****

**MATH 3305**

**PROBABILITY AND RANDOM PROCESSES**

**PROJECT: Part III**

Fethiye Gül DİKAYAK - 17070007008

Sinem VATANSEVER - 16070003004

Instructor

Prof. Volkan RODOPLU

Fall 2020

**PART 3.A**

***3.A.1.***

A

1

0

0

1

0

1

0

1

C

E

B

D

A = 0

B = 001

C = 011

D = 101

E = 111

***3.A.2.***

***3.A.3.***

a -> 000

b->  100

c->  101

d->  001

e->  010

**PART 3.B**

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

L = 3;

N = 10;

probVector = [0.4 0.05 0.2 0.15 0.2];

symbol = {'A','B','C','D','E'};

dict = huffmandict(symbol,probVector);

b = GenerateInputSymbols(N,probVector) %This function replaces the GenerateBits(N) function.

c = Compression(b, dict);

x = ChannelEncoder(c, L);

y = Channel(x);

d = ChannelDecoder(y, L);

b\_hat = Decompression(c, dict)

end

function b = GenerateInputSymbols(N,probVector)

Objects = {'A','B','C','D','E'};

idx = randp(probVector, 1, N); %https://uk.mathworks.com/matlabcentral/fileexchange/8891-randp

b = Objects(idx); %https://www.mathworks.com/matlabcentral/answers/373489-random-combinations-of-symbols-restored

end

function c = Compression(b,probVector)

c = HuffmanEncoder(b,probVector);

end

function c = HuffmanEncoder(b, dict)

c = huffmanenco(b,dict);%https://www.mathworks.com/help/comm/ref/huffmanenco.html

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,probVector)

b\_hat = HuffmanDecoder(d,probVector);

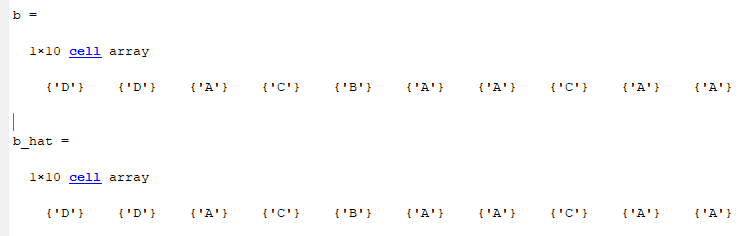
end

function b\_hat = HuffmanDecoder(d,dict)

b\_hat = huffmandeco(d,dict);%https://www.mathworks.com/help/comm/ref/huffmandeco.html

end

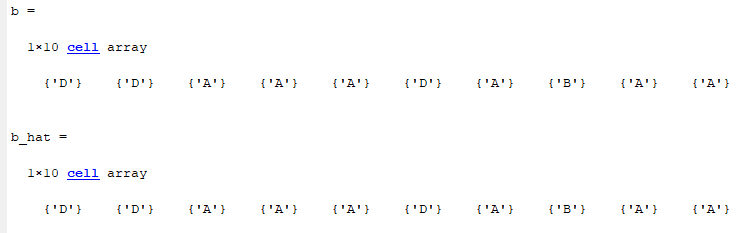
**PART 3.C**



Constructing the following system:

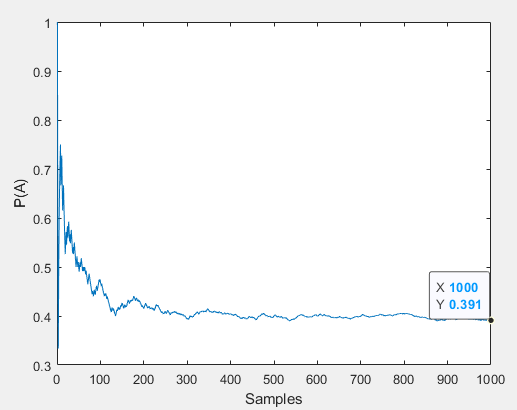
b = GenerateInputSymbols(N, probVector);

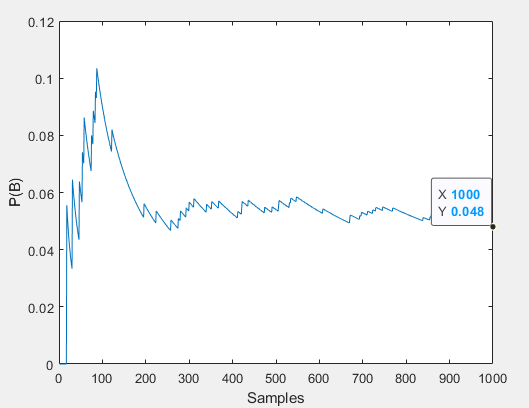
c = Compression(b, probVector);

b\_hat = Decompression(c, probVector);

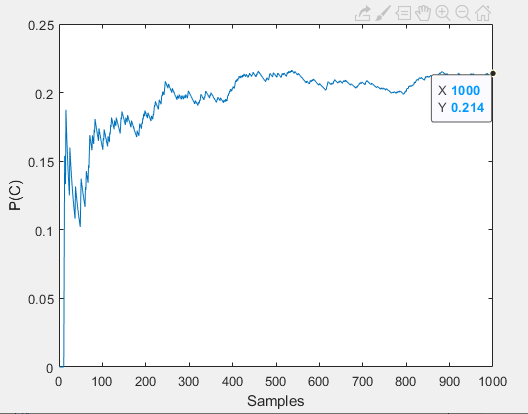
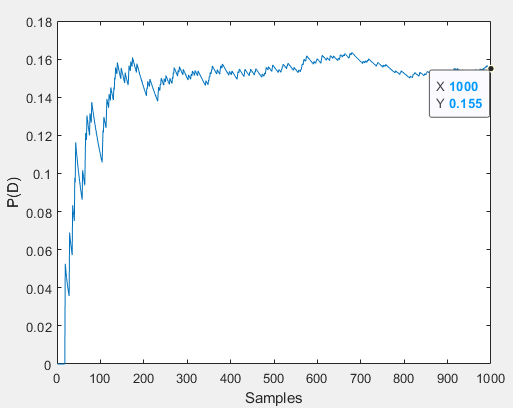
**PART 3.D.1**

To understand convergence easily N is chosen as N = 1000. Running frequency of each character is shown below and it is seen that codes are working correctly. P(A) converges to 0.4:



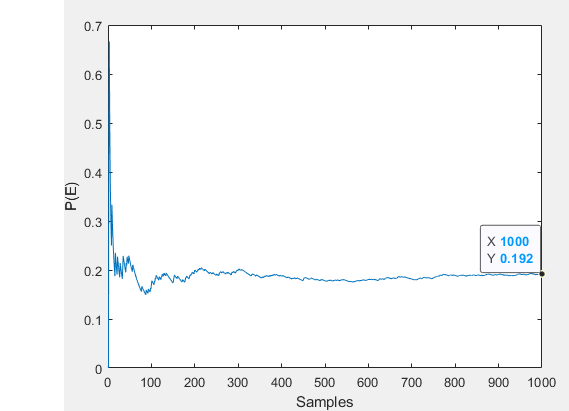


P(B) converges to 0.05:



P(D) converges to 0.15:

P(C) converges to 0.2:

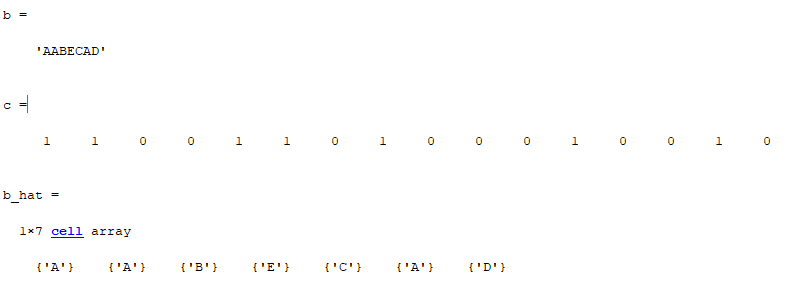


P(E) converges to 0.2:

**PART 3.D.2**



**PART 3.D.3**



huffmandict() command generate bits as:

'A' 1

'B' [0,0,1,1]

'C' [0,0,0]

'D' [0,0,1,0]

'E' [0,1]