Title:	Reassessing the MCNP Random Number Generator
Author(s):	Josey, Colin James
Intended for:	MCNP User Symposium, 2023-09-18/2023-09-21 (Los Alamos, New Mexico, United States)
Issued:	2023-09-12

Reassessing the MCNP Random Number Generator

Colin Josey, XCP-3, Monte Carlo Group

MCNP Users Symposium
September 19, 2023

LA-UR-23-3NNNN

Introduction

The random number generators used within the MCNP code are key to the quality of our results. The current generators are well-characterized and fast.

However, they:

- fail some random number tests,
- do not have all that many states, and
- require user configuration in many circumstances to ensure quality results.

This work was an investigation on if new generators could outperform our current ones in the above aspects.

Current Generators

Linear Congruential Generators

All current MCNP generators take the form of a linear congruential generator (LCG):

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

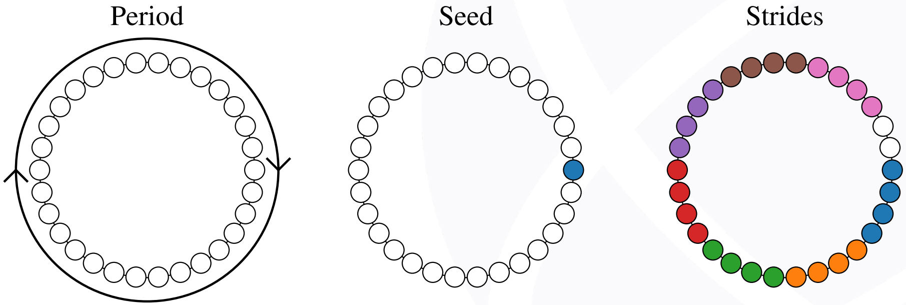
- Maximum period of m, if a and c chosen with care
- One sequence for a given a, c, m
- $\mathcal{O}(\log n)$ skip-ahead in sequence
- Quick implementation for $m = 2^p$
- Least significant bits have short period

LCG Usage in MCNP - 1

MCNP has 7 generators, with periods 2^{46} (gen 1), 2^{61} (gen 5–7) and 2^{63} (gen 2–4).

History initialization is performed by:

1. Setting the value X_0 to the seed
2. Skipping ahead (stride $\times$ i_h) where i_h is the history index



LCG Usage in MCNP - 2

Benefits:

- One seed changes all values
- Each history has $n =$ stride random numbers
- No history depends on another for parallelism

Drawbacks:

- Stride must be selected to be sufficient for a problem
- Period overflow at $\left\lceil \frac{\text{period}}{\text{stride}} \right\rceil$ histories
- Overrunning period reduces the "effective" stride
  For generator 1, after 631 billion histories, each new history is within 1 of another.
- Changing seed just changes position in same sequence

Random Number Reuse

Multiple papers indicating that short strides (< 100) affect results:

- J. S. Hendricks, "Random Number Stride in Monte Carlo Calculations," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-90-1845, Nov. 1990.
- T. E. Booth, "Bad Estimates as a Function of Exceeding the MCNP Random Number Stride," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-14-23159, May 2014.
- A. R. Hakim and D. A. Fynan, "Challenges of Near Critical-Fixed Source Monte Carlo Simulations of CANDU-6 Reactor: Bundle Power Tally Bias and Error Autocorrelation from Exceeding Random–Number Stride," in M&C 2023, Niagara Falls, ON, CA; Aug 13–17, 2023.

Does overrunning a large (152917) stride affect results?
Unknown, low probability. However, it would be nice to **eliminate the question altogether** so users do not need to be aware of the generator.

Eliminating reuse requires generator state space to be larger than the product of the maximum numbers used per history, the maximum histories, and the maximum number of independent simulations.

Looking at Other Generators

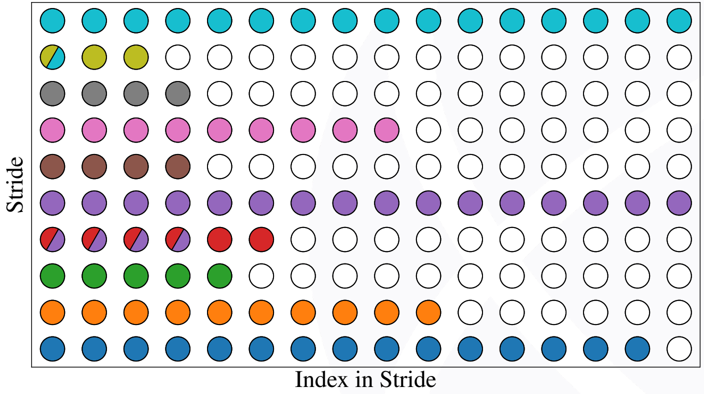
What We Need

Target is a generator that:

- can handle parallel streams easily
- can generate longer parallel streams (stride $\gg$ 152917)
- can generate more parallel streams (histories $\gg 2^{45}$)
- generates better quality bits
- performs similarly to, or faster than an LCG

Parallelism - Skip Ahead

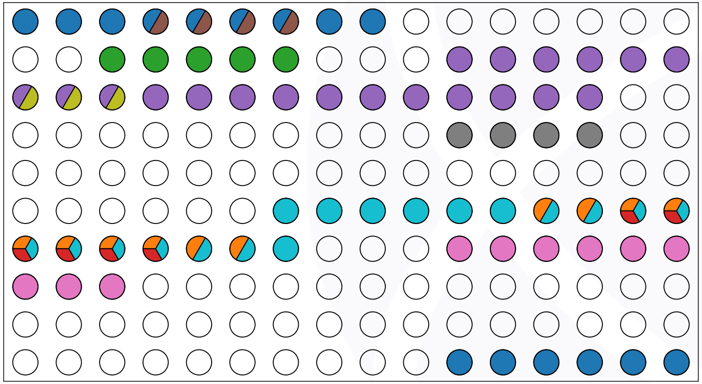
Start histories uniformly in the sequence:



Parallelism - Brute Force

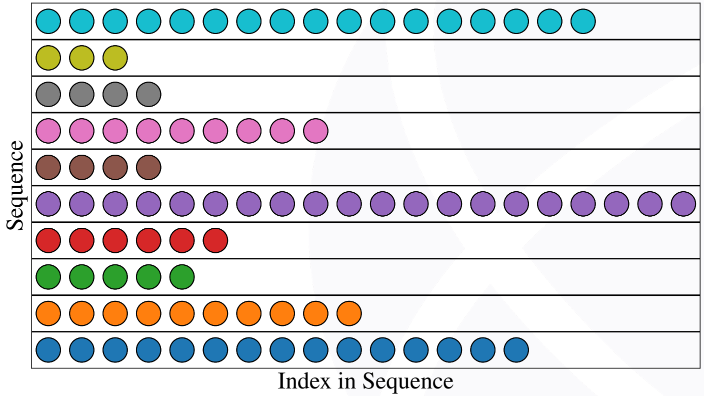
Random start state. Prob. of collision (*p* period, *n* histories, *s* values per history):

$$P_{\text{reuse}} = 1 - \frac{(p - ns - 1)!}{p^{n-1}\,(p - n(s+1))!}$$



Parallelism - Counter

Generator state contains a counter, guaranteeing *n* independent random numbers per initial state:



For some algorithms, sequences never intersect.

Choosing a Minimum State Space

Current simulations (2^{76}):

• stride $= 2^{26}$	Largest stride test problem
• histories $= 2^{40}$	Largest observed simulation
• simulations $= 2^{10}$	Enough to compute true variance to high precision

Future simulations (2^{128}):

• stride $= 2^{48}$	6 orders of magnitude increase
• histories $= 2^{64}$	Maximum allowed in the code without extensive modifications
• simulations $= 2^{16}$	Enough to compute shapes of distributions

Largest theoretical ($2^{225.6}$):

• stride $= 2^{64}$	Beyond any theoretical hardware capabilities
• histories $= 2^{97.6}$	Direct simulation of all neutrons in lifetime of reactor
• simulations $= 2^{64}$	Maximum allowed in the code without extensive modifications

Bit Quality

Cannot prove a generator good, just prove it bad (null hypothesis is generator is random). So we use 2 test suites:

- TestU01 version 1.2.3, BigCrush
  - 106 tests
  - Performed on bits both forward and backward
  - For 64-bit generators, performed on high and low bits
  - For $<$ 64-bit generators, used only high bits
- PractRand version 0.95-pre
  - Extremely powerful series of statistical tests
  - Tests powers of two bytes until correlation tests fail
  - A “pass” is typically 32 TiB (1-4 days running)

Some[1] Generator Options

[1] More were tested and not listed, such as Xoshiro256** and Chacha

Current 63-bit LCG

Period	2^{63}
Streams	As used, 1.
Bits Output	63
Parallelism	$\mathcal{O}(\log n)$ skip-ahead
Storage	8 bytes
TestU01	Failures on 10 tests
PractRand	Failed at 32 MiB
Time per Val	2.5 ns
Time Init.	115 ns

- Period insufficient
- Poor bit quality

Mersenne Twister (64-bit)

Period	$2^{19937} - 1$
Streams	1
Bits Output	64
Parallelism	For stride of 2^{64}, $P_{\text{reuse}} > 10^{-10}$ at $2^{9919.9}$ histories
Storage	2500 bytes
TestU01	Failures on 2 tests
PractRand	Failed at 512 GiB
Time per Val	3.4 ns
Time Init.	643 ns

- Period significantly exceeds needs
- Requires vectorization for performance
- Weak bit quality

PCG DXSM 128/64

Period	2^{128}
Streams	As used, 1.
Bits Output	64
Parallelism	$\mathcal{O}(\log n)$ skip-ahead
Storage	16 bytes
TestU01	Passed
PractRand	Passed to at least 32 TiB
Time per Val	4.7 ns
Time Init.	165 ns

- 2^{128} is borderline
- Successor to NumPy's default generator
- Requires 128-bit math implementation

SPECK-128/128

Period	2^{128}
Streams	2^{128}
Bits Output	128
Parallelism	Incrementing counter random access
Storage	Up to 656 bytes
TestU01	Passed
PractRand	Passed to at least 32 TiB
Time per Val	5.8 ns (12-round)
Time Init.	17.0 ns (12-round)

- High-perf cryptographic counter-based generator
- Each seed yields a new sequence
- Requires vectorization and round reduction for performance

SFC64

Period	2^{64} min., 2^{255} expected
Streams	2^{192}
Bits Output	64
Parallelism	Incrementing counter no random access
Storage	32 bytes
TestU01	Passed
PractRand	Passed to at least 32 TiB
Time per Val	2.6 ns
Time Init.	23 ns

- Part of state is a counter
- Each seed yields a new sequence
- Very high performance
- Selected for further use

SFC64 in the Code

SFC64

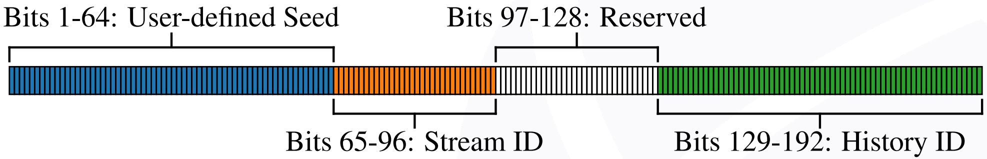
$$
\begin{aligned}
t_n &= a_n + b_n + n \\
a_{n+1} &= b_n \oplus \texttt{shift\_right}(b_n, 11) \\
b_{n+1} &= c_n + \texttt{shift\_left}(c_n, 3) \\
c_{n+1} &= \texttt{barrel\_shift}(c_n, 24) + t_n
\end{aligned}
$$

t_n output for generator.

- Added as Generator 8 for 6.3.1
- Plan to make default in 6.4

SFC64 Seeding

Seed is 192-bits:



- Seed set, generator iterated 18 times to mix bits
- Every history is a unique sequence
- Every user-seed changes all history sequences to new ones
- Multiple generator streams can be used from the same user-seed

Summary

- New random generator 8, SFC64, for 6.3.1
- As fast or faster than 63-bit LCG to generate numbers
- Faster to initialize state
- Significantly higher output bit quality
- Significantly more states (2^{192} streams of 2^{64})
- Eliminates the concept of strides and random number reuse
- No bad seed values