QPSK modulation and syndrome table-lookup decoding technique in the decoder of the receiving system. The coded scheme provided an average gain of 1.25 dB in E_b/N_0 over its un-coded equivalent scheme for a BER of order 10^{-6}, thereby showing the benefit of a forward error correction code (Hamming code) in terms of minimum requirement for the signal power to achieve a desired level of error performance for communication in a wireless environment. An interesting future work would be to extend the performance analysis with other modulation schemes and for mobile users in Rayleigh fading channel.

V. REFERENCES

[1] R.W. Hamming, "Error Detecting and Error Correcting Codes," *Bell System Technical Journal*, 29: 147-60, April 1950.

[2] B. Sklar, *Digital Communications Fundamentals and Applications*. New Jersey, Upper Saddle River: Prentice Hall Press, 2001.

Annex

% Communication system with QPSK mapping, AWGN channel, without channel coding

```
%=========================================================================
close all
clear all
clc
dataRate = 1000000;    % Transmission rate = 1Mbps
EbNoVec = [0:10];    % Eb/N0 in dB (not linear)
for i = 1:length(EbNoVec)
    No =  10^(-1 * EbNoVec(i)/10);
    datastream = randi([0 1], [1 dataRate]);
    if(mod(dataRate, 2) ~= 0)
        datastream = [datastream 0]; % Appending a dummy bit to make it
multiple of 2
    end
    qpskmodsymbols = qpskmapping(datastream); % Gray-coded QPSK mapping
    AGWNoise = sqrt(No/2)*(randn(size(qpskmodsymbols)) + 1i *
randn(size(qpskmodsymbols))); % Generating AWGN.
    receivedSymbols = qpskmodsymbols + AGWNoise; % Passing the encoded message
through AWGN channel.
    qpskdemodsymbols = qpskdemapping(receivedSymbols); % At Receiver side.
Performinng demapping.
    qpskdemodsymbols(end) = 0; % Applying known dummy bit
    BitErrorVec = (datastream ~= qpskdemodsymbols);
    BER(i) = sum(BitErrorVec)/length(BitErrorVec); % Creating a vector of BER
of the signal
    end

EbNoVecTheoretical = [0:10];    % Eb/N0 % FOR THEORETICAL PLOT
BERtheoretical = qfunc(sqrt(2* (10.^(EbNoVecTheoretical/10))));  % Theoretical
uncoded BER for QPSK mapping.

%%PLOTS
figure (3)
semilogy(EbNoVec, BER, '-r+','linewidth',1.0)
axis([0.0 11 0 1.0])
xlabel('\it E_b/N_0 \rm(dB)', 'FontName', 'Times New Roman')
ylabel('Bit Error Rate', 'FontName', 'Times New Roman')
grid on
hold on
semilogy(EbNoVecTheoretical,BERtheoretical,'-bs','linewidth',1.0)
legend('Scheme 1(a): Simulated without coding', 'Scheme 1(b): Theoretical
without coding')
hold off
```

% FUNCTIONS

```
function qpskmodsymbols = qpskmapping(codeword)  % QPSK Modulation function
y = (2 * codeword) - 1;
real = y(1:2:end);
imag = y(2:2:end);
qpskmodsymbols = real + (1i*imag);
end

 function qpskdemodsymbols = qpskdemapping(receivedSymbols) % QPSK Demodulation
function
     count = 1;
```

```
    for i=1:length(receivedSymbols)
        if real(receivedSymbols(i)) > 0
           symbol(count) = 1;
        else
            symbol(count) = 0;
        end
        if imag(receivedSymbols(i)) > 0
            symbol(count + 1) = 1;
        else
            symbol(count + 1) = 0;
        end
        count = count + 2;
    end
    qpskdemodsymbols=symbol;
end
```

% Communication system with QPSK mapping, AWGN channel,
% (15, 11) Hamming Code (for channel coding)

```
%=========================================================================
close all
clear all
clc
N = 15;  % (15, 11) Hamming code, single error correcting code => t=1
K = 11;
M = N-K;
P = [1 1 1 1;    % P = parity sub-matrix of generator matrix
     0 1 1 1;    % P = [K x M]
     1 0 1 1;
     1 1 0 1;
     1 1 1 0;
     0 0 1 1;
     0 1 0 1;
     0 1 1 0;
     1 0 1 0;
     1 0 0 1;
     1 1 0 0];
G = [P eye(K)];   % generator matrix
H = [eye(M) P'];  % parity check matrix, H = [I | P'], where P' is the
transpose of P)
t = 1;            % t = 1 as given
dataRate = 1000000;    % Transmission rate = 1Mbps
EbNoVec = [0:9 9.5 10];   % Eb/N0 in dB (not linear)
for i = 1:length(EbNoVec)
    No = 10^(-1 * EbNoVec(i)/10) * (N/K); % Computing Noise power with Eb
normalized (fixed)
    datastream = randi([0 1], [1 dataRate]); % message stream
    if(mod(dataRate, K) ~= 0)
        datastream = [datastream zeros(1, (K - mod(dataRate, K)))];  % Padding
zeros if datastream is not multiple of K=11
    end
    startframe = 0;  % index for framing
    for nf = 1:(length(datastream)/K)
        endframecount = startframe + K;
        message = datastream(startframe+1:endframecount);  % framing stream by
K=11 bits
```

```
        startframe = endframecount;
        codeword = mod((message * G), 2);  % Encoding.
        codeword = [codeword 0]; % Appending a dummy bit (zero) to make it
multiple of 2
        qpskmodsymbols = qpskmapping(codeword); % Gray-coded QPSK mapping
        AGWNoise = sqrt(No/2)*(randn(size(qpskmodsymbols)) + 1i *
randn(size(qpskmodsymbols))); % Generating AWGN.
        receivedSymbols = qpskmodsymbols + AGWNoise; % Passing the encoded
message through AWGN channel
        qpskdemodsymbols = qpskdemapping(receivedSymbols); % At Receiver side.
Performinng demapping.
        qpskdemodsymbols(end) = []; % Discarding the dummy bit
        Ie = eye(N);  % Rows of Ie are error patterns (vectors)
        syndromeTable = mod((Ie * double(H')), 2);  % Corresponding syndrome
for each single-bit error pattern/vector
        syndrome = mod((qpskdemodsymbols * double(H')), 2);  % Syndrome
        if (all(syndrome == 0))
            decodedvec = qpskdemodsymbols;  % if S = 0, then demodulated
vectors is accepted as codeword
        else
            lookSyndromeinIe = ismember(syndromeTable, syndrome, 'rows');   %
else look upp in syndrome table for match
            if (all(lookSyndromeinIe == 0))    % All-zero means that no match
is found.
                decodedvec = qpskdemodsymbols;
            else
                pidx = find(lookSyndromeinIe == 1);   % If a match is found
                decodedvec = mod((qpskdemodsymbols + Ie(pidx, :)),2);  % Adding
the corresponding e to demapped vector
            end
        end
        if((nf == length(datastream)/K) && (mod(dataRate, K) ~= 0)) % if
remainder is not zero, then datastream is not multiple of K
            decodedvec(M+mod(dataRate, K)+1:end) = 0;  % Setting the known
padded zeros for the last frame
        end
        BitErrorVec = (message ~= decodedvec(M+1:N));   % decodedvec = [parity
bits | message bits]
        BERperFrameVec(nf) = sum(BitErrorVec)/length(BitErrorVec); % Creating a
vector of BERs for all the frames
    end
    BER(i) = sum(BERperFrameVec)/length(BERperFrameVec); % Creating a vector of
BER of the signal
end

EbNoVecTheoretical = [0:10];   % Eb/N0 FOR THEORETICAL PLOT
ig=0;
for EbNoVecTheo=EbNoVecTheoretical    % Formulating the theoretical BER
formula...Series summation
    ig=ig+1;
    t=1;
    id=0;
    for ik = t+1:N
        id=id+1;
        pcoded = qfunc(sqrt(2 * (10.^(EbNoVecTheo/10)) * (K/N)));  %
Theoretical BERcoded (after encoding)
         BERcodedtheoVec(1, id) = (ik/N)*(nchoosek(N,ik)).*(pcoded.^ik).*(1-
pcoded).^(N-ik);
    end
```

```
    BERdecodedtheoretical(1,ig) = sum(BERcodedtheoVec); % The "lower bound"
overestimated BER theoretical
end


%%PLOTS
figure (2)
semilogy(EbNoVec, BER, '-r+','linewidth',1.0)
axis([0.0 11 10^-8 1.0])
xlabel('\it E_b/N_0 \rm(dB)', 'FontName', 'Times New Roman')
ylabel('Bit Error Rate', 'FontName', 'Times New Roman')
grid on
hold on
semilogy(EbNoVecTheoretical,BERdecodedtheoretical,'-bs','linewidth',1.0)
legend('Scheme 2(a): Simulated with Hamming code', 'Scheme 2(b): Theoretical
with Hamming code')
hold off
```

% FUNCTIONS

```
function qpskmodsymbols = qpskmapping(codeword)  % QPSK Modulation function
y = (2 * codeword) - 1;
real = y(1:2:end);
imag = y(2:2:end);
qpskmodsymbols = real + (1i*imag);
end

 function qpskdemodsymbols = qpskdemapping(receivedSymbols) % QPSK Demodulation
function
     count = 1;
     for i=1:length(receivedSymbols)
         if real(receivedSymbols(i)) > 0
            symbol(count) = 1;
         else
             symbol(count) = 0;
         end
         if imag(receivedSymbols(i)) > 0
             symbol(count + 1) = 1;
         else
             symbol(count + 1) = 0;
         end
         count = count + 2;
     end
     qpskdemodsymbols=symbol;
 end
```