In the name of God

University of Tehran
Faculty of Engineering
ECE

Digital Communication Systems

CA1

Reza Jahani
810198377

Spring 14001

Index

Abstract

In this project we aim to simulate sources with and without memory using the functions we code first which are meant to calculate the entropy and length of symbols. In addition, a simple comparison between obtained results will be conducted.

Problem 1) Entropy Function

In this section we are going to write a function named *entropy* which calculates the entropy of G_k using the transition probability matrix and value of k which stands for length of the generated symbol.

We already know

$$G_k = \frac{H(X_1, X_2, \dots, X_k)}{k}$$

Based on what we learned in lectures we can use this relation below:

$$H(X_1, X_2, \dots, X_k) = H(s_0) + kH(s_1|s_0)$$

We drop the third term which was $H(s_0|X_0, X_1, X_2, \dots, X_k)$ because we are simulating and not analyzing.
For this purpose, we first find the state vector and then rest of the code is left with calculation of G_K

Based on the specifications above, the function is written as *entropy*.

- Function is saved as file ***entropy.m***

Problem 2) Average Length Function

In this section we attempt to write a function to calculate the average length of words that we code a symbol chain to. For this purpose, we use *chain* and *k* as input arguments of the function respectively standing for the symbol chain and k long words. For this purpose, we employ a simple algorithm. Starting with checking if the chain is dividable by k or not, we get rid of additional symbols if necessary.
After this action, the chain is divided into k word symbols. Finding the unique symbols which can be generated by the source and calculating their probabilities we find the average length of the Huffman code using the function huffmandict.

Rest of the algorithm and the code can be observed in the file.

- The function is saved as ***average_length.m***

Problem 3) Simulating a source with memory

In this section we aim to simulate a source which has memory. Before simulation in MATLAB platform, we try to calculate the entropy of the source using analysis we learnt through the lectures. Based on the diagram presented in the problem:

$$\varphi = \begin{pmatrix} 0.5 & 0.5 \\ 0.8 & 0.2 \end{pmatrix}$$

To find out the steady state vector we will use below equation:

$$\varphi^T . P = P$$

$$\begin{pmatrix} 0.5 & 0.8 \\ 0.5 & 0.2 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \end{pmatrix} = \begin{pmatrix} P_1 \\ P_2 \end{pmatrix}$$

$$P_1 + P_2 = 1$$

Solving above equations, below results will be obtained:

$$P = \begin{pmatrix} P_1 \\ P_2 \end{pmatrix} = \begin{pmatrix} \frac{8}{13} \\ \frac{5}{13} \end{pmatrix}$$

Now that we have P, we can use below equation to find entropy of the source.

$$H(X) = \sum_i P_i H_i$$

$$H_i = -\sum_j p_{ij} \log(p_{ij})$$

Using above equations, we will get:

$$H_1 = h_b(0.5) = 1$$

$$H_2 = h_b(0.8) = 0.7129$$

$$H(X) = \frac{8}{13} * 1 + \frac{5}{13} * 0.7129 = 0.8930$$

In conclusion based on above calculations entropy of the source will be:

$$H(X) = 0.8930 \frac{bit}{symbol}$$

After this process we attend to create 10000000 symbols based on the given source in the problem. To create a random chain of symbol with length of 10000000 we employ a simple algorithm which uses a random number as the probability indicator and chooses the first state known as S_0.
After that, using another probability indicator the machine itself chooses how to transit from one state to another and generate a symbol through this transition. Using a loop which iterates 10 billion times a chain will be generated.

It is already obvious how we calculated G_K and average length using the functions we implemented before. Explained below we can observe how coding efficiency is formed and calculated.

$$\eta_N = \frac{H(X)}{\widehat{H_N}}$$

$$\widehat{H_N} = \frac{\sum_i p_i n_i}{N} \ \ where\ n_i is\ length\ and\ p_i\ is\ the\ following\ probability\ for\ symbol\ i$$

$$\sum_i p_i n_i \ \ is\ average\ code\ length$$

Using transition states and the functions we have already we can reach results below:

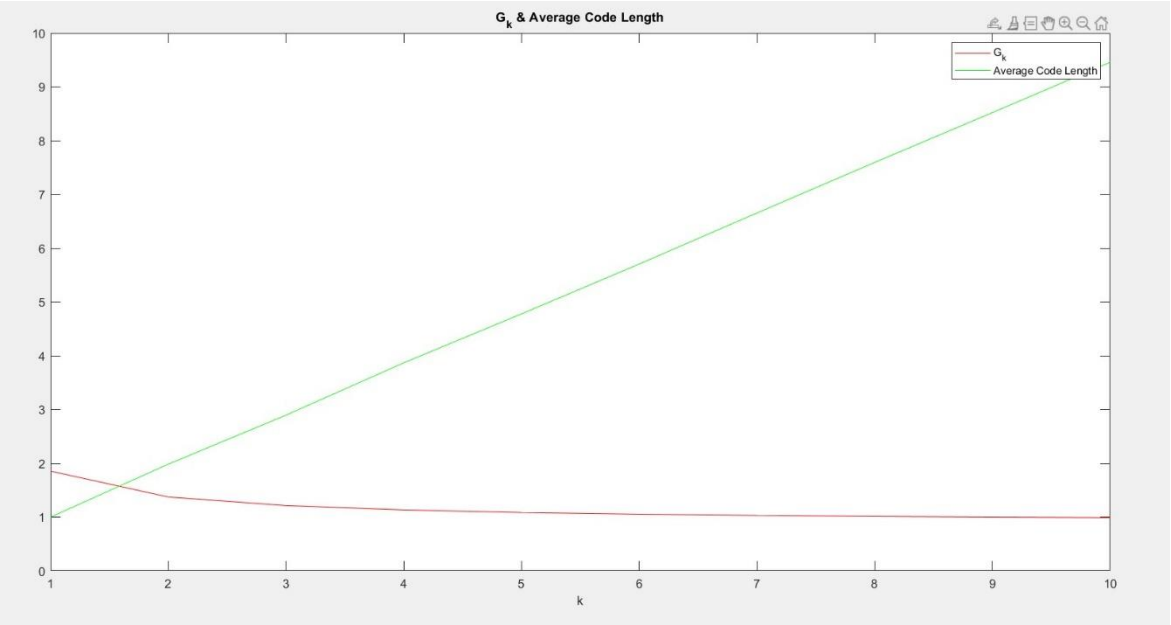

Figure 1: G_K, Average Length

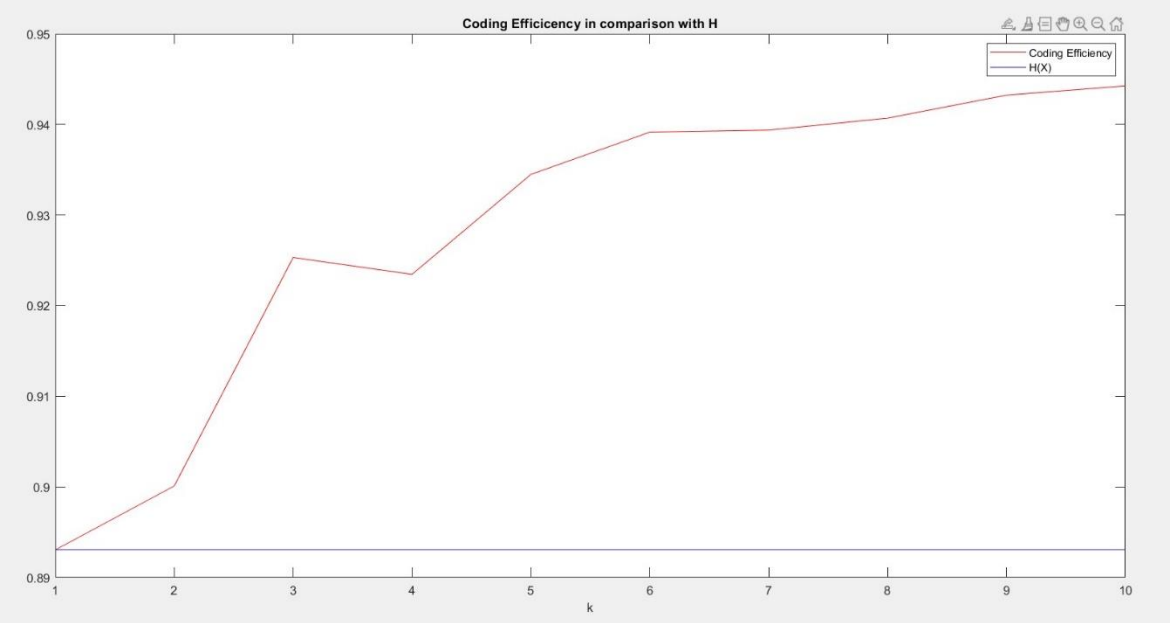

Figure 2: Coding Efficiency, H(X)

As it can be observed, coding efficiency increases with k. G_K decreases and converges to H(X) and average length of the code increases with k.

- The code file is ***P3.m***

Problem 4) Memoryless source simulation

In this section we aim to simulate a memoryless source. For this purpose, we create a random chain for each source X, X^2, X^3. To do this we calculate the probability of each symbol which can be generated by the source.

After this process we calculate the entropy of each source.

Once we have calculated entropy of each source and created a random chain generated by each of the 3 sources, we employ the functions we have already written to calculate G_K and other parameters. In the end we plot the results.

The results are shown below:

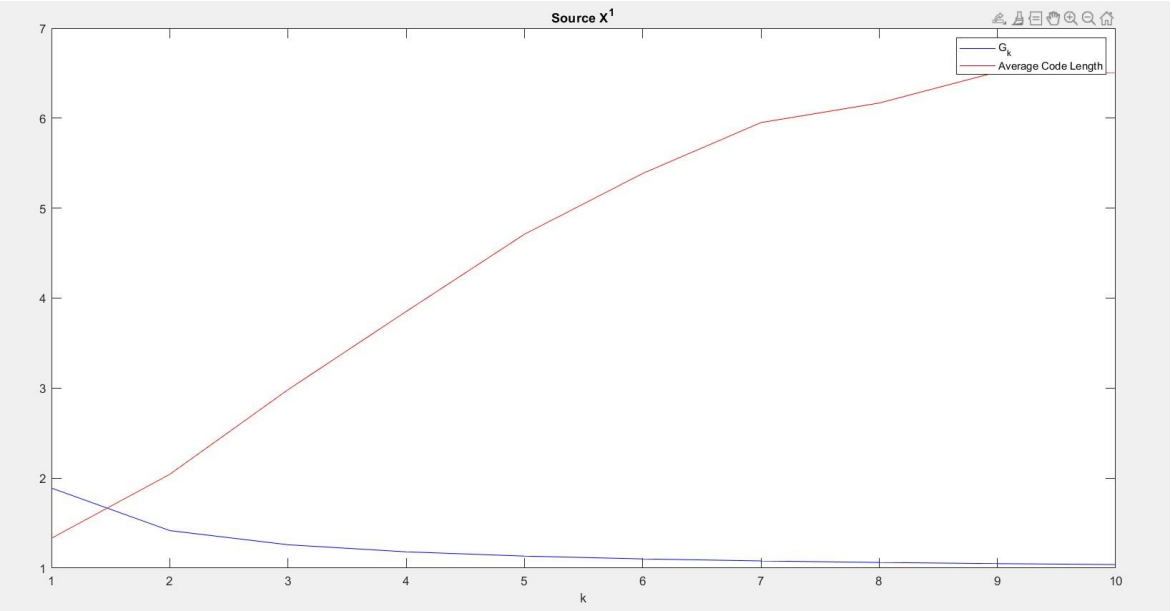

Figure 3: G_K, Average Length of X

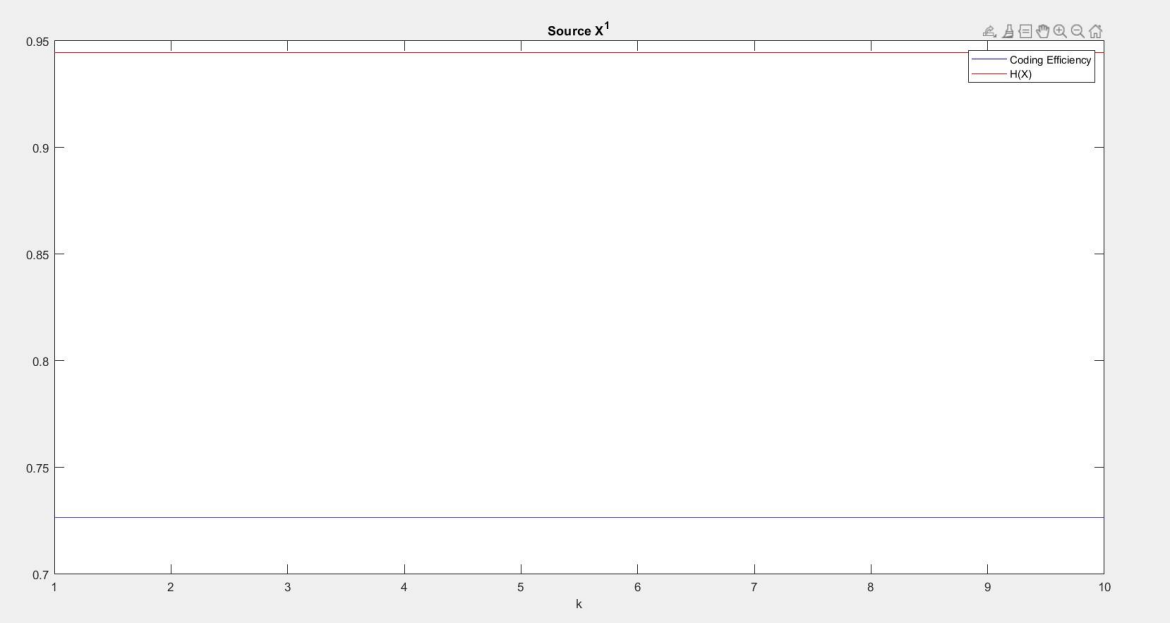

Figure 4: Efficiency and entropy for X

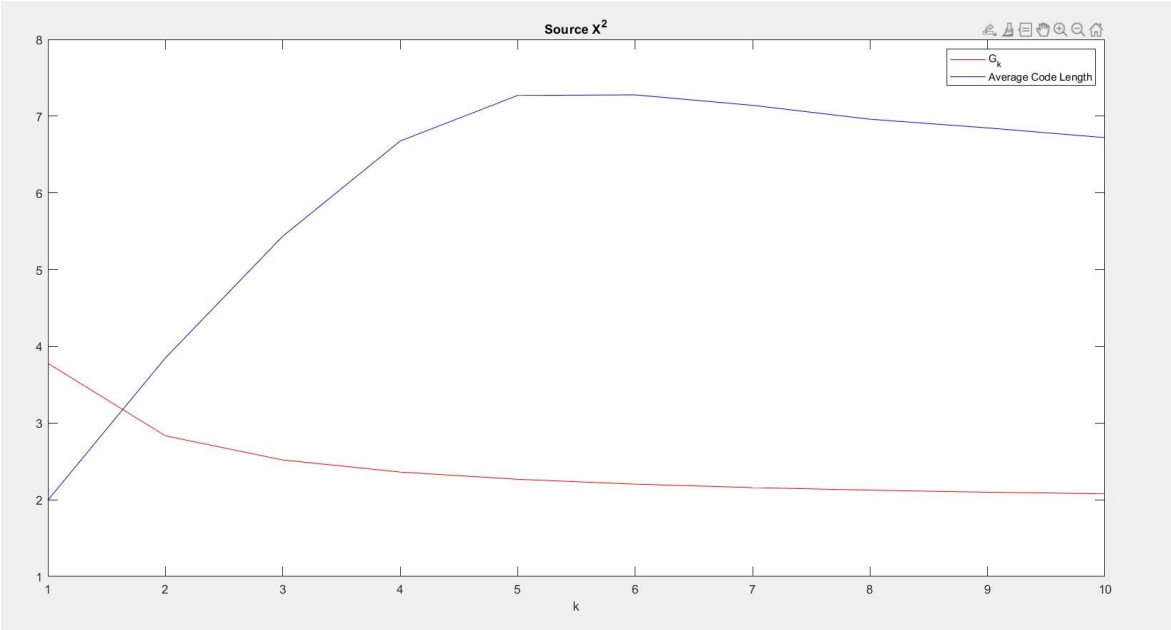

Figure 5: G_K, Average Length of code for X^2

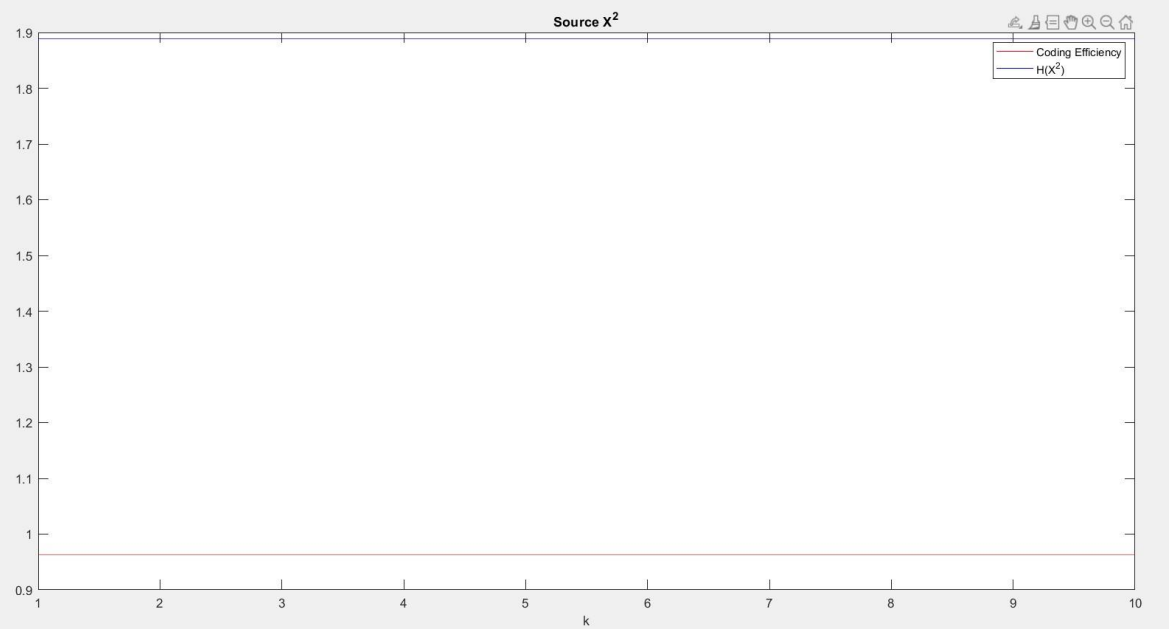

Figure 6: Coding Efficiency & Entropy for X^2

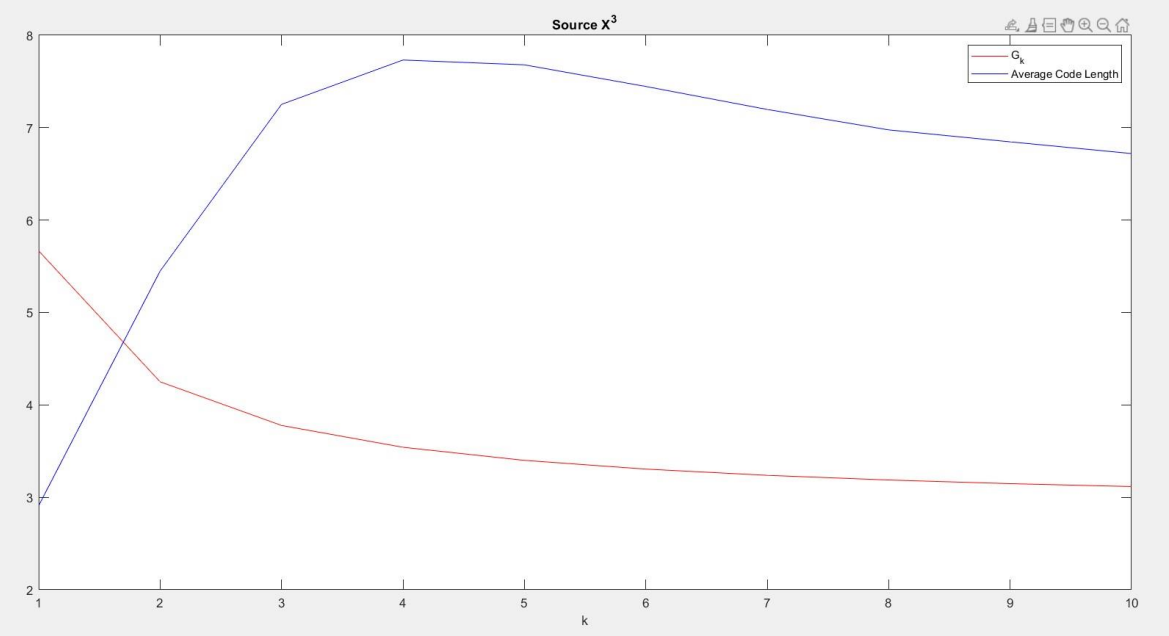

Figure 7: G_K, Average Length of code for X^3

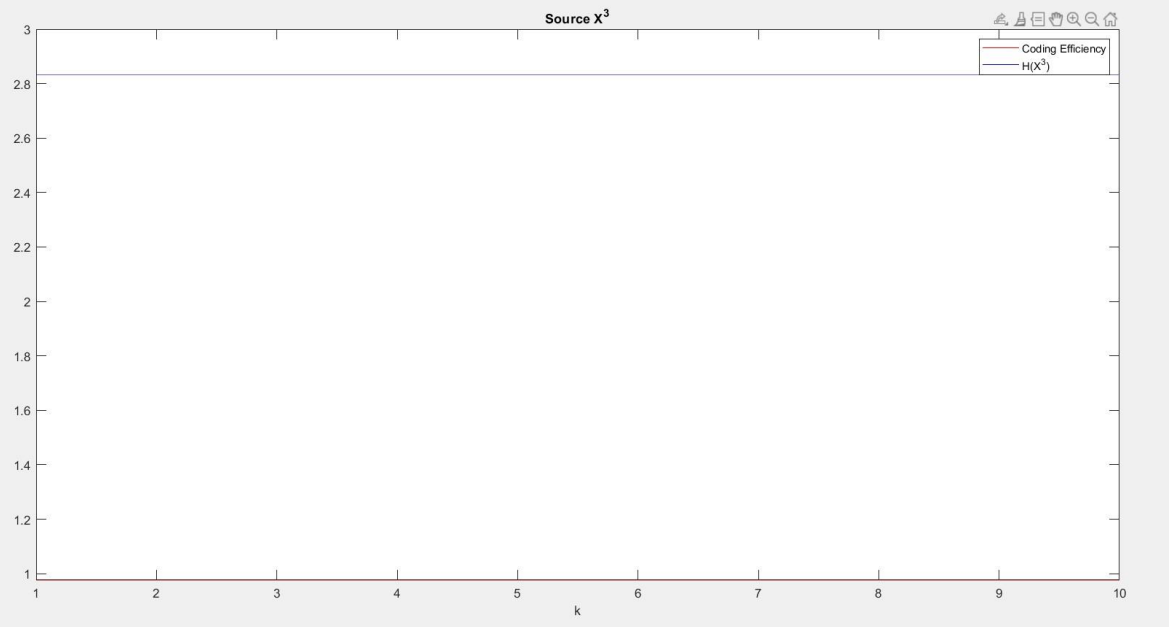

Figure 8: Coding Efficiency & Entropy for X^3

As we can observe efficiency doesn't change with k and G_K converges to H(X).

To obtain G_K we use specifications below:

$$G_k = \frac{H(s_0) + k\,H(s1|s0)}{k} = \frac{k+1}{k}H(s_0) = \frac{k+1}{k}H(X)$$

- The code is in file ***P4.m***

Problem 5) A comparison between two previous sections

In problem 4 we could deduce that for each source no matter how you separate the chain and code them, it won't affect the coding efficiency but in problem 3 which was meant for simulating a source with memory, by increasing the k value, we were able to get higher values for the coding efficiency.

In conclusion, the best we can summarize and compare memoryless and with memory sources we would get this far that k value affects the parameters of the source such as coding efficiency.

In memoryless sources, to get higher coding efficiency we'd better increase k value and try to code the chain with groups each containing more symbols.