# LAB -REPORT 4.2

**-ABHIGNA KUSUMBA(IMT2014028)**

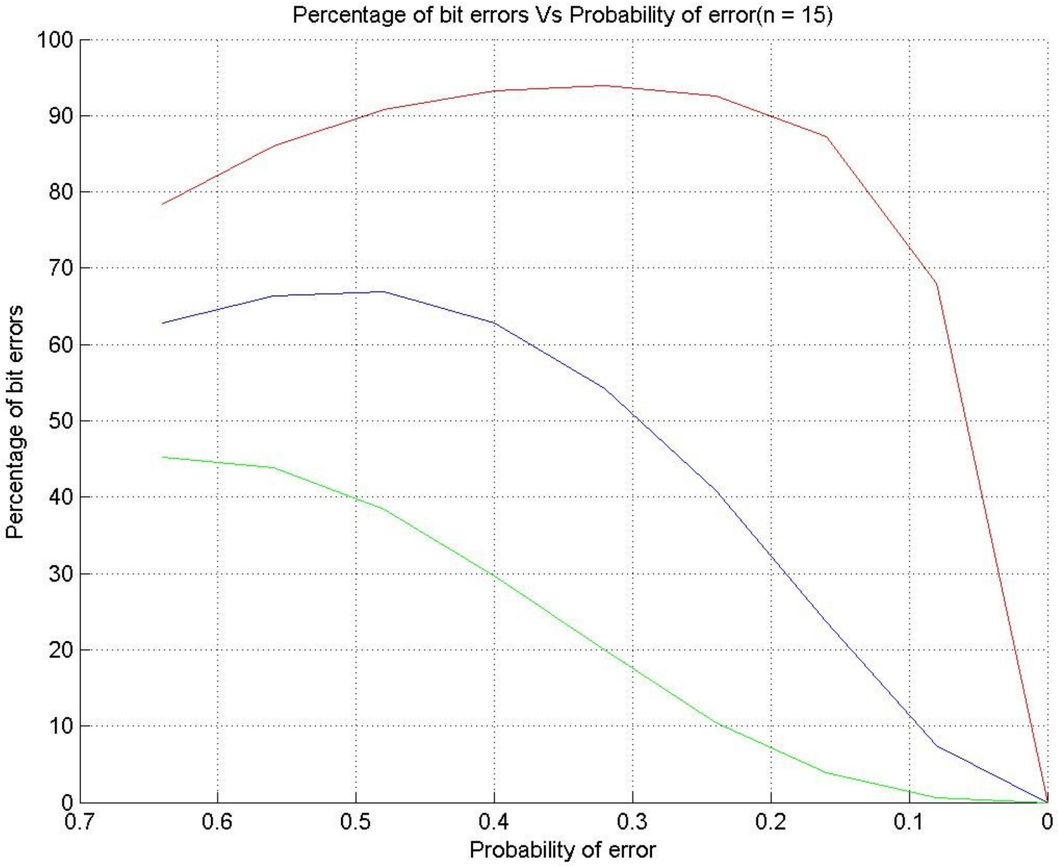
**1.** Number of one, two and three bit errors for an input size of $10^7$ were determined for different probabilities.

Red – 1 bit errors

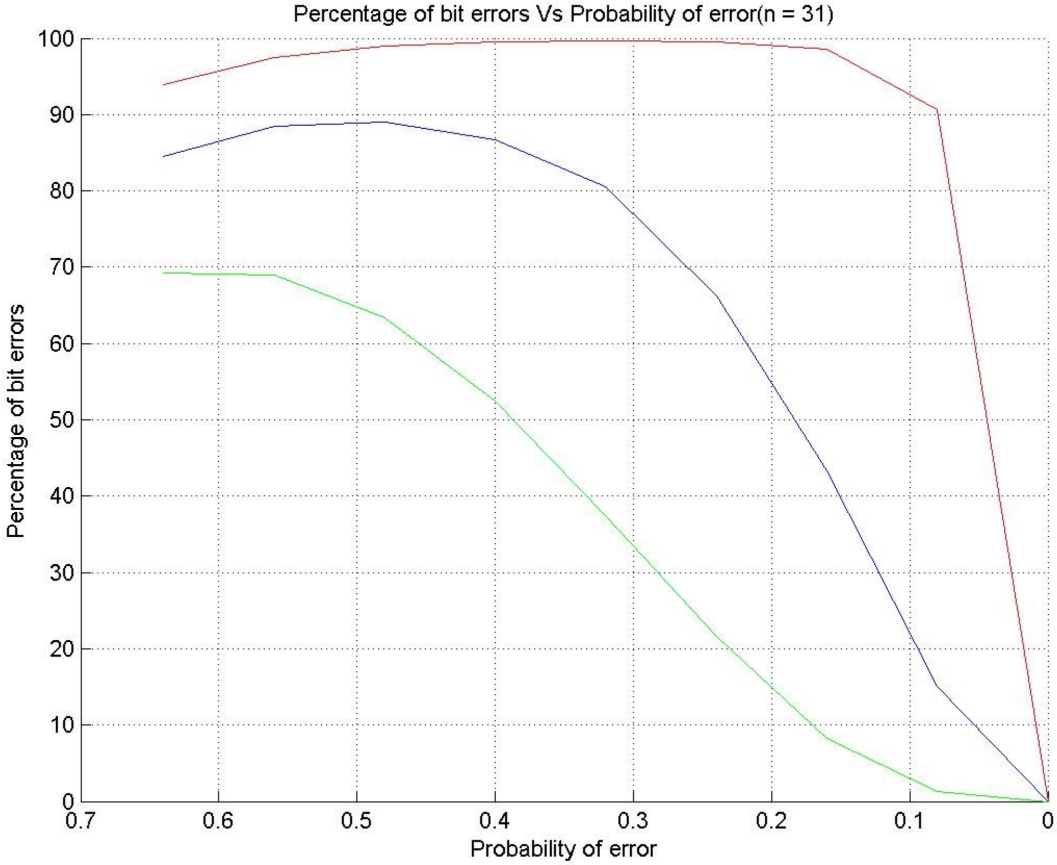
Blue – 2 bit errors

Green – 3 bit errors

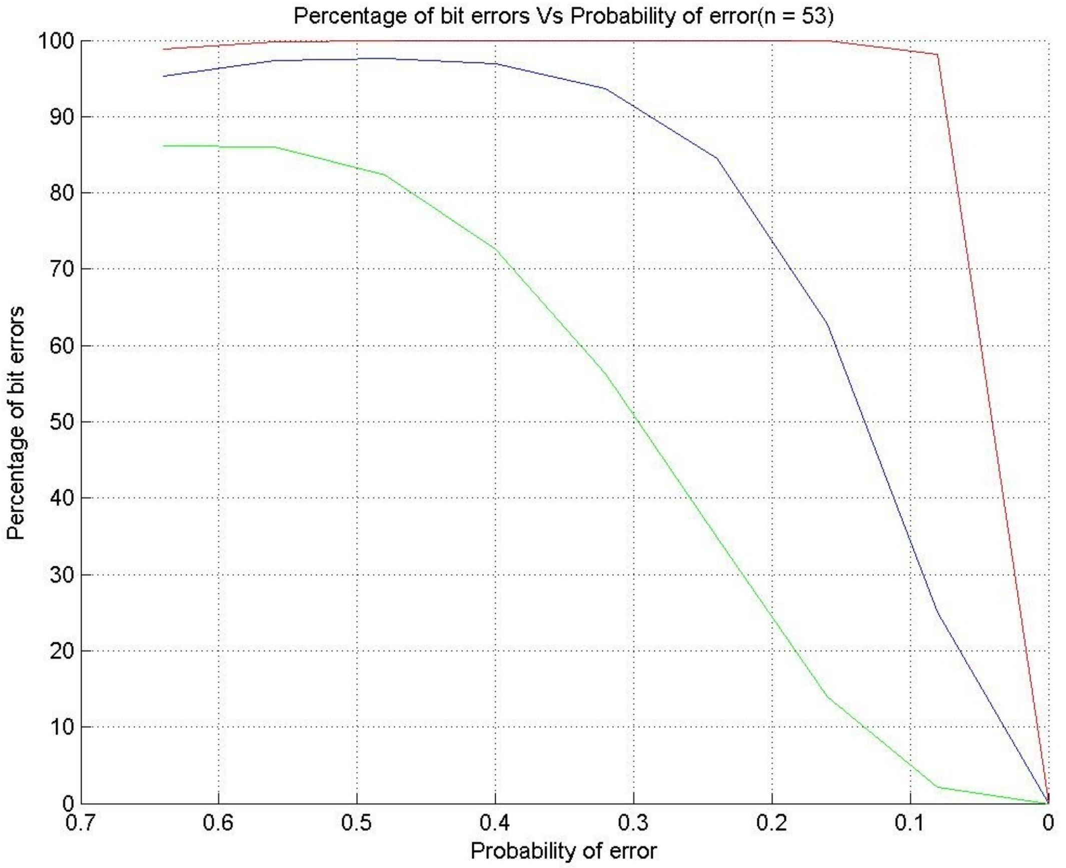
**Case-1:**



## Case-2:


Percentage of bit errors Vs Probability of error(n = 31)
Percentage of bit errors
Probability of error
100
90
80
70
60
50
40
30
20
10
0
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

**Case-3:**

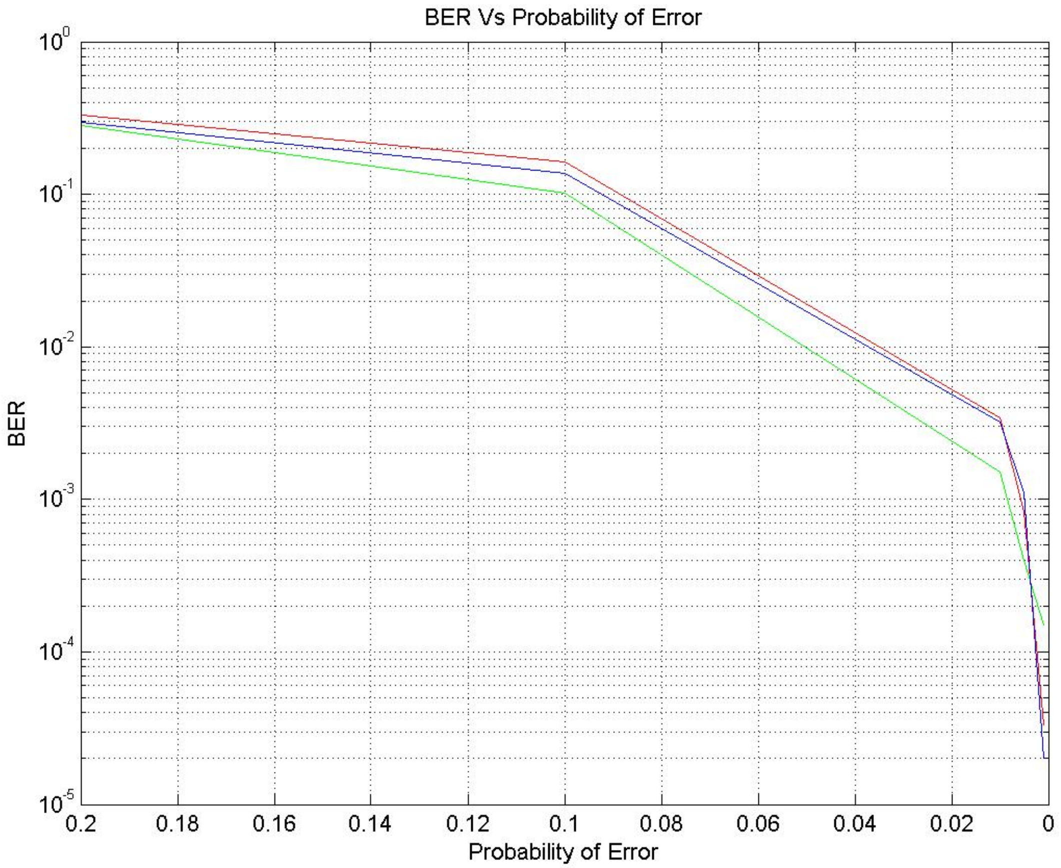


**3**. Output **BER** (semilog scale) Vs input **probability of error (deceasing order – >p = 0.2,0.1,0.01,0.005,0.001)** for each one of the **BCH** encoder-decoder pairs with **input size = $10^6$**

Red – correct 1 bit errors-(15,11)

Blue – correct 2 bit errors-(15,7)

Green – correct 3 bit errors-(15,5)



**Performance** of the 3 codes is compared above.

## Conclusions:

### Question-1:

1.As probability of error increases, percentage of bit errors increased exponentially.

2.1 bit errors are highest in all cases as its probability(p) is greater than $p^2$ ,$p^3$

### Question-2:

1. BER is much lower than probability of error for lower probability of error(<0.01).
2. BER for BCH(15,7) is better than BCH(15,11).
3. BER for BCH(15,5) performs best for lower values of probability.

**Appendix-**

**1.**

```
function []=q1(p,n)
N=[];
file=fopen('input.txt','r');
N = fscanf(file,'%d');
l=length(N);
x=transpose(N);
blocks=ceil(l/n);
percent=[0 0 0];
if ((n*blocks)~=l)
    for i=(l+1):(n*blocks)
        x(i)=0;
    end
end
    errorbits = rand(size(x)) < p;
    y=x;
    y(errorbits) = 1 - y(errorbits);
    err=xor(x,y);
for i=1:blocks
    w=err((((i-1)*n)+1):(n*(i)));
    cs = cumsum(w);
    csOnes = cs(diff([w 0]) == -1);
    seriesOnes = [csOnes(1) diff(csOnes)];
    t = tabulate(seriesOnes);
    e=t(:,1);
    for m=1:length(e)
        if m>3
            break;
        end
        if(e(m)~=0)
           percent(m)=percent(m)+1;
        end
    end
end
```

```
    p=(percent(1)/blocks)*100
    q=(percent(2)/blocks)*100
    r=(percent(3)/blocks)*100
fclose(file);
end
```

**2.**

**Encoder-**

```
function []=bch1(n,k)
u=[];
if (k==11)
    gen=[1 1 0 1];
elseif (k==7)
    gen=[1 0 0 0 1 0 1 1 1];
end
for i=1:n
   u(i)=0;
end
input=[];
file=fopen('input.txt','r');
input = fscanf(file,'%d');
l=length(input)
input=transpose(input);
blocks=ceil(l/k)
for i=(l+1):(k*blocks)
    input(i)=0;
end
indices=[];
for i=1:blocks
    for w=((i*k)-(k-1)):(i*k)
        for j=1:length(gen)
            if(gen(j)==1)
                if(w>length(input))
                    break;
                elseif(input(w)==1)
                    if(u((w-((i-1)*k))+(j-1))==1)
                        u((w-((i-1)*k))+(j-1))=0;
                    else
                        u((w-((i-1)*k))+(j-1))=1;
                    end
                end
            end
        end
    end

end
end
```

**Decoder-**

```
function m = bch_dec(g,t,u)
    r=u;
    r=fliplr(r);
    syms a;
    received_polynomial = poly2sym(r,a);

    galios = [1,2,4,8,3,6,12,11,5,10,7,14,15,13,9];
    ones = find(r==1);
    bin_sum = mod(sum(decimalToBinaryVector(galios(ones),4)),2);
    bin_sum_decimal = binaryVectorToDecimal(bin_sum);

    if(bin_sum_decimal == 1)
        error_location1 = find(galios == bin_sum_decimal);
    else
        error_location1 = find(galios == bin_sum_decimal)-1;
    end

    if(t==1)
        if(bin_sum_decimal == 0)
            disp('No errors');
            recv = fliplr(r)
            return
        end

        if(error_location1 == 1 && bin_sum_decimal == 1)
            error_location1 = 0;
            error_location = [error_location1,1];
        else
            error_location = [error_location1,1];
        end

        error_polynomial = poly2sym(error_location);
        disp('Error location is at position :');
        15-(error_location1)
        r(error_location1+1) = 1 - r(error_location1+1);
        disp('Corrected message : ');
        fliplr(r)
    end

    if(t == 2)
        S3 = mod(sum(decimalToBinaryVector(galios(mod((ones-
1)*3,15)+1),4)),2);
        if(find(galios == binaryVectorToDecimal(S3)) == 1)
            error_location2 = 1;
        else
            error_location2 = find(galios == binaryVectorToDecimal(S3))-1;
        end

        if(error_location1==1)
            s1_cube = 1;
        else
            s1_cube = error_location1*3;
        end

        if(length(mod(s1_cube,15)) == 0 && length(error_location2) == 0)
            disp('No Errors');
            recv = fliplr(r)
            return
        end
        if(mod(s1_cube,15) == 1 && error_location2 == 1)
            sigma2_vector = [0,0];
        elseif (error_location2 == 1)
            sigma2_vector = [error_location2-1 , mod(s1_cube,15)];
        elseif(mod(s1_cube,15) == 1)
            sigma2_vector = [error_location2 , mod(s1_cube,15)-1];
        else
            sigma2_vector = [error_location2 , mod(s1_cube,15)];
        end

        sigma2_result = find(galios ==
binaryVectorToDecimal( mod(sum(decimalToBinaryVector(galios(sigma2_v
ector+1),4)),2)))-1;
        if(error_location1 == 1)
            sigma2 = mod(sigma2_result,15);
        else
            sigma2 = mod(sigma2_result + 15 - error_location1,15);
        end
        error_polynomial2 = poly2sym([sigma2,error_location1,1]);
        for i = 0:14
         if(i == 0)
             if(length(sigma2) == 0)
                root_1 = 0;
             else
                  root_1 = decimalToBinaryVector(galios(sigma2+1),4);
             end
             if(error_location1 == 1)
                 root_2 = decimalToBinaryVector(galios(1),4);
             else
                 root_2 = decimalToBinaryVector(galios(error_location1+1),4);
             end

             root_3 = decimalToBinaryVector(1,4);

             if(root_1 == 0)
                root_sum_binary =
binaryVectorToDecimal(mod(root_2+root_3,2));
             else
                root_sum_binary =
binaryVectorToDecimal( mod(root_1+root_2+root_3,2));
             end

             if(root_sum_binary == 0)
                disp('Error position: ');
                15-i
                r(i+1) = 1 - r(i+1);
             end

         else
```

```
             if(length(sigma2) == 0)
                term1 = 0;
             else
                x_21 = mod(mod(i*2,15)+sigma2,15);
                term1 = decimalToBinaryVector(galios(x_21+1),4);
             end

             if(error_location1 == 1)
                x_11 = mod(mod(i*1,15),15);
             else
                x_11 = mod(mod(i*1,15)+error_location1,15);
             end

             term2 = decimalToBinaryVector(galios(x_11+1),4);
             unity = decimalToBinaryVector(1,4);
             if(term1 == 0)
                locating_polynomial_decimal =
binaryVectorToDecimal( mod(term2+unity,2));
             else
                 locating_polynomial_decimal =
binaryVectorToDecimal(mod(term1+term2+unity,2));
             end

             if(locating_polynomial_decimal == 0)
                 disp('Error position: ');
                 disp(i)
                 r(15-i+1) = 1 - r(15-i+1);
             end
         end

       end
       disp('Corrected message : ');
       recv = fliplr(r)
    end
end
```