PHYS3061 Laboratory 1 - Review on Random Number Generator

LO Man Hei

Department of Physics, The Chinese University of Hong Kong, HKSAR, China

(Dated: February 2, 2019)

This paper reviews three noticeable pseudo-random number generator, namely Middle Squared, Linear Congruential generator, and Mersenne Twister. The selected pseudo-random are being tested by (1). distribution test, (2). independence test and (3). physical system test. The Middle Squared has failed all the tests, where the Linear Congruential generator with large parameters and the Mersenne Twister have passed all the test.

PACS numbers: 02.70.c`

I. INTRODUCTION

Random number generators have many uses in many fields, including science, statistics, cryptography [1]. For example, a high quality of random number generators has played a dominant role in the *Monte Carlo* simulation, which is highly visible in science [2].

In particular, random number generators could be identified as truly random and pseudo random. The sooner ones could be achieved by measuring the fluctuating nature. Whereas the later ones, however, are implemented by a deterministic algorithm, and will produce same sequence of numbers by choosing the same seed [1]. As to clarify for the readers, only pseudo random number generators will be covered in this paper.

Despite the deterministic nature of the pseudo random number generator (PRNG), a high quality PRNG can still generate statistically random number, in a relative faster and cheaper manner compared with the truly random number generators. In addition, the deterministic nature of PRNG which producing the same sequence can also help debug. Thus, high quality PRNGs are still the first choice for *Monte Carlo* simulation and many other applications [1][2].

As aforementioned, high quality PRNGs could be especially useful. Yet the criteria of high quality are still vague and non-standardized [1][3].

Generally, the quality criteria could be classified as distribution criteria and independence criteria. Specifically, the *Kolmogorov Smirnov* test and the *Chi Squared* test are few widely accepted and simple tests on the quality [4]. On the other hand, there are also numerous other testing methods and packages existed to access the quality of a PRNG. For example, *TestU01* and *Hamming-weight Dependencies*. However, it is often happened that the tests disagreeing others' conclusion [3][4]. Therefore, for simplicity, this paper will only focus on some simple statistical tests and physical tests that will be introduced in the coming part.

II. OBJECT

Since the modern computer was invented, the algorithm of PRNGs were kept updating and inventing. According to [5], till April 2018, there are over 20 different algorithms of PRNGs existed. It is difficult to cover all of them in a single paper, therefore only some remarkable PRNGs will be tested, namely Middle Squared (MS) method, Linear Congruential generator (LCG) and Mersenne Twister (MT).

A. Middle Squared method

The MS method was invented by *J. von Neumann* in 1946, which is the first ever invented PRNG algorithm. This method needs to feed a 6-digit seed, after each iteration MS method returns the middle 6-digit of the squared seed [1]. For example:

Seed: 123456
Squared seed: $123456^2 = 15241383936$
Return number: 413839

Or:

Seed: 987654
Squared seed: $987654^2 = 975460423716$
Return number: 460423

Generally, MS method has relative short period, such that it will produce the same number after a shorter period which could be clearly seen on the test result on the later part. Moreover, the implementation of MS method on a computer is not that simple as it is look like. MS method requires to parse the number into a string, cutting the string, and then parsing back the string into number [1].

However, based on the historical reason, it is still worth to review on this first ever PRNG. The seeds of MS will be used in the later tests are 123456, 987654 and 104729.

B. Linear Congruential generator

The LCG was first proposed by *W. E. Thomson, A. Rotenberg* in 1958. It was a generalized version of *D. H. Lehmer*'s Lehmer random number generator [1], for which has the formula as below:

$$X_{n+1} = a * X_n \ mod \ m$$

where the modulus m is a prime number or a power of a prime number, the multiplier a is an element of high multiplicative order modulo m, and the seed X_0 is coprime to m. The LCG has one more term on the formula:

$$X_{n+1} = (a * X_n + c)\ mod\ m$$

where c is called the increment term. For example:

Parameter: $a = 2, c = 1, m = 5$
Seed: 3
Return number: $(2 * 3 + 1)\ mod\ 5 = 2$

The quality of LCG is heavily depend on the choice of the parameter and seed. A proper choice could yield a high speed and high quality PRNG, so it is still one of the most popular PRNG till now [5]. The coming part will test on 3 sets of parameters of LCG, {a, c, m, X_0} = 1:{69069, 0, 2^{32}, 7741}, 2:{22695477, 1, 2^{32}, 7741} and 3:{3, 2, 500, 231}.

C. Mersenne Twister

The MT was developed in 1997 by *Makoto Matsumoto* and *Takuji Nishimura*. The algorithm of MT involved some binary operators, such as bitwise-shifting, and bitwise-exclusive-or. The detail of the algorithm could refer to [1]. For some specific choice of seeds of MT could achieve an extraordinary long period of 2^{32}-1, even greater than the number of atoms existed in this universe, and the statistical quality is also satisfactory for most uses. This could probably the reason why MT is the default PRNG for many programming language, and the most popular PRNG [1][5].

In this paper will use the MT on Julia for testing, with seed equals to 104729.

III. METHOD

This paper will use the above mentioned PRNG with the specified parameter to generate a 100000-pseudo-random number sequence. Then the quality of sequence will be accessed by the following tests.

As aforementioned, there are distribution test and independence test as for the quality verification. Additionally, as this paper is aimed to serve as a reference for the choice of PRNG to be used in the CUHK PHYS3061 Computer Simulations of Physical Systems course. So testing for the PRNG on a physical system will also be included.

A. Distribution test

This paper will include 3 tests on the PRNG distribution, such are the *Kolmogorov Smirnov* test, frequency test and mean value analysis.

1. Kolmogorov Smirnov test

The *Kolmogorov Smirnov* test is a general method to investigate if a distributed data set obey certain expected distribution curve [6]. For the random number generators, it is expected the generated sequence would have equal probability on several evenly distributed interval. For example, a random number generator that will generate numbers in the range [0, 1], and generated a 100-number sequence. Then, if we set 4 bins in the [0, 1] range, such we would obtain 4 intervals: [0,0.25), [0.25,0.5), [0.5,0.75), [0.75,1]. It is supposed each bin would contain about 25 numbers, with probability of about 25% .

And then we obtain the maximum deviation (D_{max}) by comparing the obtained and expected cumulative data. To continue the last example, supposed the actual number fell on each bin were 26, 24, 27 and 23 respectively. Such that our obtained cumulative data for the probability that the number is smaller than 0.25 is 26%, smaller than 0.5 is 50%, smaller than 0.75 is 77% and the whole data is 100%. Whereas the expectation are 25%, 50%, 75% and 100% respectively. Therefore we will obtain the D_{max} of 2% in the 3rd interval. The data is said to be violated the expected distribution if the D_{max} value is larger than the critical value.

In specific, this paper will test a 100000-pseudo-random number sequence with 50000 intervals, the critical value for the test is $5.456 \cdot 10^{-3}$ at 0.10 significant level.

2. Frequency test

The frequency test is a simple statistical method accessing the number of repetition of the generated sequence. Then the period of the PRNG could be estimated by:

$$\frac{N}{R-1} > T > \frac{N}{R}$$

where N is the number of terms in the sequence, R is number of repetition and T is the period of the PRNG.

3. Mean value analysis

The mean value analysis is a simple statistical method to verify if the data is biased. The population mean could be estimated by the sample mean and sample standard deviation [6]. For example, there is a random number generator that will generate numbers in the range [0, 1], and generated a 1000-number-sequence with sample mean = 0.48 and sample standard deviation = 0.29. Such that for a 99.9% confidence level, the population mean can be estimated by:

$$\overline{X_c} = \mu \pm \frac{3.291 * s}{\sqrt{N}}$$

where $\overline{X_c}$ is the population mean estimated by the computational result, which should be 0.5, 3.291 is the

t-distribution critical value for 99.9% confidence level, μ is the sample mean, s is the sample standard deviation, and N is the number of terms. In the above example, the estimated population mean is ranged [0.45, 0.51], which included the predicted value 0.5. So that the above sample is considered as acceptable. Otherwise, there is only 0.01% the sample is not biased, and considered as not acceptable.

B. Independence test

This paper will introduce a mean different testing method on accessing the independence of the PRNG sequence.

1. *Mean different test*

The mean different test will first compute the absolute different between two consecutive terms on the sequence. Such that for a N-term sequence, there are (N-1) terms of the absolute different between two consecutive terms D, i.e. $D_n = |X_{n+1} - X_n|$, where X_n represents the n-th term on the sequence.
Supposed X_{n+1} and X_n term are generated by an ideal random number generator that generates numbers in the range [0, 1]. Then both X_{n+1} and X_n should be independent on each other and have equal probability having any values in the range [0, 1]. The expectation mean different can be calculated via:

$$\overline{D} = \int_0^1 \int_0^1 |X_{n+1} - X_n| \, dX_{n+1} \, dX_n = \frac{1}{3}$$

Whereas the generated mean different between two consecutive terms can be calculated by:

$$\overline{d} = \frac{1}{N-1} \sum_{n=1}^{N-1} |X_{n+1} - X_n|$$

where $\overline{d}$ is the mean different value of the generated sequence, N is the number of terms in the sequence, X_n the n-th term of the sequence. Meanwhile the variance of the generated different between two consecutive terms s^2 can be calculated by:

$$s^2 = \frac{1}{N-1} \sum_{n=1}^{N-1} (|X_{n+1} - X_n| - \overline{d})^2$$

Again, the validity of the mean different could be verified by the similar method with the mean value analysis. That is:

$$\overline{D_c} = \overline{d} \pm \frac{3.291 * s}{\sqrt{N}}$$

where $\overline{D_c}$ is the estimated mean different range for the population, which should be 1/3, and the other symbols follow the same meaning as aforementioned. If range of the estimated population mean different included the predicted value 1/3, then the above sample will be considered as acceptable. Otherwise, the generated sequence will be considered as biased.

C. Physical system test

For the physical system test, this paper will first setup a random walking scenario for 1000-ideal gas particles in 3-dimensions with 1000-iterations and to access the mean squared displacement value of the particles. The quality of the PRNG could be revealed on the consistence between the computational result and the expected analytical result.

1. *Random walk tests*

For every iteration Δt, each component x, y, z is added to a rescaled random term s. Where the rescaling is done by the following: (1) Normalize the PRNG by dividing the generated random number by the maximum possible number. For example, the MS PRNG has a range [000000, 999999]. The normalization could be achieved by dividing the whole sequence by 999,999, such that the resultant range is [0, 1]. (2) Then minus 0.5 for all the terms, such that the range becomes [-0.5, 0.5].

If the rescaled random numbers are evenly distributed and with little dependence, then the increment of squared displacement of the n-th particle ΔS_n after each iteration could be shown as a constant value:

$$\Delta S_n = \int_{-0.5}^{0.5} \int_{-0.5}^{0.5} \int_{-0.5}^{0.5} ({s_x}^2 + {s_y}^2 + {s_z}^2) ds_x \, ds_y \, ds_z = \frac{1}{4}$$

where s_x, s_y, s_z are the rescaled random term on x, y, z direction respectively.

With such property, it can be deduced that the expected squared displacement of each particle S_i would show a linear relation with the number of iterations I:

$$S_i(t_0 + I\Delta t) = I/4$$

where t_0 is the initial time, Δt is the time step of each iteration. Obviously, the expected mean squared displacement value of the system obey the same relation, viz:

$$\overline{S}(t_0 + I\Delta t) = I/4$$

where $\overline{S}$ is the expected mean squared displacement value. While the computational result is analyzed via:

$$\overline{S_c}(t) = \frac{1}{N} \sum_{n=1}^{N} (X_n - X_{0n})^2 + (Y_n - Y_{0n})^2 + (Z_n - Z_{0n})^2$$

where $\overline{S_c}(t)$ represents the computational mean squared displacement value at time t, N is the total

number of particles, X_n is the x-position of the n-th particle at time t, X_{0n} is the initial x-position of the n-th particle, Y and Z follow the same pattern. The validity of the computational result, which could be determined by the coefficient of correlation of the linear regression method.

Besides the regression method, using *Chi Squared* test to access the appearance of the particles in each octant could be an alternative analysis way.

In specific, the initial position of each particle will be considered as its origin of frame. Certain time later, if n particles appear in the first octant, such that the coordinates of x, y, z are all positive value with respective to their own frame, then the appearance of particles in the first octant is recognized as n. The other octants follow the same concept as the first octant.

Then the result of the 8 octants will be test by the *Chi Squared* method. For which, the null hypothesis is the 8 octants obtain equal amounts of particles. For 99.9% cumulative probability, the *Chi Squared* critical value is 24.3 [6], that means if the obtained *Chi Squared* value is larger than 24.3, there is only 0.1% that the particles are evenly distributed, and hence the PRNG will be considered as not random.

Whereas the *Chi Squared* value is calculated by:

$$\chi^2 = \sum_{i=1}^{8} \frac{(R_i - E_i)^2}{E_i}$$

where R_i is the obtained value on the i-th octant, while E_i is the expectation value on the i-th octant. In this test, the 1000-particles system has $E_i = 125$ for all i.

IV. RESULT

In this section will only present the numerical results of the tests. The graphical results of the tests are attached to APPENDIX I.

A. Middle-Square method

1. Distribution test

The distribution tests results of the sequences generated by MS method with seeds 123456, 987654 and 104729 are shown as following:

TABLE I. the *Kolmogorov Smirnov* test result for MS.

Seed	D_{max}	Occurred at	Validity
123456	0.04393	0.07850	Fail
987654	0.20445	0.53818	Fail
104729	0.04413	0.07850	Fail

TABLE II. the frequency test result for MS.

Seed	Repetition	Estimated Period
123456	381	(262.5, 263.2)
987654	4160	(24.0, 24.0)
104729	383	(261.1, 261.8)

TABLE III. the mean value analysis result for MS.

Seed	Mean	Standard deviation	Validity
123456	0.49489	0.28763	Fail
987654	0.56121	0.26700	Fail
104729	0.49480	0.28759	Fail

2. Independence test

The independence test results of the sequences generated by MS method with seeds 123456, 987654 and 104729 is shown as following:

TABLE IV. the mean different test result for MS.

Seed	Mean	Standard deviation	Validity
123456	0.36332	0.22660	Fail
987654	0.33705	0.23630	Fail
104729	0.36343	0.22648	Fail

3. Physical system test

The tests results on the random walk system using MS method with seeds 123456, 987654 and 104729 are shown as following:

TABLE V. the regression results of the mean squared displacement data generated by MS.

Seed	Coefficient of correlation
123456	0.84195
987654	$<<1$
104729	0.84233

TABLE VI. the regression result of the mean squared displacement data generated by MS.

Seed	χ^2 value	Validity
123456	7000.0	Fail
987654	1000.0	Fail
104729	7000.0	Fail

B. Linear Congruential generator

1. Distribution test

The distribution tests results of the sequences generated by LCG method with the 3 set of parameters are shown as following:

TABLE VII. the *Kolmogorov Smirnov* test result for LCG.

Set	D_{max}	Occurred at	Validity
1	0.00258	0.47860	Pass
2	0.00227	0.45624	Pass
3	0.01000	0.99000	Fail

TABLE VIII. the frequency test result for LCG.

Set	Repetition	Estimated Period
1	No repeat	>100000
2	No repeat	>100000
3	1000	(100, 100.1)

TABLE IX. the mean value analysis result for LCG.

Set	Mean	Standard deviation	Validity
1	0.50071	0.28832	Pass
2	0.50048	0.28837	Pass
3	0.49801	0.28857	Pass

2. Independence test

The independence test results of the sequences generated by LCG method with the 3 set of parameters is shown as following:

TABLE X. the mean different test result for LCG.

Set	Mean	Standard deviation	Validity
1	0.33356	0.23548	Pass
2	0.33211	0.23531	Pass
3	0.27568	0.18155	Fail

3. Physical system test

The tests results on the random walk system using LCG method with the 3 set of parameters are shown as following:

TABLE XI. the regression result of the mean squared displacement data generated by LCG.

Set	Coefficient of correlation
1	0.99866
2	0.99747
3	<<1

TABLE XII. the regression result of the mean squared displacement data generated by LCG.

Set	χ^2 value	Validity
1	2.9120	Pass
2	8.4960	Pass
3	219.20	Fail

C. Mersenne Twister

1. Distribution test

The distribution tests results of the sequences generated by MT method with seed 104729 are shown as following:

TABLE XIII. the *Kolmogorov Smirnov* test result for MT.

Seed	D_{max}	Occurred at	Validity
104729	0.00404	0.63374	Pass

TABLE XIV. the frequency test result for MT.

Seed	Repetition	Estimated Period
104729	No repeat	>100000

TABLE XV. the mean value analysis result for MT.

Seed	Mean	Standard deviation	Validity
104729	0.49840	0.28851	Pass

2. Independence test

The independence test results of the sequences generated by MT method with seed 104729 is shown as following:

TABLE XVI. the mean different test result for MT.

Seed	Mean	Standard deviation	Validity
104729	0.33316	0.23577	Pass

3. Physical system test

The tests results on the random walk system using MT method with seed 104729 are shown as following:

TABLE XVII. the regression result of the mean squared displacement data generated by MT.

Seed	Coefficient of correlation
104729	0.99356

TABLE XVIII. the regression result of the mean squared displacement data generated by MT.

Seed	χ^2 value	Validity
104729	3.3120	Pass

V. DISCUSSION

Based on the analysis in the previous section, the

PRNG implemented by MS method failed all the indicators. Refer to the graphs in the APPENDIX I, Figure 1, 2 and 3 show that the MS method would converge to a repeating pattern with a very short period. Noticeably, both MS seeded with 123456 and 104729 are converged to the same repeating pattern, referring to Figure 1 and 3. Moreover, the discrepancy of the repeating pattern is large. For example, refer to Figure 2, the result generated by MS with seed 987654, there are 3 major gaps occurred in the repeating pattern with width of about 0.25, means that it will skip most values in the desired range. Hence using MS method for physical simulation is not reliable.

As for the PRNG implemented by LCG method, the result revealed that the parameters with large value of a and m could generate high quality random number sequences that are able to pass all the mentioned tests, which can also simulate the random walk system with acceptable result. Whereas the parameters with small value of a and m, however, has similar defect as the MS method. It will converge to a repeating pattern and has discrepancy in the desired range. All in all, implement the PRNG by LCG method will some large value of parameters could produce high quality random number in an efficient manner.

As for the PRNG implemented by MT method, the result is like the LCG method with large parameters. It passed all the tests despite the mean value and the mean different are slightly smaller than the expected value. Surprisingly, this slightly deviation would accumulate during the random walk simulation. As shown in Figure 21, the computational result of the mean squared displacement tends to be more negative deviated form the expected value, which does not occur in the system generated by LCG with large parameters. However, with the consideration of its high efficiency and extremely long period. MT is still a good choice of PRNG for the CUHK PHYS3061 Computer Simulations of Physical Systems course.

As for the concern on the consistence of the tests, despite the tests on LCG with small parameters, all other tests results are consistent with each other. Which means using some but not all above tests could still draw a similar conclusion.

VI. CONCLUSION

To conclude, both PRNG implemented by the LCG with large parameters, and the MT method are good enough to finish the coursework from the CUHK PHYS3061 Computer Simulations of Physical Systems course.

VII. REFERENCE

[1] P. L'Ecuyer, History of Uniform Random Number Generation, IEEE Press, 202–230 (2017).

[2] D. P. Kroese et al., Why the Monte Carlo method is so important today, WIREs Comput Stat, **6**, 6, 386–392 (2014).

[3] P. L'Ecuyer et al., TestU01: A C library for empirical testing of random number generators, ACM Trans, Math. Softw, 33, **4**, 22 (2007)

[4] Jr. Warren et al., Hacker's Delight (2 ed.). Addison Wesley - Pearson Education Inc., 81–96 (2013)

[5] G. Marsaglia, Yet another RNG, sci.stat.math (2018).

[6] R. V. Hogg et al., Probability and Statistical Inference (8 ed.), Pearson Education International, 3,6-8, 141-156, 273-461 (2010).

APPENDIX I. : GRAPHICAL RESULT

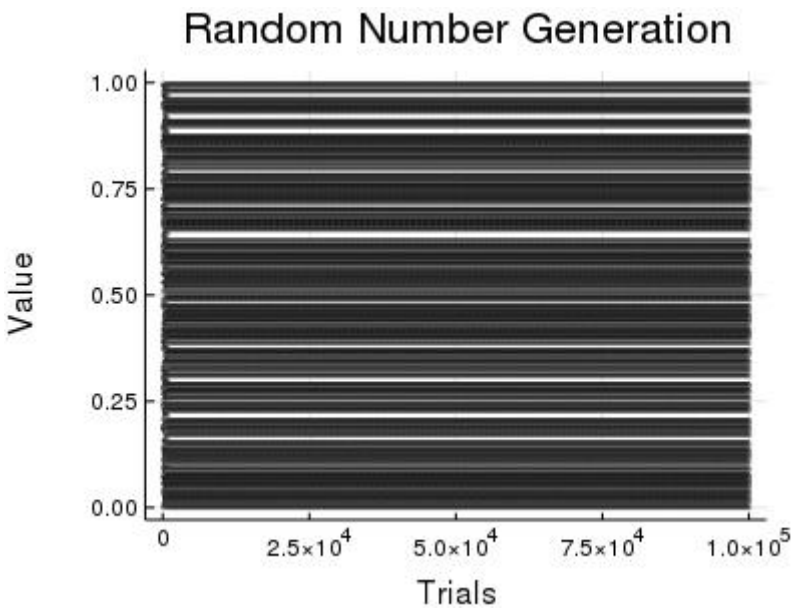

Figure 1. The plot of the values on the sequence generated by MS method with seed 123456.

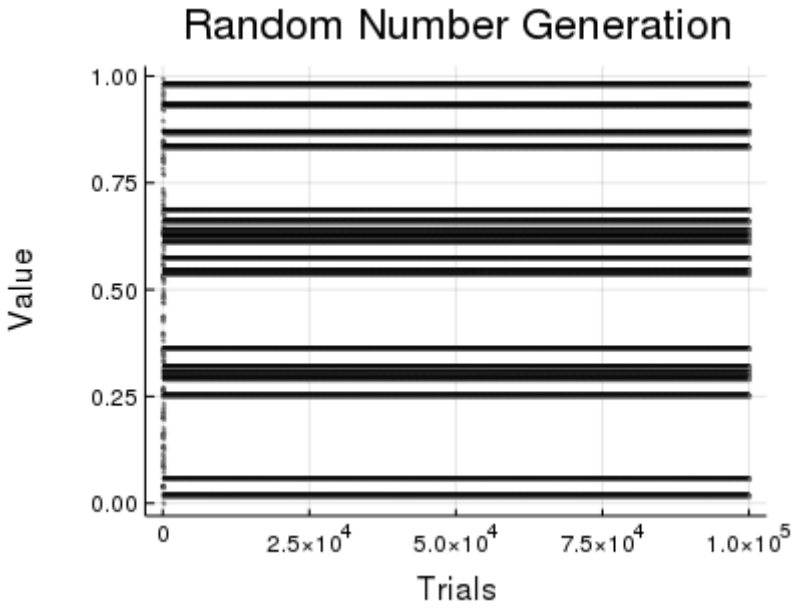

Figure 2. The plot of the values on the sequence generated by MS method with seed 987654.

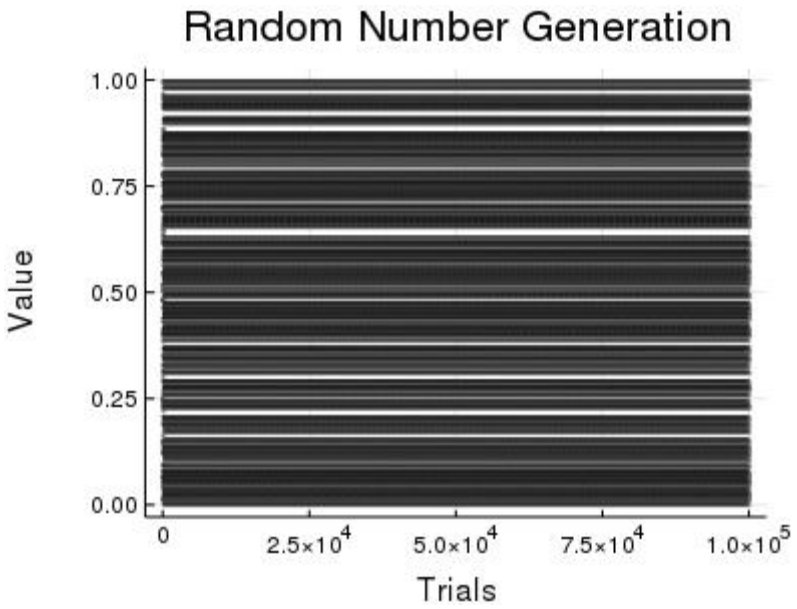

Figure 3. The plot of the values on the sequence generated by MS method with seed 104729.

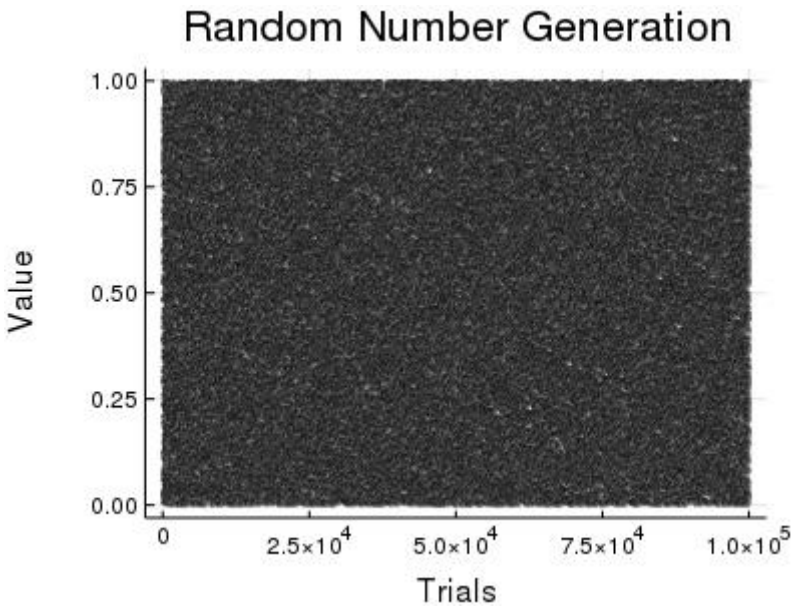

Figure 4. The plot of the values on the sequence generated by LCG method with the set 1 parameters.

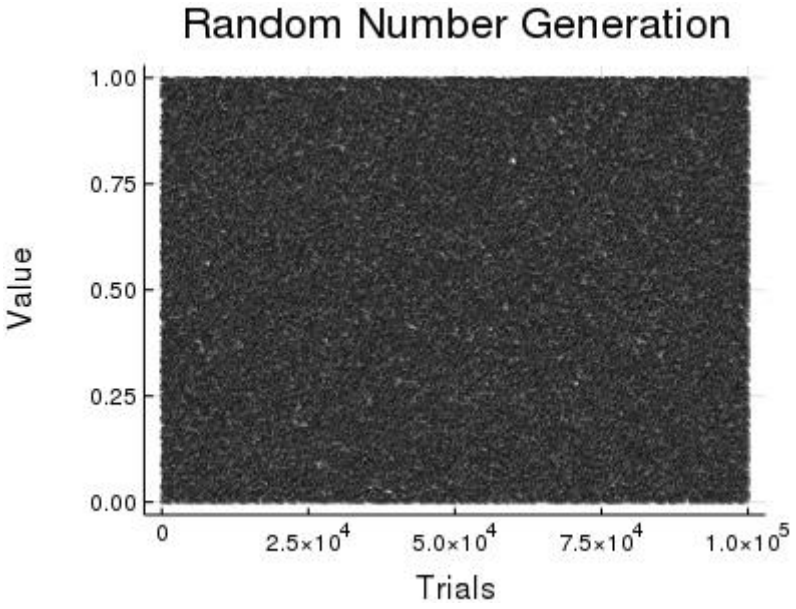

Figure 5. The plot of the values on the sequence generated by LCG method with the set 2 parameters.

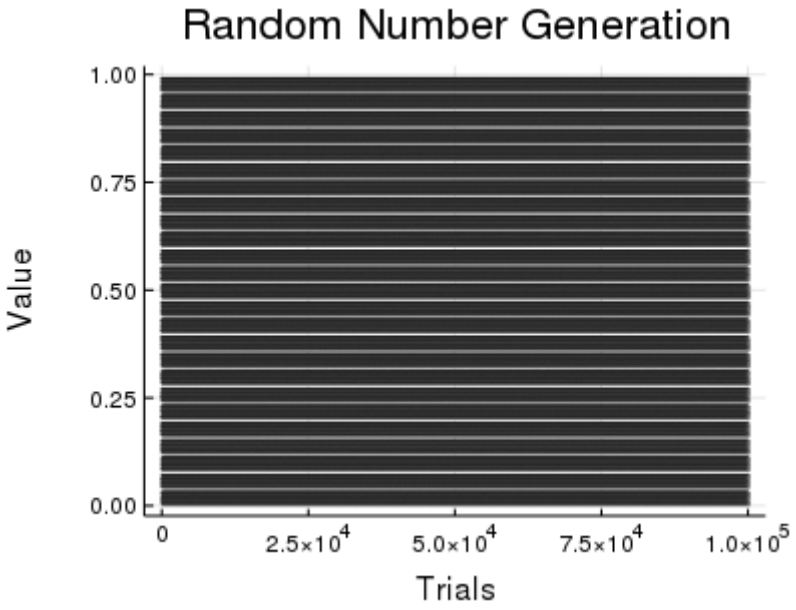

Figure 6. The plot of the values on the sequence generated by LCG method with the set 3 parameters.

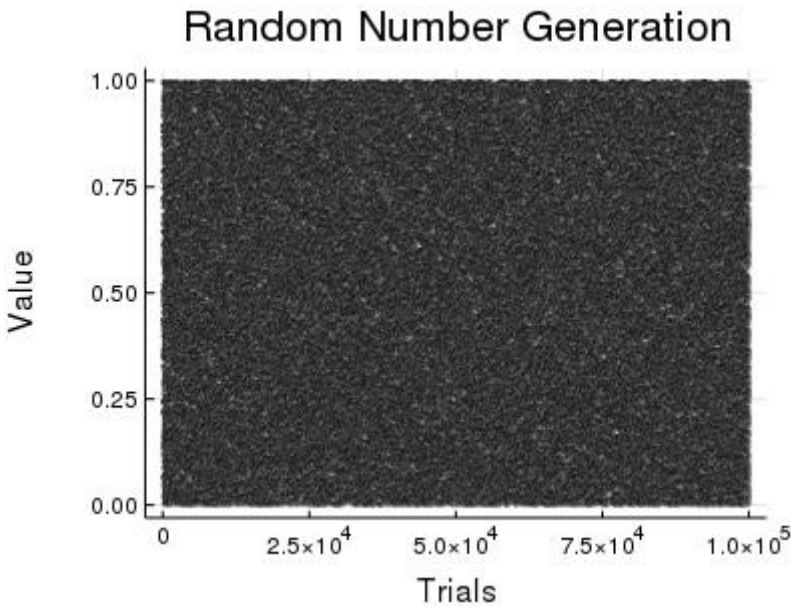

Figure 7. The plot of the values on the sequence generated by MT method with seed 104729.

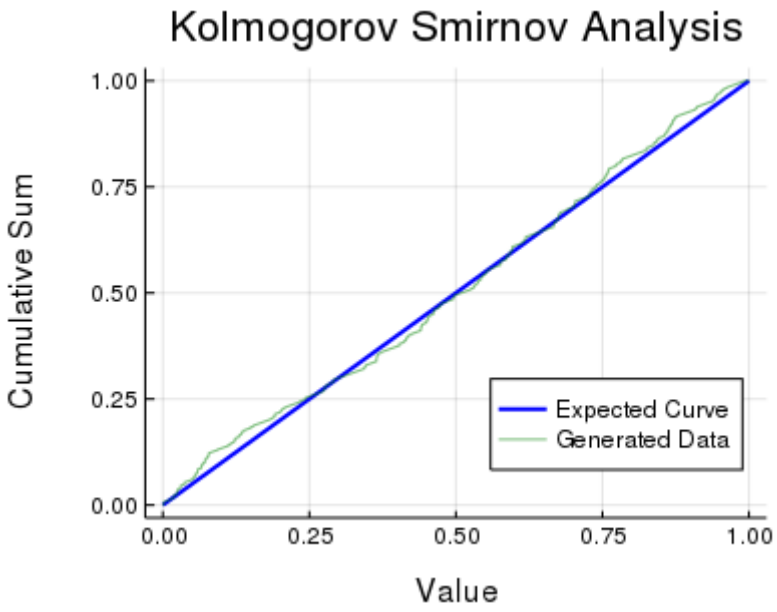

Figure 8. The plot of the Kolmogorov Smirnov test result on the sequence generated by MS method with seed 123456.

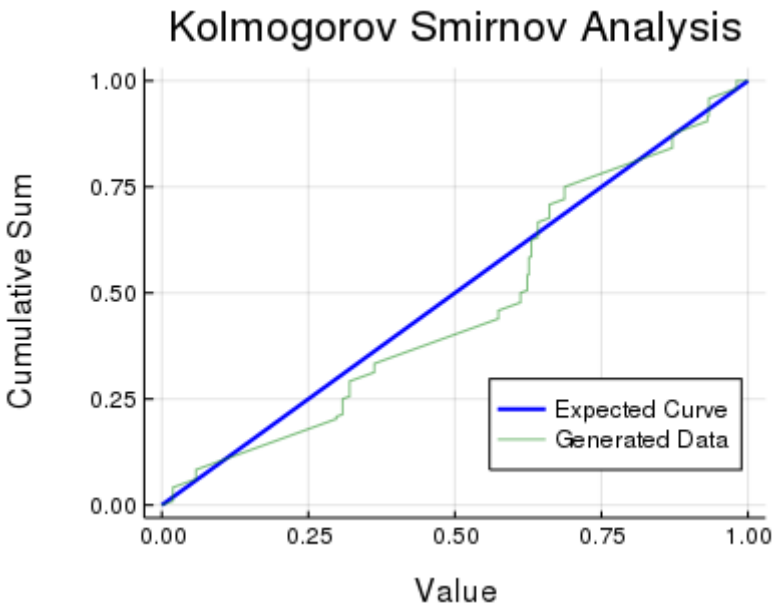

Figure 9. The plot of the Kolmogorov Smirnov test result on the sequence generated by MS method with seed 987654.

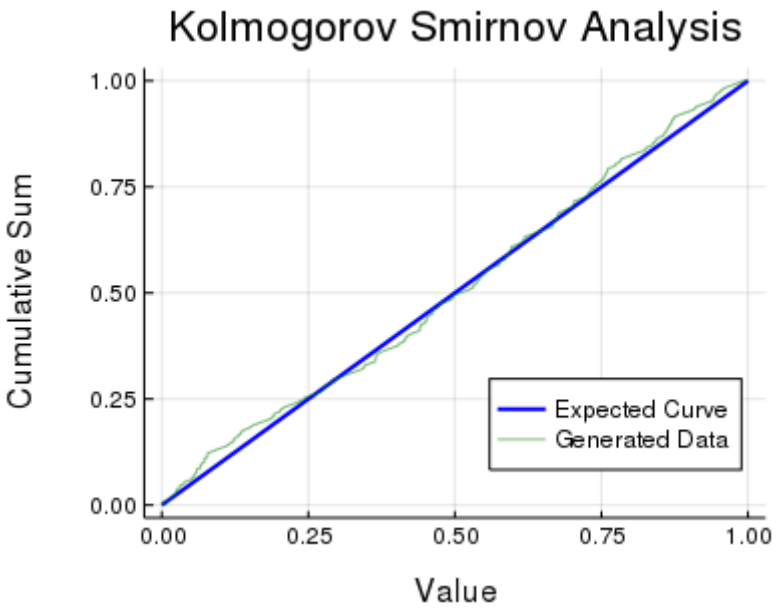

Figure 10. The plot of the Kolmogorov Smirnov test result on the sequence generated by MS method with seed 104927.

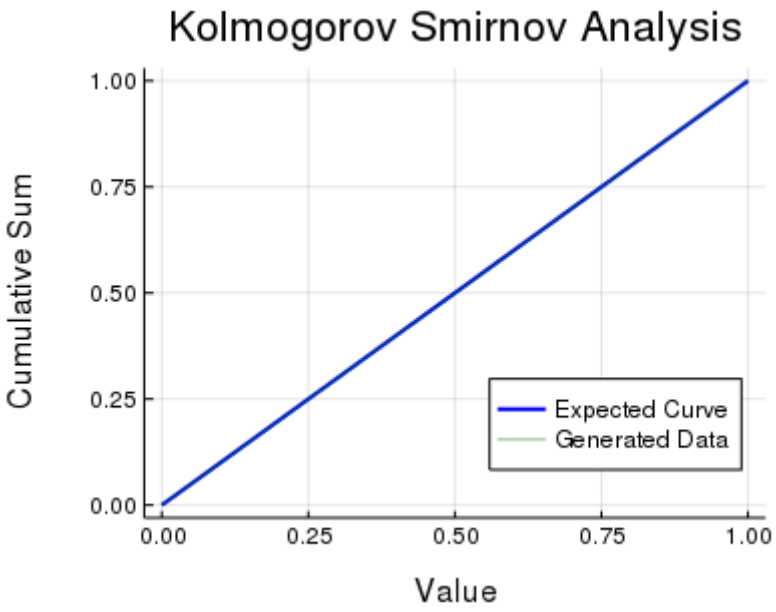

Figure 11. The plot of the Kolmogorov Smirnov test result on the sequence generated by LCG method with the set 1 parameters.

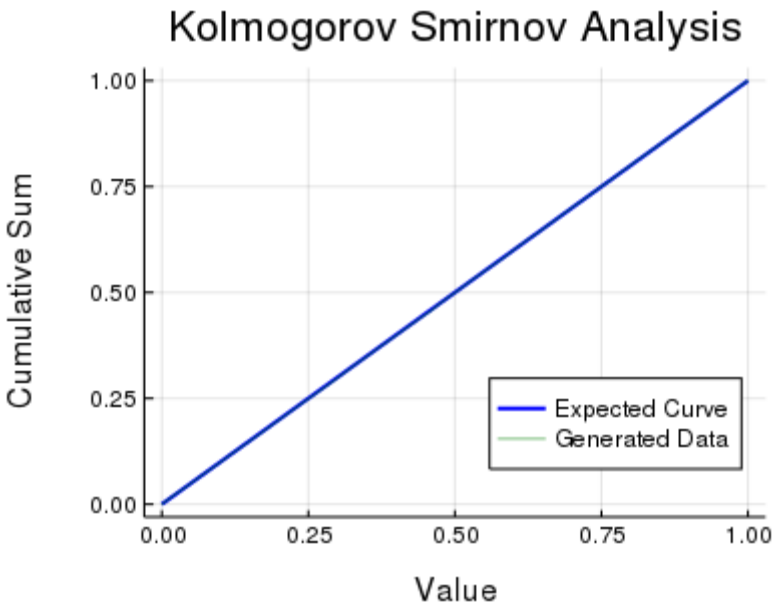

Figure 12. The plot of the Kolmogorov Smirnov test result on the sequence generated by LCG method with the set 2 parameters.

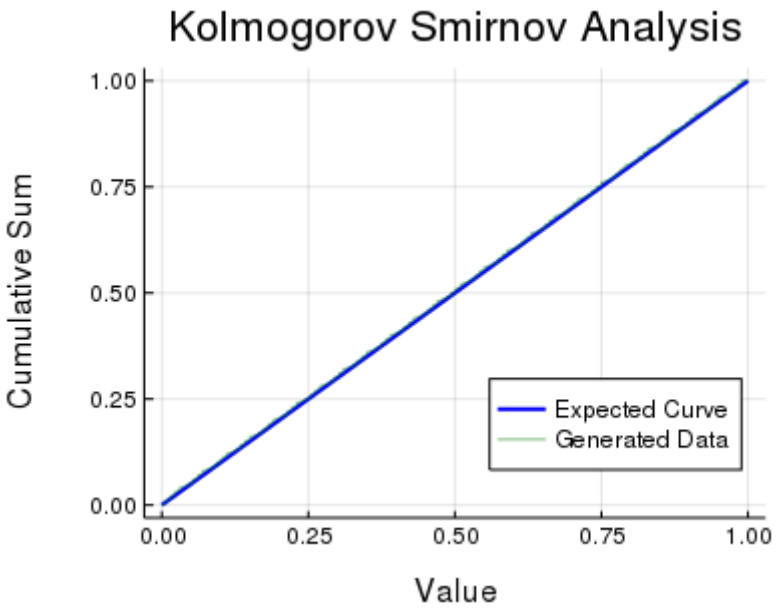

Figure 13. The plot of the Kolmogorov Smirnov test result on the sequence generated by LCG method with the set 3 parameters.

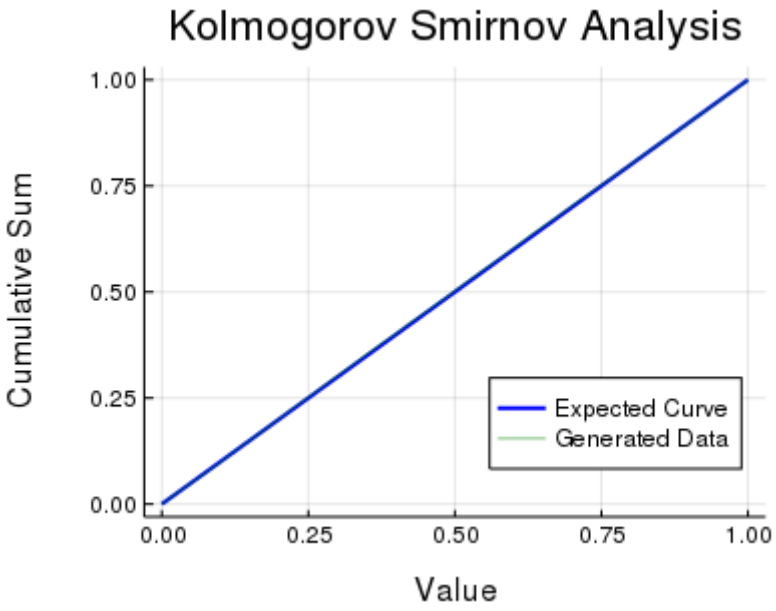

Figure 14. The plot of the Kolmogorov Smirnov test result on the sequence generated by MT method with seed 104729.

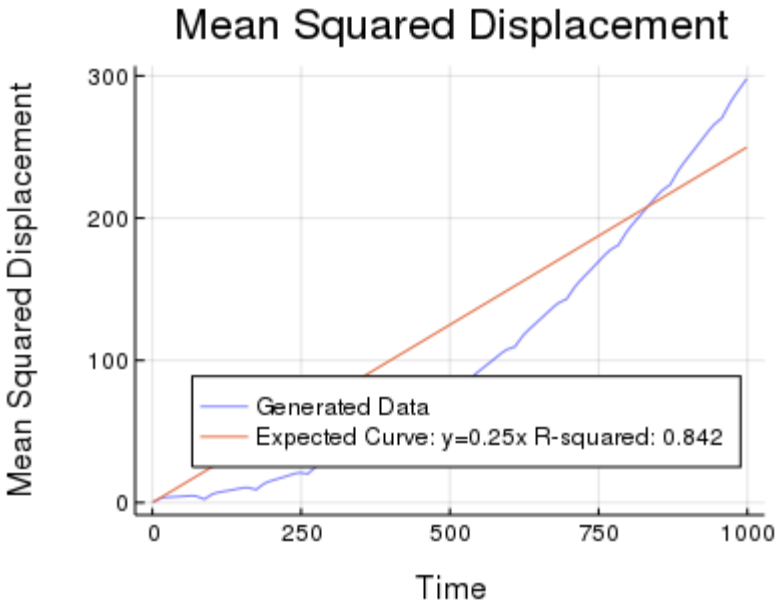

Figure 15. The plot of the mean squared displacement of the system generated by MS method with seed 123456. The horizontal axis (Time) represents the number of iterations.

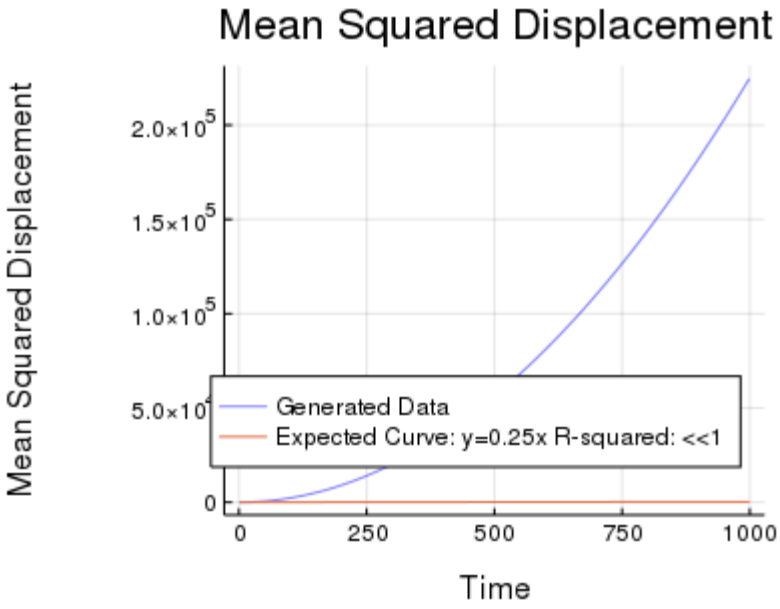

Figure 16. The plot of the mean squared displacement of the system generated by MS method with seed 987654. The horizontal axis (Time) represents the number of iterations.

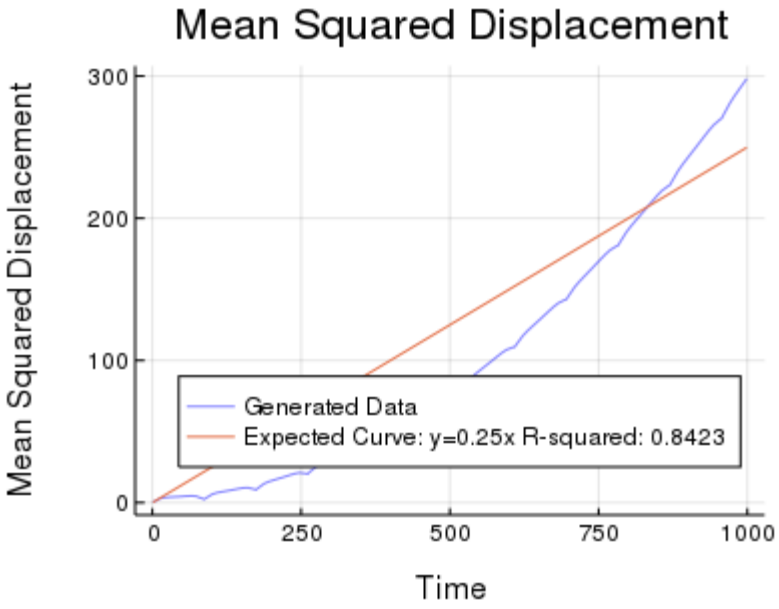

Figure 17. The plot of the mean squared displacement of the system generated by MS method with seed 104729. The horizontal axis (Time) represents the number of iterations.

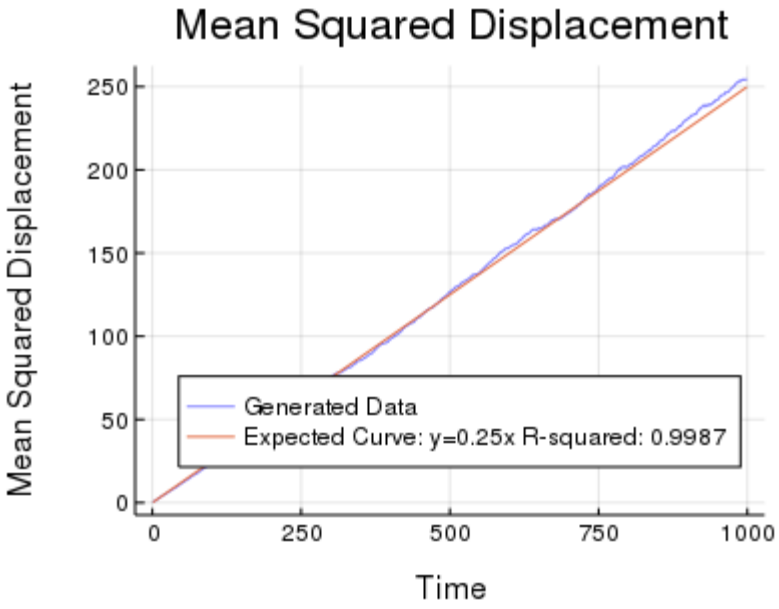

Figure 18. The plot of the mean squared displacement of the system generated by LCG method with set 1 parameters. The horizontal axis (Time) represents the number of iterations.

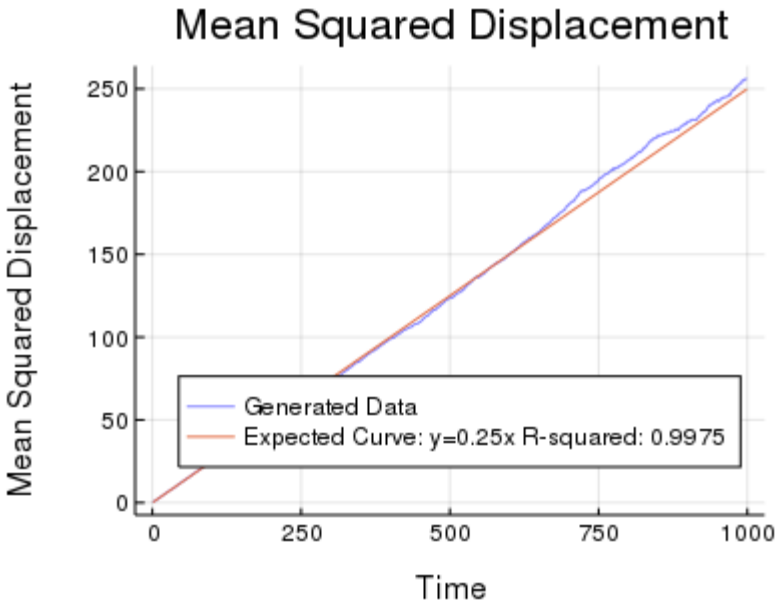

Figure 19. The plot of the mean squared displacement of the system generated by LCG method with set 2 parameters. The horizontal axis (Time) represents the number of iterations.

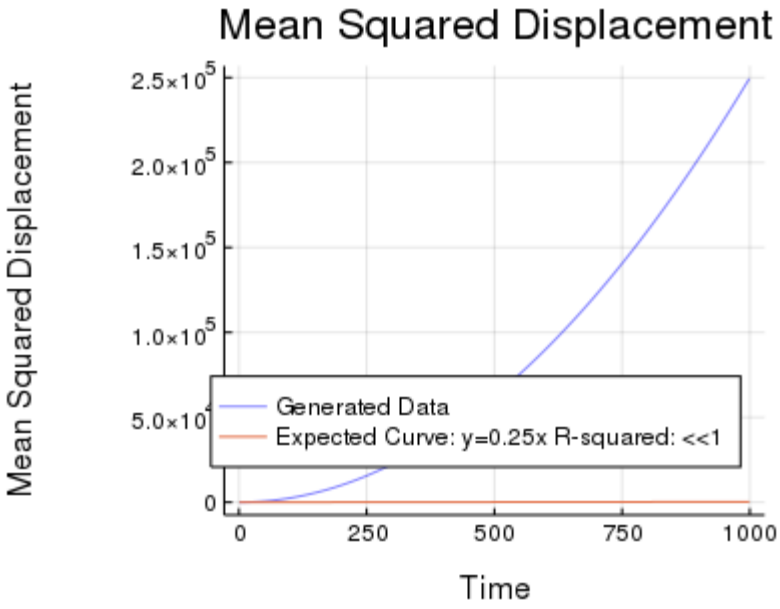

Figure 20. The plot of the mean squared displacement of the system generated by LCG method with set 3 parameters. The horizontal axis (Time) represents the number of iterations.

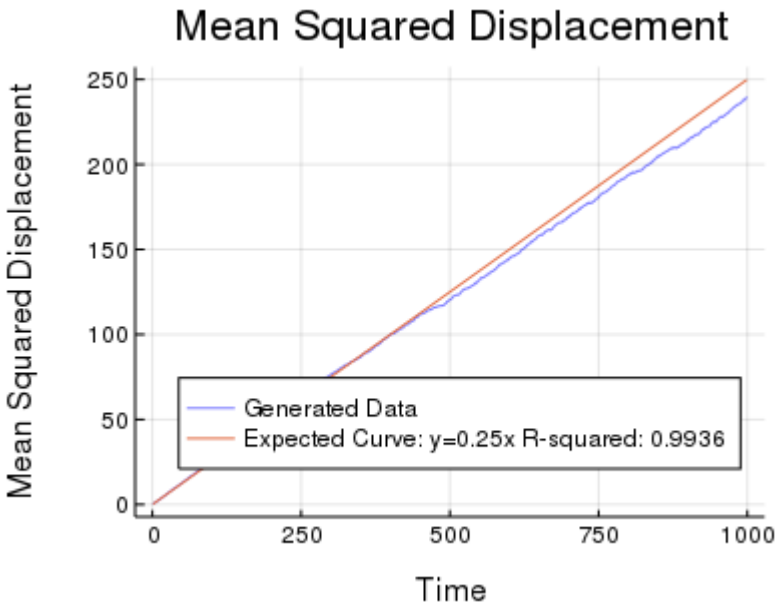

Figure 21. The plot of the mean squared displacement of the system generated by MT method with seed 104729. The horizontal axis (Time) represents the number of iterations.

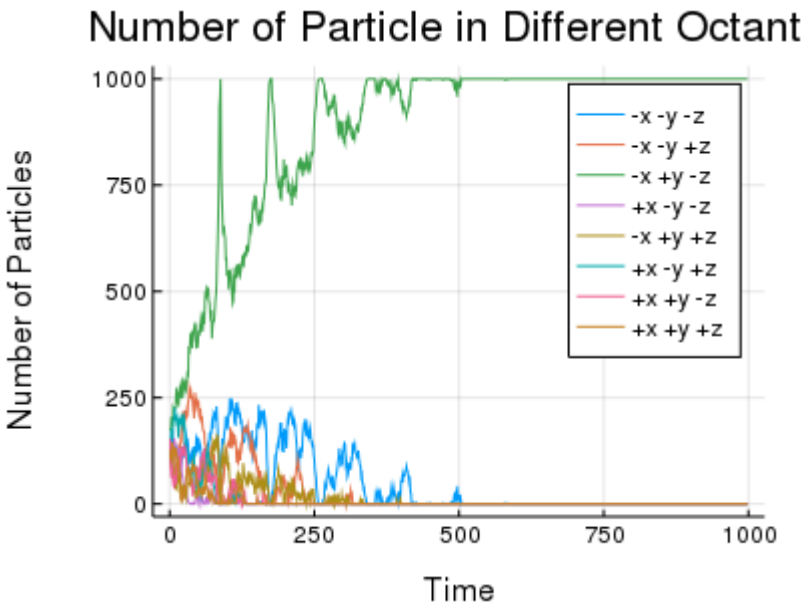

Figure 22. The plot of the number of particles in different octant of the system generated by MS method with seed 123456. The horizontal axis (Time) represents the number of iterations.

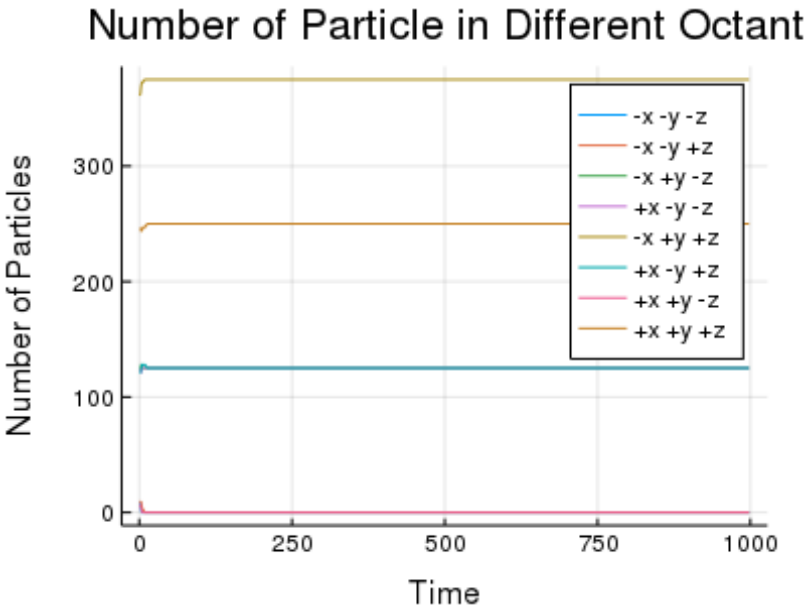

Figure 23. The plot of the number of particles in different octant of the system generated by MS method with seed 987654. The horizontal axis (Time) represents the number of iterations.

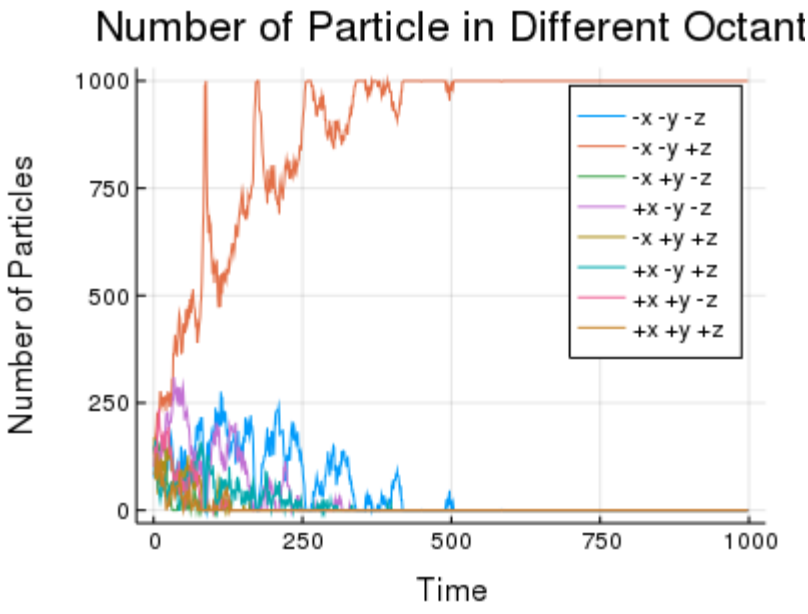

Figure 24. The plot of the number of particles in different octant of the system generated by MS method with seed 104729. The horizontal axis (Time) represents the number of iterations.

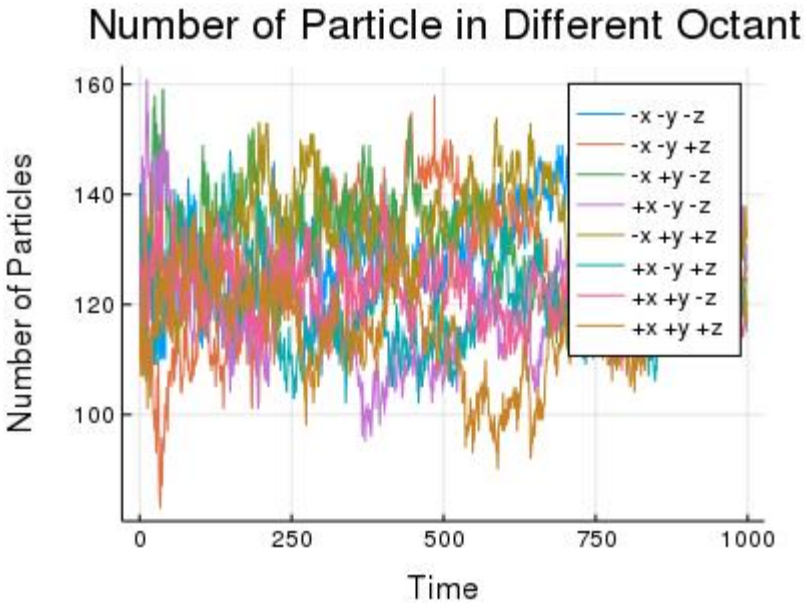

Figure 25. The plot of the number of particles in different octant of the system generated by LCG method with set 1 parameters. The horizontal axis (Time) represents the number of iterations.

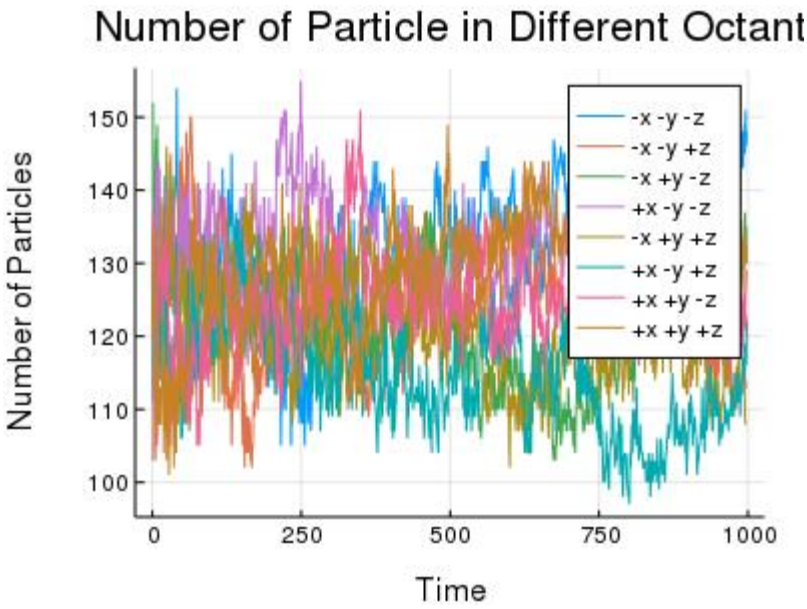

Figure 26. The plot of the number of particles in different octant of the system generated by LCG method with set 2 parameters. The horizontal axis (Time) represents the number of iterations.

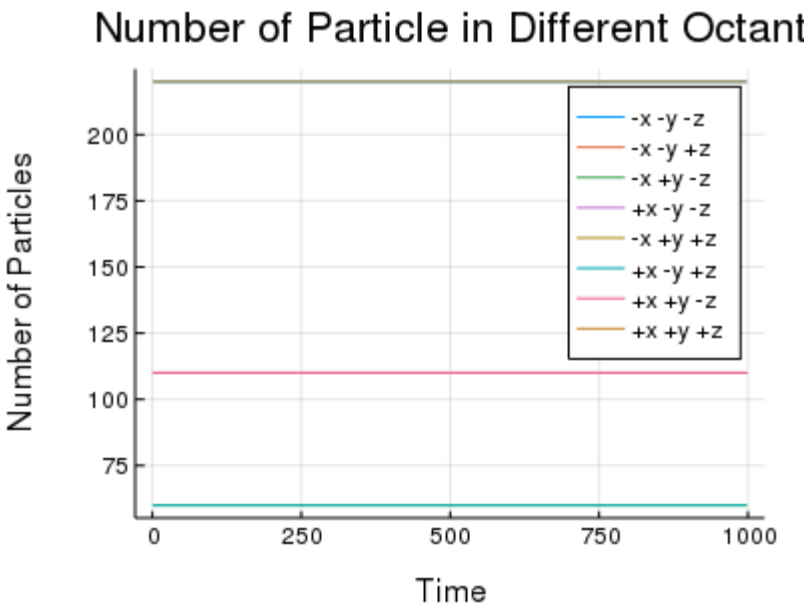

Figure 27. The plot of the number of particles in different octant of the system generated by LCG method with set 3 parameters. The horizontal axis (Time) represents the number of iterations.

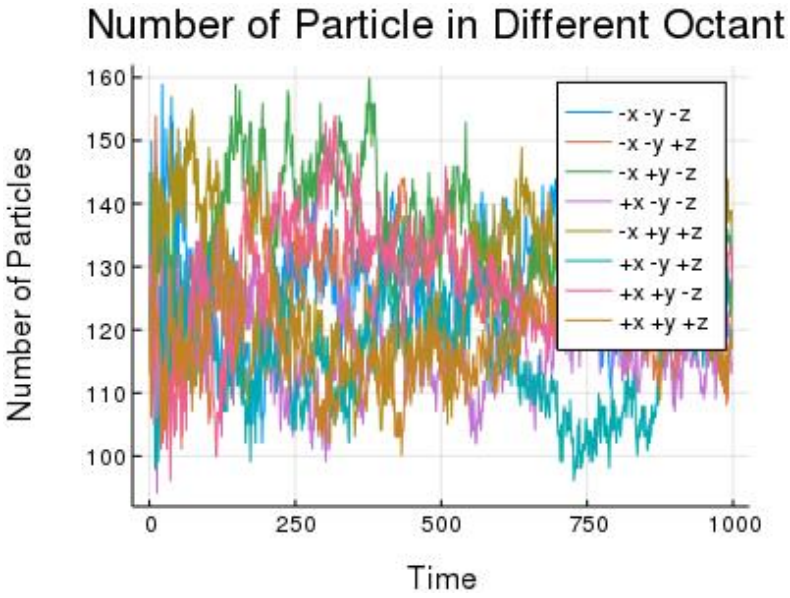

Figure 28. The plot of the number of particles in different octant of the system generated by MT method with seed 104729. The horizontal axis (Time) represents the number of iterations.