LA-UR-02-3782

Approved for public release;
distribution is unlimited.

Title: THE MCNP5 RANDOM NUMBER GENERATOR

Author(s): Forrest B. Brown,
Yasunobu Nagaya

Submitted to: American Nuclear Society
2002 Winter Meeting
November 17-21, 2002
Washington, DC

Los Alamos
NATIONAL LABORATORY

The MCNP5 Random Number Generator

Forrest B. Brown & **Yasunobu Nagaya**
Diagnostics Applications Group (X-5) Department of Nuclear Energy System
Los Alamos National Laboratory Japan Atomic Energy Research Institute
<fbrown@lanl.gov> <nagaya@mike.tokai.jaeri.go.jp>

INTRODUCTION

MCNP [1] and other Monte Carlo particle transport codes use random number generators to produce random variates from a uniform distribution on the interval [0,1). These random variates are then used in subsequent sampling from probability distributions to simulate the physical behavior of particles during the transport process. This paper describes the new random number generator developed for MCNP Version 5 [2]. The new generator will optionally preserve the exact random sequence of previous versions and is entirely conformant to the Fortran-90 standard, hence completely portable. In addition, new skip-ahead algorithms have been implemented to efficiently initialize the generator for new histories, a capability that greatly simplifies parallel algorithms. Further, the precision of the generator has been increased, extending the period by a factor of 10^5. Finally, the new generator has been subjected to 3 different sets of rigorous and extensive statistical tests to verify that it produces a sufficiently random sequence.

BACKGROUND

The random number generator in MCNP and most other Monte Carlo codes for particle transport (e.g,, RACER, MORSE, KENO, VIM, RCP, MCPP) is based on algorithms called linear congruential generators (LCGs) [3]. The basic LCG in these codes has been in use for at least 40 years, and has several desirable properties:

1. The sequence is deterministic, so that repeated calculations will produce identical results.
2. LCGs are very fast, involving only a small number of arithmetic operations.
3. Initialization is trivial, and the state information to specify the sequence for a history is small (1 word).
4. A simple algorithm exists for skipping ahead to any given point in the random sequence.
5. If at least 48 bits of precision are used in the LCG, the period is large ($>10^{14}$) and serial correlation is entirely negligible.
6. The algorithm is robust – it cannot fail.
7. An extensive body of literature exists for LCGs, providing a sound theoretical basis and guidance for the proper choice of algorithm parameters.

The LCG traditionally used by MCNP and other codes has the form

$$\begin{aligned} S_{k+1} &= g \cdot S_k + c \mod 2^M \\ r_{k+1} &= S_{k+1} / 2^M \end{aligned} \tag{1}$$

where S_k, g, and c are integers expressible in M bits or fewer, and r_{k+1} is a floating pointing number – the "random number" in the interval [0,1). The initial value of S_k, S_0, is called the initial seed for the generator. If c=0 and (g mod 8) = 3 or 5, then the generator has a period 2^{M-2}. If $c\neq 0$ and (g mod 4)=1 and c is odd, then the period is 2^M. The traditional LCG for MCNP uses $g=5^{19}$, c=0, $S_0=5^{19}$, and M=48. We will refer to generators in the form LCG(g, c, S_0, M), so that the traditional MCNP generator is LCG(5^{19}, 0, 5^{19}, 48).

Repeated application of Eq. (1) permits expressing the k^{th} seed in terms of the initial seed:

$$S_k = g^k \cdot S_0 + c \cdot \frac{g^k - 1}{g - 1} \mod 2^M \tag{2}$$

Eq. (2) must be computed using exact integer arithmetic. A simple algorithm for doing so has been previously presented in [4]. Eq. (2) can be used to skip ahead in the random sequence by an arbitrary

number of steps. This is particularly useful in initializing the random seed for a history in MCNP, since the skip-distance, or "stride" between starting random seeds for successive histories is fixed.

NEW RANDOM NUMBER GENERATOR

As part of a general upgrade to the MCNP code, a new random number package has been developed and implemented in MCNP Version 5. Because of the long experience with LCGs and the desirable features noted in the previous section, this new package is based on the LCG algorithm. This new package preserves all previously existing capabilities and provides many important new features:

Coding Considerations

The coding is entirely standard Fortran-90 and is completely portable. It has been tested on SGI, Sun, IBM, Compaq, HP, and Intel systems using a variety of Fortran-90 compilers. No special options or "#ifdefs" are required, and identical coding executes successfully on all systems. The generator has been implemented in a modular fashion, such that all parameters are private to the module and not subject to inadvertent side effects from other portions of the code. The module is thread-safe for parallel calculations using OpenMP threading. The module contains the LCG parameters, functions for generating random numbers, functions for skip-ahead in the random sequence and initialization for histories, unit tests for verifying correctness of the functions, and reference test results.

Algorithm Considerations

Parameters for 13 standard LCGs are included in the new random number package. These are listed in Table (1). The default LCG parameters are those for the traditional MCNP random number generator, set 1 in the table. Using the default parameters, the standard 48-bit LCG algorithm from previous versions of MCNP is preserved, yielding a bit-for-bit identical stream of random numbers for verification against previous versions of the code.

The standard 48-bit LCG algorithm in MCNP has a period of 7×10^{13}. With modern parallel computers, large calculations may simulate 10^9 or more particle histories (with $\sim 10^5$ random numbers per history), resulting in the reuse of portions of the random number sequence. To avoid this problem, the LCG algorithm has been extended to use up to 63-bits and incorporate an additive term, so that a period of 9.2×10^{18} is achieved. The use of a 63-bit integer LCG algorithm, rather than 64-bits, was deliberate, to avoid coding complications arising from the 2's complement form for integers (i.e., the leading bit is interpreted as a sign bit for Fortran on most systems). The LCG parameter sets 2-7 in Table (1) are natural extensions of the traditional 48-bit MCNP LCG to 63 bits. The LCG parameter sets 8-13 are recommended based on [5], where a search for "best" parameters was performed.

A new algorithm for fast "skip-ahead" using arbitrary strides in the sequence [4] has been implemented, greatly simplifying the LCG initialization for new histories. This feature is invaluable for reducing the complexity of parallel calculations.

Testing

Both the standard and extended LCG algorithms have been subjected to thorough testing, including:

1. The Unix utility "bc" was used to perform extended-precision integer arithmetic in order to generate reference values for the random streams generated by Eq. (1) for each of the sets of LCG parameters listed in Table (1). These reference values were compared to those generated by the Fortran-90 random number module. In addition, the reference values were included in the module data section for use in unit testing routines.
2. The standard statistical tests for random number generators described by Knuth [3] were applied to the random streams generated by each of the LCGs listed in Table (1), using test parameters from [6,7]. Coding for these tests was obtained from the SPRNG package [8]. The tests and parameters used are given in Table (2). In a few cases, a single test was flagged as suspect. Repeating the test with a different initial seed (which should not affect test results) resulted in

passing, indicating that some tests are sensitive to statistical fluctuations. None of the LCGs from Table (1) consistently failed any of the tests for randomness. (Note that these statistical tests cannot be used to prove randomness; consistent failure of several tests, however, is a good indicator of non-randomness.)

3. Marsaglia's DIEHARD test suite for random number generators [9] was applied to each of the LCGs listed in Table (1). This test suite involves running over 200 variations on the statistical tests listed in Table (3). Only one of the LCGs from Table (1) failed any of the tests, LCG set 1, the traditional 48-bit MCNP generator. For 3 tests – the overlapping pairs sparse occupancy, the overlapping quadruples sparse occupancy test, and the DNA test – the generator failed when the least-significant 10-12 bits of the random numbers were used for testing. The generator passed these tests when higher-order bits were used. These failures are not deemed serious, since it is well-known and understood that the least-significant bits of LCGs may be non-random and should not be used. In fact, these bits are not used directly in any portion of MCNP; only the higher-order bits are important.

CONCLUSIONS

As a result of the above testing, we believe that any of the 13 LCGs listed in Table (1) may be reliably used for Monte Carlo particle transport calculations. The traditional MCNP generator will remain the default, to provide consistency with older calculations. For new work, we are currently recommending sets 5-7 and 11-13. Any of these should be satisfactory and robust. These sets include an additive constant, so that the period of the generator will be 2^{63}, a factor of $\sim 10^5$ longer than the traditional MCNP generator.

The extended LCG algorithm is the foundation for future planned work in MCNP – providing independent random number generators for different particle types (in order to allow reproducibility when particle physics options are turned on/off). The use of different carefully selected additive constants has been shown both in theory and practice to be a reliable method of producing different independent and uncorrelated random streams.

The new random number generator package for MCNP is entirely self-contained and portable. It can be used directly in other Monte Carlo codes or stand-alone packages; there is no dependence on other portions of MCNP. It is highly recommended that other code developers make use of this thoroughly tested, well-proven package if new random number generator capabilities are needed.

REFERENCES

[1] J.F. Briesmeister, Ed., "MCNP – A General Monte Carlo N-Particle Transport Code – Version 4C," LA-13709-M, Los Alamos National Laboratory (March, 2000)

[2] F.B. Brown, et al., "MCNP Version 5," (this meeting), (Nov., 2002)

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

[4] F.B. Brown, "Random Number Generation with Arbitrary Strides," *Trans. Am. Nucl. Soc.* (Nov., 1994)

[5] P. L'Ecuyer, "Tables of Linear Congruential Generators of Different Sizes and Good Lattice Structure," *Math. of Comp.*, **68**, 225, pp 249-260 (1999)

[6] P. L'Ecuyer, "Efficient and Portable Combined Random Number Generators," *Comm. ACM* **31**(6): 742-749, 774 (1988)

[7] I. Vattulainen, et al., "A comparative study of some pseudorandom number generators," *Comput. Phys. Comm*. **86**, 209 (1995)

[8] M. Mascagni, et al., "The Scalable Parallel Random Number Generators Library (SPRNG) for ASCI Monte Carlo Computations," http://sprng.cs.fsu.edu

[9] G.S. Marsaglia, "The DIEHARD Battery of Tests of Randomness," http://stat.fsu.edu/pub/diehard

Index	g	c	M	S_0	stride	Comments
1	5^{19}	0	48	5^{19}	152,917	Traditional MCNP generator
2	5^{19}	0	63	5^{19}	152,917	
3	5^{23}	0	63	5^{19}	152,917	
4	5^{25}	0	63	5^{19}	152,917	
5	5^{19}	1	63	5^{19}	152,917	
6	5^{23}	1	63	5^{19}	152,917	
7	5^{25}	1	63	5^{19}	152,917	
8	3512401965023503517	0	63	1	152,917	From Ref. [5]
9	2444805353187672469	0	63	1	152,917	From Ref. [5]
10	1987591058829310733	0	63	1	152,917	From Ref. [5]
11	9219741426499971445	1	63	1	152,917	From Ref. [5]
12	2806196910506780709	1	63	1	152,917	From Ref. [5]
13	3249286849523012805	1	63	1	152,917	From Ref. [5]

Table 1. Standard LCG parameters for new MCNP5 random number generator

Knuth Statistical Tests	L'Ecuyer's test parameters	Vattulainen's test parameters
Equidistribution test	$N=10^4$, $n=10^3$, d=64	$N=10^4$, $n=10^4$, d=128
	$N=10^4$, $n=10^4$, d=256	$N=10^4$, $n=10^5$, d=256
Serial test	$N=10^3$, $n=10^5$, d=64	$N=10^3$, $n=10^5$, d=100
Gap test	$N=10^3$, $n=10^4$, a=0, b=.05, t=15	$N=10^3$, $n=2.5x10^4$, a=0, b=.05, t=30
	$N=10^3$, $n=10^4$, a=.95, b=1.0, t=15	$N=10^3$, $n=2.5x10^4$, a=.45, b=.55, t=30
	$N=10^3$, $n=10^4$, a=1/3, b=2/3, t=10	$N=10^3$, $n=2.5x10^4$, a=.95, b=1.0, t=30
Poker test	$N=10^3$, $n=10^4$, k=4, d=4	
	$N=10^3$, $n=10^4$, k=6, d=8	
	$N=10^3$, $n=10^4$, k=8, d=16	
Coupon test	$N=10^3$, $n=10^4$, d=5, t=25	
Permutation test	$N=10^3$, $n=10^4$, t=3	
	$N=10^3$, $n=10^4$, t=5	
Runs-up test	$N=10^3$, $n=10^5$, t=6	$N=10^3$, $n=10^5$, t=6
Maximum of t	$N=10^3$, $n=10^4$, t=8	$N=10^3$, $n=2x10^3$, t=5
		$N=10^3$, $n=2x10^3$, t=3
Collision test	$N=10^2$, $n=2x10^3$, logmd=6, logd=3	$N=10^3$, $n=2^{14}$, logmd=2, logd=10
	$N=10^2$, $n=2x10^4$, logmd=10, logd=2	$N=10^3$, $n=2^{14}$, logmd=4, logd=5
	$N=10^2$, $n=2x10^4$, logmd=20, logd=1	$N=10^3$, $n=2^{14}$, logmd=10, logd=2

Table 2. Knuth's statistical tests for random number generators [3, 6, 7]

DIEHARD Statistical Tests
Birthday spacings test
Overlapping 5-permutation test
Binary rank test for 31x31 matrices
Binary rank test for 32x32 matrices
Binary rank test for 6x8 matrices
Bitstream test
Overlapping pairs sparse occupancy
Overlapping quadruples sparse occupancy
DNA test
Count the 1's in a stream of bytes
Count the 1's for specific bytes
Parking lot test
Minimum distance test
3D spheres test
Squeeze test
Overlapping sums test
Runs test
Craps test

Table 3. DIEHARD Test Suite for Random Number Generators [9]