LA-UR-11-04859

Approved for public release;
distribution is unlimited.

Title:	Testing MCNP Random Number Generators
Author(s):	Yasunobu Nagaya & Forrest B. Brown
Intended for:	MCNP References, historical document from 2002 unpublished memos

Testing MCNP Random Number Generators

Forrest B. Brown
Diagnostics Applications Group (X-5)
Los Alamos National Laboratory
<fbrown@lanl.gov>

Yasunobu Nagaya
Visiting Scientist - LANL
Japan Atomic Energy Research Institute
<nagaya@lanl.gov>

Testing MCNP Random Number Generators

- Introduction
  - History
  - Requirements
- MCNP-5 RN Generator
  - Algorithm
  - Coding
  - Skip-ahead
  - Parallel considerations
- RN Generator Parameters
  - Traditional generators – MCNP, RACER, RCP, MORSE, KENO, VIM, EGS
  - Extended generators – 63-bits
- RN Generator Testing
  - Knuth statistical tests
  - Marsaglia's DIEHARD test suite
  - Spectral test
  - Results
- Future Plans

Introduction

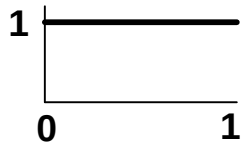
Introduction

- **Monte Carlo Simulation:**
  - Random sampling to model the outcome of physical events
  - Ray tracing through 3-D computational geometry

- **Random Number Generators**
  - **Numbers** are not random; a **sequence** of numbers can be
  - Repeatable (deterministic)
  - Pass statistical tests for randomness

  - Function which generates a sequence of numbers which appear to have been randomly sampled from a uniform distribution on (0,1)

  - Probability density function



  - Typical use in codes: r = rang( )

Remarks: History

- **MCNP & related precursor codes**
  - 40+ years of intense use
  - Many different computers & compilers
  - Modern versions are parallel: MPI + threads
  - History based: Consecutive RNs used for primary particle, then for each of it's secondaries in turn, etc.
  - RN generator is small fraction of total computing time (~ 5%)

- **Traditional MCNP RN Algorithm**
  - Linear congruential, multiplicative

$$S_{n+1} = g\, S_n \mod 2^{48}, \qquad g = 5^{19}$$

  - 48-bit integer arithmetic, carried out in 24-bit pieces
  - Stride for new histories: 152,917
  - Skip-ahead: crude, brute-force
  - Period / stride = 460×10^6 histories
  - Similar RN generators in RACER, RCP, MORSE, KENO, VIM

Remarks: Requirements

- **Algorithm**
  - Robust, well-proven
  - Long period: 10^9 particles x stride 152,917 = 10^{14} RNs
  - >10^9 parallel streams
  - High-precision is **not** needed, low-order bits not important
  - Reasonable theoretical basis, no correlation within or between histories

- **Coding**
  - Robust !!!! Must never fail.
  - Rapid initialization for each history
  - Minimal amount of state information
  - Fast, but portable – must be exactly reproducible on any computer/compiler

MCNP-5
RN Generator

MCNP-5 RN Generator: Algorithm

- **Linear congruential generator (LCG)**

$$S_{n+1} = g\,S_n + c \mod 2^m,$$

Period = 2^m (for c>0) or 2^{m-2} (for c=0)

Traditional MCNP:	m=48, c=0	Period=10^{14}, 48-bit integers
MCNP-5:	m=63, c=1	Period=10^{19}, 63-bit integers

How to pick g and c ???

- **RN Sequence & Particle Histories**

1 2 3 etc.

– Stride for new history: 152,917

MCNP-5 RN Generator: Skip-ahead

- To skip ahead k steps in the RN sequence:

$$S_k = g\,S_{k-1} + c \mod 2^m$$

$$= g^k S_0 + c\,(g^k-1)/(g-1) \mod 2^m$$

- Negative skip k equivalent to positive skip [period-k]

- Can skip from any seed to any other
  - Initial seed → i^{th} seed for j^{th} particle on m^{th} processor in k^{th} generation
  - Particle i → particle j
  - Batch i → batch j

- Need a fast way to compute $g^k \mod 2^m$ & $c(g^k-1)/(g-1) \mod 2^m$ in O(m) steps, rather than O(k) steps

MCNP-5 RN Generator: Skip-ahead

- Computing $G = g^k \bmod 2^m$

$G \leftarrow 1, \quad h \leftarrow g, \quad i \leftarrow k+2^m \bmod 2^m$

While $i > 0$

if i = odd: $G \leftarrow G\,h \bmod 2^m$

$h \leftarrow h^2 \bmod 2^m$

$i \leftarrow \lfloor i / 2 \rfloor$

Used in: RACER, VIM, KENO-Va (Spain), MCNP-5

- Computing $C = c(g^k-1)/(g-1) \bmod 2^m$

$C \leftarrow 0, \quad f \leftarrow c, \quad h \leftarrow g, \quad i \leftarrow k+2^m \bmod 2^m$

While $i > 0$

if i = odd: $C \leftarrow C\,h + f \bmod 2^m$

$f \leftarrow f\,(h+1) \bmod 2^m$

$h \leftarrow h^2 \bmod 2^m$

$i \leftarrow \lfloor i / 2 \rfloor$

Reference: F.B. Brown, "Random Number Generation with Arbitrary Strides", *Trans. Am. Nucl. Soc.* (Dec 1994)

MCNP-5 RN Generator: Coding

- **RN Generation in MCNP-5**
  - RN module, entirely replaces all previous coding for RN generation
  - Fortran-90, using INTEGER(I8) internally, where I8=selected_int_kind(18)
  - All parameters, variables, & RN generator state are PRIVATE, accessible only via “accessor” routines
  - Includes “new” skip-ahead algorithm for fast initialization of histories, greatly simplifies RN generation for parallel calculations
  - Portable, standard, thread-safe
  - Built-in unit test, compile check, and run-time test
  - Developed on PC, tested on SGI, IBM, Sun, Compaq

MCNP5 RN Generator: Coding

```
Module  mcnp_random
  . . . . .
  integer(I8), PRIVATE, SAVE ::       &
    &  RN_MULT,                       &        ! Multiplier
    &  RN_ADD,                        &        ! Adder
    &  RN_MASK,                       &        ! Mask, to get lower bits
  . . . . .
  real(R8),    PRIVATE, SAVE ::       &
    &  RN_NORM                                 ! norm. to (0,1)

  ! Private data for a single history
  !-----------------------------------
  integer(I8), PRIVATE ::             RN_SEED, RN_COUNT, RN_NPS
  common                 /RN_THREAD/  RN_SEED, RN_COUNT, RN_NPS
  !$OMP THREADPRIVATE ( /RN_THREAD/ )
  . . . . .
CONTAINS

  function rang( )
    ! MCNP5 random number generator
    implicit none
    real(R8) ::  rang

    RN_SEED  = iand( RN_MULT*RN_SEED, RN_MASK )
    RN_SEED  = iand( RN_SEED+RN_ADD,  RN_MASK )
    rang     = RN_SEED * RN_NORM
    RN_COUNT = RN_COUNT + 1
    return
  end function rang
  . . . . .
```

MCNP5 RN Generator: Coding

```
Program mcnp5
  . . . . .
  ! Initialize RN parameters for problem
  call RN_init_problem( new_standard_gen= 2,    &
    &                   new_seed= ProblemSeed   )

  . . . . .

  do nps = 1, number_of_histories

    ! Analyze one particle history
    call RN_init_particle( nps )

    . . . . .
    if( rang()>xs ) . . .
    . . . . .

    ! Terminate history
    call RN_update_stats
    . . . . .
```

MCNP–5 Random Number Generation & Testing

- **Introduction**
  - ✓ History
  - ✓ Requirements
- **MCNP-5 RN Generator**
  - ✓ Algorithm
  - ✓ Coding
  - ✓ Skip-ahead
  - ✓ Parallel considerations
- **RN Generator Parameters**
  - Extended generators – 63-bits
  - L'Ecuyer's 63-bit generators
- **RN Generator Testing**
  - Knuth statistical tests
  - Marsaglia's DIEHARD test suite
  - Spectral test
  - Performance test
  - Results
- **Future Plans**

RN Generator Parameters

Extended generators : 63-bit LCGs

- Selection of multiplier, increment and modulus

$$S_{n+1} = 5^{19} S_n + 0 \mod 2^{48} \text{ (MCNP4)}$$

$5^{23}, 5^{25}$ $\quad$ 1 $\quad$ 2^{63}

- Multiplicative LCG(g, 0, 2^{β})

$g \equiv \pm 3 \mod 8, S_0 = \text{odd} \Rightarrow$ Period : $2^{\beta-2}$

- Mixed LCG(g, c, 2^{β})

$g \equiv 1 \mod 4, c = \text{odd} \Rightarrow$ Period : 2^{β}

- Extension of multiplier
  - 5^{19} = 45-bit integer in the binary representation
  - 5^{19} seems to be slightly small in 63-bit environment.
  - Odd powers of 5 satisfy both conditions above.

L'Ecuyer's 63-bit LCGs

- L'Ecuyer suggested 63-bit LCGs with good lattice structures. Math. Comp., **68**, 249-260 (1999)
- Good multipliers are chosen based on the spectral test.
- Multiplicative LCGs
  - LCG(3512401965023503517, 0, 2^{63})
  - LCG(2444805353187672469, 0, 2^{63})
  - LCG(1987591058829310733, 0, 2^{63})
- Mixed LCGs
  - LCG(9219741426499971445, 1, 2^{63})
  - LCG(2806196910506780709, 1, 2^{63})
  - LCG(3249286849523012805, 1, 2^{63})
- Tested RNGs
  - Traditional MCNP RNG
  - 6 Extended 63-bit LCGs
  - L'Ecuyler's 63-bit LCGs above
  - 13 LCGs were tested.

RN Generator Testing

Tests for RNGs

- Theoretical tests :
  - Analyzing the algorithm of RNGs based on the number theory and the theory of statistics.
  - Theoretical tests depend on the type of RNGs. (LCG, Shift register, Lagged Fibonacci, etc.)
  - LCG : **Spectral test**

- Empirical tests :
  - Analyzing the uniformity, patterns, etc. of RNs generated by RNGs.
  - **Standard tests** (reviewed by D. Knuth) : SPRNG test routines
  - Bit level tests (**DIEHARD test** proposed by G. Marsaglia) : more stringent
  - Physical tests : RNGs are used in a practical application. The exact solutions for the tests are known. (not performed in this work)

Standard test suite in SPRNG

- SPRNG (Scalable Parallel Random Number Generators)
  - Test programs are available. http://sprng.cs.fsu.edu

- Standard test suite
  - Equidistribution, Serial, Gap, Poker, Coupon collector's, Permutation, Runs-up , Maximum-of-t, Collision tests

- Choice of test parameters
  - L'Ecuyer's test suite : Comm. ACM **31** p.742 (1988)
  - Vattulainen's test suite : Comp. Phys. Comm. **86** p.209 (1995)
  - Mascagni's test suite : Submitted to Parallel Computing

Equidistribution test

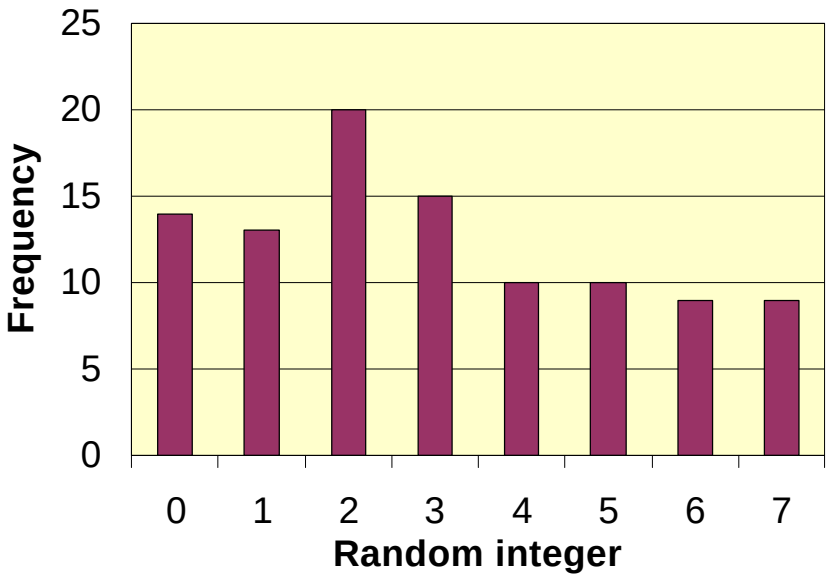
- Check whether RNs are uniformly generated in [0, 1).
- Generate random integers in [0,d-1].
- Each integer must have the equal probability 1/d.

0.10574, 0.66509, 0.46622, 0.93925, 0.26551, 0.11361, …

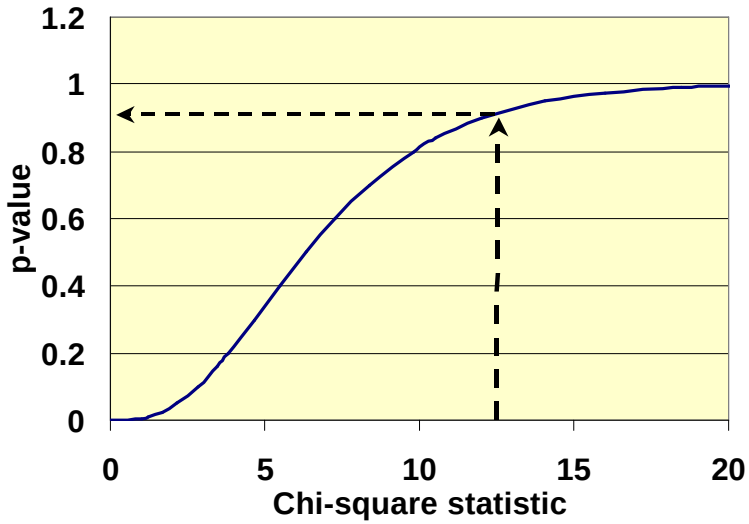
$\lfloor d * x_i \rfloor$

0, 5, 3, 7, 2, 0, 2, 3, 1, 4, …

Count frequencies of 0 ~ d-1.



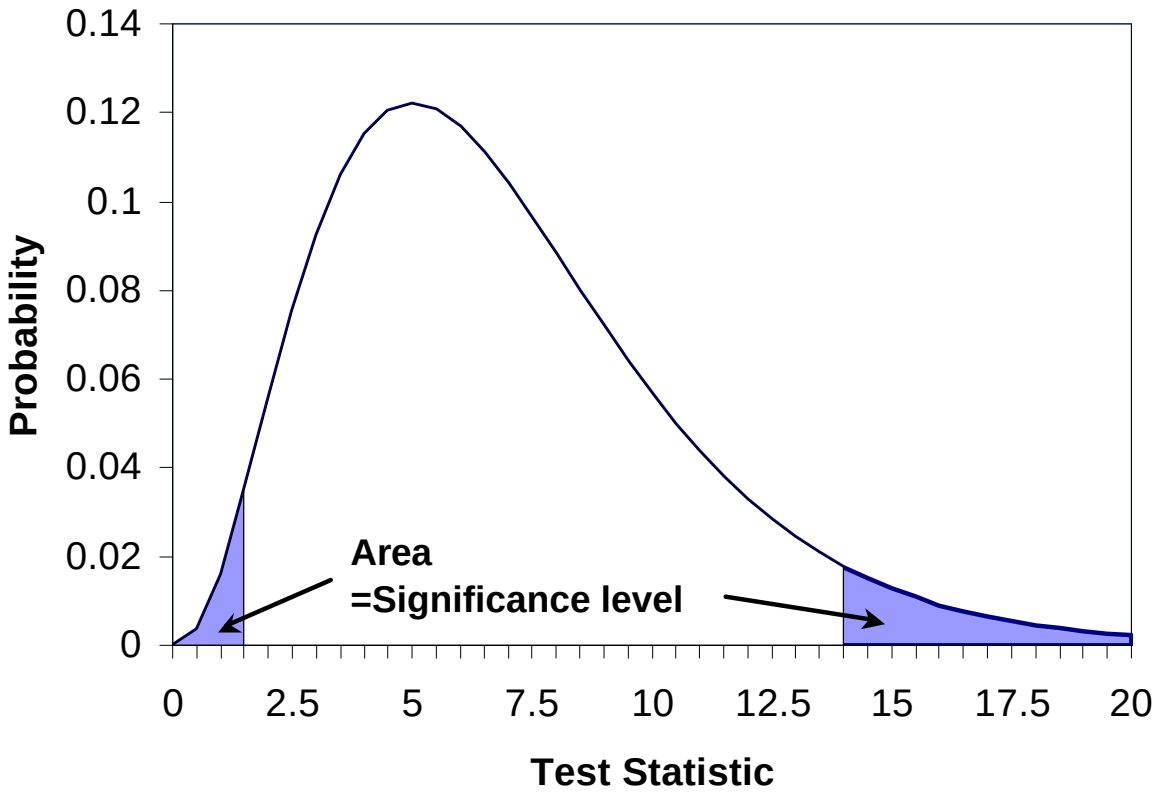
Cumulative chi-square distribution



$$V = \sum_{s=1}^{k} \frac{(Y_s - np_s)^2}{np_s}$$

Criterion of "Pass or Failure"

- All empirical tests score a statistic.
- A goodness-of-fit test is performed on the test statistic and yield a p-value. (Chi-sqaure or Kolmogorov-Smirnov test)
- If the p-value is close to 0 or 1, a RNG is suspected to fail.
- Significance level : 0.01(1%)
- Repeat each test 3 times.
- All 3 p-values are suspicious, then the RNG fails.



DIEHARD test suite

- DIEHARD test
  - A battery of tests proposed by G. Marsaglia.
  - Test all bits of random integers, not only the most significant bits.
  - More stringent than standard tests.
  - Test programs are available. http://stat.fsu.edu/~geo/diehard.html

- Included tests
  - Birthday spacings, Overlapping 5-permutation, Binary rank, Bitstream, Overlapping-pairs-sparse-occupancy (OPSO), Overlapping-quadruples-sparse-occupancy (OQSO), DNA, Count-the-1's test on a stream of bytes, Count-the-1's test for specific bytes, Parking lot, Minimum distance, 3-D spheres, Squeeze, Overlapping sums, Runs, Craps

- Test Parameters
  - Default test parameters were used in this work.

Overlapping-pairs-sparse-occupancy test (1)

- OPSO = Overlapping-Pairs-Sparse-Occupancy test
- Preparation of 32-bit integers

0.10574, 0.66509, 0.46622, 0.93925, 0.26551, 0.11361, …

$\lfloor 2^{32} * x_i \rfloor$

454158374, 2856527213, 2002411287, 4034027575, …

Binary representation

11011000100011110100000100110,

10101010010000110010010101101101, …

- Letter : a designated string of consecutive 10 bits

11011000100011110100000100110,

10101010010000110010010101101101, …

Letter : 2^{10} = 1024 patterns

(letters)

Overlapping-pairs-sparse-occupancy test (2)

- 2-letter words are formed from an alphabet of 1024 letters.

  0000100110, 0101101101, 1100010111, 0000110111, …

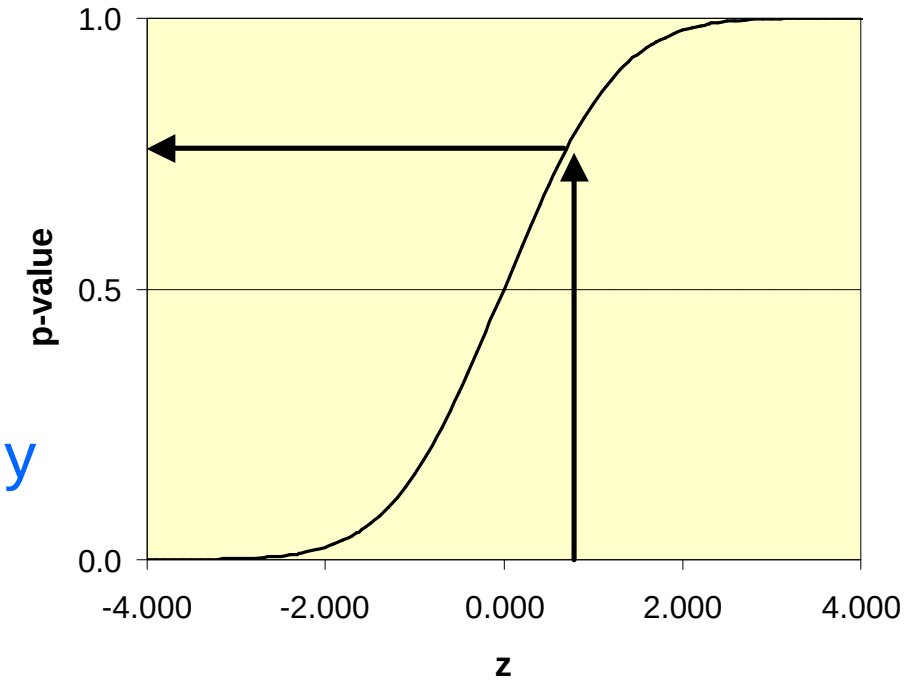
  Decimal representation

  38, 365, 791, 55, …

  2-letter word 2-letter word

- Count the number of missing words (=j).
- The number of missing words should be very closely normally distributed with mean 141,909, standard deviation 290.

Cumulative normal distribution



$$z = \frac{j - 141909}{290}$$

Overlapping-quadruples-sparse-occupancy test

- OQSO = Overlapping-Quadraples-Sparse-Occupancy test
- Similar to the OPSO test.
- Letter : a designated string of consecutive 5 bits

  11011000100011110100000100110,

  10101010010000110010010101101101, …

  Letter : 2^5 = 32 letters

- 4-letter words are formed from an alphabet of 32 letters.

  00110, 01101, 10111, 10111, …

  4-letter word

- The number of missing words should be very closely normally distributed with mean 141909, standard deviation 295.

DNA test

- Similar to the OPSO and OQSO tests.
- Letter : a designated string of consecutive 2 bits

  11011000100011110100000100110,

  10101010010000110010010101101101, …

  Letter : 2^2 = 4 letters

- 10-letter words are formed from an alphabet of 4 letters.

  10, 1,11,11,11, 1,10, 0,11,10, …

  10-letter word

- The number of missing words should be very closely normally distributed with mean 141909, standard deviation 399.

Criterion for DIEHARD test

- If the p-value is close to 0 or 1, a RNG is suspected to fail.
- Significance level : 0.01(1%)
- A RNG fails the test if we get six or more p-values less than 0.01 or more than 0.99.

Results for standard & DIEHARD tests

- All 13 RNGs pass all standard tests with L'Ecuyer's, Vattulainen's and Mascagni's test parameters.
- Extended and L'Ecuyer's 63-bit LCGs pass all the DIEHARD tests.
- The traditional MCNP RNG fails the OPSO, OQSO and DNA tests in the DIEHARD test suite.

Result of OPSO test for traditional MCNP RNG

Tested bits	p-value	Tested bits	p-value
bits 23 to 32	**0.0000**	bits 11 to 20	0.7457
bits 22 to 31	**0.0000**	bits 10 to 19	0.0598
bits 21 to 30	**0.0000**	bits 9 to 18	0.1122
bits 20 to 29	**0.0000**	bits 8 to 17	0.4597
bits 19 to 28	**0.0001**	bits 7 to 16	**0.0011**
bits 18 to 27	0.6639	bits 6 to 15	0.6319
bits 17 to 26	0.0445	bits 5 to 14	0.7490
bits 16 to 25	0.0125	bits 4 to 13	0.2914
bits 15 to 24	0.7683	bits 3 to 12	0.1792
bits 14 to 23	0.9712	bits 2 to 11	0.3253
bits 13 to 22	0.1077	bits 1 to 10	0.7277
bits 12 to 21	0.0717		

Result of OQSO test for traditional MCNP RNG

Tested bits	p-value	Tested bits	p-value
bits 28 to 32	1.0000	bits 14 to 18	0.6487
bits 27 to 31	1.0000	bits 13 to 17	0.5575
bits 26 to 30	1.0000	bits 12 to 16	0.1634
bits 25 to 29	1.0000	bits 11 to 15	0.6600
bits 24 to 28	1.0000	bits 10 to 14	0.2096
bits 23 to 27	1.0000	bits 9 to 13	0.3759
bits 22 to 26	0.0000	bits 8 to 12	0.9191
bits 21 to 25	0.0000	bits 7 to 11	0.8554
bits 20 to 24	0.0000	bits 6 to 10	0.5535
bits 19 to 23	0.1906	bits 5 to 9	0.4955
bits 18 to 22	0.0011	bits 4 to 8	0.0868
bits 17 to 21	0.3823	bits 3 to 7	0.1943
bits 16 to 20	0.8394	bits 2 to 6	0.8554
bits 15 to 19	0.2518	bits 1 to 5	0.7421

Result of DNA test for traditional MCNP RNG

Tested bits	p-value	Tested bits	p-value	Tested bits	p-value
bits 31 to 32	1.0000	bits 20 to 21	0.4937	bits 9 to 10	0.4550
bits 30 to 31	1.0000	bits 19 to 20	0.0613	bits 8 to 9	0.4737
bits 29 to 30	1.0000	bits 18 to 19	0.2383	bits 7 to 8	0.7834
bits 28 to 29	1.0000	bits 17 to 18	0.4831	bits 6 to 7	0.4063
bits 27 to 28	1.0000	bits 16 to 17	0.0925	bits 5 to 6	0.8959
bits 26 to 27	0.1777	bits 15 to 16	0.0197	bits 4 to 5	0.3438
bits 25 to 26	0.0000	bits 14 to 15	0.7377	bits 3 to 4	0.3972
bits 24 to 25	0.0000	bits 13 to 14	0.7171	bits 2 to 3	0.8986
bits 23 to 24	0.0000	bits 12 to 13	0.0309	bits 1 to 2	0.5407
bits 22 to 23	0.0000	bits 11 to 12	0.2803		
bits 21 to 22	0.0000	bits 10 to 11	0.8440		

Comments on results for OPSO, OQSO, DNA

- Less significant (lower) bits of RNs fail the tests.

- These failures in less significant bits are caused by the shorter period than the significant bits.

Drawback of LCGs with power-of-two moduli

> The (r+1)-th most significant bit has period length at most 2^{-r} times that of the most significant bit.

- However, these failures do not have a significant impact in the practical use.

Spectral test

- LCGs have regular patterns (lattice structures) when overlapping t-tuples of a random number sequence are plotted in a hypercube. (Marsaglia, 1968).

- all the t-tuples are covered with families of parallel (t-1)-dimensional hyperplanes.

- The spectral test determines the maximum distance between adjacent parallel hyperplanes.

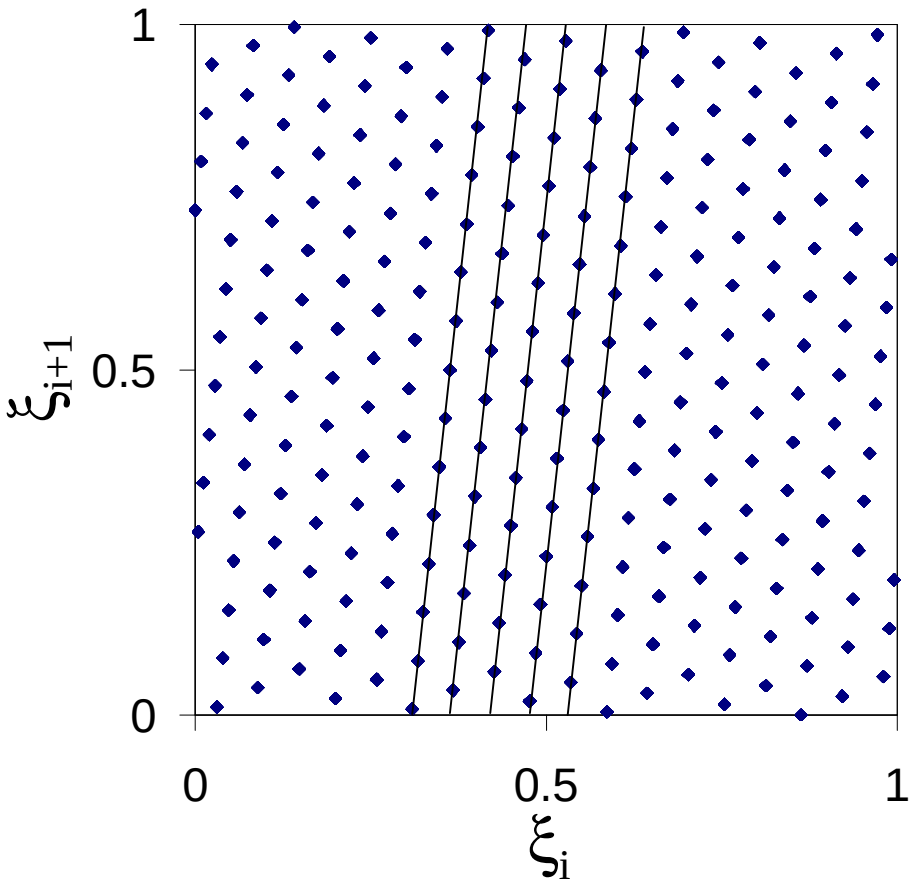
Illustration of the spectral test

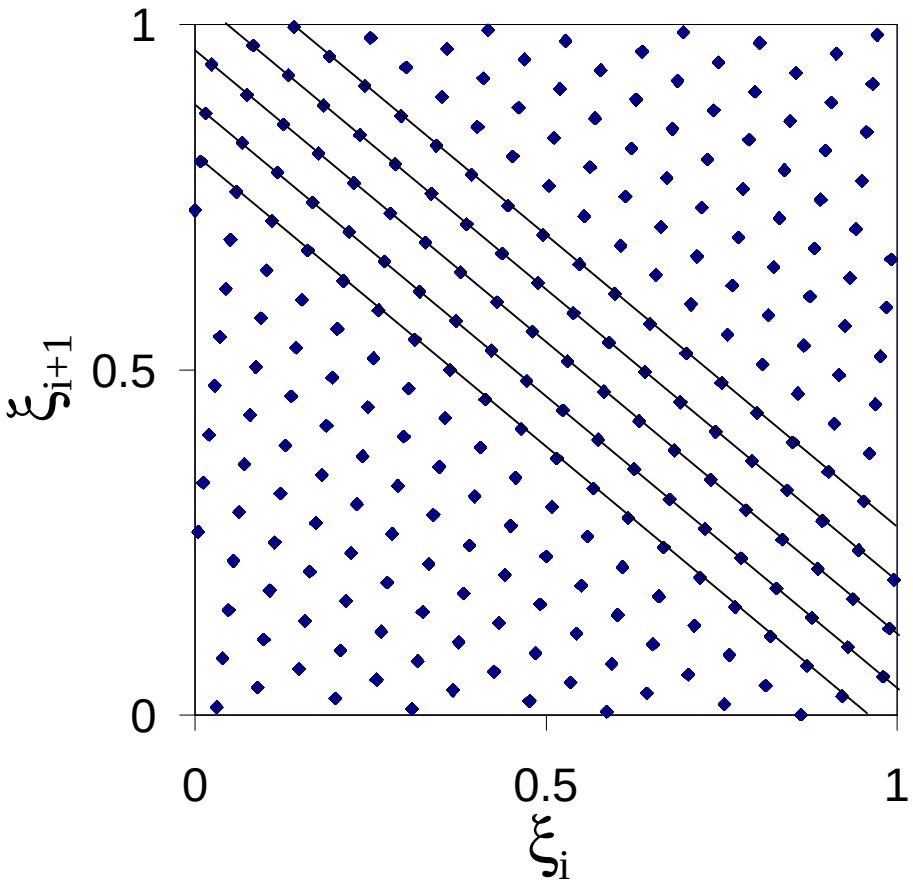
- Example: $S_{n+1} = 137\ S_n + 187 \bmod 256$

pair pair

0.26562, 0.12109, 0.32031, 0.61328, 0.75000, ...

pair pair





Measures for criterion & ranking

- μ value proposed by Knuth
  - Represent the effectiveness of a multiplier.

Knuth's criterion

$m_t(m,g)$ for $2 \le t \le 6$	Result
$m_t(m,g) \ge 1$	Pass with flying colors
$0.1 \le m_t(m,g) < 1$	Pass
$m_t(m,g) \le 0.1$	Fail

- S value
  - Normalized maximum distance.

$$S_t = \frac{d_t^*(m)}{d_t(m,g)}$$

$d_t(m,g)$: Maximum distance between adjacent parallel hyperplanes.

$d_t^*(m)$: Lower bound on $d_t(m,g)$.

  - The closer to 1 the S value is, the better the RNG is.

Results of spectral test

- Results for the traditional MCNP RNG

Dimension(t)	2	3	4	5	6	7	8
$\mu_t(m,g)$	3.0233	0.1970	1.8870	0.9483	1.8597	0.8802	1.2931
$S_t(m,g)$	0.9129	0.3216	0.6613	0.5765	0.6535	0.5844	0.6129

- All extended 63-bit LCGs fail with Knuth's criterion.
- All L'Ecuyer's 63-bit LCGs pass with flying colors.
- Comparison of minimum S values

RNG	Minimum $S_t(m,g)$
LCG(5^{19},0,2^{48})	0.3216
LCG(351240196502350 3517,0,2^{63})	0.7493
LCG(244805353187672469,0,2^{63})	0.7094
LCG(198759105882931 0733,0,2^{63})	0.6449
LCG(921974142649997 1445,1,2^{63})	0.7371
LCG(280619691050678 0709,1,2^{63})	0.6967
LCG(324928684952301 2805,1,2^{63})	0.6451

Performance test

- Test program

```
 . . . . .
 integer(8) :: i
 integer(8), parameter :: NumGeneratedRNs = 1000000000
!real(8)    :: rang ! For MCNP4
 real(8)    :: RN_initial, RN_last
 real(8)    :: dummy
 . . . . .

!call random ! For MCNP4
 call RN_init_problem( new_standard_gen = 1 )

 RN_initial = rang()

 do i = 2, NumGeneratedRNs-1
    dummy = rang()
 end do

 RN_last = rang()
 . . . . .
```

Results of performance test

- Comparison between MCNP-4 and –5
- Generate 1 billion RNs.

	MCNP4	MCNP5	MCNP4/MCNP5
CPU (sec) No optimization (/optimization:0)	290.0	97.1	3.0
CPU (sec) Local optimization (/optimization:1)	191.7	77.2	2.5
CPU (sec) Full optimization (/optimization:4)	188.4	78.1	2.4

Platform : Windows 2000, Intel Pentium III 1GHz

Compiler : Compaq Visual Fortran Ver.6.6

Summary

- The traditional MCNP RNG fails the OPSO, OQSO and DNA tests in the DIEHARD test suite.
- The 63-bit LCGs extended from the MCNP RNG fail the spectral test.
- L'Ecuyer's 63-bit LCGs pass all the tests and their multipliers are excellent judging from the spectral test.
- These 63-bit LCGs are implemented in the RNG package for MCNP Ver.5.
- The MCNP-5 RNG is ~2.5 times faster than the MCNP-4 RNG.

Future Work

Plans for MCNP RN Generation

- For now, stick with existing RN algorithm – LCG
  - Today's longest problems use ~10^9 histories, for a total of ~10^{14} RN's
  - The period of the RN generator in MCNP5 has been extended by a factor of 10^5 from $2^{46} = 7 \times 10^{13}$ to $2^{63} = 9.2 \times 10^{18}$
- Eventually, will need an even longer period.
  - ASCI: 30 T computer this year, 100 T in a few years, & then ….
  - More histories + RN streams by particle type → need longer period
- Desirable to modify MCNP5 so that separate particle types (neutrons, photons, electrons, …) have separate RN streams
  - Want particle behavior to be identical & reproducible if physics options involving other particle types are turned on/off
  - For example, neutron behavior for collisions, tracking, tallies, etc., should be the same if a problem is run with
    - Neutrons only
    - Neutrons + photons
    - Neutrons + photons + electrons

Plans for MCNP RN Generation

- For independent particle streams, could use a different RN additive constant for each particle type:

Neutrons: $S_{N,n+1} = g\, S_{N,n} + c_N \mod 2^m$

Photons: $S_{P,n+1} = g\, S_{P,n} + c_P \mod 2^m$

Electrons: $S_{E,n+1} = g\, S_{E,n} + c_E \mod 2^m$

etc.

Percus & Kalos have proven that the streams would be independent.

- For a longer period:
  - Could extend RN generator to use more than 64-bits
    - Straightforward coding extensions to existing generator
    - Retain “tried & true” mixed LCG scheme
    - Need new multiplier, adder, modulus, & extensive testing
  - Could use different RN algorithm with longer period
    - Combined LCG’s seems a good bet
    - Retain existing coding & algorithm, combine 2 LCG’s
    - Needs a lot of thought, plus advice from experts

End

Appendices

Spectral test for extended Multiplicative LCGs

Dimension(t)	2	3	4	5	6	7	8
LCG(5^{19},0,2^{63})							
$\mu_t(m,g)$	1.7321	2.1068	2.7781	1.4379	0.0825	2.0043	5.9276
$S_t(m,g)$	0.6910	0.7085	0.7284	0.6266	0.3888	0.6573	0.7414
LCG(5^{23},0,2^{63})							
$\mu_t(m,g)$	0.0028	1.9145	2.4655	5.4858	0.3327	0.2895	6.6286
$S_t(m,g)$	0.0280	0.6863	0.7070	0.8190	0.4906	0.4986	0.7518
LCG(5^{25},0,2^{63})							
$\mu_t(m,g)$	0.3206	1.8083	0.0450	3.0128	0.3270	3.1053	0.4400
$S_t(m,g)$	0.2973	0.6733	0.2598	0.7265	0.4892	0.6998	0.5356

Spectral test for extended Mixed LCGs

Dimension(t)	2	3	4	5	6	7	8
LCG(5^{19},1,2^{63})							
$\mu_t(m,g)$	1.7321	2.9253	2.4193	0.3595	0.0206	0.5011	1.6439
$S_t(m,g)$	0.6910	0.7904	0.7036	0.4749	0.3086	0.5392	0.6316
LCG(5^{23},1,2^{63})							
$\mu_t(m,g)$	0.0007	2.8511	2.5256	3.1271	4.5931	1.8131	4.2919
$S_t(m,g)$	0.0140	0.7837	0.7112	0.7319	0.7598	0.6480	0.7121
LCG(5^{25},1,2^{63})							
$\mu_t(m,g)$	0.0801	3.4624	1.3077	1.0853	1.4452	0.7763	1.3524
$S_t(m,g)$	0.1486	0.8361	0.6033	0.5923	0.6266	0.5740	0.6163

Spectral test for L'Ecuyer's Multiplicative LCGs

Dimension(t)	2	3	4	5	6	7	8
LCG(3512401965023503517,0,2^{63})							
$\mu_t(m,g)$	2.9062	2.9016	3.1105	4.0325	5.3992	6.7498	7.2874
$S_t(m,g)$	0.8951	0.7883	0.7493	0.7701	0.7806	0.7818	0.7608
LCG(2444805353187672469,0,2^{63})							
$\mu_t(m,g)$	2.2588	2.4430	6.4021	2.9364	3.0414	5.4274	4.6180
$S_t(m,g)$	0.7891	0.7443	0.8974	0.7228	0.7094	0.7579	0.7186
LCG(1987591058829310733,0,2^{63})							
$\mu_t(m,g)$	2.4898	3.4724	1.7071	2.5687	2.1243	2.0222	4.1014
$S_t(m,g)$	0.8285	0.8369	0.6449	0.7037	0.6682	0.6582	0.7080

Spectral test for L'Ecuyer's Mixed LCGs

Dimension(t)	2	3	4	5	6	7	8
LCG(9219741426499971445,1,2^{63})							
$\mu_t(m,g)$	2.8509	2.8046	3.5726	3.8380	3.8295	6.4241	6.8114
$S_t(m,g)$	0.8865	0.7794	0.7757	0.7625	0.7371	0.7763	0.7544
LCG(2806196910506780709,1,2^{63})							
$\mu_t(m,g)$	1.9599	4.0204	4.4591	3.1152	3.0728	3.0111	3.7947
$S_t(m,g)$	0.7350	0.8788	0.8199	0.7314	0.7106	0.6967	0.7012
LCG(3249286849523012805,1,2^{63})							
$\mu_t(m,g)$	2.4594	2.4281	3.7081	2.8333	3.7633	3.0844	1.9471
$S_t(m,g)$	0.8234	0.7428	0.7829	0.7176	0.7350	0.6991	0.6451