Assignment-1

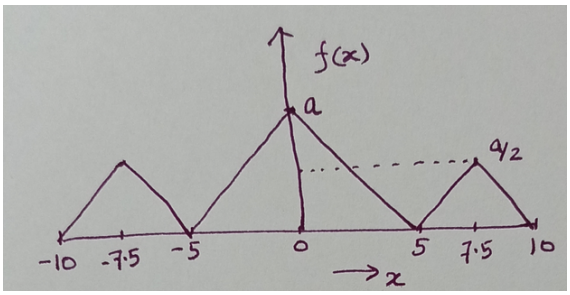
Objective:-To generate a discrete-time source output Y(k) using a random variable X(k) which follows the below pdf f(x) and then quantize and encode the obtained Y(k) and encode it using different encoding techniques.

Name: Harshavardhan Alimi

Roll No.: 18EC10021

Problem Statement:

Assume the following PDF f(x)



1.Compute the value of a.

2.Generate the continuous random variable X~f(X) for N= 10^6 time points.

3.The discrete-time source output is given by

Y(k)=X(k)+ 0.5 * X(k-1)

4.Quantize Y such that MSE=10^{-4}

5.Compute the entropy rate H(Y|S).

6.Encode Y using fixed length binary encoding.Compute the average bitrate.

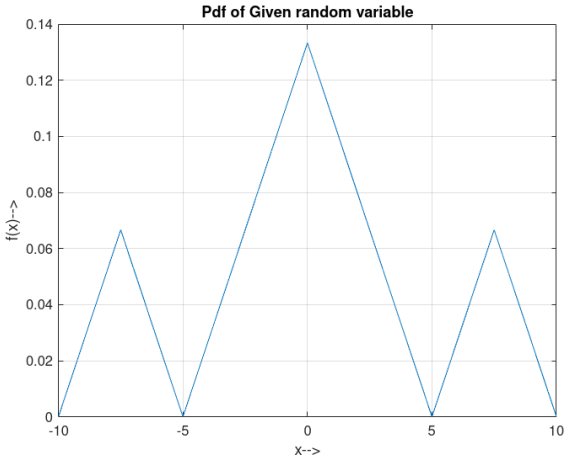
7.Encode Y using Hoffmann encoding for stationary source.Compute the average bitrate.

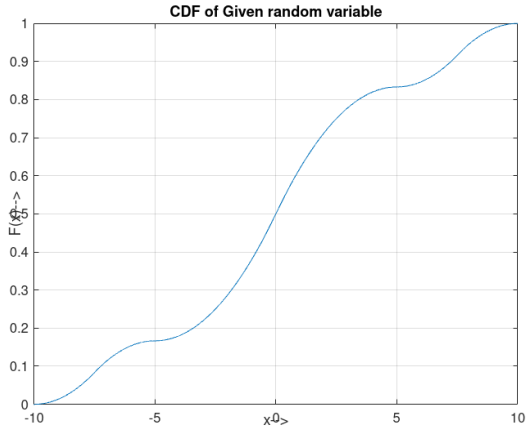
8.Encode Y using LZ77 encoding.Compute the average bitrate.

Results and discussion:

1. a = **2/15** (since,area under pdf=1)

$$\int_{-\infty}^{\infty} f(x)dx = 1$$
$$\implies \int_{-10}^{10} f(x)dx = 1$$
$$\implies 5a + \frac{5a}{2} = 1$$
$$\implies a = \frac{2}{15}$$
$$\implies a = 0.1333$$





2. Generating continuous random variable: x range : [-10,10]
   a. The random numbers are generated by the transformation of uniformly distributed pseudorandom numbers to the desired pdf.
   b. First, the pdf is normalized and then linearly interpolated for better integration.
   c. Computing the cumulative distribution function of the pdf and removing those parts which are parallel to X-axis.
   d. Then generating the uniformly distributed random numbers using the rand function and distributing those numbers using the CDF (which would be done by interpolation).

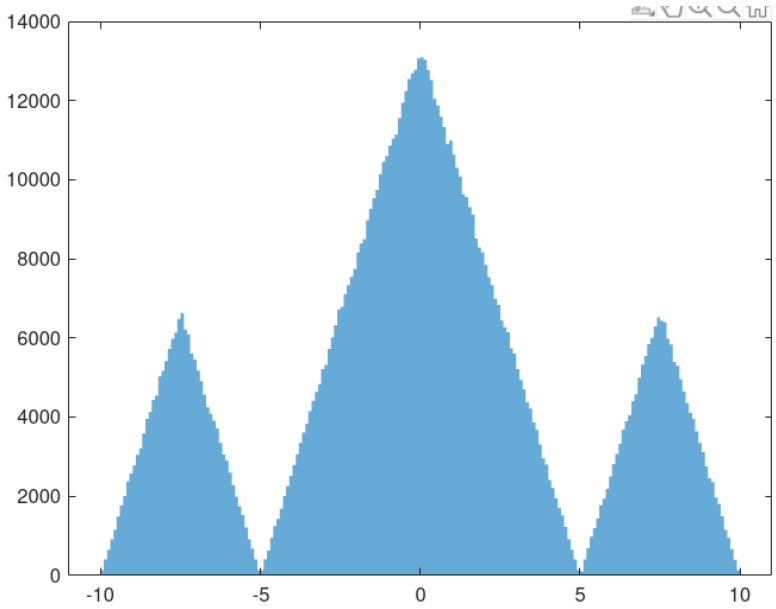


Fig.: Histogram of the Generated random variable X

3. Y range : [-15,15] - it is generated using the formulae y(k) = x(k) + x(k-1)

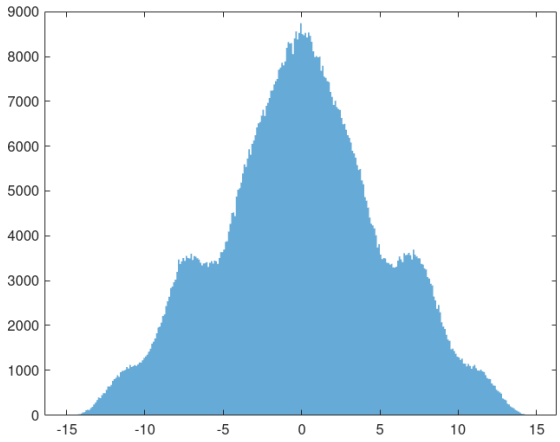


Fig.: Histogram of the Generated random variable Y

4. Quantization:
   a. We assume MSE = $delta^2$ / 12
   b. delta=**0.0346**(from : delta=sqrt(12*MSE))

      → number of states = **867** (30(15-(-15):range of random values of Y) / delta)

   c. The range [-15,15] is split into 867 partitions with each partition having length of delta and assigning midvalue for that partition(while quantising),each partition is a state.
   d. MSE obtained by doing the above process: **9.9953e-05.**
   e. The probability vector is found using the histogram and its bins(867 is set as the number of bins, where each bin represents the number of occurrences of that state from which we can calculate the probability of each state){The below graph depicts the same}

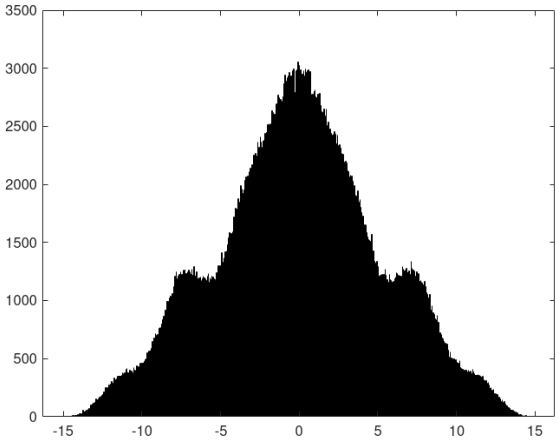


Fig.: Histogram of the random variable Y by taking a number of bins equal to a number of partitions.

5. Entropy rate = **9.2573**, it is calculated using the probabilities which are obtained as mentioned above.
6. Average bit rate in fixed-length binary encoding= **10 bits per symbol**
   a. As there are 867 states then to represent the states in fixed-length binary encoding we require 10bits(ceil(log2[867])).
7. Average bit rate in Huffman encoding = **9.2959 bits per symbol**
   a. Encoding was done using the Huffman algorithm (i.e. gives a low number of bits to the highest probability state and gives a high number of bits to the lowest probability state).
   b. The minimum number of bits used in Huffman encoding of the given random process Y is **8**, and a minimum number of bits(8-bits) was used in encoding **67** states.
   c. The maximum number of bits used in Huffman encoding of the given random process Y is **44**, and a maximum number of bits(44-bits) was used in encoding **2** states, they are [-15,-15+Δ] & [-15+Δ,-15+2Δ].
8. Average bit rate in LZ77 encoding:-
   a. Calculated average bit rate = **9.0524 bits/s** (when w = 2^{19})
   b. Calculated average bit rate = **9.4986 bits/s** (when w = $2^{19.5}$)

Technically Challenges Faced:

- The first basic challenge we faced was to generate random numbers which follow a particular distribution.
- There will be difficulty in forming the Huffman tree as we have to sort and select(selecting the lower two probabilities at every sort time) 867 times where the tree can be in any fashion.
- After successfully coding the algorithm and obtaining the results, we realized the importance of Huffman encoding and how it reduces the bitrate by assigning a lower number of bits for the most probable outcome and a higher number it's for the least probable outcome.
- While encoding using the LZ77 algorithm, we realized that the sample length was too small for the algorithm to work and give good results.
- As we know that for the algorithm to work properly:

$$Window\,length(w) \approx n.2^{nH(Y/S)}\ Where\ n \geq 2$$

Therefore by substituting n = 2 and H(Y/S) = 9.25

We get :

w should be at least $2^{19.5}$ = 741456.

Making no. of bits assigned to u = [log2(74145)] = 20

And also we can only guarantee n = 2 during the running of the algorithm.

Making total number of bits required to encode 2 symbols =[2*log2(n)] - 1 + [log2(w)] = 2*2 - 1 + 20 = 23 bits

Which is more than the number of bits required for fixed-length encoding i.e 20 bits.

- As expected the bit rate is higher than the theoretical bit rate i.e entropy rate.(i.e 9.2584)
- Hence, in order to optimize it, we took w = 2^{19} instead of $2^{19.5}$ making u = [log2(w)] = 19 which significantly decreased the bit rate to 9.026 which is better than the expected bitrate (i.e 9.2584).
- Which also decreased the probability of having n always greater than 2. But the tradeoff was good enough.
- Also, another reason for the decrease of bitrate can be that when window length was decreased we made room for more number of symbols to be compressed.
- Also, we assumed that the window length is sufficient to have always i.e at least two consecutive elements match inside the window(even after decreasing 'w').

Conclusion:

After encoding the output signal in all three ways (fixed length, Huffman, LZ77) Here are the few points we realized:

- To encode fixed length is just faster and easier than the other two but it gives the worst bitrate possible.
- Huffman encoding is better than LZ77 encoding for low numbers of sample data (as we can see in our case).
- Also, Huffman encoding is faster and computationally less taxing than LZ77.
- But for a higher number of samples, we can clearly conclude that LZ77 is a better algorithm for achieving the best bitrates even the encoding using the LZ77 algorithm requires much more time than Huffman encoding.