Linear Congruential Random Number Generator (LCG) and its Further Applications

Rajdeep Tah, 1911124[1,*]

[1] *School of Physical Sciences, National Institute of Science Education and Research, HBNI, Jatni-752050, India*

(Dated: November 15, 2021)

In this project, I begin with a brief introduction to the working algorithm of a Linear Congruential Generator (LCG) which is in fact regarded as one of the best known methods for generating pseudo-random numbers. Then we create a Random Number Generator (RNG) by combining four LCGs (to increase the efficiency) and we evaluate the accuracy of it along with visualizing the periodicity of the constructed RNG. Later we utilize it to construct a Hangman game involving the numbers generated from our RNG and verify whether our RNG is working as expected or not, by performing the famous Kolmogorov-Smirnov Test of Normality.

I. THEORY AND ALGORITHMS

A brief description of the theory and algorithms which I used for designing my project are discussed below.

A. Linear Congruential Generator

A Random Number Generator mainly works on the principle of **congruence** which is dealt mainly in the field of Number Theory but a basic definition of it is given as

"*If* $\boldsymbol{n}$ *is a positive integer, we say the integers* $\boldsymbol{a}$ *and* $\boldsymbol{b}$ *are congruent modulo* $\boldsymbol{n}$*, and write* $\boldsymbol{a \equiv b (mod\ n)}$*, if they have the same remainder on division by* $\boldsymbol{n}$."
This can be easily expressed as:

$$\mathrm{a} \equiv \mathrm{b}(mod\ \mathrm{n}) \quad \text{iff } n \mid (a-b) \tag{1}$$

By using Eq.(1), we can have a recurrence relation for the LCG as [1]:

$$X_{n+1} = (a \cdot X_n + c) \mathrm{mod}\ m \tag{2}$$

where X is the sequence of pseudo-random numbers and we have the parameters as:

- m is the modulus value and it always have a positive non-zero value (i.e. $m > 0$).
- a is defined as the multiplier and it always lies in between 0 and m (i.e. $0 < a < m$).
- c is defined as the increment and it is always a positive number, maybe 0, but less than m (i.e. $0 \leq c < m$).
- Finally, X_0 is the initial or seed value for the LCG and it is always a positive number, maybe 0, but less than m (i.e. $0 \leq X_0 < m$).

We are not taking the $c = 0$ case here, cause it will lead to another algorithm associated to the **Multiplicative Congruential Generator** (MCG), better known as **Lehmer RNG**, and we keep ourself limited to LCG.

Also, we must know how to choose the parameters of a LCG to get a period of maximal period length and they are known as **Hull-Dobell Theorem**. The conditions which must be satisfied are:

- m and c must be relatively prime.
- $(a-1)$ must be divisible by all prime factors of m.
- $(a-1)$ must be divisible by 4 if m is divisible by 4.

This is necessary so that we get a particular number once during each period and this forms a main criterion of shuffling in the RNGs. More details about **Hull-Dobell Theorem** (from where I took help) can be found **here**.

B. Combination of four LCGs

In our project, after discussing the LCGs, we move onto construct a combination of four LCGs or a **Combined linear congruential generator**, where the algorithm is similar to that of a LCG but with certain modifications. By combining two or more LCGs, random numbers with a longer period and better statistical properties can be created. The algorithm is given as [2]:

$$X_i = \left(\sum_{j=1}^{k} (-1)^{j-1} Y_{i,j} \right) (\mathrm{mod}\ m_1 - 1) \tag{3}$$

where we have;

- m_1 is the modulus of the 1^{st} LCG.
- $Y_{i,j}$ is the i^{th} input from the j^{th} LCG.
- X_i is the i^{th} generated random integer.

Now, to be accurate and ease our calculations, we can create a CLCG whose output lies between 0 and 1. This

* rajdeep.tah@niser.ac.in

can be achieved by defining:

$$R_i \equiv \begin{cases} \dfrac{X_i}{m_1} & \text{for } X_i > 0 \\ \left(\dfrac{X_i}{m_1}\right) + 1 & \text{for } X_i < 0 \\ \dfrac{(m_1 - 1)}{m_1} & \text{for } X_i = 0 \end{cases} \tag{4}$$

where R_i is the uniformly distributed random number between 0 and 1. The period of the CLCG for our case is given as [2]:

$$P = \frac{(m_1 - 1)(m_2 - 1)(m_3 - 1)(m_4 - 1)}{2^{4-1}} \tag{5}$$

$$\Rightarrow P = \frac{(m_1 - 1)(m_2 - 1)(m_3 - 1)(m_4 - 1)}{8} \tag{6}$$

More details about the properties of **CLCG**, from where I took help during this project, can be found **here**.

Our case, $c = 0$ ***and*** $m = 2^n$ ***for*** $n \in N$

If m, a and c are properly chosen, then the period will be of maximum length. So we choose m to be a power of 2 since this allows the modulus operation to be computed by simply truncating the binary representation. We know that if k random numbers are used to plot points (with each coordinate between 0 and 1) in a k-dimensional space, at a given instance, then the points will tend to not fill up the k-dimensional space but rather lie on $(k-1)$-dimensional planes and the number of such planes will be at most about $m^{1/k}$. So, we need to choose the values of m, a and c very carefully.

The number I chose is usually close to the machine's largest representable integer so it will be easy for the computer to compute such values. Also we know that in a computer, the least significant bit has a period of at most 2, second one at most 4 and so on. So, choosing m as power of 2 will yield accurate and efficient results. The form which I mentioned has a maximal period of $\frac{m}{4}$ iff $a \equiv 3$ or $a \equiv 5 \pmod 8$. I took the values as:

- $a_1 = 8003$, $a_2 = 8005$, $a_3 = 8083$, $a_4 = 8085$

I have tried for other values of a, c and m by taking help from the reference material given **here**.

C. The Kolmogorov-Smirnov Test of Normality

The Kolmogorov-Smirnov Test is basically an example of Frequency test which is used to determine the uniformity and agreement between the distribution of a sample of generated random numbers and the theoretical uniform distribution. It is mainly based on the null hypothesis of no significant difference between the sample distribution and the theoretical distribution.

In this test, we compare the cumulative distribution function (CDF) of a uniform distribution $F(x)$ to the empirical CDF $S_N(x)$ of the sample of N-observations.

The steps for this test are:

1. We define $F(x) = x \ \forall \ x \in [0, 1]$.

2. We have $S_N(x) = \dfrac{no.\ of\ R_1, R_2, ...R_N \leq x}{N}$ where R_N are the data values obtained from the CLCG.

3. We observe that as N becomes larger, the value of $S_N(x) \to F(x)$.

4. The absolute difference D (which is a random variable) is given as:

$$D = max \mid F(x) - S_N(x) \mid \tag{7}$$

5. We then sort the data ($R_1,\ R_2, ...,\ R_N$) in ascending order and calculate the values of D^+ and D^- given as:

$$D^+ = max\left\{\frac{i}{N} - R_i\right\} \ \forall\ i \in [1, N] \tag{8}$$

$$D^- = max\left\{R_i - \frac{(i-1)}{N}\right\} \ \forall\ i \in [1, N] \tag{9}$$

6. From Eq.(8) and (9), we compute D as:

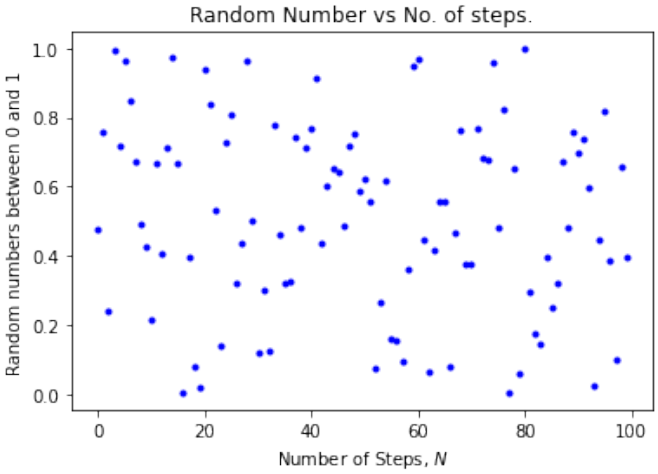
$$D = max\ (D^+,\ D^-) \tag{10}$$

7. Finally, from a reference table [3], we determine the critical value, D_α, for $\alpha = 0.05$ and N (attached in code). We can say that if the sample statistic D is greater than the critical value D_α, the null hypothesis that the sample data is from a uniform distribution is rejected and if $D \leq D_\alpha$, then there is no evidence to reject it.

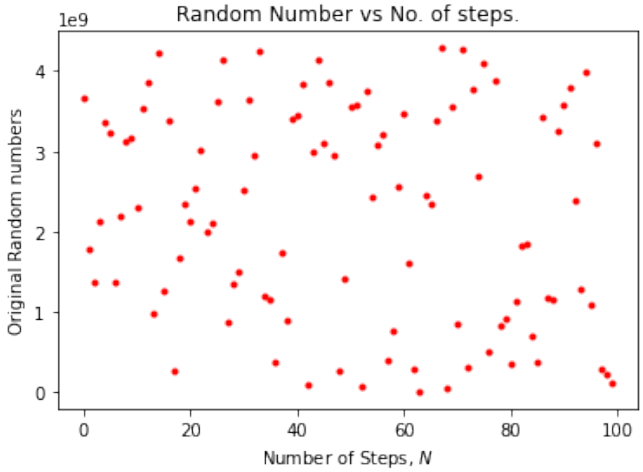
II. BRIEF DETAILS ABOUT THE PROJECT

A. For the LCG

In this project, initially I have put forward a simple version of the LCG where I have kept the values of the different parameters $m = 2^32$, $a = 1664525$ and $c = 1013904223$ and I have put the values of $N = 100, 200, 300, 400$ and 500. To reduce the complexity, I have generated the numbers between 0 and 1 although I am attaching one plot for the actual numbers for reference. The plots are given below.



(a) The numbers are between 0 and 1.



(b) The original randomly generated numbers.

FIG. 1: For $N = 100$, both the plots.

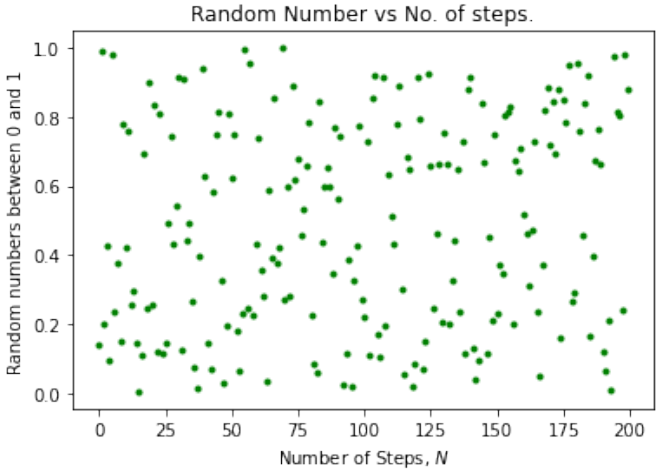


FIG. 2: For $N = 200$, plot for the generated numbers between 0 and 1.

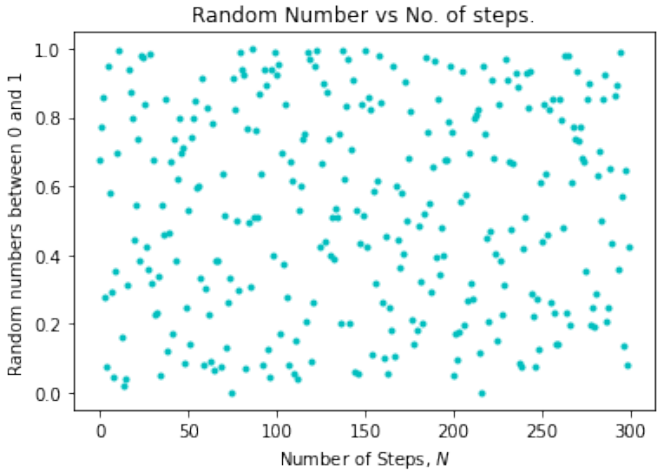


FIG. 3: For $N = 300$, plot for the generated numbers between 0 and 1.

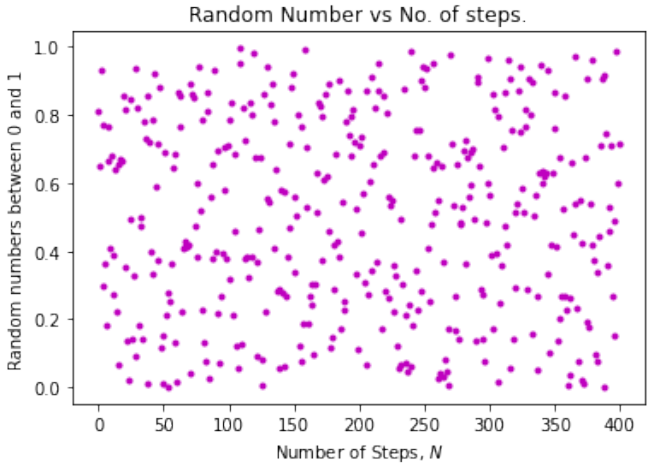


FIG. 4: For $N = 400$, plot for the generated numbers between 0 and 1.

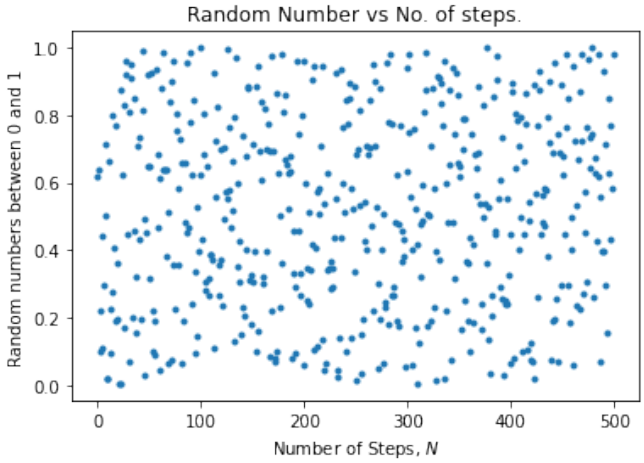


FIG. 5: For $N = 500$, plot for the generated numbers between 0 and 1.

All the plots given above satisfied the Kolmogorov-Smirnov Test for normality and they are given in the code.

B. For the CLCG and the Hangman Game

As described earlier, I have taken **four** LCGs and formed a Combined Linear Congruential Generator where I have taken the parameters for the individual LCGs in regards to the criterions as mentioned in the previous sections. The different values of m, a, c for $k = 4$ are:

- For first LCG, $a_1 = 8003$, $m_1 = 2^{64}$ and $c_1 = 0$.
- For second LCG, $a_2 = 8005$, $m_2 = 2^{32}$ and $c_2 = 0$.
- For third LCG, $a_3 = 8083$, $m_3 = 2^{16}$ and $c_3 = 0$.
- For fourth LCG, $a_4 = 8085$, $m_4 = 2^{8}$ and $c_4 = 0$.

The whole thing is clearly explained in the code. Like before, to reduce the complexity, I have generated the numbers between 0 and 1 by scaling down the values by using the expressions in Eq.(4) but the numbers which are getting as input in the Hangman game are the original randomly generated ones. The plot for the same are attached below.

The Hangman game just takes a random number out of the list of the original generated numbers and starts the game by providing a number of error chances (60% of the actual number length).

In this case also, all the plots given below satisfied the Kolmogorov-Smirnov Test for normality and they are given in the code.

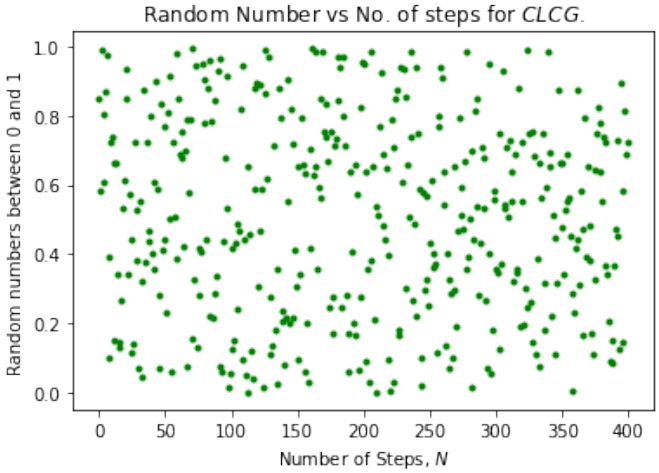


FIG. 6: For $N = 400$, plot for the generated numbers between 0 and 1 for CLCG.

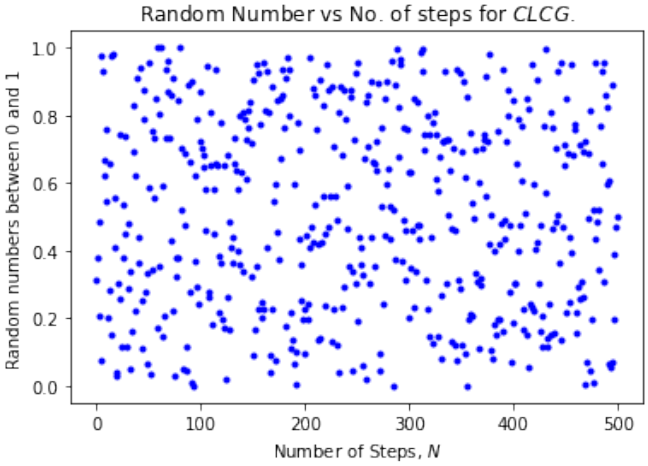


FIG. 7: For $N = 500$, plot for the generated numbers between 0 and 1 for CLCG.

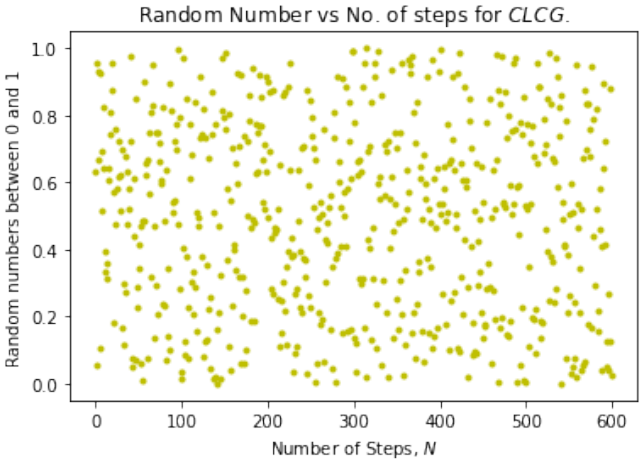


FIG. 8: For $N = 600$, plot for the generated numbers between 0 and 1 for CLCG.

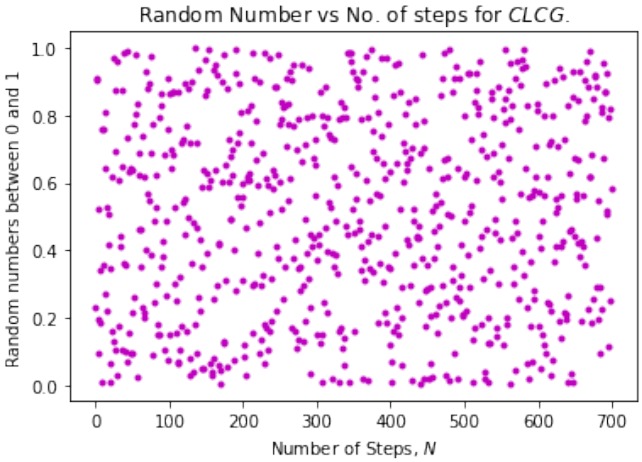


FIG. 9: For $N = 700$, plot for the generated numbers between 0 and 1 for CLCG.

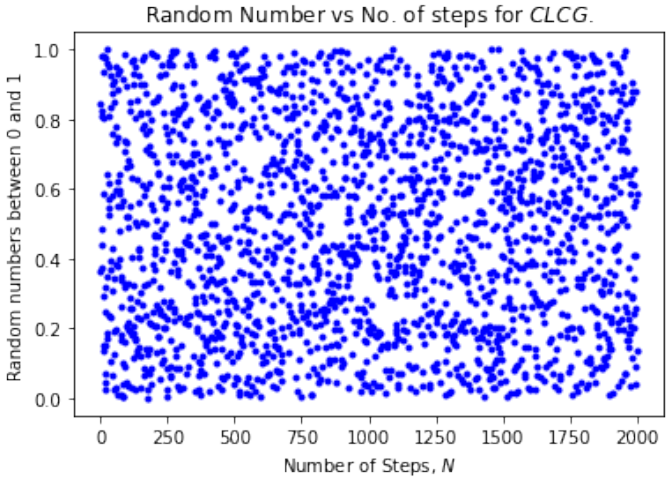


FIG. 10: For $N = 2000$, plot for the generated numbers between 0 and 1 for CLCG.

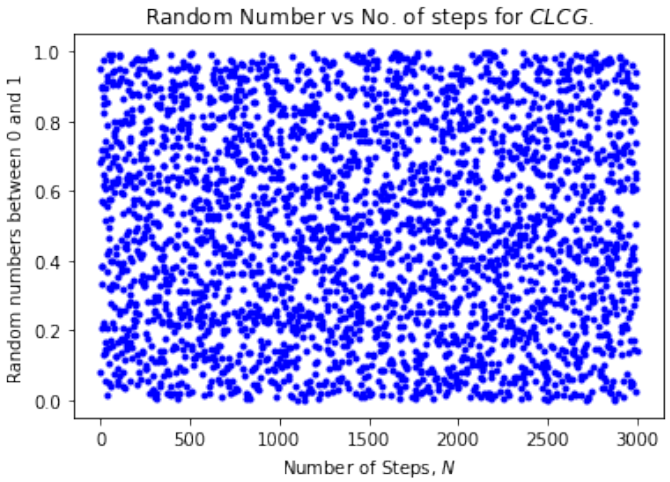


FIG. 11: For $N = 3000$, plot for the generated numbers between 0 and 1 for CLCG.

III. VERIFICATION OF THE PROPER WORKING OF THE CLCG

Using external software, I have calculated the efficiency of my constructed CLCG and found out that the efficiency of it is within appreciable range. The output from the software is given below. We needed to show that our data was not normally distributed and this indeed proved it.

FIG. 12: The data is not normally distributed.

[1] Linear Congruential Generator. https://en.wikipedia.org/wiki/Linear_congruential_generator
[2] Combined Linear Congruential Generator https://en.wikipedia.org/wiki/Combined_linear_congruential_generator
[3] Critical values (D_α) for Kolmogorov-Smirnov test https://blogs.sas.com/content/iml/2019/05/20/critical-values-kolmogorov-test.html