

Parallel Monte Carlo Particle Transport and the Quality of Random Number Generators: How Good is Good Enough?

Richard Procassini and Bret Beck
Lawrence Livermore National Laboratory



Monte Carlo 2005 Meeting
18 April 2005

University of California



Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94551



Introduction and Motivation

- Recent advances in the speed of parallel computers and the development of parallel algorithms, have prompted discussion of the “quality” of the (pseudo-)random numbers required in order to produce accurate, unbiased results in Monte Carlo transport simulations.
- Several theoretical and empirical tests have been designed to test the *quality* of random number generator (RNG) algorithms [1].
- Such tests are not sufficient to determine the level of RNG quality that is required to produce unbiased *application* results.
- Each application must diagnose the results of simulations in order to determine if the quality of the RNG being employed is sufficient.
- One might assume that use of an RNG producing a stream of random numbers with a “long” repetition period is crucial in order to produce unbiased results.



Introduction and Motivation

- For some Monte Carlo applications it is extremely important to avoid the reuse of random numbers:
 - These applications employ RNGs to randomly sample a small number of distributions (usually one)
 - It is important to avoid correlations in the sampled quantity
 - *Example:* Numerical integration
- In contrast, Monte Carlo transport applications use RNGs to randomly sample from several distributions, which may or may not be correlated:
 - Spatial, energy or angular distributions of sources
 - Distance to collision
 - Isotope with which the particle interacts and the resulting reaction
 - Energy and angle of secondary particles

Introduction and Motivation



- While it is also desirable to avoid reuse of random numbers in Monte Carlo transport simulations, it is not clear, *a priori*, whether such reuse will produce biased results.
- For example, a given random number may be used to sample the energy of one particle which is created in an external source, and the next use of that random number may occur many time steps and particles later during a collision sampling event.
- The non-linear nature of particle transport, along with the multiple types of distributions sampled in Monte Carlo transport, *may* be beneficial since the RNG period required for unbiased simulations *may* be longer than that required for single sample applications.
- This paper explores the issue of RNG quality in the context of parallel, Monte Carlo transport simulations in order to determine how “good” is “good enough”.
- This study employs the MERCURY Monte Carlo particle transport code [2], which incorporates the CNPRNG library [3] for the generation of (pseudo-)random numbers.

Overview of the MERCURY Monte Carlo Transport Code

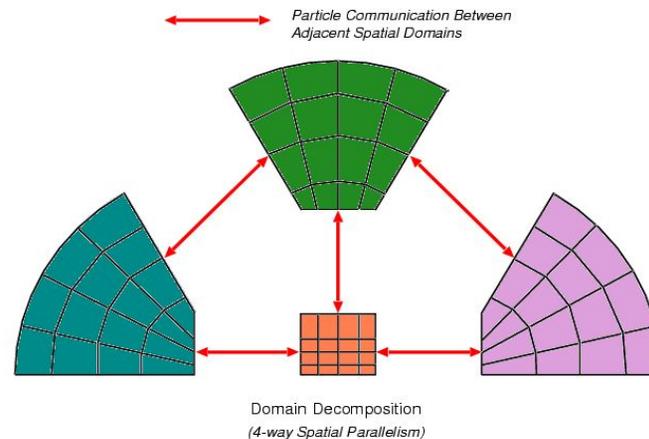


- MERCURY [2] is:
 - ◆ A modern, Monte Carlo transport code being developed at the Lawrence Livermore National Laboratory (LLNL)
 - ◆ Intended to replace the legacy codes TART and COG as the next-generation radiation transport tool at LLNL
- MERCURY's features include:
 - ◆ Time-dependent tracking of neutrons (n), gammas (γ) and light charged ions (1H , 2H , 3H , 3He , 4He)
 - ◆ Support of various types of geometry, including 1-D, 2-D and 3-D meshes, as well as 3-D combinatorial geometries
 - ◆ Modular design and implementation allows for easy replacement of components, such as collision, nuclear data and RNG libraries
 - ◆ A flexible parallel-run model allows the code to operate on a wide variety of parallel computing platforms

The MERCURY Parallel Programming Model



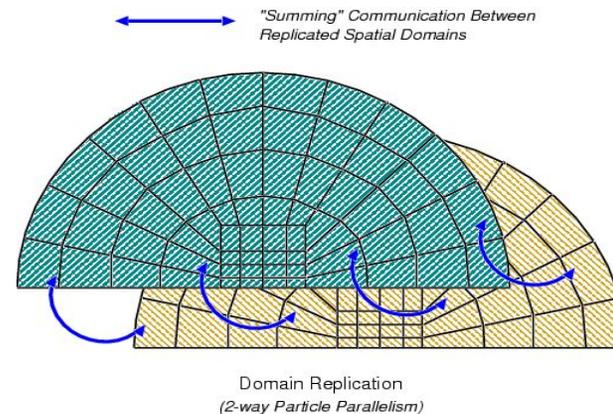
- A three-pronged approach to parallelism has been adopted within MERCURY:
 - *Domain Decomposition*: The problem geometry is spatially partitioned across multiple processors.
 - This form of *spatial* parallelism allows the code to transport particles through geometries containing a large number of cells
 - As particles track to a facet that lies on a domain boundary, it must be sent to the adjacent domain before it can continue its trajectory
 - Particle transfers between adjacent domains is accomplished via point-to-point communication of particle buffers



The MERCURY Parallel Programming Model



- ◆ *Domain Replication*: Particles are distributed across multiple copies of the problem geometry, which are assigned to specific processors.
 - This form of *particle* parallelism allows the code to efficiently transport large numbers of particles
 - Once the calculation is complete on each copy of the domain, partial results must be summed in order to produce the complete result
 - This transfer of data is accomplished via collective communication between the multiple copies of the domain



- ◆ *Task Decomposition*: The main particle loop is decomposed by assigning individual particles, or *tasks*, to threads on a shared-memory parallel computer.
 - This represents another form of particle parallelism

The Use of Random Numbers in MERCURY



- A key design goal is that MERCURY be capable of reproducing the same result, regardless of the chosen mode of operation (serial, parallel with any of the three modes, individually or in combination).
- In an attempt to guarantee reproducibility, a *hierarchy* of random number seeds has been implemented in MERCURY, making use of the spawning feature of the CNPRNG library:
 - *Simulation Seed*: The primordial seed from which all other seeds are spawned.
 - *Source Seed* and *Domain Seed(s)*: These are used to spawn the seeds for all particles generated in external sources, and in cell-based, physics sources (such as a fusion source) on a given spatial domain.
 - *Particle Seed*: Once spawned from the Source Seed or a Domain Seed, the Particle Seed controls the behavior of the particle. The Particle Seed is an inherent attribute of the particle. The particle carries its random number seed along with it as it tracks through the problem geometry and is communicated between adjacent spatial domains.

The Use of Random Numbers in MERCURY



- This approach to parallelism and random numbers is different from others in which only the processors are assigned unique random number seeds.
- Compact storage of the random number seed is crucial for applications using per-particle seeds.
- The per-particle random number seed should be a small fraction of the overall storage cost of each particle:
 - The storage required in MERCURY for all particle attributes *except* the random number seed is 120 bytes.
 - The seed of a random number stream in the CNPRNG library is defined by a single 64-bit integer (8 bytes). This increases the cost of storing each particle by only 6.67%.
 - The compact nature of this seed is the reason that this RNG library was chosen over others, such as the SPRNG library [4].

Testing Random Number Quality in Parallel Calculations



- The aim of this study is to determine the length of the RNG period required to prevent correlations in the random number streams, and thus, biases in the results.
- Four problems (2 shielding and 2 criticality calculations), derived from real world applications of interest to the nuclear engineering and transport communities, are run with several RNGs of varying period.
- For each pairing of problem and RNG period, an ensemble of 25 calculations is run where the initial random number seed is changed for each calculation, in order to determine the mean and standard deviations of each result.
- The CNPRNG library [3] used in MERCURY is a collection of Linear Congruential Generators (LCGs) of the form: $x_{n+1} = (a x_n + b) \text{ mod } m$ where x_n and x_{n+1} are successive random numbers in the sequence, a and b are constants, and m is the modulus of the RNG. The LCGs used for this study have moduli that are a power of 2: $m = 2^p$ with powers of $p = 4, 8, 12, 16, 24, 32, 48$ or 64 . This results in LCGs with periods between $m = 16$ (short) and $m = 1.84 \times 10^{19}$ (long).

Testing Random Number Quality in Parallel Calculations

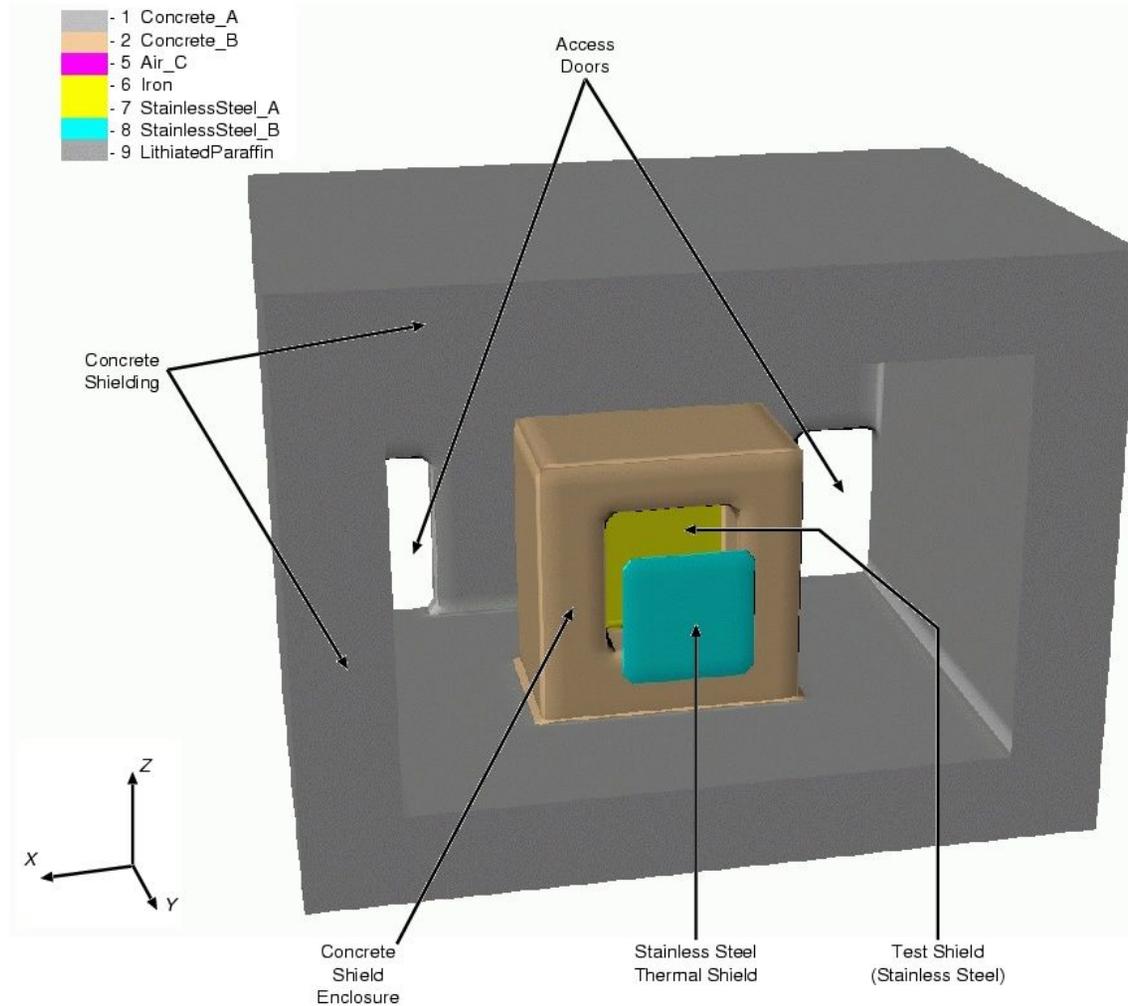


- Each of the calculations in this study are run with 10 million particle histories:
 - 10 million injected particles in each source problem
 - 25000 particles are run for 400 “equilibrium settle cycles” (generations or time steps) in each of the criticality problems
- This number of histories was chosen as a reasonable balance between accuracy and run time.
- The number of random numbers used by these calculations was (unfortunately) not edited, but one can assume that each particle history requires (at least) hundreds of random numbers.
- All calculations were run on 16 processors in the form of eight 2-way symmetric multi-processor (SMP) nodes on a parallel Linux cluster.
- These parallel calculations employed only Domain Replication which provides 16-way particle parallelism.

Testing Random Number Quality in Parallel Calculations

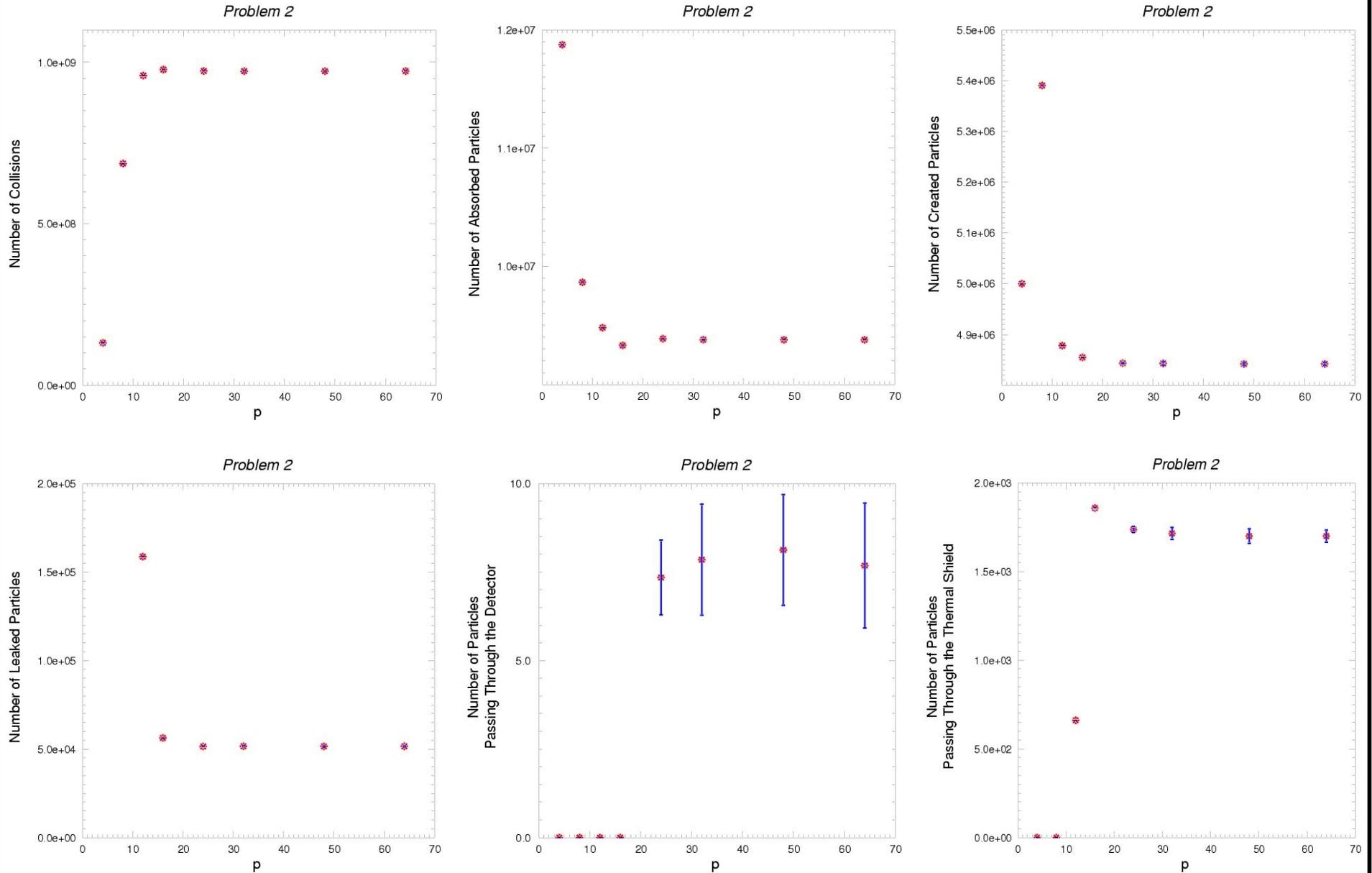


Source Problem: A Fusion Neutron Shielding Test Facility



See Reference [5].

Testing Random Number Quality in Parallel Calculations

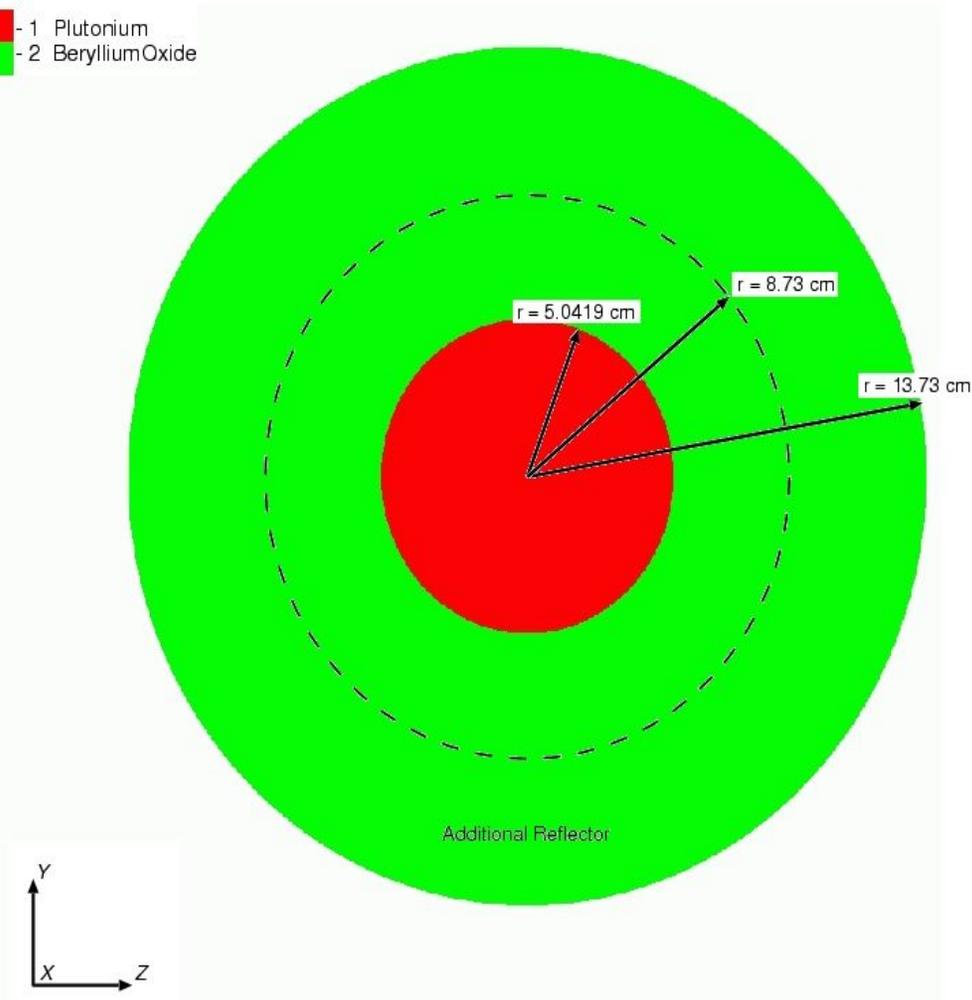


Testing Random Number Quality in Parallel Calculations



Criticality Problem: A (Gedanken) Supercritical Assembly

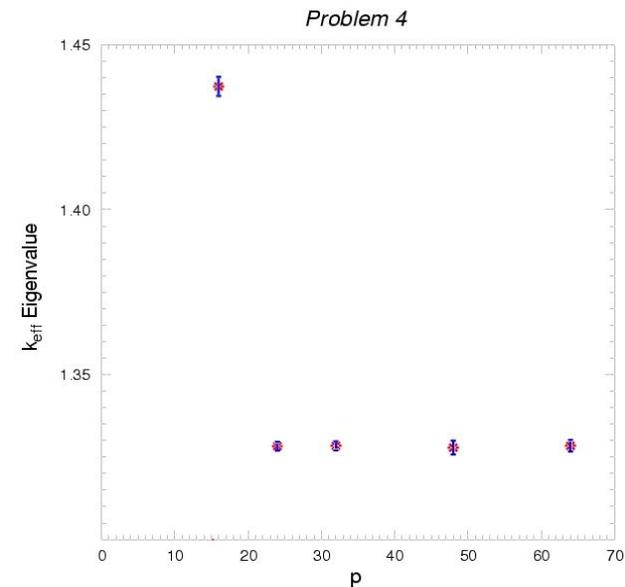
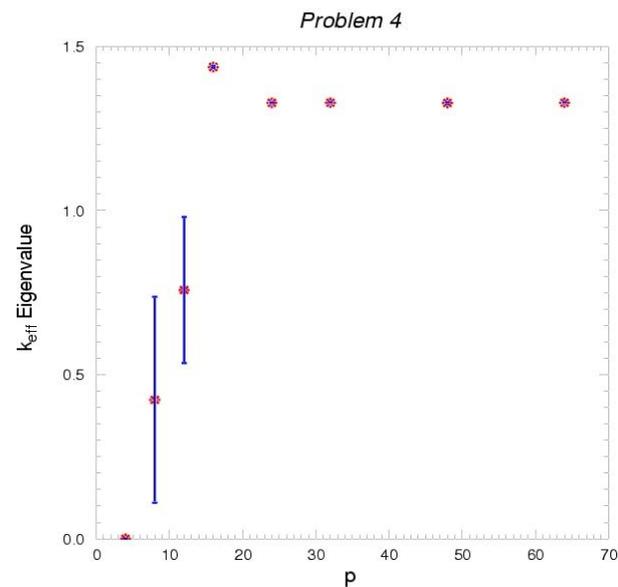
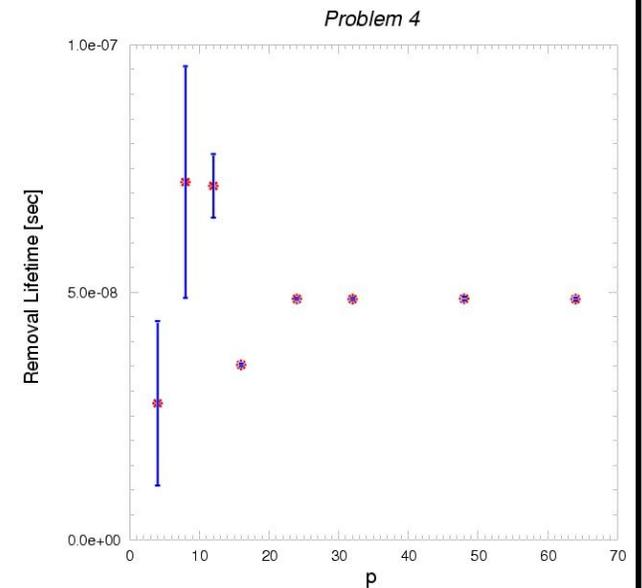
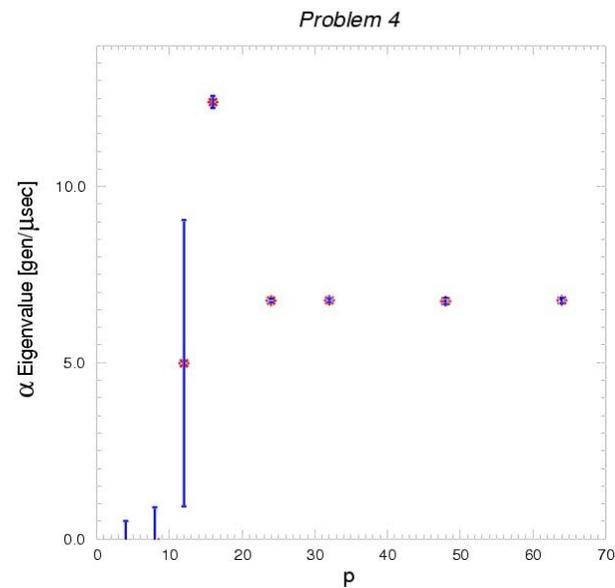
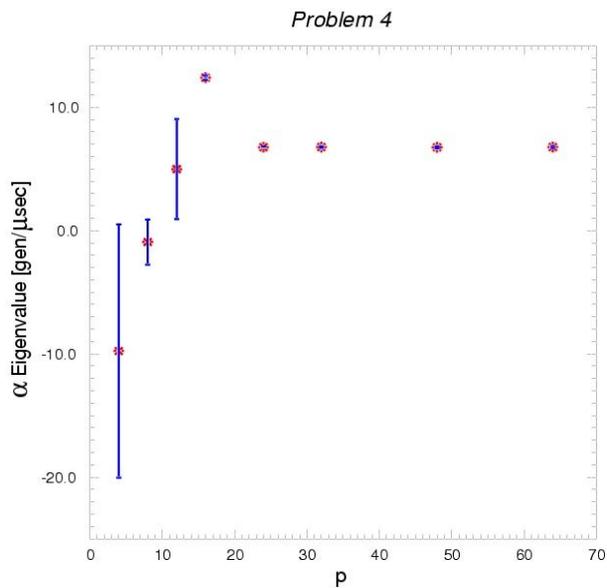
- 1 Plutonium
- 2 BerylliumOxide



$$\alpha = 6.6 \text{ gen}/\mu \text{ sec}$$

The dashed line represents the *actual* radius of the benchmark critical assembly PU-MET-FAST-018-001 [6].

Testing Random Number Quality in Parallel Calculations



Summary and Conclusions



- The accuracy of MERCURY Monte Carlo particle transport calculations has been studied in the context of a variable period (“quality”) random number generator (RNG):
 - Four problems were modeled (2 source and 2 criticality problems)
 - For each problem:
 - A series of calculations were made by varying the modulus, and hence the period, of the linear congruential generator (LCG) in the range $2^4 \leq m \leq 2^{64}$
 - For each of these problem-period pairs, an ensemble of 25 runs were performed by varying the initial random number seeds
 - Each of these runs employed 10 million particle histories
- The average results of several tallies were then compared as a function of m in order to determine the minimum RNG period necessary for unbiased results from parallel, Monte Carlo transport calculations.

Summary and Conclusions



- The main conclusion is that RNG periods as low as $m > 2^{16} = 65,536$ are sufficient to produce unbiased results.
- **Note:** These results were obtained from a parallel code that utilizes *per-particle* random number seeds, and should not be considered a universal guideline.
- A parallel transport code which employs *per-processor* random number seeds *may* require RNG periods that are larger by the ratio of the number of particles per processor.
- An important conclusion is that the current form of random number parallelism used within MERCURY does *not* require us to develop a new RNG library that supports 128-bit LCGs, which would require an additional 64-bit integer (8 bytes) of per-particle storage.



Future Directions

- Repeat this series of calculations with the prime-modulus LCGs that are available within the CNPRNG library:
 - ◆ The prime-number-modulus LCGs have been shown [7] to produce streams of random numbers with fewer correlations in the bit patterns than comparable period power-of-two-modulus LCGs
 - ◆ Determine if the added cost of calculating random numbers with the prime-modulus LCGs would produce superior results to those obtained in this study
- Modify the CNPRNG library to allow MERCURY to use random number streams that are assigned on a *per-processor* basis:
 - ◆ Determine if the *per-particle-seed* approach to Monte Carlo transport is truly superior to the *per-processor-seed* approach for a given RNG period

References



- [1] J. E. Gentle, *Random Number Generation and Monte Carlo Methods (Second Edition)*, Springer-Verlag, New York, USA (2003).
- [2] R. J. Procassini and J. M. Taylor, *MERCURY User Guide (Version b.6)*, Lawrence Livermore National Laboratory, Report UCRL-TM-204296 (2004).
- [3] B. R. Beck and E. D. Brooks III, *The RNG Random Number Library*, Lawrence Livermore National Laboratory, Internal Report (2000).
- [4] "SPRNG: Scalable Parallel Pseudo Random Number Generators Library", <http://sprng.cs.fsu.edu> (2002).
- [5] R. T. Santoro, R. G. Alsmiller, J. M. Barnes and G. T. Chapman, "Calculation of Neutron and Gamma Ray Spectra for Fusion Reactor Shield Design: Comparison with Experiment", *Nucl. Sci. and Eng.*, **78**, pp. 259-272 (1981).
- [6] International Criticality Safety Benchmark Evaluation Program, *International Handbook of Evaluated Criticality Safety Benchmark Experiment*, Nuclear Energy Agency, Report NEA/NSC/DOC(95)03/1 (2003).
- [7] B. R. Beck, *Linear Congruential Random Number Generators with Prime Moduli*, Lawrence Livermore National Laboratory, Report UCRL-JC-145424 (2000).

Acknowledgments



The authors would like to acknowledge discussions with and suggestions by Eugene Brooks and Jim Rathkopf of the Lawrence Livermore National Laboratory. This work was performed under the auspices of the U.S. Department of Energy at the Lawrence Livermore National Laboratory under Contract Number W-7405-Eng-48.