VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Information Theory and Coding

ECE4007 – F2+TF2

BER PERFORMANCE ANALYSIS OF LINEAR BLOCK CODES

By

Shikhar Chandra – 18BEC0146

Srivathsan V K K – 18BEC0341

Harish K – 18BEC0353

Submitted to

Thanikaiselvan V

Associate Professor Sr.

School of Electronics Engineering

INDEX

ABSTRACT

Channel coding is an important step in the process of transmission and reception of a digitally modulated signal. To ensure that the communication of information is reliable and free of errors, a channel encoder and decoder is incorporated at the transmitter and receiver respectively, in a digital communication system. One such channel coding technique called Linear Block Coding is discussed in this project. The performance of three different sub-types of Linear Block codes, namely, (3,1) repetition code, (5,1) repetition code and (7,4) Hamming code is analyzed for BPSK digital modulation scheme and under AWGN channel noise conditions. The algorithm is simulated in MATLAB tool and the results are obtained in the form of BER curves and are compared individually for different cases of decoding techniques and uncoded technique.

DESCRIPTION OF PROJECT

In this project, a comparative analysis of the Bit Error Rate performance is done on different types of linear block codes.

BER is used as an important parameter in characterizing the performance of data channels. When data is transmitted over a data link, there is a possibility of errors being introduced into the system. If errors are introduced into the data, then the integrity of the system may be compromised. As a result, it is necessary to assess the performance of the system. A Bit Error Rate is defined as the rate at which errors occur in a transmission system. This can be directly translated into the number of errors that occur in a string of a stated number of bits. The definition of bit error rate can be translated into a simple formula:

BER = Errors / Total Number of Bits

For a given pass-band modulation scheme, i.e., Binary Phase Shift Keying (BPSK) and under a random channel noise of the type Additive White Gaussian Noise (AWGN), the performance of different linear block codes is studied. A block code is a rule for converting a sequence of source bits, of length K, say, into a transmitted sequence of length N bits, where, in order to add redundancy, N will of course be greater than K. Linear block codes are a type of channel codes in which the codewords are blocks of symbols having added redundant bits which are linear combinations of the message bits.

The linear block codes demonstrated in this project are:

1. (3,1) Repetition Code
2. (5,1) Repetition Code
3. (7,4) Hamming Code

The (3,1) Repetition code and (5,1) Repetition code follow the same logic of repeating symbols a specific number of times in a block of data.

Repetition Code	Message Bit	Encoded Data
(3,1)	0	000
	1	111
(5,1)	0	00000
	1	11111

The (7,4) Hamming code adds 3 redundant bits to a message block of 4 bits. The redundant bits are parity bits determined from the Parity Matrix of the code.

Given Data bits as d1, d2, d3, and d4,

A (7, 4) Hamming code may define parity bits p1, p2, and p3 as:

$$p1 = d2 + d3 + d4$$

$$p2 = d1 + d3 + d4$$

$$p3 = d1 + d2 + d4$$

where all additions are modulo-2 additions

A Generator matrix that produces code words with the bits ordered [d1, d2, d3, d4, p1, p2, p3] (4 data bits followed by 3 parity bits), is as follows:

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

For decoding the received encoded word, a Parity Check Matrix is used:

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

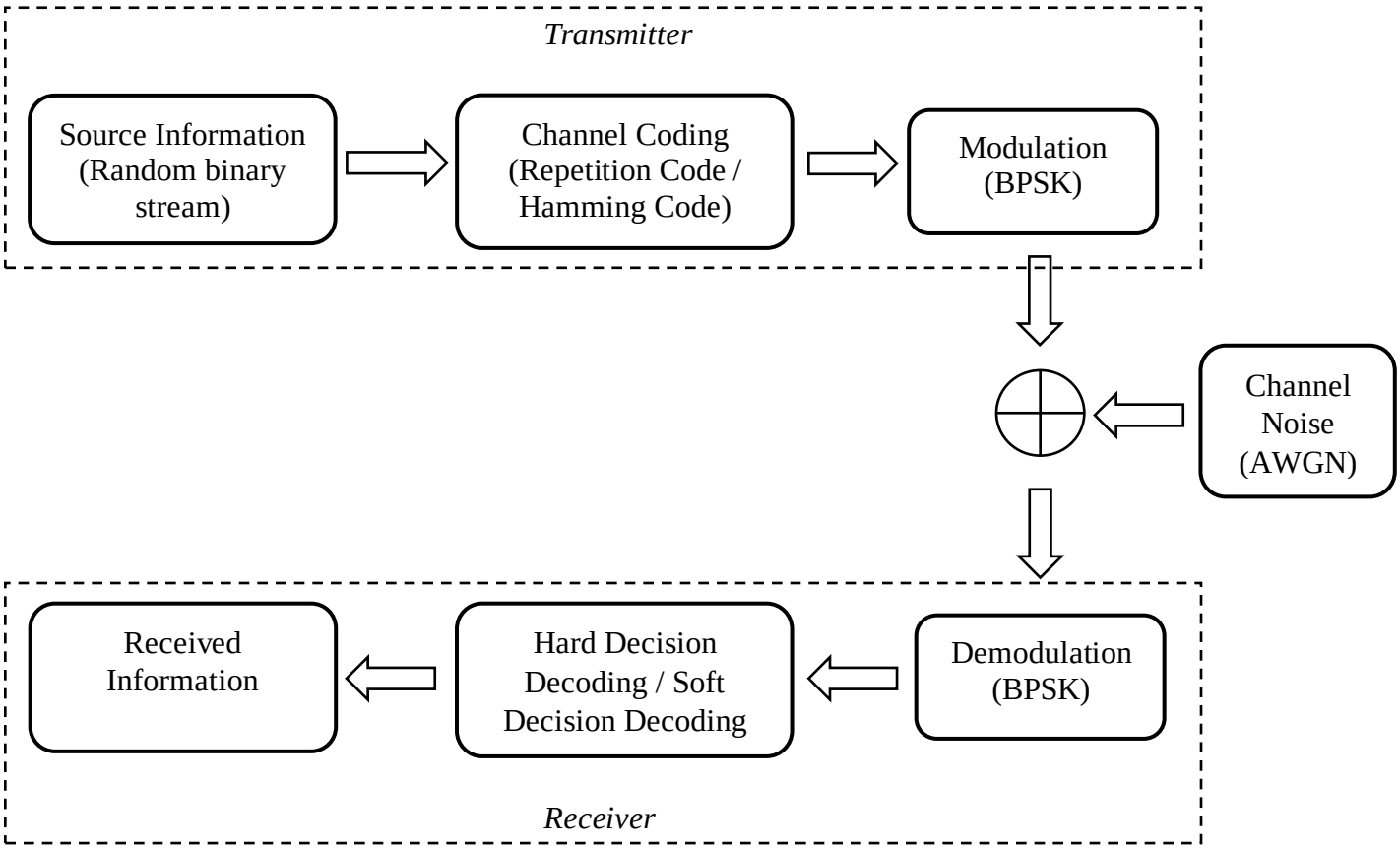
Multiplying the received codeword columns with the transpose of Parity Check Matrix, yields the ‘Syndrome’.

If the syndrome is all zeros, the encoded data is error free. If the syndrome has a non-zero value, flipping the encoded bit that is in the position of the column in H that matches the syndrome will result in a valid code word.

At the receiver side, the decoding is done either by Hard Decision decoding technique or Soft Decision Decoding technique.

The algorithms for these channel coding techniques are developed using MATLAB and their performance is studied by plotting the Bit Error Rate (BER) curves.

BLOCK DIAGRAM



Overview of Block Diagram:

- Source Information: This is a random stream of binary symbols (1s and 0s). This denotes the original information to be transmitted from the source and can be either text, voice, or image encoded in binary format. It is assumed that Source coding is also done at this block.

- Channel Coding: To ensure error detection and correction at receiver end, the information is channel coded using linear block codes. This can be either one of Repetition code or Hamming code. This step adds redundant bits to the blocks of data and hence the size of the data increases.

- Modulation (BPSK): The BPSK modulation maps the symbols as "1" to "1" and "0" to "-1". Hence the output of this block is a stream of 1s and -1s.

- Channel Noise (AWGN): Additive White Gaussian Noise is added by the channel with variance adjusted to specific Signal to Noise ratio (SNR).

- Demodulation (BPSK): The demodulation involves a decision device (D.D) which decides that the received symbol is "0" if it is negative, and "1" if the received symbol is positive.

- Hard Decision Decoding / Soft Decision Decoding: For channel decoding, either Hard Decision Decoding, or Soft Decision Decoding is performed. In Hard Decision Decoding (HDD), the decoding of a particular bit in the received block, is not dependent on the other bits in the block. On the other hand, in Soft Decision Decoding (SDD), the decoding of a bit is dependent upon the neighboring bits and their distance from the decision boundary.

- Received Information: The decoded symbols are finally received at the receiver. This information is bound to have some errors due to the channel noise and some of these errors can be detected and corrected depending upon the type of channel coding used. The discrepancy between the original information and received information is measured by calculating the Bit Error Rate.

ALGORITHM

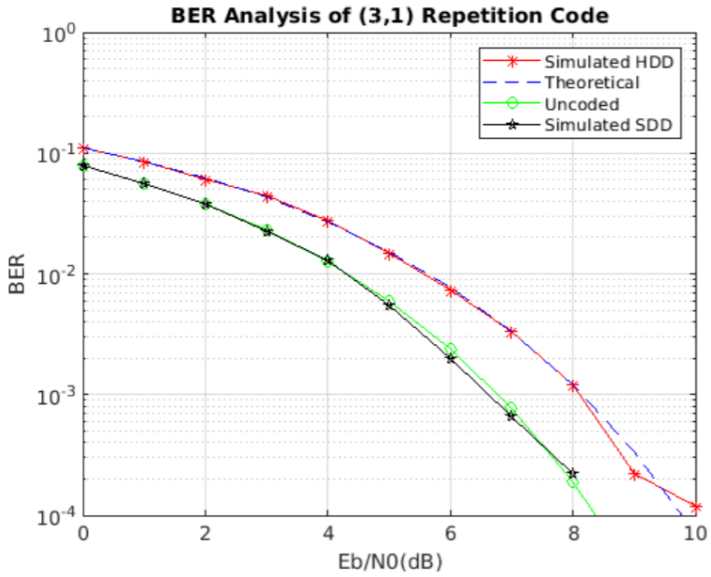
(n, 1) Repetition Code:

1. Initializing ‘m’ number of message bits.
2. Repeat the respective bits ‘n’ times for repetition coding.
3. Take the output after repetition coding and apply BPSK modulation where +1 will be as +1 and 0 will be made to -1.
4. Add noise (AWGN) to the output of the BPSK modulated data.
5. Send it to the Decision decoding channel by setting the threshold as 0 and get the estimated code vector.
6. Now send it to the Hard and Soft decision decoding and get the estimated message vector.
7. Now compare the estimated message vector with the input message bits and analyze the BER performance.
8. In hard decision decoding we will take the estimated code vector and divide by the number of repetitions and get the estimated message vector whereas in soft we will take each and every code bit and compare.
9. In the case of Error rate calculations, we have compared if the estimated message vector bit crosses a threshold of (n+1)/2 to consider it as 1 or 0.
10. Then we have manually calculated the different probabilities in error for the (3,1,3) and (5,1,5) and inputted it with the single bit probability.
11. Calculate the BER for different SNR values and plot the curve.

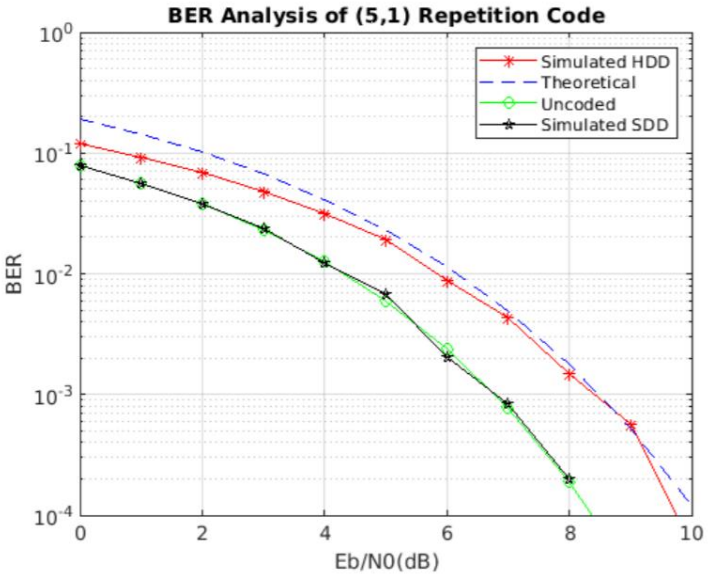
(n, k) Hamming Code:

1. Initializing ‘m’ number of message bits.
2. Assume a Parity Matrix P for parity bits determination.
3. Create Generator Matrix G = [I_k | P], where P = Parity Matrix
4. Create Parity Check Matrix H = [P^T | I_{n-k}]
5. Create Codeword C = [D]*[G] for every 4-bit data vector D.
6. Take the output codewords and apply BPSK modulation where +1 will be as +1 and 0 will be made to -1.
7. Add noise (AWGN) to the output of the BPSK modulated data.
8. R = 7-bit received vector corresponding to a message vector.
9. Calculate Syndrome S = [R]*[H^T]
10. If S = 0, then No error
11. Else If S ≠ 0, then flipping the encoded bit that is in the position of the column in H that matches the syndrome will result in a valid code word. This is Hard Decision Decoding.
12. For soft decision decoding, find the Hamming distance between the received codeword with all the possible codewords. The decoded word is the codeword with minimum Hamming distance with the received word.
13. Compare the decoded words with input message words to calculate errors.
14. Calculate the BER for different SNR values and plot the curve.

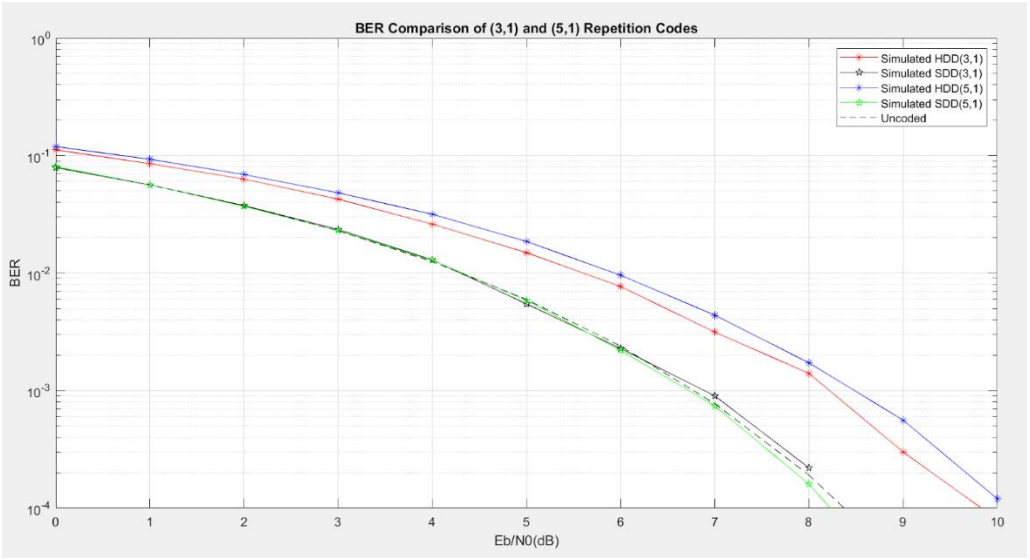
RESULTS AND INFERENCES



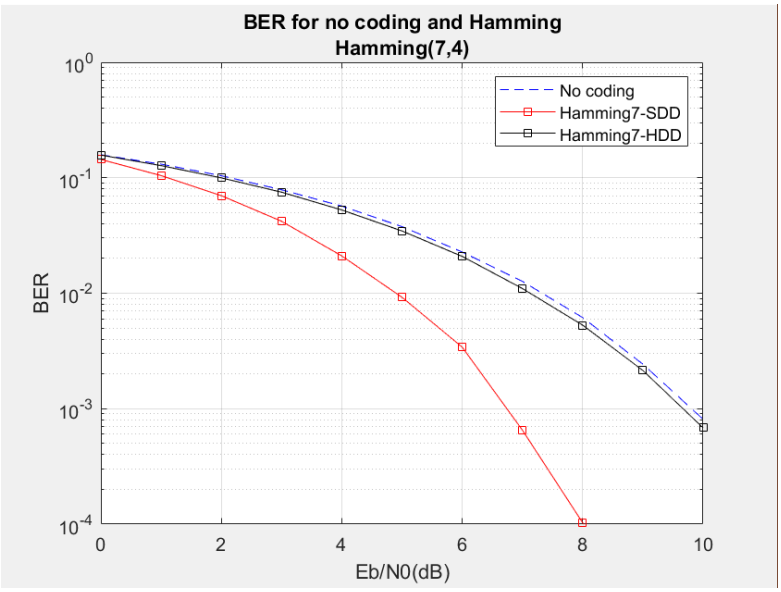
It is observed that the BER of Hard Decision Decoded (3,1) repetition coded data is rather poor than the Uncoded data. Also, the Hard Decision Decoded data has almost the same performance as theoretically coded data. The Soft Decision Decoded data has slightly better performance than the Uncoded data.



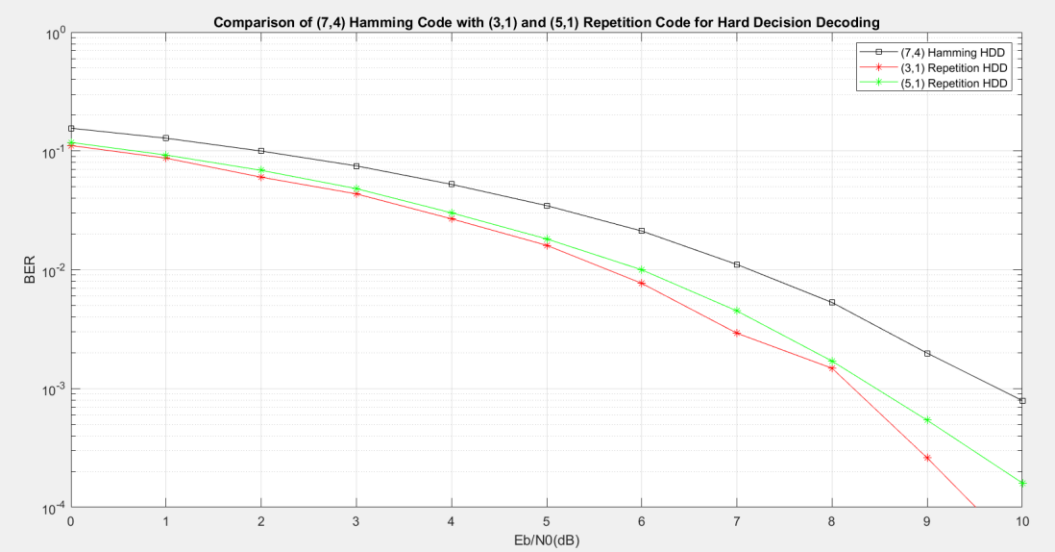
It is observed that the BER of Hard Decision Decoded (5,1) repetition coded data is also poor than the Uncoded data. But the Hard Decision Decoded data has better performance than theoretically coded data. The Soft Decision Decoded data has slightly better performance than the Uncoded data and large improvement over Hard Decision Decoded data.



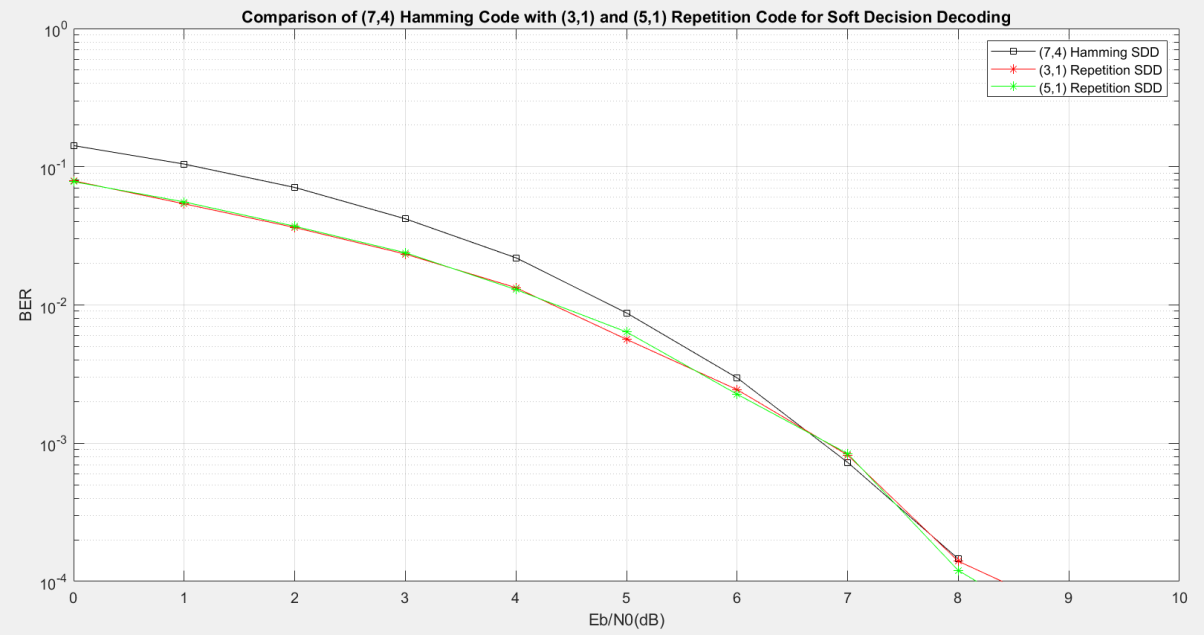
On comparing the BER performance of (3,1) repetition code with (5,1) repetition code, it is observed that (3,1) code has better performance than (5,1) code for the case of Hard Decision Decoding (HDD). However (5,1) code is slightly better than (3,1) code for the case of Soft Decision Decoding (SDD).



The (7,4) Hamming code has an overall better performance than an uncoded BPSK transmission system. The Soft Decision Decoded (SDD) data has less errors compared to the Hard Decision Decoded (HDD) data, hence it has a better error correcting performance.



On plotting the BER curves for all coding techniques, i.e. (7,4) Hamming code, (3,1) Repetition code and (5,1) Repetition code, for the case of Hard decision decoding, we observe that (3,1) Repetition code performs best while (7,4) Hamming code performs worst.



For the case of Soft Decision Decoding, the performance of (7,4) Hamming code is again the worst but as the Signal to Noise ratio increases, all the coding techniques converge to a have approximately the same BER performance.

CONCLUSION

We have successfully analyzed the performance of Repetition Codes and Hamming Code by plotting their BER curves. The algorithm for the different linear block codes were developed simulated in MATLAB. From Repetition Codes, it is observed that (3,1) code has better performance than (5,1) code for the case of Hard Decision Decoding (HDD). However (5,1) code is slightly better than (3,1) code for the case of Soft Decision Decoding (SDD).

It is also concluded that (7,4) Hamming code has a significantly better BER performance than uncoded data, which is an advantage over the repetition codes. Comparing all three coding techniques, it is concluded that (7,4) Hamming code has the worst performance while (3,1) and (5,1) Repetition codes have nearly similar performance. For all the linear block coding techniques, soft decision decoding has an overall better BER performance than other techniques.

As a future improvement to this project, different modulation schemes other than BPSK can be analyzed to measure the performance. The channel noise can also be varied according to different channels such as Rician, Rayleigh etc. to have a better comparative analysis.

CODE

(3, 1) Repetition Code:

```
N=50000; %Length of data bit stream
m = randi([0 1],1,N); %Random 0s and 1s
c=[];
%Repetition Coding
for i=1:N
    c=[c m(i) m(i) m(i)]; %Repeated symbol 3 times
end
x=[];
%BPSK Mapping
for i=1:length(c)
    if c(i)==0
        x(i)= -1;
    else
        x(i)= 1;
    end
end
r=1/3; %Code rate
rep = 3;%Block size
BER_sim=[];
BER_th_C = [];
BER_UC = [];
BER_sim_SDD=[];
for EbN0dB=0:10
    EbN0=10.^(EbN0dB/10);
    sigma = sqrt(1/(2*r*EbN0)); %Noise Standard deviation
    n = sigma.*randn(1,length(x));%Random Noise(AWGN) with adjusted
variance
    y=x+n; %Received symbol = Transmitted + Noise
    c_cap=(y>0);%If y is positive, c_cap=1, otherwise c_cap=0
    m_cap=[];
    % Hard Decision Coding (decision of a bit is not dependent on
other
    % bits)
    m_cap_SDD=[];%Soft Decision Coding (Dependent on neighbouring
bits)
    for j=1:(length(c_cap)/rep)
        code = c_cap((j-1)*rep+1:j*rep); %Storing one block of
symbols in single variable (3 bits)
        if sum(code)>=2
            code1=1;
        else
            code1=0;
        end
    m_cap = [m_cap code1];
    code_SDD=y((j-1)*rep+1:j*rep);
        if sum(code_SDD)>0
            code2=1;
        else
            code2=0;
        end
            m_cap_SDD=[m_cap_SDD code2];
```

```
    end
    noe = sum(m~=m_cap); %Number of Errors HDD
    ber_sim1 = noe/N;
    BER_sim=[BER_sim ber_sim1]; %Appending the BER values in an
array
    noe1= sum(m~=m_cap_SDD); %Number of Errors SDD
    ber_sim2= noe1/N;
    BER_sim_SDD=[BER_sim_SDD ber_sim2]; %Appending the BER values in
an array
    p = qfunc(sqrt(2*r*EbN0)); %Single bit Probability of Error in
Coded BPSK
    ber_th_q= 3*p*p*(1-p)+p^3; %2-bit Error + 3-bit Error
    BER_th_C = [BER_th_C ber_th_q]; %Theoretical BER for Coded BPSK
    BER_UC = [BER_UC 0.5*erfc(sqrt(EbN0))]; %BER for Uncoded BPSK
end
EbN0dB = 0:10;
semilogy(EbN0dB,BER_sim,'r*-',EbN0dB,BER_th_C,'b--
',EbN0dB,BER_UC,'go-',EbN0dB,BER_sim_SDD,'b^-');
title("BER Analysis of (3,1) Repetition Code");
xlabel('Eb/N0(dB)');
ylabel('BER');grid on;
legend("Simulated HDD","Theoretical","Uncoded","Simulated SDD");
axis([min(EbN0dB) max(EbN0dB) 10^-4 10^0]);
```

(5, 1) Repetition Code:

```
N=50000; %Length of data bit stream
m = randi([0 1],1,N); %Random 0s and 1s
c=[];
%Repetition Coding
for i=1:N
    c=[c m(i) m(i) m(i) m(i) m(i)]; %Repeated symbol 3 times
end
x=[];
%BPSK Mapping
for i=1:length(c)
    if c(i)==0
        x(i)= -1;
    else
        x(i)= 1;
    end
end
r=1/5; %Code rate
rep = 5;%Block size
BER_sim=[];
BER_th_C = [];
BER_UC = [];
BER_sim_SDD=[];
for EbN0dB=0:10
    EbN0=10.^(EbN0dB/10);
    sigma = sqrt(1/(2*r*EbN0)); %Noise Standard deviation
    n = sigma.*randn(1,length(x));%Random Noise(AWGN) with adjusted
    variance
    y=x+n; %Received symbol = Transmitted + Noise
```

```
    c_cap=(y>0);%If y is positive, c_cap=1, otherwise c_cap=0
    m_cap=[];
    % Hard Decision Coding (decision of a bit is not dependent on
other bits)
    m_cap_SDD=[];%Soft Decision Coding (Dependent on neighbouring
bits)
    for j=1:(length(c_cap)/rep)
    code = c_cap((j-1)*rep+1:j*rep); %Storing one block of symbols
in single variable (3 bits)
    if sum(code)>=3
        code1=1;
    else
        code1=0;
    end
    m_cap = [m_cap code1];
    code_SDD=y((j-1)*rep+1:j*rep);
        if sum(code_SDD)>0
            code2=1;
        else
            code2=0;
        end
    m_cap_SDD=[m_cap_SDD code2];
    end
noe = sum(m~=m_cap); %Number of Errors HDD
ber_sim1 = noe/N;
BER_sim=[BER_sim ber_sim1]; %Appending the BER values in an array
noe1= sum(m~=m_cap_SDD); %Number of Errors SDD
ber_sim2= noe1/N;
BER_sim_SDD=[BER_sim_SDD ber_sim2]; %Appending the BER values in an
array
p = qfunc(sqrt(2*r*EbN0)); %Single bit Probability of Error in Coded
BPSK
ber_th_q= 5*(p^4)*(1-p) + p^5+ 10*(1-(p^2))*(p^3); %3-bit Error +
4-bit Error
BER_th_C = [BER_th_C ber_th_q]; %Theoretical BER for Coded BPSK
BER_UC = [BER_UC 0.5*erfc(sqrt(EbN0))]; %BER for Uncoded BPSK
end
EbN0dB = 0:10;
semilogy(EbN0dB,BER_sim,'r*-',EbN0dB,BER_th_C,'b--
',EbN0dB,BER_UC,'go-',EbN0dB,BER_title("BER Analysis of (5,1)
Repetition Code");
xlabel('Eb/N0(dB)');
ylabel('BER');grid on;
legend("Simulated HDD","Theoretical","Uncoded","Simulated SDD");
axis([min(EbN0dB) max(EbN0dB) 10^-4 10^0]);
```

(7, 4) Hamming Code:

```
close all;
clear;
clc;

N=50000; %number of iterations
n1=7; k1=4; %Hamming (7,4)

%----------------------Hamming (7,4)-------------------------
[h,g]=hammgen(n1-k1); %create hamming parity and generator matrix
dictII=mod(de2bi(0:(2^k1)-1)*g,2); %create dictionary with all
possible messages and their parity bits matrix(2^k1,n1)

% Create random bitstreams messages matrix N streams-messages of
size k
msg4=(sign(randn(N,k1))+1)/2;
msg4_m=msg4; %modulated messages
msg4_m(msg4_m==0)=-1; %BPSK Mapping
% Create hamming coded messages
coded_w7=mod(msg4*g,2);
coded_w7(coded_w7==0) =-1; % modulation one bit per symbol
EbN0dB= [0:1:10]; %SNR vector represented in dB
EbN0=10.^(EbN0dB. /10);

%---BER vectors----
ber_noCoding=zeros(1,length(EbN0));
ber_7sdd=zeros(1,length(EbN0));
ber_7hdd=zeros(1,length(EbN0));

for i=1:length(EbN0)
  yI=(sqrt(EbN0(i))*msg4_m)+randn(N,k1);
  yII=(sqrt(EbN0(i))*coded_w7)+randn(N,n1); % channel out signal
with AWGN (Nxn1) matrix
  for j=1:N
    %%--------No coding case----------
    yI_hd=yI(j,:);
    yI_hd(yI_hd>0)=1;
    yI_hd(yI_hd<=0)=0;

ber_noCoding(i)=ber_noCoding(i)+length(find(yI_hd~=msg4(j,:)))/k1;

    %Soft Decision Decoding
    for m=1:length(dictII)
      distanceII(m)=norm(yII(j,:)-dictII(m,:));%Hamming distance
between received word and coded word
    end
    [minv,mini]=min(distanceII);
    decoded_w7=dictII(mini,:); %get the coded word with minimum
distance
    cw7=coded_w7(j,:);
    cw7(cw7==-1)=0; %revert modulation to calculate error between
decoded and original coded message
    ber_7sdd(i)=ber_7sdd(i)+length(find(decoded_w7~=cw7))/n1;
```

```
    % Hard Decision Decoding
    yII_hd=yII(j,:);
    yII_hd(yII_hd>0)=1;
    yII_hd(yII_hd<=0)=0;
    zII=h*yII_hd';% calculate syndrome
    for g1=1:length(yII_hd)
      if (zII==h(:,g1))
        yII_hd(g1)=~yII_hd(g1); %fix mistake using syndrome
      end
    end
    ber_7hdd(i)=ber_7hdd(i)+length(find(yII_hd~=cw7))/n1;
  end
end
%%__|Final average BER|_______
ber_noCoding=ber_noCoding/N;
ber_7sdd=ber_7sdd/N;
ber_7hdd=ber_7hdd/N;
figure(1)
semilogy(EbN0dB,ber_noCoding,"b--")
hold on;
semilogy(EbN0dB,ber_7sdd,"r-s")
hold on;
semilogy(EbN0dB,ber_7hdd,"k-s")
grid on;
title({"BER for no coding and Hamming","Hamming(7,4)"});
ylabel({"BER"});
xlabel({"Eb/N0(dB)"});
legend("No coding","Hamming7-SDD","Hamming7-HDD")
axis([min(EbN0dB) max(EbN0dB) 10^-4 10^0]);
```

REFERENCES

1. Kaur, Amandeep & Kaur, Gagandeep & Singh, Navdeep. (2017). A Review: BER Performance Analysis of Linear Block codes used in FEC over AWGN channel.
2. P. R. Kamala and R. V. S. Satyanarayana, "Optimal linear block code modeling and performance analysis over various channels for use in wireless communications," 2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS), 2016, pp. 1-6, doi: 10.1109/ICETETS.2016.7603038.
3. A. K. Singh, "Error detection and correction by hamming code," 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), 2016, pp. 35-37, doi: 10.1109/ICGTSPICC.2016.7955265.
4. A. A. Ali and I. A. Al-Kadi, "On the use of repetition coding with binary digital modulations on mobile channels," in IEEE Transactions on Vehicular Technology, vol. 38, no. 1, pp. 14-18, Feb. 1989, doi: 10.1109/25.31130.
5. http://www.inference.org.uk/mackay/itprnn/1997/l1/node7.html
6. http://www.inference.org.uk/mackay/itprnn/1997/l1/node6.html