****

**MATH 3305**

**PROBABILITY AND RANDOM PROCESSES**

**PROJECT: Part II**

Fethiye Gül DİKAYAK - 17070007008

Sinem VATANSEVER - 16070003004

Instructor

Prof. Volkan RODOPLU

Fall 2020

**PART II-A:**

*Main code:*

function [] = Simulation()%This is the main function

L = 3;

N = 14;

b = GenerateBits(N); %Generates N independent bits {0,1} with equal probability

c = Compression(b); %Makes b = c

x = ChannelEncoder(c,L); %Copies the bits 3 times

y = Channel(x) ; %wrapper function

d = ChannelDecoder(y,L);

b\_hat = Decompression(d); %makes d = b\_hat

end

*Other functions in Simulation Code:*

function b = GenerateBits(N)

b = randi([0 1],1,N);

end

function [b] = Compression(c)

b=c;

end

function x = ChannelEncoder(c,L)

x = RepetitionEncoder(c, L);

end

function x = RepetitionEncoder(c,L)

x = reshape(repmat(c,L,1),1,[]); %Retrieved from: https://in.mathworks.com/matlabcentral/answers/57388-is-there-a-function-in-matlab-that-creates-a-binary-repetition-coder

end

function y = Channel(x)

y = BinarySymmetricChannel(x);

end

function [output] = BinarySymmetricChannel(x)

E = 0;

output = bsc(x,E); %"Retrieved from: https://ch.mathworks.com/help/comm/ref/bsc.html#:~:text=ndata%20%3D%20bsc(data%2Cprobability)%20passes%20the%20binary%20input,array%20in%20GF(2)."

end

function d = ChannelDecoder(y,L)

d = MajorityDecoder(y, L);

end

function d=MajorityDecoder(y,L)

a = reshape(y,L,[]);

m = 0;

for i = 1:length(y)/L

m(i) = mode(a(:,i));

end

d = m;

end

function b\_hat = Decompression(d)

b\_hat = d;

end

**PART II-B:**

***For Part II-B.(A): Testing the MajorityDecoder function individually***

L1 = 1;

y1 = [1];

d1 = MajorityDecoder(y1,L1)

L2 = 3;

y2 = [0 0 0 1 1 1 0 0 0];

y3 = [0 1 0 1 1 1 0 0 0];

y4 = [0 1 0 0 1 1 0 0 1];

d2 = MajorityDecoder(y2,L2)

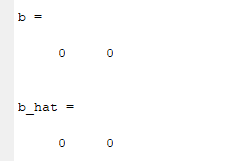
d3 = MajorityDecoder(y3,L2)

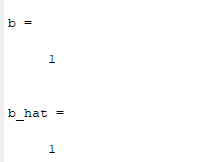
d4 = MajorityDecoder(y4,L2)

*Results:*

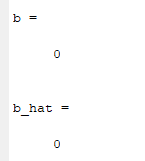
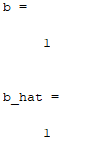


***For Part II-B.(B): Testing the MajorityDecoder function within Simulation***

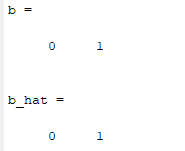
*Results of when N = 1, L = 1, E = 0: Results of when N = 1, L = 1, E = 0:*

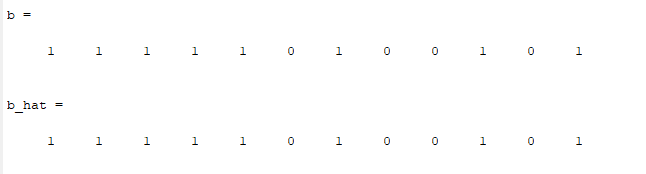


*Results of when N = 1, L = 3, E = 0: Results of when N = 1, L = 5, E = 0:*

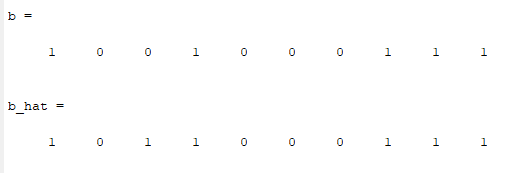


*Results of when N = 2, L = 3, E = 0:*

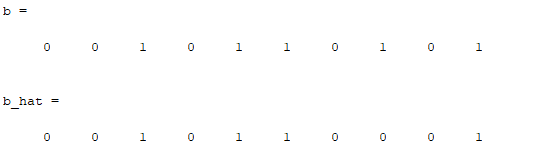


*Results of when N = 12, L = 5, E = 0:*

*Results of when N = 10, L = 1, E = 0.1:*



*Results of when N = 10, L = 3, E = 0.1:*



**PART II-C:**

N = 100;

L = 9;

b = GenerateBits(N);

c = Compression(b);

x = ChannelEncoder(c,L);

y = Channel(x) ;

d = ChannelDecoder(y,L);

b\_hat = Decompression(d);

Matrix = RunningFrequencyComputation(b,b\_hat);

figure(1),

plot(Matrix)

title('Bit Eror Rate Graph')

ylabel('Eror Value')

xlabel('Generated Vector')

function [computation] = RunningFrequencyComputation(b,x)

computation = zeros(1,length(b));

sum = 0;

for i=1:length(b)

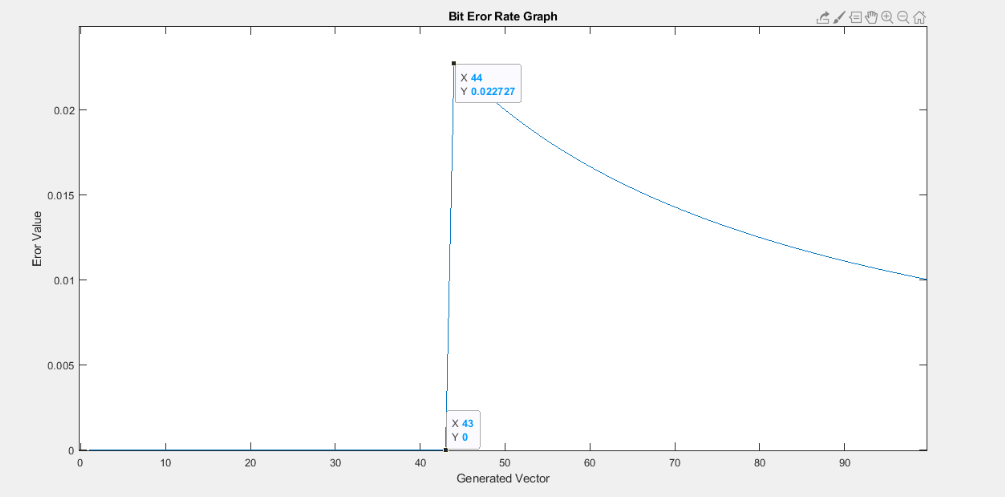
if b(i) ~= x(i)

sum = sum + 1;

end

computation(i) = sum / i;

end



**PART II-D:**

P(E)= P{0 was received | 1 was sent} \* P{1 was sent} + P{1 was received | 0 was sent} \* P{0 was sent}. The GenerateBits function generates 0 and 1 with equal probability, the P{1 was sent} = P{0 was sent} = 1/2 in this case.

N: number of bits to get 0.

p: probability of 0 was received and 1 was sent.

L = 1.

When transmitting a single bit, the probability that the bit will receive an error is 0.1 (the communication system has a bit error rate of 0.1). A three-bit repeat code is used to transmit messages, while L= 3, i.e., 111 is transmitted to send message 1 and 000 is transmitted to send message 0. In the receiver, if two or more 1s are received, the majority decoder decides that message 1 has been sent; if two or more zeros are received, it decides that message 0 has been sent.

L = 3.

While L= 5, i.e., 11111 is transmitted to send message 1 and 00000 is transmitted to send message 0. In the receiver, if three or more 1s are received, the majority decoder decides that message 1 has been sent; if three or more zeros are received, it decides that message 0 has been sent.

L = 5.

* There must be multiple n trials.
* Each of these trials must be independent.
* There will be a success with a probability value of p.

Since these three conditions are met, we have formed the formulation according to the binomial distribution.

, where k=(L+1)/2.

*Matlab Code:*

function out = ErrorofProbability(L,p)

k = (L+1)/2;

sum = 0;

for i = k:L

P\_e(i) = nchoosek(L,i)\*p^i\*(1-p)^(L-i); %https://ch.mathworks.com/help/matlab/ref/nchoosek.html

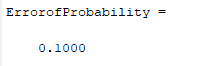
sum = sum + P\_e(i);

end

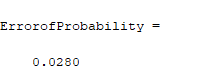
out = sum;

end

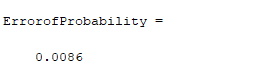
ErrorofProbability = ErrorofProbability(1,0.1)



ErrorofProbability = ErrorofProbability(3,0.1)



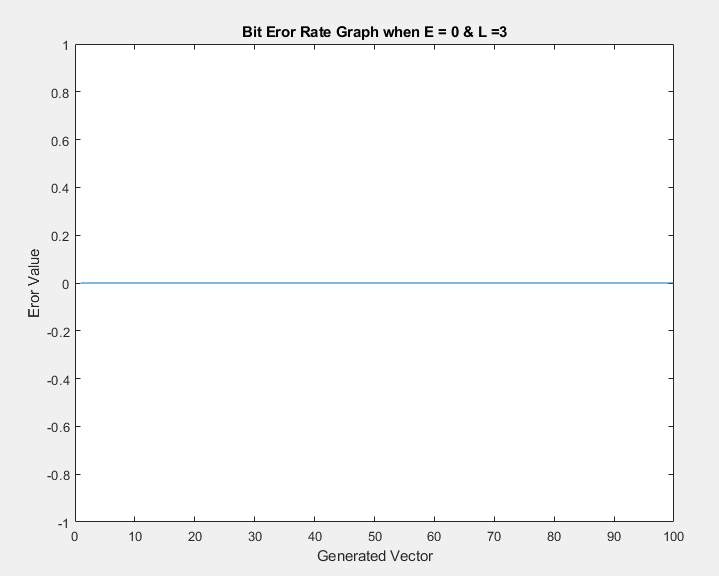
ErrorofProbability = ErrorofProbability(5,0.1)



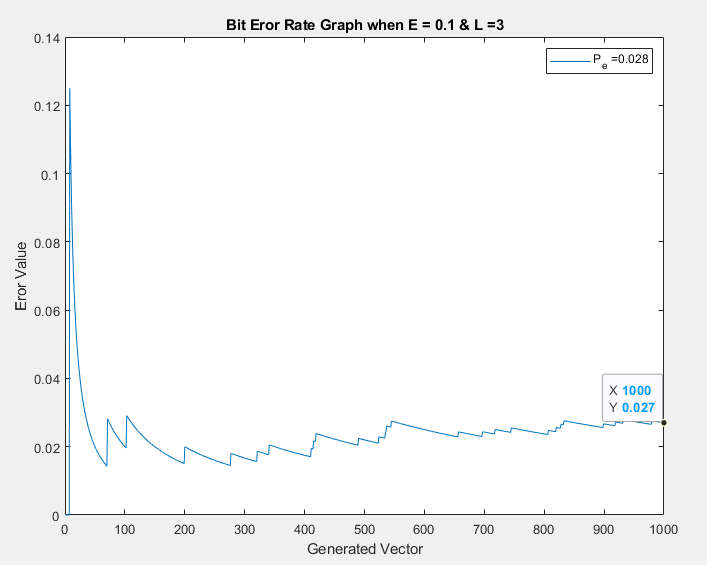
**PART II-E:**

***(A)***We use the code of Part C&D for the evaluation of BER and error of probability.

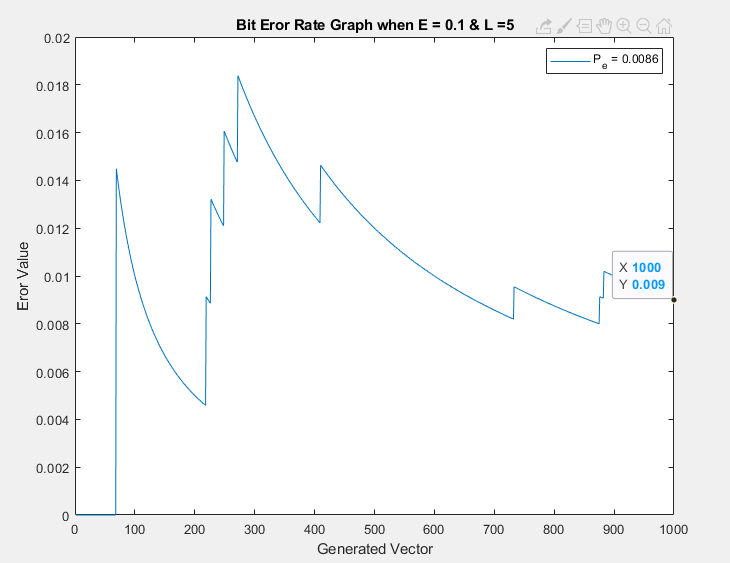
(a) epsilon = 0,    L = 3.



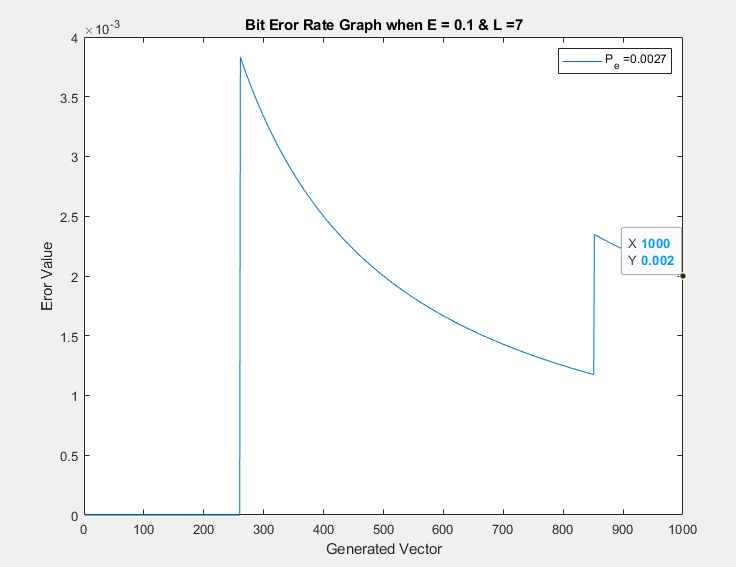
(b) epsilon = 0.1,   L = 3.



(c) epsilon = 0.1, L = 5.



(d) epsilon = 0.1,   L = 7



It was observed that for fixed epsilon, as L becomes larger, the probability of error decreases.

***(B)***

Increasing L means that raised the average number of information bits sent per channel uses. Repetition coding over a long block can decrease data rate and it goes to zero with increasing block length. We can say the data rate and L are inversely proportional. Also, as we observed in Part D the probability of error value and L are inversely proportional, too.

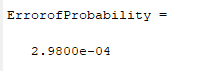
***(C)***

To achieve a probability of error of 0.001 while epsilon = 0.01, we use trial and error method in Matlab by using code of Part-D.

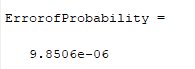
L=1,



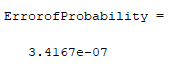
L = 3,



L=5,



L=7,



Therefore; the best option was L = 3.