DIGITAL COMMUNICATION LAB

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

Bachelor of Technology
in
Communication and Computer Engineering

by

Abhay Agrawal - Roll No. 20UCC002

Course Coordinator
Mr. Dharm Pal Yadav

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

November 2022

Contents

Chapter *1*

Experiment - 8

1.1 AIM

Performance analysis of Linear Block Codes/Repetition Coding.

1.2 Software USED

MATLAB

1.3 Theory

1.3.1 Linear Block Codes

Linear block codes are so named because each code word in the set is a linear combination of a set of generator code words. If the messages are k bits long, and the code words are n bits long (where n ¿ k), there are k linearly independent code words of length n that form a generator matrix. To encode any message of k bits, you simply multiply the message vector u by the generator matrix to produce a code word vector v that is n bits long. Linear block codes are very easy to implement in hardware, and since they are algebraically determined, they can be decoded in constant time. They have very high code rates, usually above 0.95. They have low coding overhead, but they have limited error correction capabilities. They are very useful in situations where the BER of the channel is relatively low, bandwidth availability is limited in the transmission, and it is easy to retransmit data. One class of linear block codes used for high-speed computer memory are SEC/DED (single-errorcorrecting/double-error-detecting) codes. In high speed memory, bandwidth is limited because the cost per bit is relatively high compared to low-speed memory like disks . The error rates are usually low and tend to occur by the byte so a SEC/DED coding scheme for each byte provides sufficient error protection. Error coding must be fast in this situation because high throughput is desired. SEC/DED codes are extremely simple and do not cause a high coding delay.

1.4 CODE and RESULT

```
clc;
clear all;
close all;
data = randi([0 1],16,4); %message bit generation
G = [1 1 0 1 0 0 0; 0 1 1 0 1 0 0; 1 1 1 0 0 1 0; 1 0 1 0 0 0 1] ;

for i=1:16
    ones = 0;
    for j=1:4
        if data(i,j) == 1
            ones = ones+1;
        end
    end
    if ones == 0
        data(i,3) = 1;
    end

end

B = mod(data*G,2) ; %codeword generation
%Hamming Wt and min Hamming wt
hamWt = zeros(1,16);
minWt = 10;
for i=1:16
    ones = 0;
    for j=1:7
        if B(i,j) == 1
            ones = ones+1;
        end
    end
    hamWt(i) = ones;
end
N = randi([0 1], 16, 7) ;
%R = mod(B+N, 2) ;
R = [1 0 1 1 0 0 0] ;
I = eye(3) ;
P = zeros(4, 3) ;
for i = 1:4
    for j = 1:3
        P(i,j) = G(i,j) ;
    end
end
H = [I, transpose(P)] ; %generate H Matrix
H_ = transpose(H) ; %generate H(transpose) Matrix
syn = mod(R*H_, 2) ; %syndrome generation
R1 = R ;
%code for error generation
for i = 1:length(H_)
    flag = 1 ;
```

```
    for j = 1:3
        if (H_(i,j) ~= syn(j))
        flag = 0 ;
        break ;
        end
    end
    if flag == 1
        Rl(i) = mod(1+Rl(i),2) ;
        disp("6) Error on index ") ; %error position
        disp(i) ;
        break ;
    end
end
disp("1) Message bit") ;
disp(data) ;
disp("2) Codeword") ;
disp(B) ;
disp("3)Hamming Wt: ");
disp(hamWt);
disp("10) min Hamming Wt: ");
disp(min(hamWt));
disp("4) H Matrix") ;
disp(H) ;
disp("5) H Transpose") ;
disp(H_) ;
disp("7) Syndrome") ;
disp(syn) ;
disp("8) R") ;
disp(R) ;
disp("9) R'") ;
disp(Rl) ;
```

```
6) Error on index
     5

1) Message bit
     1     0     1     0
     1     1     1     0
     1     1     1     1
     1     0     0     0
     0     0     1     1
     0     0     0     1
     1     0     1     1
     0     0     1     0
     0     1     1     0
     1     0     0     0
     1     0     1     1
     1     1     0     1
     0     0     1     0
     0     0     0     1
     0     1     0     1
     1     0     0     1

2) Codeword
```

```
     0     0     1     1     0     1     0
     0     1     0     1     1     1     0
     1     1     1     1     1     1     1
     1     1     0     1     0     0     0
     0     1     0     0     0     1     1
     1     0     1     0     0     0     1
     1     0     0     1     0     1     1
     1     1     1     0     0     1     0
     1     0     0     0     1     1     0
     1     1     0     1     0     0     0
     1     0     0     1     0     1     1
     0     0     0     1     1     0     1
     1     1     1     0     0     1     0
     1     0     1     0     0     0     1
     1     1     0     0     1     0     1
     0     1     1     1     0     0     1

3)Hamming Wt:
  Columns 1 through 13

     3     4     7     3     3     3     4     4     3     3     4     3     4

  Columns 14 through 16

     3     4     4

10) min Hamming Wt:
     3

4) H Matrix
     1     0     0     1     0     1     1
     0     1     0     1     1     1     0
     0     0     1     0     1     1     1

5) H Transpose
     1     0     0
     0     1     0
     0     0     1
     1     1     0
     0     1     1
     1     1     1
     1     0     1

7) Syndrome
     0     1     1

8) R
     1     0     1     1     0     0     0

9) R'
     1     0     1     1     1     0     0
```

1.5 CONCLUSION

1. Linear block code is very easy to implement then Convolutional block code.

2. Convolutional code is better than linear block code and concatenated code.

1.6 PRECAUTION

1. Check the connections before switching on the kit.

2. Connections should be done properly.

3. Observation should be taken properly.