# Performance of Bit-Interleaved Coded Modulation over Rayleigh Fading Channel

Honesh Roopchand
Student ID: 40071769

and

Yukta Thapliyal
Student ID: 40137818

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

*Abstract*— **This paper presents the effect of bit-interleaver/de-interleaver in the Bit Error Rate (BER) performance of a coded wireless transmission system and provides comparison of the BER performance for the Bit-Interleaved Coded Modulation (BICM) with Hamming code against its counterpart system without interleaver. The BICM scheme provides a significant gain of 14 dB in $E_b/N_0$ for a BER of order $10^{-4}$ in Rayleigh flat and slow fading channel.**

***Keywords — BICM, Hamming code, syndrome decoding, Rayleigh fading, 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. Bit-interleaver has been widely used along with forward error correction codes for wireless transmission in fading channels such as Rayleigh [2]. The interleaver simply shuffles the bits from row-wise to column-wise arrangement to help combat burst errors that strike the transmitted stream in the fading environment [3].

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. Bit-interleaving is also employed to combat burst errors in the harsh Rayleigh fading environment. QPSK mapping is used to modulate the data bits into signal for transmission over the fading 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 into the encoder with rate, $r = k/n = 11/15$ and the corresponding code words, $v$ that are generated from the (15,11) Hamming code are fed into the bit-interleaver (scrambler) whose serial output is then passed into the QPSK mapper which converts each pair of bits to a symbol, $x_i$. The encoder, scrambler and mapper is generally referred to as Bit-Interleaved Coded Modulation (BICM) as shown in Figure 1. Following which, the signal, $x$ is transmitted through the Rayleigh fading channel with Gaussian noise (*noise*) added as well to produce the corrupted signal, $y$ at the receiver side as follows.

$$y_i = \alpha_i x_i + noise$$

where $\{\alpha_i\}$ are the Rayleigh fading coefficients with unit variance.

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 code word, $v$ 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, $P$.

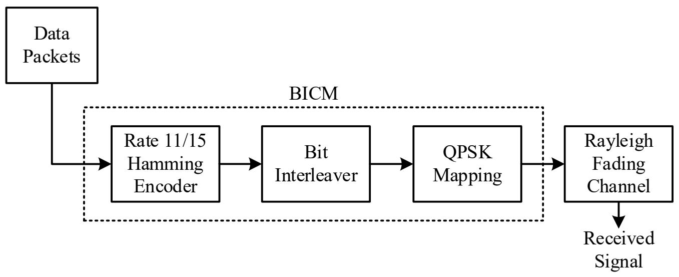

**Figure 1:** Transmitter system

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($2^1$).

The number of rows in the interleaver is called as the interleaver depth, $d$ and it corresponds to the maximum burst error length in bits that the channel can incur for the given parameters. Therefore, the maximum contiguous bits in the coded bit stream that can be affected at once is given by, $d = \left\lceil \frac{T_{coh}}{T_{b\_coded}} \right\rceil$ where, the coherence time, $T_{coh}$ and the duration of a bit after encoding, $T_{b\_coded}$ are computed as $T_{coh} = \frac{9}{16\pi f_d}$ and $T_{b_{coded}} = 1/(R_b \times n/k)$ respectively, given that the maximum Doppler shift, $f_d = \frac{V}{\lambda}$ and wavelength $\lambda = \frac{c}{f_c}$, ($f_c$ is the carrier frequency). The signal has already been mentioned as flat fading and further calculations find the signal undergoing through slow fading as well. Slow fading takes effect when signal transmission time is less than coherence time, i.e. $T_{sig} \ll T_{coh}$, where $T_{sig} = 1/(R_s \times n/k)$. The signal rate is determined through $R_s = \frac{R_b}{\log_2 M}$. The number of mapped symbols that are affected ($N_{sym\_affected}$) by same channel coefficient $\alpha_i$ is obtained as follows.

$$N_{sym\_affected} = \left\lceil \frac{T_{coh}}{T_{sig}} \right\rceil.$$

### 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 demodulated signal is then de-interleaved. De-interleaving involves the received signal being re-arranged to its original code word sequence, $v$. 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.

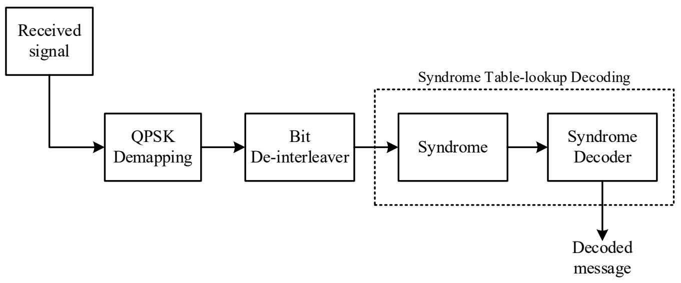

**Figure 2:** Receiver system

## 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 Rayleigh fading channel, without the Hamming Encoder and Interleaver at the transmitter side and without the De-interleaver and Syndrome Decoder at the receiver. The relevant transmitter and receiver blocks are described in section II.

*Scheme 1(b): Theoretical without coding*
It is simply the theoretical version of Scheme 1(a), computed from theoretical formula of the BER.

*Scheme 2: Simulated with Hamming code*
The Hamming coded QPSK system without the interleaver/de-interleaver simulated over Rayleigh fading channel. The concerned transmitter and receiver blocks are discussed in section II.

*Scheme 3: Simulated with BICM (interleaver)*
The bit-interleaved coded modulation (BICM) version simulated over Rayleigh fading channel. The transmitter and receiver frameworks are given in Figure 1 and 2.

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.

The BER, $P_{b\_uncoded}$ for Scheme 1(b): Theoretical without coding is computed as follows:

$$P_{b\_uncoded} = \frac{1}{2}\left(1 - \sqrt{\frac{E_b/N_0}{1 + E_b/N_0}}\right)$$

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.

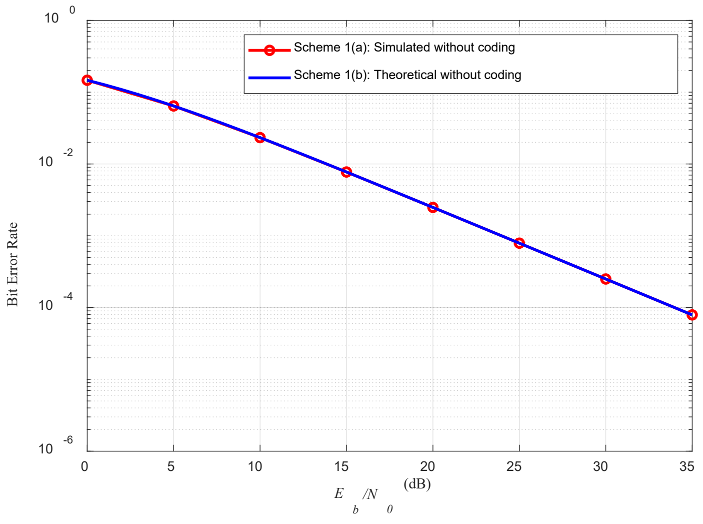

**Figure 3:** Error performance of system without coding (Schemes 1(a) and 1(b))

The graphs of BER against $E_b/N_0$ for Schemes 2 and 1(b) are presented in Figure 4. It is intuitively expected that by introducing an error correcting code in the communication system, one would get a better bit error rate performance at least for mid-to-high range values of $E_b/N_0$ (and inferior performance for low SNR range). However, it is observed that under a Rayleigh fading channel — whose characteristics approaches a realistic wireless channel more than the AWGN — the performance of the coded system is inferior to that of the un-coded one for all values of $E_b/N_0$ employed. This is because the energy per each bit of the codeword (i.e. with coding) is always smaller than that of un-coded bit (i.e. without coding) for the same value of SNR since the same energy available has to be shared amongst more bits (message and parity bits) for the codeword. In this case, since a single error correcting Hamming code is employed, even at high SNR 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, because the signal is affected more — it is the consequence of the Rayleigh fading channel coefficients — and the presence of burst errors makes this single error correcting code ineffective for such environment.

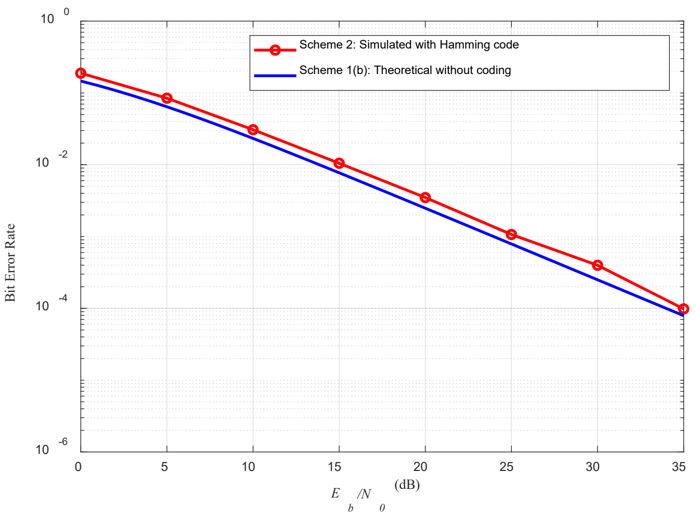

**Figure 4:** Error performance of system with Hamming code (Schemes 2 and 1(b))

The performances of Schemes 3, 2 and 1(b) are presented in Figure 5 for comparative analysis. It is seen that the Bit-Interleaved Coded Modulation (BICM) system outperforms the coded system without interleaver/de-interleaver 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 significant gain of approximately 14 dB in $E_b/N_0$ is obtained for a BER of order $10^{-4}$ when the scrambler is applied to the coded system. However, for low values of the ratio of bit energy to noise power ($E_b/N_0 < 7$ dB), the un-coded system performs better than the BICM one for BER > $3\times10^{-2}$. This is due to the fact that at low SNR, the BICM decoder (single-bit correction in this case) is unable to compensate for the dispensing of same *Eb* with parity and data bits as explained above as well as the high probability of having more than one bit in error at low SNR — effect of burst errors per codeword will be significant and more bits are prone to errors due to additive white Gaussian noise as well.

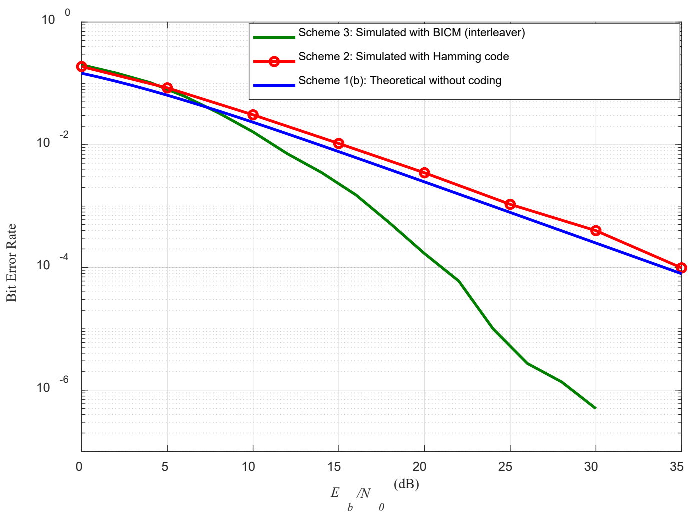

**Figure 5:** Bit error performance of BICM system and with and without coding (Schemes 3, 2 & 1(b))

It is interesting to note that if we use a more robust error correcting code with a soft-decoding algorithm, the BER performance of the coded system will be far superior compared to its un-coded counterpart system even in the absence of the interleaver/de-interleaver.

## IV. Conclusion

This project presented the simulation of a bit-interleaved coded modulation system for wireless communication. Hamming code was employed for forward error correction in a Rayleigh slow and flat fading channel. Simulations were performed with QPSK modulation and syndrome table-lookup decoding technique in the decoder of the receiving system. The bit-interleaving process provides significant improvement in the BER performance of a coded system while combatting for burst errors in this type of channel. Consequently, an important gain of 14 dB in $E_b/N_0$ is achieved with the introduction of the scrambler in the coded system. Additionally, its benefit of rendering the lowest error correcting code (single-bit correction) effective was shown and discussed. An interesting future work would be to investigate on the performance of the BICM with Turbo or LDPC codes and adaptive modulation coding 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.

[3] Barnali das, Manash P. Sharma and Kandarpa kumar Singh, “Implementation of a Block Interleaver Structure for use in Wireless Channels,” *16th International Conference on automatic control, modelling & simulation (acmos '14)*

# Annex

## % Communication system with QPSK mapping, Rayleigh fading channel, without channel coding

```
%==============================================================================
close all
clear all
clc

ModOrder = 4; % QPSK => Modulation Order = 4
BitRate = 10^6;  % Transmission bit rate = Rb
fc = 10*10^9; % Carrier frequency
velocity = (60*10^3)/(3600);  % v = 60km/h = (50/3) m/s
velocityOfLight = 3*10^8;
wavelength = velocityOfLight/fc;
fd = velocity/wavelength;  % fd is the maximum Doppler shift
Tsig = 1/(BitRate/log2(ModOrder));  % Tsignal is duration of symbol after
modulation and hence, after encoding as well.
Tcoh = 9/(16*pi*fd);  % Tcoh is coherence time
NsymbolsAffected = ceil(Tcoh/Tsig);  % Number of symbols affected by same
fading coefficient (aplhai)

InputDataSize = 162*10^4;   % Data bitstream length size
Nchannelcoeff = (InputDataSize/log2(ModOrder))/NsymbolsAffected;  % Number of
fading coefficients required for the modulated symbols stream.

EbNoVec = 0:5:30;   % Eb/N0 in dB (not linear)
BER = zeros(1, round(length(EbNoVec)));  % Pre-allocating BER vector for fast
execution only.
for i = 1:length(EbNoVec)
    No =  10^(-1 * EbNoVec(i)/10);
    datastream = randi([0 1], [1 InputDataSize]);
    qpskmodsymbols = qpskmapping(datastream); % Gray-coded QPSK mapping
    alphavector = sqrt(0.5)*(randn(Nchannelcoeff) + 1i*randn(Nchannelcoeff));
% alphavector stores all the fading coefficients required.
    alphaMagnitude = abs(alphavector);
    receivedSymbols = zeros(1,round(length(qpskmodsymbols)));  % Pre-allocation
of vector for improving execution speed in the for loop.
    for indexalpha = 1:length(alphaMagnitude)
        FadingGroupedSymbols = qpskmodsymbols(1,((indexalpha-
1)*NsymbolsAffected)+1:(indexalpha)*NsymbolsAffected);
        FadingPartialSignal =
alphaMagnitude(1,indexalpha)*FadingGroupedSymbols;  % This is the (alpha*x) in
y=alpha*x + AWGnoise. But here it is part of whole x.
        AGWNoise = sqrt(No/2)*(randn(size(FadingGroupedSymbols)) + 1i *
randn(size(FadingGroupedSymbols))); % Generating AWGN for each group of
NsymbolsAffected with same alpha.
        receivedGroupedSymbols = FadingPartialSignal + AGWNoise; % Passing the
encoded message through AWGN channel.
        EqualizedReceivedSymbols = receivedGroupedSymbols /
alphaMagnitude(1,indexalpha);
        receivedSymbols(1,((indexalpha-
1)*NsymbolsAffected+1:indexalpha*NsymbolsAffected)) = EqualizedReceivedSymbols;
% Whole signal which is the combined version all the partial signals (grouped
symbols).
    end
    qpskdemodsymbols = qpskdemapping(receivedSymbols); % Performinng demapping.
    BitErrorVec = (datastream ~= qpskdemodsymbols);
    BER(i) = sum(BitErrorVec)/length(BitErrorVec); % Creating a vector of BER
of the signal
end

EbNoVecTheoretical = 0:30;   % Eb/N0 % FOR THEORETICAL PLOT
BERuncodedTheoretical = 0.5 .* (1 - sqrt((10.^(EbNoVecTheoretical/10))./(1 +
(10.^(EbNoVecTheoretical/10)))));   % Theoretical uncoded BER for QPSK in fast
and slow Rayleigh fading channel

%%PLOTS
figure (2)
semilogy(EbNoVec, BER, '-r+','linewidth',1.0)
axis([0.0 35 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,BERuncodedTheoretical,'-b','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;
     modulOrder = 4; % Modulation order for QPSK = 4.
     demappedBitVec = zeros(1, length(receivedSymbols)*log2(modulOrder)); %
Pre-allocating its size for fast execution of the program only. Other than
this, it is not really required.
     for i=1:length(receivedSymbols)
         if real(receivedSymbols(i)) > 0
            demappedBitVec(count) = 1;
         else
             demappedBitVec(count) = 0;
         end
         if imag(receivedSymbols(i)) > 0
             demappedBitVec(count + 1) = 1;
         else
             demappedBitVec(count + 1) = 0;
         end
         count = count + 2;
     end
     qpskdemodsymbols=demappedBitVec;
 end
```

**% Communication system with QPSK mapping, Rayleigh fading 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

ModOrder = 4; % QPSK => Modulation Order = 4
BitRate = 10^6;  % Transmission bit rate = Rb
fc = 10*10^9; % Carrier frequency
velocity = (60*10^3)/(3600);  % v = 60km/h = (50/3) m/s
velocityOfLight = 3*10^8;
wavelength = velocityOfLight/fc;
fd = velocity/wavelength;  % fd is the maximum Doppler shift
Tsig = 1/((BitRate*(N/K))/log2(ModOrder));  % Tsignal is duration of symbol
after modulation and hence, after encoding as well.
Tcoh = 9/(16*pi*fd);  % Tcoh is coherence time
NsymbolsAffected = ceil(Tcoh/Tsig);  % Number of symbols affected by same
fading coefficient (aplhai)

InputDataSize = 11*22*10^4;    % Input data bitstream. It can be different from
Rb.
EbNoVec = 0:5:35;   % Eb/N0 in dB (not linear)
BER = zeros(1, round(length(EbNoVec)));  % Pre-allocating BER vector for fast
execution only.
datastream = randi([0 1], [1 InputDataSize]); % message stream
for i = 1:length(EbNoVec)
    No = 10^(-1 * EbNoVec(i)/10) * (N/K); % Computing Noise power with Eb
normalized (fixed)
    codedstream = zeros(1, round(length(datastream)*(N/K)));  % Pre-allocation
of vector for improving execution speed in the for loop.
    flag = 0;
    startframe = 0;  % index for framing
    for indexMsgFrame = 1:(length(datastream)/K) % index for moving frame-wise
for message bits
        endframecount = startframe + K;
        message = datastream(startframe+1:endframecount);  % framing stream by
K=11 bits
        startframe = endframecount;
        codeword = mod((message * G), 2);  % Encoding.
        codedstream(1,((indexMsgFrame-1)*N+1:indexMsgFrame*N)) = codeword;  % 
Creating a vector of the whole coded stream of message bits.
    end
    if(mod(((length(codedstream)/log2(ModOrder))/NsymbolsAffected),2) ~= 0)   % 
if symbols stream (signal) is not multiple of NsymbolsAffected (after encoding 
and modulation), then we pad zeros (dummy bits).
        flag = 1;
        Ndummysymbols = NsymbolsAffected - 
mod((length(codedstream)/log2(ModOrder)), NsymbolsAffected);
        Ndummybits = log2(ModOrder)* Ndummysymbols;
        codedstream = [codedstream zeros(1, Ndummybits)];
    end
    qpskmodsymbols = qpskmapping(codedstream); % Gray-coded QPSK mapping
    Nchannelcoeff = length(qpskmodsymbols)/NsymbolsAffected;   % Number of 
channel (fading) coefficients that will be required to cater for all groups of 
'NsymbolsAffected' symbols which will be affected by same alpha.
    alphavector = sqrt(0.5)*(randn(Nchannelcoeff) + 1i*randn(Nchannelcoeff));
% alphavector stores all the fading coefficients (alphai) required in complex 
form.
    alphaMagnitude = abs(alphavector);
    receivedSymbols = zeros(1,round(length(qpskmodsymbols)));  % Pre-allocation 
of vector for improving execution speed in the for loop.
    for indexalpha = 1:length(alphaMagnitude)  % index for moving alphai-wise.
        FadingGroupedSymbols = qpskmodsymbols(1,((indexalpha-
1)*NsymbolsAffected)+1:(indexalpha)*NsymbolsAffected);
        FadingPartialSignal = 
alphaMagnitude(1,indexalpha)*FadingGroupedSymbols;  % This is the (alpha*x) in 
y = alpha*x + AWGnoise. But here it is part of the whole x.
        AGWNoise = sqrt(No/2)*(randn(size(FadingGroupedSymbols)) + 1i * 
randn(size(FadingGroupedSymbols))); % Generating AWGN for each group of 
NsymbolsAffected with same alpha.
        receivedGroupedSymbols = FadingPartialSignal + AGWNoise; % Passing the 
encoded message through AWGN channel.
        EqualizedReceivedSymbols = receivedGroupedSymbols / 
alphaMagnitude(1,indexalpha);
        receivedSymbols(1,((indexalpha-
1)*NsymbolsAffected+1:indexalpha*NsymbolsAffected)) = EqualizedReceivedSymbols;
% Creating a vector for the whole signal which is the combined version all the 
partial signals (grouped symbols).
    end
    qpskdemodsymbols = qpskdemapping(receivedSymbols); % Performinng demapping. 
qpskdemodsymbols includes the dummy bits, if they were added before. Hence need 
to remove them, if any.
    if(flag == 1)  % If flag==1, then it means dummy bits were added before.
        qpskdemodsymbols(end+1-Ndummybits:end) = [];  % Discarding the dummy 
bits
    end
    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
    startframe = 0;  % index for start of frame
    clear endframecount;
    BERperFrameVec = zeros(1,(length(qpskdemodsymbols)/N));  % Pre-allocating 
its size with zeros for fast execution of the program only.
    for indexRXcodwrd = 1:(length(qpskdemodsymbols)/N)  % index for moving 
codeword-wise in the long demapped stream
        endframecount = startframe + N;
        demappedCodeword = qpskdemodsymbols(startframe+1:endframecount);  %
framing stream by K=11 bits
        syndrome = mod((demappedCodeword * double(H')), 2);  % Syndrome
        if (all(syndrome == 0))
            decodedvec = demappedCodeword;  % if S = 0, then demodulated
vectors is accepted as codeword
        else
            lookSyndromeinIe = ismember(syndromeTable, syndrome, 'rows');   %
else look up in syndrome table for match
            if (all(lookSyndromeinIe == 0))    % All-zero means that no match
is found.
                decodedvec = demappedCodeword;
            else
                pidx = find(lookSyndromeinIe == 1);   % If a match is found
                decodedvec = mod((demappedCodeword + Ie(pidx, :)),2);  % Adding
the corresponding e to demapped vector
            end
        end
        decodedstream((indexRXcodwrd-1)*K+1:
indexRXcodwrd*K)=decodedvec(M+1:N);
        startframe = endframecount;
    end
    BER(i) = sum(mod((decodedstream + datastream),2))/length(datastream);
end
EbNoVecTheoretical = 0:35;   % Eb/N0 % FOR UN-CODED THEORETICAL PLOT
BERuncodedTheoretical = 0.5 .* (1 - sqrt((10.^(EbNoVecTheoretical/10))./(1 +
(10.^(EbNoVecTheoretical/10)))));   % Theoretical uncoded BER for QPSK in fast
and slow Rayleigh fading channel

%%PLOTS
figure (2)
semilogy(EbNoVec, BER, '-r+','linewidth',1.0)
axis([0.0 55 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,BERuncodedTheoretical,'-b','linewidth',1.0)
legend('Scheme 2(a): Simulated with Hamming code', '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;
     modulOrder = 4; % Modulation order for QPSK = 4.
     demappedBitVec = zeros(1, length(receivedSymbols)*log2(modulOrder)); %
Pre-allocating its size for fast execution of the program only.
     for i=1:length(receivedSymbols)
```

```
        if real(receivedSymbols(i)) > 0
           demappedBitVec(count) = 1;
        else
            demappedBitVec(count) = 0;
        end
        if imag(receivedSymbols(i)) > 0
            demappedBitVec(count + 1) = 1;
        else
            demappedBitVec(count + 1) = 0;
        end
        count = count + 2;
    end
    qpskdemodsymbols=demappedBitVec;
end
```

**% Communication system with Bit-Interleaved Coded Modulation (BICM),**
**% Rayleigh fading channel, QPSK, (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
     0 1 1 1;
     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'];
t = 1;             % t = 1 as given

ModOrder = 4; % QPSK => Modulation Order = 4
BitRate = 10^6;  % Transmission bit rate = Rb
fc = 10*10^9; % Carrier frequency
velocity = (60*10^3)/(3600);  % v = 60km/h = (50/3) m/s
velocityOfLight = 3*10^8;
wavelength = velocityOfLight/fc;
fd = velocity/wavelength;  % fd is the maximum Doppler shift
Tsig = 1/((BitRate*(N/K))/log2(ModOrder));
Tcoh = 9/(16*pi*fd);
Tbcoded = (1/BitRate)*(K/N);
NsymbolsAffected = ceil(Tcoh/Tsig);  % Number of symbols affected by same
fading coefficient
interleaverDepth = ceil(Tcoh/Tbcoded); % Interleaver Depth
interleaverSerialLength = interleaverDepth * N;  % The total number of bits in
a block interleaver, that is its length when arranged serially.

InputDataSize = 22*10^5;    % Input data bitstream.
EbNoVec = 0:2:34;   % Eb/N0 in dB (not linear)
BER = zeros(1, round(length(EbNoVec)));  % Pre-allocating BER vector for fast
execution only.
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 InputDataSize]); % message stream
    codedstream = zeros(1, round(length(datastream)*(N/K)));  % Pre-allocation
of vector for improving execution speed in the for loop.
    flag = 0;
    startframe = 0;  % index for framing
    for indexMsgFrame = 1:(length(datastream)/K) % index for moving frame-wise
for message bits
        endframecount = startframe + K;
        message = datastream(startframe+1:endframecount);  % framing stream by
K=11 bits
```

```
        startframe = endframecount;
        codeword = mod((message * G), 2);  % Encoding.
        codedstream(1,((indexMsgFrame-1)*N+1:indexMsgFrame*N)) = codeword;  %
Creating a vector of the whole coded stream of message bits.
    end
    if(mod((length(codedstream)/interleaverSerialLength),2) ~= 0)   % Padding
zeros if codestream (bits) is not multiple of interleaverDepth
        flag = 1;
        Ndummybits = interleaverSerialLength -
mod(length(codedstream),interleaverSerialLength);
        codedstream(1, (end+1):(end+Ndummybits)) = zeros(1, Ndummybits);
    end

    % BIT-INTERLEAVING
    NinterleaverBlocks = length(codedstream)/interleaverSerialLength;   %
NinterleaverBlocks is number of block interleavers required.
    interleavedCodedstream = zeros(1, round(length(codedstream)));
    for indexBlock = 1:NinterleaverBlocks
        partialSerialCodedstreamVec = codedstream(1,(((indexBlock-
1)*interleaverSerialLength)+1:indexBlock*interleaverSerialLength));
        transposedBlockInterleaverMatrix = reshape(partialSerialCodedstreamVec,
N, []);
        blockInterleaverMatrix = transposedBlockInterleaverMatrix';
        partialInterleavedSerialBits = reshape(blockInterleaverMatrix, 1, []);
        interleavedCodedstream(1,(((indexBlock-
1)*interleaverSerialLength)+1:indexBlock*interleaverSerialLength))=
partialInterleavedSerialBits;  % Creating a vector of the complete interleaved
bits of the coded stream
    end
    qpskmodsymbols = qpskmapping(interleavedCodedstream); % Gray-coded QPSK
mapping
    Nchannelcoeff = length(qpskmodsymbols)/NsymbolsAffected;   % Number of
channel (fading) coefficients that will be required to cater for all groups of
'NsymbolsAffected' symbols which will be affected by same alpha.
    alphavector = sqrt(0.5)*(randn(Nchannelcoeff) + 1i*randn(Nchannelcoeff));
% alphavector stores all the fading coefficients (alphai) required in complex
form.
    alphaMagnitude = abs(alphavector);
    receivedSymbols = zeros(1,round(length(qpskmodsymbols)));  % Pre-allocation
of vector for improving execution speed in the for loop.
    for indexalpha = 1:length(alphaMagnitude)  % index for moving alphai-wise.
        FadingGroupedSymbols = qpskmodsymbols(1,((indexalpha-
1)*NsymbolsAffected)+1:(indexalpha)*NsymbolsAffected);
        FadingPartialSignal =
alphaMagnitude(1,indexalpha)*FadingGroupedSymbols;  % This is the (alpha*x) in
y = alpha*x + AWGnoise. But here it is part of the whole x.
        AGWNoise = sqrt(No/2)*(randn(size(FadingGroupedSymbols)) + 1i *
randn(size(FadingGroupedSymbols))); % Generating AWGN for each group of
NsymbolsAffected with same alpha.
        receivedGroupedSymbols = FadingPartialSignal + AGWNoise; % Passing the
encoded message through AWGN channel.
        EqualizedReceivedSymbols = receivedGroupedSymbols /
alphaMagnitude(1,indexalpha);
        receivedSymbols(1,((indexalpha-
1)*NsymbolsAffected+1:indexalpha*NsymbolsAffected)) = EqualizedReceivedSymbols;
% Creating a vector for the whole signal which is the combined version all the
partial signals (grouped symbols).
    end
    qpskdemodsymbols = qpskdemapping(receivedSymbols); % Performinng demapping.
    % BIT DE-INTERLEAVING
    DeInterleavedDemappedstream = zeros(1, round(length(qpskdemodsymbols)));  %
For pre-allocation
    for indexBlock = 1:NinterleaverBlocks
        partialSerialDemappedstreamVec = qpskdemodsymbols(1,(((indexBlock-
1)*interleaverSerialLength)+1:indexBlock*interleaverSerialLength));
        blockDeInterleaverMatrix = reshape(partialSerialDemappedstreamVec,
interleaverDepth, []);
        transposedBlockDeInterleaverMatrix = blockDeInterleaverMatrix';
        partialDeInterleavedSerialBits =
reshape(transposedBlockDeInterleaverMatrix, 1, []);
        DeInterleavedDemappedstream(1,(((indexBlock-
1)*interleaverSerialLength)+1:indexBlock*interleaverSerialLength)) =
partialDeInterleavedSerialBits;  % Creating a vector of the complete de-
interleaved bits of the demapped stream
    end
    % We must remove the dummy bits added before interleaving (which was done
to make multiple of row*column of the block interleaver).
    if(flag == 1)  % If flag==1, then it means dummy bits were added before.
        DeInterleavedDemappedstream(end+1-Ndummybits:end) = [];  % Discarding
the dummy bits
    end
    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
    startframe = 0;  % index for start of frame
    clear endframecount;
    BERperFrameVec = zeros(1,(length(DeInterleavedDemappedstream)/N));  % Pre-
allocating its size with zeros for fast execution of the program only.
    for indexRXcodwrd = 1:(length(DeInterleavedDemappedstream)/N)  % index for
moving codeword-wise in the long demapped stream
        endframecount = startframe + N;
        demappedCodeword =
DeInterleavedDemappedstream(startframe+1:endframecount);  % framing stream by
K=11 bits
        syndrome = mod((demappedCodeword * double(H')), 2);  % Syndrome
        if (all(syndrome == 0))
            decodedvec = demappedCodeword;  % if S = 0, then demodulated
vectors is accepted as codeword
        else
            lookSyndromeinIe = ismember(syndromeTable, syndrome, 'rows');   %
else look up in syndrome table for match
            if (all(lookSyndromeinIe == 0))    % All-zero means that no match
is found.
                decodedvec = demappedCodeword;
            else
                pidx = find(lookSyndromeinIe == 1);   % If a match is found
                decodedvec = mod((demappedCodeword + Ie(pidx, :)),2);  % Adding
the corresponding e to demapped vector
            end
        end
        BitErrorVec = (datastream((indexRXcodwrd-1)*K+1: indexRXcodwrd*K) ~=
decodedvec(M+1:N));  % decodedvec = [parity bits | message bits]. So comparing
decoded message bits to transmitted message bits (NOT parity bits)
        BERperFrameVec(indexRXcodwrd) = sum(BitErrorVec)/length(BitErrorVec); %
Creating a vector of BERs for all the frames
        startframe = endframecount;
    end
    BER(i) = sum(BERperFrameVec)/length(BERperFrameVec); % Creating a vector of
BER of the signal
end
EbNoVecTheoretical = 0:35;   % Eb/N0 % FOR UN-CODED THEORETICAL PLOT
BERuncodedTheoretical = 0.5 .* (1 - sqrt((10.^(EbNoVecTheoretical/10))./(1 +
(10.^(EbNoVecTheoretical/10)))));   % Theoretical uncoded BER for QPSK in fast
and slow Rayleigh fading channel

%%PLOTS
figure (2)
semilogy(EbNoVec, BER, '-r+','linewidth',1.0)
axis([0.0 37 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,BERuncodedTheoretical,'-b','linewidth',1.0)
legend('Scheme 3: Simulated with BICM', '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;
     modulOrder = 4; % Modulation order for QPSK = 4.
     demappedBitVec = zeros(1, length(receivedSymbols)*log2(modulOrder)); %
Pre-allocating its size for fast execution of the program only.
     for i=1:length(receivedSymbols)
         if real(receivedSymbols(i)) > 0
            demappedBitVec(count) = 1;
         else
             demappedBitVec(count) = 0;
         end
         if imag(receivedSymbols(i)) > 0
             demappedBitVec(count + 1) = 1;
         else
             demappedBitVec(count + 1) = 0;
         end
         count = count + 2;
     end
     qpskdemodsymbols=demappedBitVec;
 end
```