Improved Linear Congruential Random Number Generators

SHEN Huayun, ZHANG Peng and WANG Kan
Department of Engineering Physics,
Tsinghua University, Beijing, 100084, P. R. China
chen-hy03@mails.tsinghua.edu.cn,
f-z03@mails.tsinghua.edu.cn,
wangkan@tsinghua.edu.cn.

Abstract

The random number generator in MCNP and most other Monte Carlo codes for particle transport (e.g, RCP, RACER, MORSE, KENO, VIM, MCPP) is based on algorithms called linear congruential random number generators(LCGs). However, all LCGs have a defect that cannot be removed by adjusting their parameters. The problem lies in the "crystalline" nature of LCGs. This article presents the improved linear congruential random number generators (ILCGs), which preserve the advantage of rapidity, and the statistical tests indicate that the statistical qualities have been obviously improved.

1. Introduction

A random number sequence is essential for solving neutron transport by Monte Carlo method. To generate such a sequence, there are three generators which are physical random number generators, random number table and mathematic random number generators. Compare with the first two, the mathematic method has a simple, efficient and repeatable character. Although the sequence generated by the mathematic method is pseudorandom, it can be used in the same way if the sequence passes the randomness tests as the real random numbers.

According to the choice of the recursion formula, there are many random number generators, such as linear congruential random number generators (LCGs), quadratic congruential random number generators and additive generators [1]. By far, the LCGs which are introduced by D.H. Lehmer in 1949, are the most popular random number generators. They are adopted by many Monte Carlo codes for particle transport (e.g, MCNP, RCP, PACER, MORSE, KENO, VIM, and MCPP). This article is introducing these generators and discussing the improvement for them.

2. Linear congruential random number generators (LCGs)

The recursion formula for LCGs is as follows,

$$\begin{aligned} x_n &= a x_{n-1} + c \pmod{M} \\ \xi_n &= x_n / M \qquad\qquad n \geq 1 \end{aligned} \tag{1}$$

where,

M, the modulus; $M > 0$.
a, the multiplier; $0 \le a < M$.
c, the increment; $0 \le c < M$.
x_0, the initial seed; $0 \le x_0 < M$.
ξ_n, the random number; $0 \le \xi_n < 1$.

On how to select the parameters, many papers have given the detailed description. Especially, sets of parameters for LCGs of different sizes and good performance with respect to the spectral test are presented in References 2 and 3.The form LCGs(M, a, c, x_0) is used to denote the generator, so that the traditional MCNP generator is LCGs(2^{48}, 5^{19}, 0, 5^{19})[2].

3. Improved linear congruential random number generators (ILCGs)

Although LCGs are simple and effective and have successfully applied for various problems, all LCGs have a defect that cannot be removed by adjusting their parameters. The problem lies in the "crystalline" nature of LCGs[4]. And each random number in the sequence by LCGs is decided by the former, so the randomness is doubtful. Thereby, randomizing by shuffling are presented and two algorithms are introduced in the reference 1 as follows,

Algorithm 1: Generate two sequences $<\xi_n>$ and $<\xi_n'>$ from two different LCGs. Use an auxiliary table $V[0]$, $V[1]$, $\cdots$, $V[k\text{-}1]$, where k is some number chosen for convenience, and fill the V-table with the first k values of the ξ_n-sequence.

A1. Set ξ_n and ξ_n' equal to the next members of the sequences $<\xi_n>$ and $<\xi_n'>$, respectively.

A2. Set $j \leftarrow \lfloor k\xi_n' \rfloor$; that is , j is a random value, $0 \le j < k$, determined by ξ_n'.

A3. Output $V[j]$ and then set $V[j] \leftarrow \xi_n$.

Algorithm 2: Generate a sequence $<\xi_n>$ from a given LCGs. Use an auxiliary table $V[0]$, $V[1]$, $\cdots$, $V[k\text{-}1]$ as in Algorithm 1. Fill the V-table with the first k values of the ξ_n-sequence and set an auxiliary variable Y equal to the (k+1)st value.

A1. Set $j \leftarrow \lfloor kY \rfloor$; that is , j is a random value, $0 \le j < k$, determined by Y.

A2. Set $Y \leftarrow V[j]$, output Y, and then set $v[j]$ to the next number of the sequence $<\xi_n>$

From the two algorithms, it is obvious that the key of randomizing by shuffling is how to obtain the variable j. Considering the computing rapidity, the second is better than the first. Still, it is significative to discuss if there are more effective approaches for the variable j.

In order to enhance the performance of LCGs, its modulus is usually equal to 2^S. In this case, the first expression of the formulas (1) can be replaced by bit manipulations. Likewise, can the variable j be obtained by bit manipulations? It is possible. Before introducing the algorithm, let's review the storage format of an integer in the binary computer, see figure 1.

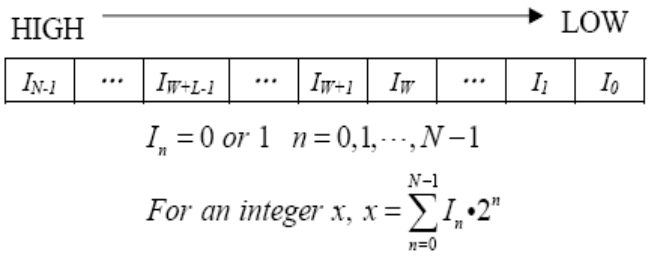

Figure 1 The sketch map of the storage format of an integer in the binary computer.

Looking back on the recursion formula of LCGs, two integers, ax_{n-1} and x_n, will be obtained before each random number ξ_n. Certainly, they will be kept in the computer according to the format of figure 1. So, a new approach of randomizing by shuffling can be adopted.

Algorithm 3: Generate a sequence $<\xi_n>$ from a given LCGs. Use an auxiliary table $V[0]$, $V[1]$, ⋯, $V[k\text{-}1]$ as in Algorithms 1 and 2, and set $L \leftarrow \lceil Log_2(k) \rceil$. Fill the V-table with the first k values of the ξ_n-sequence.

A1. Generate the next numbers ax_{n-1}, x_n and ξ_n by the recursion formula.

A2. Set $j \leftarrow \sum_{l=0}^{L-1} I_{W+l} \bullet 2^l$, where, the meaning of I_{W+l} is shown in the figure 1, and the integer x in the figure 1 is equal to ax_{n-1} or x_n. If $j \geq k$, set $j \leftarrow j - k$.

A3. Output $V[j]$ and then set $V[j] \leftarrow \xi_n$.

In the same way as $M=2^S$ in the LCGs, the parameter k is always equal to 2^L. Then the second step of the algorithm 3 can be simplified, and changed into 'A2. Set $j \leftarrow \sum_{l=0}^{L-1} I_{W+l} \bullet 2^l$ '.That is, the inequation $j<k$ is always satisfied, so the following judgement can be omitted. And what's more, the process A2 can be implemented by bit manipulations, so the effect for the efficiency is very small, which has been verified by the testing result in section 4.1.

Similarly, the form ILCGs(M, a, c, x_0, W, L, T) is used to denote the improved generators, where T=1 indicates $x=x_n$, and T=2 indicates $x=ax_{n-1}$ in the step A2. It is vital that the step A2 can be completed by bit manipulations, so the ILCGs are as fast as the LCGs, which is testified by the testing results.

About the choice of the parameters W and L, there are some qualitative conclusions. If W is too small, it is not suitable because of the periodicity of the number composed of the several low bits of the integer x_n[1]. Considering the required memory spaces of V-table, it is advisable that L is not larger than 10. Finally, it is necessary to carry out some special statistical tests for a choice of the two parameters plus another parameter T.

4. Testing

To testify the capability of ILCGs, comparison tests have been processed for LCGs and ILCGs.

4.1 Traditional MCNP LCGs(2^{48}, 5^{19}, 0, 5^{19}) VS ILCGs(2^{48}, 5^{19}, 0, 5^{19}, 27, 3, 1)

To generate 10^9 random numbers, the two generators consume 76.1 seconds and 76.2 seconds, respectively.

Marsaglia's DIEHARD test suite for random number generators [5] was applied to the given LCGs and ILCGs. This test suite involves running over 200 variations on the statistical tests listed in Table 1. Reference 2 refers that for 3 tests, the overlapping pairs sparse occupancy, the overlapping quadruples sparse occupancy test and the DNA test, the traditional MCNP generator failed when the 10-12 bits of the random numbers were used for testing. Whereas, the ILCGs above passes all tests.

4.2 LCGs(2^{32}, 2891336453, 1, 1) from reference 3 VS ILCGs(2^{32}, 2891336453, 1, 1, 51, 9, 2)

Although this ILCG is a 32-bit generator, the product $a \bullet x_{n-1}$ is a 64-bit integer. Because the last parameter of the ILCG is equal to 2, it is reasonable to set the *W* equal to 51. Results of statistical tests by DIEHARD are listed on the following table.

DIEHARD Statistical Tests	LCG	ILCG
Birthday spacings test	***FAILED***	***PASSED***
Overlapping 5-permutation test	PASSED	PASSED
Binary rank test for 31x31 matrices	PASSED	PASSED
Binary rank test for 32x32 matrices	PASSED	PASSED
Binary rank test for 6x8 matrices	FAILED	FAILED
Bitstream test (Repeat twenty times)	***10 times PASSED***	***8 times PASSED***
Overlapping pairs sparse occupancy	***FAILED***	***IMPROVED***
Overlapping quadruples sparse occupancy	***FAILED***	***PASSED***
DNA test	***FAILED***	***PASSED***
Count the 1's in a stream of bytes	***FAILED***	***PASSED***
Count the 1's for specific bytes	***FAILED***	***PASSED***
Parking lot test	PASSED	PASSED
Minimum distance test	PASSED	PASSED
3D spheres test	PASSED	PASSED
Squeeze test	PASSED	PASSED
Overlapping sums test	PASSED	PASSED
Runs test	PASSED	PASSED
Craps test	PASSED	PASSED

Table 1 Results of statistical tests by DIEHARD.

5. Conclusion

As a result of the above testing, we believe that ILCGs are reliable methods of producing independent and uncorrelated random streams, which will exceed LCGs', although the adopted

approach seems very simple. Moreover, compared with the combination of random number generators, ILCGs preserve the advantage of rapidity, almost the same as the traditional LCGs.

6. ACKNOWLEDGMENTS

The research work was supported by the 973 Program（2007CB209800）and Project 10775081 supported by NSFC in China.

7. References

[1]. D.E. KNUTH, The Art of Computer Programming Volume 2, Seminumerical Algorithms, 3rd Edition, pp. 1-170, Addison-Wesley (1991).

[2]. F. B. BROWN and Y. NAGAYA, “The MCNP5 Random Number Generator,” Trans. Am. Nucl. Soc. 87, 230-232 (2002).

[3]. P. L’ESUYER, “Tables of Linear Congruential Generators of Different Sizes and Good Lattice Structure,” Math. of Comp., 68, 225, 249-260 (1999).

[4]. G. MARSAGLIA, “Random Numbers Fall Mainly in the Planes.” Proc. Nat. Acad. Sci., 1, 25-28. (1968).

[5]. G.S. MARSAGLIA, “The DIEHARD Battery of Tests of Randomness,” Available on the Internet at <http://stat.fsu.edu/pub/diehard>.