

# Reproducibility in Stochastic Simulation

Prof. Michael Mascagni

Department of Computer Science  
Department of Mathematics  
Department of Scientific Computing  
Graduate Program in Molecular Biophysics  
Florida State University, Tallahassee, FL 32306 **USA**  
*AND*

Applied and Computational Mathematics Division, Information Technology Laboratory  
National Institute of Standards and Technology, Gaithersburg, MD 20899-8910 **USA**

E-mail: [mascagni@fsu.edu](mailto:mascagni@fsu.edu) or [mascagni@nist.gov](mailto:mascagni@nist.gov)

URL: <http://www.cs.fsu.edu/~mascagni>

Joint work with Profs. Hao Ji at Cal Poly Pomona and Yaohang Li at Old Dominion University

# Contents

## Introduction

- What are Monte Carlo Methods?

- Some Illustrative Monte Carlo Applications

- Various Types of Random Numbers

## Reproducibility for Random Numbers

- Difficult Situations for RNG Reproducibility

- SPRNG

- Forensic Reproducibility

## Another Approach to Reproducibility in Stochastic Computations

- A Major Problem

- Potential Solutions

## Conclusions and Ongoing Work

# Contents

## Introduction

- What are Monte Carlo Methods?

- Some Illustrative Monte Carlo Applications

- Various Types of Random Numbers

## Reproducibility for Random Numbers

- Difficult Situations for RNG Reproducibility

- SPRNG

- Forensic Reproducibility

## Another Approach to Reproducibility in Stochastic Computations

- A Major Problem

- Potential Solutions

## Conclusions and Ongoing Work

# Contents

## Introduction

- What are Monte Carlo Methods?

- Some Illustrative Monte Carlo Applications

- Various Types of Random Numbers

## Reproducibility for Random Numbers

- Difficult Situations for RNG Reproducibility

  - SPRNG

  - Forensic Reproducibility

## Another Approach to Reproducibility in Stochastic Computations

- A Major Problem

- Potential Solutions

## Conclusions and Ongoing Work

# Contents

## Introduction

- What are Monte Carlo Methods?

- Some Illustrative Monte Carlo Applications

- Various Types of Random Numbers

## Reproducibility for Random Numbers

- Difficult Situations for RNG Reproducibility

  - SPRNG

  - Forensic Reproducibility

## Another Approach to Reproducibility in Stochastic Computations

- A Major Problem

- Potential Solutions

## Conclusions and Ongoing Work

# Monte Carlo Methods: Numerical Experimental that Use Random Numbers

## **A Monte Carlo method is any process that consumes random numbers**

1. Each calculation is a numerical experiment
  - ▶ Subject to known and unknown sources of error
  - ▶ Should be reproducible by peers
  - ▶ Should be easy to run anew with results that can be combined to reduce the variance
2. Sources of errors must be controllable/isolatable
  - ▶ Programming/science errors under your control
  - ▶ Make possible RNG errors approachable
3. Reproducibility
  - ▶ Must be able to rerun a calculation with the same numbers
  - ▶ Across different machines (modulo arithmetic issues)
  - ▶ Parallel and distributed computers?

# Integration: The Classic Monte Carlo Application

1. Consider computing  $I = \int_0^1 f(x) dx$
2. Conventional quadrature methods:

$$I \approx \sum_{i=1}^N w_i f(x_i)$$

- ▶ Standard quadrature is of this form with deterministic error bounds
  - ▶ If we hold work,  $f(x_i)$ , constant as dimension increases we see the MC advantage vs. the curse of dimensionality
3. Monte Carlo method has two parts to estimate a numerical quantity of interest,  $I$ 
    - ▶ The random process/variable:  $x_i \sim U[0, 1]$  i.i.d.
    - ▶ The estimator or score:  $f(x_i)$
    - ▶ One averages and uses a confidence interval for an error bound

$$\bar{I} = \frac{1}{N} \sum_{i=1}^N f(x_i), \quad \text{var}(I) = \frac{1}{N-1} \sum_{i=1}^N (f(x_i) - \bar{I})^2 = \frac{1}{N-1} \left[ \sum_{i=1}^N f(x_i)^2 - N\bar{I}^2 \right],$$

$$\text{var}(\bar{I}) = \frac{\text{var}(I)}{N}, \quad I \in \bar{I} \pm k \times \sqrt{\text{var}(\bar{I})}$$

## Other Early Monte Carlo Applications

- ▶ Numerical linear algebra based on sums:  $S = \sum_{i=1}^M a_i$ 
  1. Define  $p_i \geq 0$  as the probability of choosing index  $i$ , with  $\sum_{i=1}^M p_i = 1$ , and  $p_i > 0$  whenever  $a_i \neq 0$
  2. Then  $a_i/p_i$  with index  $i$  chosen with  $\{p_i\}$  is an unbiased estimate of  $S$ , as
 
$$E[a_i/p_i] = \sum_{i=1}^M \left(\frac{a_i}{p_i}\right) p_i = S$$
- ▶ Can be used to solve linear systems of the form  $x = Hx + b$
- ▶ Consider the linear system:  $x = Hx + b$ , if  $\|H\| = \mathbb{H} < 1$ , then the following iterative method converges:

$$x^{n+1} := Hx^n + b, \quad x^0 = 0,$$

and in particular we have  $x^k = \sum_{i=0}^{k-1} H^i b$ , and similarly the Neumann series converges:

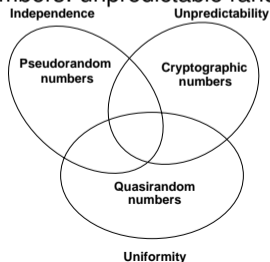
$$N = \sum_{i=0}^{\infty} H^i = (I - H)^{-1}, \quad \|N\| = \sum_{i=0}^{\infty} \|H^i\| \leq \sum_{i=0}^{\infty} \mathbb{H}^i = \frac{1}{1 - \mathbb{H}}$$

- ▶ Formally, the solution is  $x = (I - H)^{-1}b$



# Various Types of Random Numbers

1. There are ideal random numbers that serve as a powerful abstract for testing, and abstraction
2. There are several types of computational random numbers optimized for various aspects of randomness
  - ▶ “Real” random numbers: use a ‘physical source’ of randomness
  - ▶ Pseudorandom numbers: deterministic sequence that passes tests of randomness
  - ▶ Quasirandom numbers: well distributed (low discrepancy) points
  - ▶ Cryptographic random numbers: unpredictable random numbers



# Random Numbers and Reproducibility

- ▶ In general, all computations that consume random numbers benefit from being reproducible
  1. Reproducibility aids considerably in permitting program debugging, and other careful testing and measurement functions
  2. Unpredictable random numbers are specifically designed not to be reproducible mostly to enhance cryptographic security
  3. Unpredictable random numbers are often produced with the assistance of “real” sources of randomness such as radioactive decay or thermal noise
- ▶ Computational reproducibility is a concept that makes the most sense in simulation, and so it concerns mostly
  1. Pseudorandom numbers: deterministic sequences of random numbers that pass tests of randomness
  2. Quasirandom numbers: highly uniform and deterministic random numbers (not considered further)

# Reproducibility for Monte Carlo Computations and Simulation

- ▶ As mentioned previously, Monte Carlo computations and other stochastic simulations are numerical experiments
  1. You publish your computations as a confidence interval using the sample mean and variance
  2. If you use different random numbers, you get different results (mean and variance)
  3. Error estimation is stochastic
- ▶ Since Monte Carlo computations and simulations are computations, they need to be done in an environment that permits reproducibility
  1. Publication and other scientific communication requires reproducibility
  2. Reproducibility of Monte Carlo computations and other stochastic simulations ultimately rests on the reproducibility of the random numbers used
  3. Sometimes a computation may fail due to bad interaction with a particular RNG and one would like to rerun with a completely different type of RNG to probe the failure

# Requirements for a Reproducible Random Number Generator

- ▶ Reproducibility of pseudorandom numbers rests on reconstruction of the random number generator (RNG) state
- ▶ Consider the simple linear congruential generator (LCG)

$$x_{n+1} = ax_n + b \pmod{m}$$

1. We denote this as  $\text{LCG}(a, b, m; x_0)$ , where there are parameters; state/seed
  2. By knowing the parameters and restoring the state/seed one can regenerate the same random numbers from  $\text{LCG}(a, b, m; x_0)$  as often as you like
- ▶ Absolute Reproducibility for a RNG is still challenging (remember that this is an integer sequence)
    1. An RNG running on a single processor
    2. An RNG running on a large distributed memory multiprocessor system (SPRNG)
    3. There are many situations where RNGs running on complicated and unreplicable hardware are very hard to make reproducible

## Difficult Situations for RNG Reproducibility

- ▶ Many RNG reproducibility issues follow from the desire to run a computation previously run with the same random numbers on a different architecture
  1. A computation run on a multiprocessor recomputed on a single processor: support for RNG serialization
  2. A computation run on differently configured multiprocessors (ignoring accelerators) is currently supported in `SPRNG`
  3. A computation run on unpredictable hardware: example being an OpenMP code running on a multiprocessor with unknown numbers of threads
- ▶ Absolute Reproducibility for a RNG is still desirable, but we are working in `SPRNG` to ensure a new type of reproducibility
  1. Most computations do not need to be reproducible, as they are part of development or proof-of-concept
  2. Reproducibility is necessary when a computation will be used for publication

# SPRNG is Based on Parameterization

1. SPRNG is based on parameterized RNGs, which was invented by the SPRNG authors
2. Advantages of using parameterized generators
  - ▶ Each unique parameter value gives an “independent” stream
  - ▶ Each stream is uniquely numbered
  - ▶ Numbering allows for absolute reproducibility, even with MIMD queuing
  - ▶ Effective serial implementation + enumeration yield a portable scalable implementation
  - ▶ Provides theoretical testing basis
3. Implementation details
  - ▶ Generators mapped canonically to a binary tree
  - ▶ Extended seed data structure contains current seed and next generator
  - ▶ Spawning uses new next generator as starting point: assures no reuse of generators
4. A good application that provides motivation is executing parallel neutronics in an absolutely reproducible way

## A New Concept: Forensic Reproducibility

- ▶ **Forensic Reproducibility:** The ability to annotate a computation in a manner that the same computation can be redone using the annotations, and new software tools; however, the computation may not be feasible on the same architectures
- ▶ **Forensic reproducibility in random number generation is a major new focus for further SPRNG development**
  1. The SPRNG RNG data type will be expanded to record the parameters and state/seed of all the generators
  2. This will be communicated to a central file created during the computation
  3. This file can be used, with new SPRNG utilities, to recompute using the same RNGs
  4. At present, this recomputation will only be supported on a single processor in SPRNG
- ▶ **Absolute reproducibility will still be possible on SPRNG as long the computation does not have**
  1. Serialization
  2. Unreproducible hardware

# Many Types of Computations Have Had Reproducibility Issues

Numerous scientific applications have reported failure of numerical reproducibility:

- ▶ Modeling deformation of metal sheets [1];
- ▶ Compressible fluid dynamics simulations [2];
- ▶ Hydrodynamic finite element simulations [3];
- ▶ Climate simulations [4];
- ▶ Molecular dynamics simulations [5];
- ▶ Neural network simulation [8];



# A Major Problem

- ▶ Scientific applications have different requirements on numerical reproducibility
  1. Strictly bit-by-bit identical
  2. Reproducible within satisfactory bounds
  3. Reproducible if the results are statistically distributed in a manner consistent with the problem
- ▶ The major problem lies in
  1. How to define appropriate measures of numerical reproducibility
  2. How to estimate to what extent a reproduced numerical operation can be deemed acceptable to scientific computing applications
- ▶ A good measure of numerical reproducibility allows us to identify critical components within a scientific computing program that mostly affect the overall numerical reproducibility



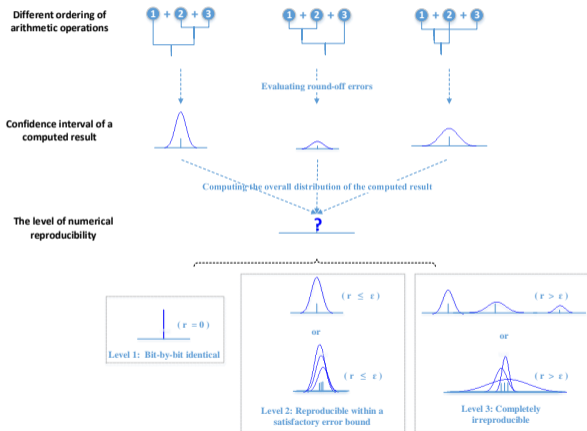
# Potential Solutions

- ▶ Measuring numerical reproducibility problem from both deterministic and statistical aspects
- ▶ Interval algorithms [6], which deal with intervals instead of numbers and obtain guarantees that the exact results are within the intervals.
- ▶ The following needed to be investigated:
  1. The propagation of the intervals along computation path under different orders
  2. How different floating point execution orders impact the result intervals
  3. How well these intervals contain the results from different execution order
  4. Determine whether interval arithmetic suffers from underestimate or overestimate problems

## Potential Solutions

- ▶ Stochastic methods [7], which is based on random sampling to approximate round-off error accumulated in scientific codes
- ▶ Numerical reproducibility can be quantified by confidence intervals, which are estimated by the distribution of randomly perturbed execution order
- ▶ The solution of dynamical system can have stable long-term structure that is relatively immune to perturbations due to unreproducible accumulation of round-off errors
  1. ODEs that have nonlinearities can have solutions that exhibit threshold behavior
  2. Accumulated round-off errors can lead to quantitatively different solutions if execution order is changed
  3. The qualitative nature of the solutions is not changed
  4. Characterization of the qualitative nature of a solution for the purposes of a phase-space reproducible solution is a possible approach

# Potential Solutions











**Figure:** Computing the Overall Confidence Interval of the Computed Result of a Summation of Floating-point Numbers using Stochastic Arithmetic.

# Conclusions and Ongoing Work

- ▶ Reproducible RNGs
  1. Reproducibility for RNGs is very different than other computational tools
  2. Absolute reproducibility, while very desirable, is sometimes not feasible
  3. Forensic reproducibility is a more tractable goal
- ▶ Future Work
  1. Studying two numerical problems, integration and matrix-matrix multiplication, to evaluate the effectiveness of various deterministic and stochastic measures
  2. Comparing the computational efficiency, accuracy, and coverage of the deterministic and stochastic measures of numerical reproducibility on various numerical operations

# References

-  K. DIETHELMR, *The limits of reproducibility in numerical simulation*, Computing in Science and Engineering, 14(1): 64-72, 2012.
-  R. W. ROBey, J. M. ROBey, R. AULWES, *In search of numerical consistency in parallel programming*, Parallel Computing, 37(4): 217-229, 2011.
-  P. LANGLOIS, R. NHEILI, C. DENIS, *Numerical Reproducibility: Feasibility Issues*, Proceedings of 7th IFIP International Conference on New Technologies, Mobility and Security, 2015.
-  Y. HE, C. H. DING, *Using accurate arithmetics to improve numerical reproducibility and stability in parallel applications*, Proceedings of the 14th international conference on Supercomputing, 2000.
-  M. TAUFER, O. PADRON, P. SAPONARO, S. PATEL, *Improving numerical reproducibility and stability in large-scale numerical simulations on GPUs*, Proceedings of IEEE International Symposium on Parallel and Distributed Processing, 2010.
-  R. MOORE, R. B. KEARFOTT, M. J. CLOUD, *Introduction to interval analysis*, SIAM, 2009
-  J. VIGNES, *A stochastic arithmetic for reliable scientific computation*, Mathematics and computers in simulation 35(3): 233-261, 1993.
-  W. BLANCO, *Personal communication*, 2016.

# Questions?

**Questions/Comments/Offers?**

**Thank You**



# Copyright Notice

**© Michael Mascagni, 2016**

**All Rights Reserved**

