LA-UR-02-0632

Approved for public release;
distribution is unlimited.

Title:	MCNP RANDOM NUMBER ISSUES
Author(s):	Forrest B. Brown
Submitted to:	SPRNG Workshop February 11-12, 2002 Sandia National Laboratory Albuquerque, NM

MCNP Random Number Issues

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

SPRNG Workshop
Sandia National Laboratory

February 11-12, 2002

MCNP Random Number Issues

- **Remarks**
  - History
  - Requirements
- **MCNP5 RN Generator**
  - Algorithm
  - Skip-ahead
  - Coding
- **Plans**
  - Longer period
  - Testing
  - Independent streams for particle types
  - Other RN generators

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% ?)
  - New Fortran-90 RN module for MCNP5 (release mid-2002)

- **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
  - Inter-history correlation due to RN generator has never been observed
  - Used by MCNP, RACER, 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

MCNP5 RN Generator: Algorithm

- Linear congruential, multiplicative

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

  Period = $2^{46} = 7 \times 10^{13}$

- RN Sequence & Particle Histories

1 2 3 etc.

  - Stride for new history: 152,917
  - Period / stride = 460×10^6 histories
  - Adequate for nearly all problems, but some now use $>10^9$ histories

MCNP5 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

MCNP5 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), MCNP5

- 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)

MCNP5 RN Generator: Coding

- MCNP5 is a rewrite of MCNP4C
  - Entire code is standard Fortran-90
  - Standard parallel coding: MPI (message-passing) + OMP (threads)
  - Fortran-90 dynamic memory allocation
  - Vastly improved modern coding style:
    spaces, blank lines, modules, replaced many thousands of GOTO's, …..
  - Some new features & new physics
  - To be released in mid-2002

MCNP5 RN Generator: Coding

- RN Generation in MCNP5
  - RN module, entirely replaces all previous coding for RN generation
  - Fortran-90, using INTEGER(I8) internally, where I8=selected_int_kind(15)
  - 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          = FIVE**19,   &       ! multiplier
    &  RN_MASK          = TWO**48-1,  &       ! mask, to get lower 48 bits
  . . . . .
  real(R8),    PRIVATE, parameter ::  &
    &  RN_NORM          = 1.0_R8/TWO**48      ! norm. to (0,1)

  ! Private data for a single particle
  !-------------------------------------
  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 )
    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_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
    . . . . .
```


Plans for MCNP RN Generation

- For now, stick with existing RN algorithm – linear congruential
- Very minor modifications needed to lengthen the period
- Thorough testing is needed, even though changes are trivial
- Desirable to modify MCNP5 so that separate particle types (neutrons, photons, electrons, …) have separate RN streams
- Eventually, will have to change to a new RN algorithm to get even longer period.

Plans: Longer Period

- Use of 48-bit algorithm limits the period
  - Period = 2^{46} = 7×10^{13}
  - With stride=152,917, only 460×10^{6} histories before "wrap-around" in RN's
  - Wrap-around is probably not a concern, since (period/stride)≠integer
  - Adequate for nearly all problems, but some now use >10^{9} histories
- To increase period: Use 63-bit algorithm
  - Period = $2^{61} = 2.3 \times 10^{18}$
  - Requires changing only 3 parameters in existing coding: multiplier, mask, norm
  - Why 63-bits, instead of 64-bits?
    - Portability – standard Fortran-90, avoiding machine/compiler quirks with integer sign
  - Choice of multiplier ? 5^{25} ? SPRNG multiplier? ??
- To increase period: Use mixed-congruential algorithm
  - $S_{n+1} = g\,S_n + c \mod 2^{63}$, Period = $2^{63} = 9.2 \times 10^{18}$
  - Requires changing only a few lines of coding (add constant, skip-ahead)
  - Choice of adder ? c=1, or anything else should be fine

Plans: Testing

- Only trivial coding changes are needed to increase the period of the MCNP5 RN generator by a factor of 10^{5} – 1 hour's work
- Major hurdle is testing – 1 month's work or more
  - Theory, given new multiplier, adder, modulus
  - Empirical testing to ensure no correlation either within or between histories
  - Need to run & document standard suite of RN tests
  - Looking for a GRA or summer student to do this

Plans: Independent Streams for Particle Types

- Desirable to modify RN usage in MCNP so that RN usage for each particle type is independent of other particle types
  - 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
- Could be accomplished with minor changes to current scheme through partitioning RN stride by particle type:

	nnnnnpppppeeeee	nnnnnpppppeeeee	nnnnnpppppeeeee	
History:	1	2	3	etc.

where n = RN's used for neutrons
p = RN's used for photons
e = RN's used for electrons

- What if more particle types are needed? (They are!)

Plans: Other RN Generators

- Motivation for Continued Work on MCNP RN Generator
  - Today, the longest problems use 10^9 histories with 10^5 RN's each, for a total of 10^{14} RN's
  - The period of the existing RN generator in MCNP5 can be trivially extended by a factor of 10^5 from $2^{46} = 7 \times 10^{13}$ to $2^{63} = 9.2 \times 10^{18}$
  - ASCI is getting a 30 TeraOp computer this year, 100 TeraOp computer within a few years, & then ….
  - More histories + separate RN streams for particle types → need longer period
- For an even 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