Digital Communication Lab

Laboratory report submitted for the partial fulfillment
of the requirements for the degree of

Bachelor of Technology
in
Electronics and Communication Engineering

by

Mohit Akhouri - 19ucc023

Course Coordinator
Dr. Nikhil Sharma

LNMIIT
The LNM Institute of Information Technology

Department of Electronics and Communication Engineering
The LNM Institute of Information Technology, Jaipur

September 2021

Contents

Chapter *7*

Experiment - 7

7.1 Name of the Experiment

To perform encoding and decoding for a (7,4) Hamming code

7.2 Software Used

- MATLAB

7.3 Theory

7.3.1 About Hamming Code :

In computer science and telecommunication, **Hamming codes** are a family of **linear error-correcting codes**. Hamming codes can detect **one-bit** and **two-bit** errors, or correct one-bit errors without detection of uncorrected errors. By contrast, the simple parity code cannot correct errors, and can detect only an odd number of bits in error. Hamming codes are perfect codes, that is, they achieve the highest possible rate for codes with their block length and **minimum distance of three**. **Richard W. Hamming** invented Hamming codes in 1950 as a way of automatically correcting errors introduced by punched card readers. In his original paper, Hamming elaborated his general idea, but specifically focused on the **Hamming(7,4) code** which adds **three parity bits** to **four bits of data**.

In mathematical terms, Hamming codes are a class of **binary linear code**. For each integer r $>=$ 2 there is a code with block length n = 2^r - 1 and message length k = 2^r - r - 1. Hence the rate of Hamming codes is R = k / n = 1 - r / (2^r - 1), which is the highest possible for codes with minimum distance of three (i.e., the minimal number of bit changes needed to go from any code word to any other code word is three) and block length 2^r - 1.

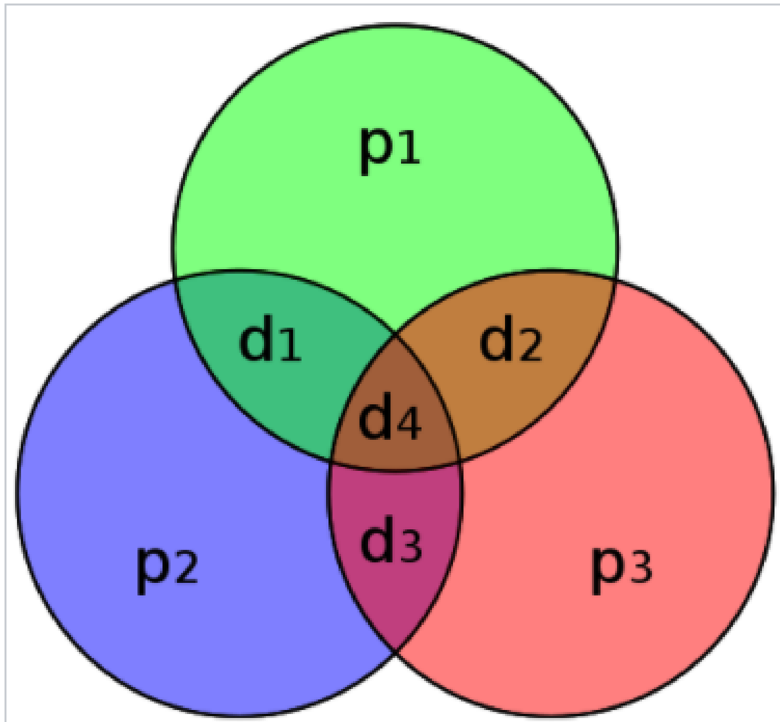

Figure 7.1 Graphical depiction of the four data bits and three parity bits - Hamming(7,4)

7.3.1.1 About (7,4) Hamming Code :

In **1950**, Hamming introduced the [7,4] Hamming code. It encodes **four data bits** into **seven bits** by adding **three parity bits**. It can detect and correct single-bit errors. With the addition of an overall parity bit, it can also detect (but not correct) double-bit errors.

The (7,4) Hamming Code is given by the generator matrix :

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{7.1}$$

The matrix **G** is called the **generator** of **canonical** matrix of linear (n,k) code. The matrix **H** is called **parity-check matrix**. The matrix **H** is given as :

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{7.2}$$

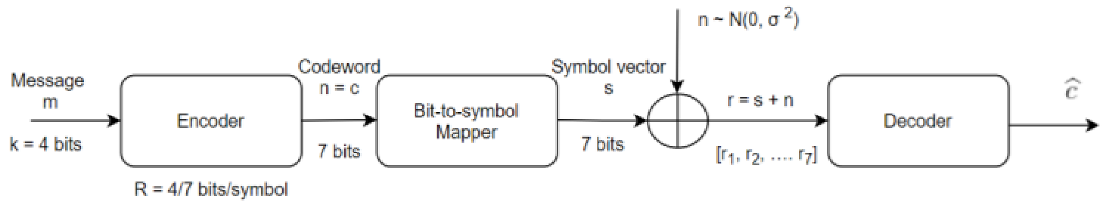

Figure 7.2 Block diagram of hamming code

7.3.1.2 About Hard Decision Decoder :

In **hard decision decoder** , we find codeword closest in **Hamming distance**. In this type of decoder , we will find distance of **b** from 2^k codeword. In this type of decoder , **complexity** increases **exponentially** with k.

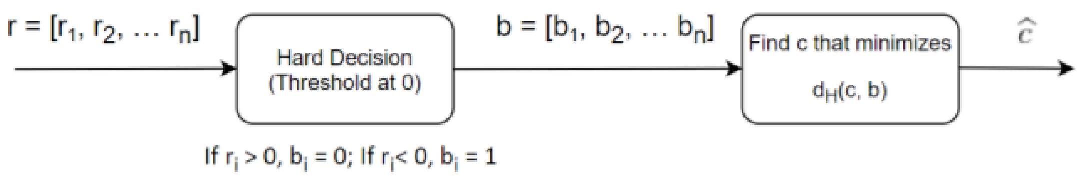

Figure 7.3 Hard Decision Decoder

7.3.1.3 About Soft Decision Decoder :

In **soft decision decoder** , we find codeword closest in **Euclidean distance**. In this type of decoder , we will find distance with 2^k code-symbol vectors . In this type of decoder , **Complexity** is **more** than hard decision decoder.

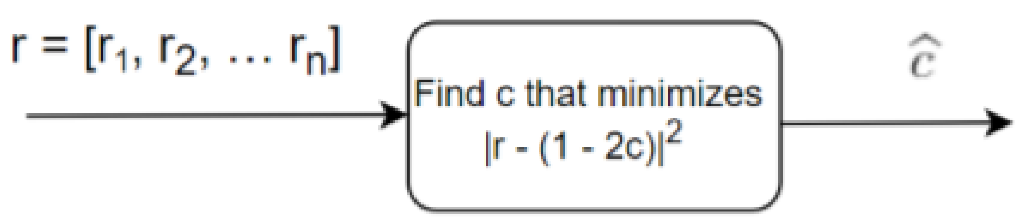

Figure 7.4 Soft Decision Decoder

7.4 Code and Results

7.4.1 Encoding and Decoding of (7,4) Hamming Code :

```
% 19ucc023
% Mohit Akhouri
% Observation - Performing the encoding and decoding for a (7,4)
 Hamming
% Code and finding BER for different values of SER

clc;
clear all;
close all;

% This code will perform the encoding and decoding of (7,4) Hamming
 Code
% The Hamming code is given by the generator function G and we will
 use the
% block diagram of encoder and decoder to perform the operations. We
 will
% use two types of decoder - SOFT DECISION decoder and HARD DECISION
% decoder.

% Finally we will find the values of BER for different SER and also
 plot
% the graph between SER and BER for hard and soft decision decoder.

N = 10000; % Size of input sequence

n = 7; % columns of Generating matrix G for Hamming Code
k = 4; % rows of Generating matrix G for Hamming Code

G = zeros(4,7); % Initializing the generating matrix G
G=[1 0 0 0 1 0 1 ; 0 1 0 0 1 1 1 ; 0 0 1 0 1 1 0 ; 0 0 0 1 0 1 1]; %
 Defining the generating matrix G

SER_dB = (0:7); % Defining the array to store SER values in dB from
 0dB to 7dB
SER_array = zeros(1,8); % array to store the SER values converted from
 dB to unitless

% Loop to calculate unitless value of SER
for i=1:size(SER_dB)
    SER_array(i) = 10^(SER_dB(i)/10);
end

size_SER = size(SER_array,2); % to store the number of elements in the
 SER_array

message = zeros(16,4); % to store the Message bits ( 0000 to 1111 )

% Loop to calculate the message in terms of bits and storing them in
% message matrix for further computation
for i=1:16
    str = dec2bin(i-1,4); % To get the binary equivalent of decimal
 number'i-1'
```

Figure 7.5 Part 1 of the Code for Encoding and Decoding of (7,4) Hamming Code

```
    temp_array = zeros(1,4); % To store the bits of the binary
 equivalent obtained from above

    % Loop to store the bits in the correct position of temp_array
    for j=1:size(str,2)
        if(str(j)=='0')
            temp_array(j) = 0;
        else
            temp_array(j) = 1;
        end
    end
    message(i,:) = temp_array; % to store the encoded bits in the
 message matrix
end

codeword = mod(message*G,2); % To store the corresponding codewords to
 each bit sequence of message matrix

% Displaying the bit sequence and corresponding codeword
disp('The codewords are as follows :');
disp(sprintf('%-15s \t %-15s','Message','Codeword'));
for i=1:16
    disp(sprintf('%-15s \t
 %-15s',int2str(message(i,:)),int2str(codeword(i,:))));
end

codeword_modf = codeword; % Initializing array to store the modified
 codewords
codeword_modf(codeword_modf==0) = -1; % modify the codewords by
 comparing them to 0 and then assigning -1

INFO_mat = randi([0,1],N,4,size_SER); % to store the information to be
 transmitted through AWGN channel

H = [G(:,k+1:n)',eye(n-k)]; % To store the H matrix obtained by
 transpose of G matrix and multiplying with Identity matrix I

% Loop to calculate the information ( bit sequence ) to be transmitted
for i=1:size_SER
    code_transm(:,:,i) = mod((INFO_mat(:,:,i)*G),2);
end

code_transm_modf = code_transm; % To store the modified transmitted
 codeword
code_transm_modf(code_transm_modf==0) = -1; % Modification of
 transmitted codeword done by comparing to 0 and then assigning -1

% Loop to determine the information ( bit sequence ) received
for i=1:size_SER
    code_recv(:,:,i) = awgn(code_transm_modf(:,:,i),SER_dB(i));
end

hard_dec_decoder = ones(size(code_recv)); % Initializing array to
 store the decoded bit sequence from hard decision decoder
```

Figure 7.6 Part 2 of the Code for Encoding and Decoding of (7,4) Hamming Code

```
hard_dec_decoder(code_recv<0) = 0; % Parity checking and modifying the
 hard_dec_decoder array

INFO_hard_dec = zeros(size(INFO_mat)); % To store the received
 information through hard decision decoder
% Main loop algorithm for calculation of bit error rate in case of
 hard
% decision decoder by comparing the distances
for h=1:size_SER
    for i=1:N
        dist = zeros(1,2^k); % to store the distance of codewords

        % loop to calculate the distance of codewords
        for j=1:2^k
            dist(j) = norm(mod(codeword(j,:) +
 hard_dec_decoder(i,:,h),2),1);
        end

        % finding minimum distance index
        [min_elem,ind] = min(dist);
        INFO_hard_dec(i,:,h) = message(ind,:);
    end
    BER_hard_dec(h) = length(find(INFO_hard_dec(:,:,h) -
 INFO_mat(:,:,h)))/(4*N); % Computing the bit error rate for hard
 decision decoder
end

% Displaying the values obtained for the BER for different values of
 SER
% (in dB ) for hard decision decoder
disp('BER value for different values of SER for hard decision decoder
 is as follows :');
disp(sprintf('%-8s \t %-8s','SER (dB)','BER'));
for i=1:8
    disp(sprintf('%-8d \t %-8f',SER_dB(i),BER_hard_dec(i)));
end

INFO_soft_dec = zeros(size(INFO_mat)); % To store the received
 information through soft decision decoder
% Main loop algorithm for calculation of bit error rate in case of
 soft
% decision decoder by comparing the distances
for h=1:size_SER
    for i=1:N
        dist = zeros(1,2^k); % to store the distance of codewords

        % loop to calculate the distance of codewords
        for j=1:2^k
            dist(j) = norm((codeword_modf(j,:) - code_recv(i,:,h)),2);
        end

        % finding minimum distance index
        [min_elem,ind] = min(dist);
        INFO_soft_dec(i,:,h) = message(ind,:);
```

Figure 7.7 Part 3 of the Code for Encoding and Decoding of (7,4) Hamming Code

```
    end
    BER_soft_dec(h) = length(find(INFO_soft_dec(:,:,h) -
 INFO_mat(:,:,h)))/(4*N); % Computing the bit error rate for soft
 decision decoder
end

% Displaying the values obtained for the BER for different values of
 SER
% (in dB ) for soft decision decoder
disp('BER value for different values of SER for soft decision decoder
 is as follows :');
disp(sprintf('%-8s \t %-8s','SER (dB)','BER'));
for i=1:8
    disp(sprintf('%-8d \t %-8f',SER_dB(i),BER_soft_dec(i)));
end

% Plotting graph of Bit error rate ( BER ) vs. Symbol error rate
 ( SER )
figure;
semilogy(SER_dB,BER_hard_dec,'red');
hold on;
semilogy(SER_dB,BER_soft_dec,'blue')
xlabel('SER (dB) ->');
ylabel('BER (dB) ->');
title('19ucc023 - Mohit Akhouri','Plots of Bit Error Rate ( BER )
 vs. Symbol Error Rate ( SER in dB ) for HARD and SOFT decision
 decoders');
legend('Hard Decision Decoder','Soft Decision Decoder');
grid on;
hold off;
```

Figure 7.8 Part 4 of the Code for Encoding and Decoding of (7,4) Hamming Code

```
The codewords are as follows :
Message              Codeword
0  0  0  0           0  0  0  0  0  0  0
0  0  0  1           0  0  0  1  0  1  1
0  0  1  0           0  0  1  0  1  1  0
0  0  1  1           0  0  1  1  1  0  1
0  1  0  0           0  1  0  0  1  1  1
0  1  0  1           0  1  0  1  1  0  0
0  1  1  0           0  1  1  0  0  0  1
0  1  1  1           0  1  1  1  0  1  0
1  0  0  0           1  0  0  0  1  0  1
1  0  0  1           1  0  0  1  1  1  0
1  0  1  0           1  0  1  0  0  1  1
1  0  1  1           1  0  1  1  0  0  0
1  1  0  0           1  1  0  0  0  1  0
1  1  0  1           1  1  0  1  0  0  1
1  1  1  0           1  1  1  0  1  0  0
1  1  1  1           1  1  1  1  1  1  1
```

Figure 7.9 Display of **Message** and corresponding **codeword**

```
BER value for different values of SER for hard decision decoder is as follows :
SER (dB)     BER
0            0.139775
1            0.100200
2            0.072525
3            0.047625
4            0.022400
5            0.011400
6            0.003600
7            0.001700
```

Figure 7.10 Display of the BER values for different SER values (in dB) for Hard Decision Decoder

```
BER value for different values of SER for soft decision decoder is as follows :
SER (dB)     BER
0            0.102750
1            0.063375
2            0.041000
3            0.023775
4            0.008925
5            0.003400
6            0.000900
7            0.000300
```

Figure 7.11 Display of the BER values for different SER values (in dB) for Soft Decision Decoder

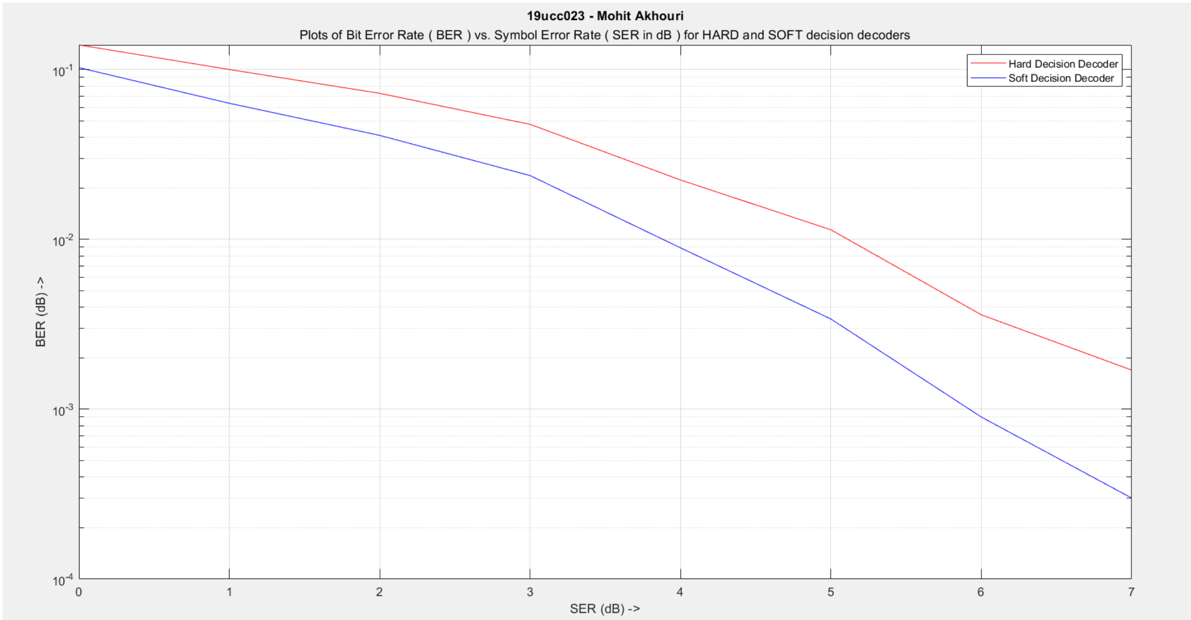

Figure 7.12 Plot of the BER vs. SER (dB) for Hard Decision Decoder and Soft Decision Decoder

7.4.2 Observation Table for BER vs. SER (dB) for Hard Decision and Soft Decision Decoders :

SER (in dB)	BER for Hard Decision Decoder	BER for Soft Decision Decoder
0	0.139775	0.102750
1	0.100200	0.063375
2	0.072525	0.041000
3	0.047625	0.023775
4	0.022400	0.008925
5	0.011400	0.003400
6	0.003600	0.000900
7	0.001700	0.000300

Table 7.1 BER for different values of SER for Hard and Soft Decision Decoders

7.5 Conclusion

In this experiment , We learnt about the **(7,4) Hamming Code** and how to implement it in MATLAB. We learnt about the block diagram of the Hamming Code for **encoding**. We also learnt about the **received** code after passing through AWGN channel and about two types of decoders - **Hard Decision Decoder** and **Soft Decision Decoder**. We also learnt about the concepts of **Euclidean distance** and **Hamming distance** and how they can be utilized in the calculation of distance. We also found BER values for different SER values (in dB) for both hard and soft decision decoders. We also plotted the graph between SER and BER. We also learnt about new MATLAB functions like **norm**,**mod** and **awgn** to name a few.