Comparison of Pseudo Random Number Generator Algorithms

BY: LUKE FETCHKO

CSCI U599

DR. ZHONG

Introduction

- Randomness is a property needed for many different applications in the fields of science, mathematics, and technology. Computer programs or simulations that require randomness depend on certain means of producing numbers that are either random or appear to be random.
- There are two distinct methods of producing random numbers. These methods are the use of True Random Number Generators and the use of Pseudo Random Number Generators.
- Pseudo Random Number Generators are distinct from TRNGs since they are not truly random. PRNGs are classified as deterministic. PRNGs start with an initial seed value which is used as input for an arithmetical process (or algorithm) of generating random numbers.
- Another important property of PRNGs is efficiency. PRNGs are typically much faster at producing results compared to TRNGs. Applications requiring fast generation of random numbers greatly benefit from the efficiency of PRNGs.

Literature Review

- Computer scientists and engineers have worked on creating and improving upon PRNG algorithms for many years. Their goal is to continuously improve the results of generating numbers which appear to be truly random. Many studies conducted that focus on one approach and do heavy statistical analysis on the results leave the researchers and observers with many questions at the end. These studies may conclude that statistical results are not desirable or that generating truly random numbers may be possible with the current approach [3].

- A previous study conducted on random number generation using bioelectrical and physical signals followed this pattern. This study provides great background information on random number generators including both PRNGs and TRNGs. This article clearly lays out advantages and disadvantages of both approaches to generating random numbers. As I mentioned, this study explains that TRNGs are commonly much slower than PRNGs and are largely hardware dependent.

- Explanation of the bioelectrical and physical signals used in the random number generation technique is elaborate and valuable to understanding the overall operation of the random number generator. Data used for this study included EEG, EMG, EOG, blood volume pulse, galvanic skin response, and respiration readings [3]. This raw data was then transformed into binary number representations using a method involving modulo arithmetic [3]. Many random number generators appear random due to modulo arithmetic.

- The conclusion of the mentioned study on bioelectrical and physical signals concluded that the numbers obtained are suitable for use in many areas including game programming, authentication, and simulation [3]. The mentioned study also concluded that it is possible to generate truly random numbers using this approach.

Methodology

Linear Congruential Method

$$X_{i+1} = aX_i + c \mod m$$

X, is the sequence of pseudo-random numbers
m, (> 0) the modulus
a, (0, m) the multiplier
c, (0, m) the increment
X0, [0, m) – Initial value of sequence known as seed

m, a, c, and X0 should be chosen appropriately to get a period almost equal to m.

Middle Square Method

The algorithm is implemented as follows: take any four-digit number, square it, remove the middle digits of the resulting number as the "random number", then use that number as the seed for the next iteration.

Implementation

- Implementation of these Pseudo Random Number Generator algorithms is done using Python
- Python 3.9
- Using the IDLE integrated development environment which comes bundled with Python upon installation
- Driver will import certain modules like timeit to calculate execution time, statsmodels to execute a Runs Test for Randomness, and Matplotlib to plot results
- Will have one script called DataAnalysis.py which contains both algorithms and the needed modules and code to conduct data analysis
- This script will show the random numbers that were produced and the desired statistics for evalution

Dataset and Experimental Setup

- No external dataset is used for this study, but the seed values for each test run will determine or alter the resulting random numbers to be generated. This experiment uses a single seed value, and then based on the specific algorithm will generate the 50 random numbers and those results will be analyzed.

- By default, the script is programmed to generate 50 random numbers. The user of the driver may enter a desired number of random numbers to generate if wanted. For the experiment, the default value of 50 random numbers to be generated will be used. Each algorithm will be tested 5 times for a total of 10 tests. For both algorithms, these tests will use different user defined seed values which will be noted.

- One evaluation metric used to analyze the different methods are the execution time (speed) in milliseconds since quickly producing the random numbers is an important aspect of using PRNGs. This is done using the timeit module in Python.

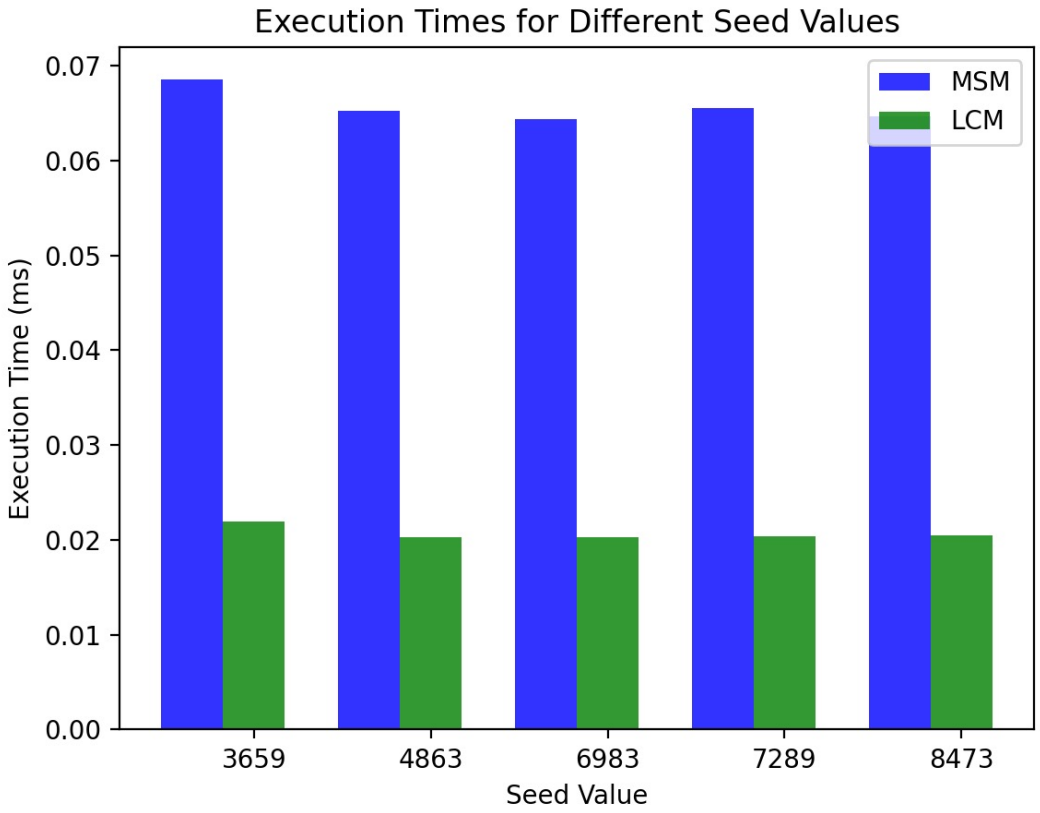
- Another evaluation metric being used is the corresponding p-value from the Runs Test for Randomness. The p-value from the Runs Test will be compared with the significance level (alpha) as 0.05. If the p-value is greater than or equal to the significance level, there is enough evidence to say that the numbers were produced using a random process. The Runs Test for Randomness is conducted using the statsmodels.sandbox.stats module in Python.

- The resulting p-values will be transferred to a table which lists which method was used, which seed value was used, and the corresponding p-value from the Runs Test for Randomness

- The execution times in milliseconds for each seed value used will be graphed on a bar graph comparing the two methods, the MSM and the LCM. Graphing is done using the matplotlib module in Python.

Results Analysis

- Middle Square Method took much longer than the Linear Congruential Method to execute in terms of milliseconds. Around double the amount of time to execute.
- This was the execution time for generating 50 random numbers, each with a different user defined seed value.
- As mentioned earlier, PRNGs are useful since they should generate many random numbers very quickly and the execution time is important for this reason.



Results Analysis (cont.)

Null hypothesis H_0: The data was produced in a random manner

Alternative hypothesis H_1: The data was not produced in a random manner

Significance level (alpha) = 0.05

Runs Test for Randomness was conducted on each dataset produced by the different seed values for each method.

Method	Method	Seed	Runs Test p-value	Random? (p-value >= 0.05)
Middle Square Method	Middle Square Method	3659	0.7750513736700824	Yes (fail to reject H_0)
		4863	0.5676284992633359	Yes (fail to reject H_0)
		6983	0.4886081662662083	Yes (fail to reject H_0)
		7289	0.03079306420403246	No (reject H_0)
		8473	0.9158685490944402	Yes (fail to reject H_0)
Linear Congruential Method	Linear Congruential Method	3659	0.0650902667816322	Yes (fail to reject H_0)
		4863	0.5966099949082471	Yes (fail to reject H_0)
		6983	0.1783756136768152	Yes (fail to reject H_0)
		7289	0.4139656223656427	Yes (fail to reject H_0)
		8473	0.8090679300618948	Yes (fail to reject H_0)

Conclusion

Based on the results from the experiment, the Linear Congruential Method was able to generate random numbers much more quickly than the Middle Square Method was. It took about double the time for the Middle Square Method to generate the same quantity of random numbers with the same seed values used.

Additionally, one of the Middle Square Method datasets produced a p-value which fell under the significance level which lets us conclude that the data was not produced in a random manner.

For these reasons, the better method of generating random numbers appears to be using the Linear Congruential Method. This was the expected result as the Middle Square Method is known to be one of the oldest methods for generating random numbers.

References

1. Behnia, S., Akhavan, A., Akhshani, A., & Samsudin, A. (2011). A novel dynamic model of pseudo random number generator. Journal of Computational and Applied Mathematics, 235(12), 3455-3463. doi:10.1016/j.cam.2011.02.006

2. Yang, YG., Zhao, QQ. Novel pseudo-random number generator based on quantum random walks. *Sci Rep* **6,** 20362 (2016). https://doi.org/10.1038/srep20362

3. Seda Arslan Tuncer, Turgay Kaya, "True Random Number Generation from Bioelectrical and Physical Signals", *Computational and Mathematical Methods in Medicine*, vol. 2018, Article ID 3579275, 11 pages, 2018. https://doi.org/10.1155/2018/3579275

4. Anikin, I. V., & Alnajjar, K. (2018). Increasing the quality of pseudo-random number generator based on fuzzy logic. *Journal of Physics: Conference Series, 1096*, 012193. doi:10.1088/1742-6596/1096/1/012193

5. K. Jasim Mohammad, O., Abbas, S., M. El-Horbaty, E., & M. Salem, A. (2014). A new trend of pseudo random number generation using qkd. International Journal of Computer Applications, 96(3), 13-17. doi:10.5120/16773-6344