Monte Carlo Methods: Early History and The Basics

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, ITL
National Institute of Standards and Technology, Gaithersburg, MD 20899-8910 **USA**
E-mail: mascagni@fsu.edu or mascagni@math.ethz.ch
or mascagni@nist.gov
URL: http://www.cs.fsu.edu/∼mascagni

Research supported by ARO, DOE, NASA, NATO, NIST, and NSF
with equipment donated by Intel and Nvidia

Outline of the Talk

Early History of Probability Theory

- Probability was first used to understand games of chance

Early History of Probability Theory

- Probability was first used to understand games of chance
  1. Antoine Gombaud, chevalier de Méré, a French nobleman called on Blaise Pascal and Pierre de Fermat were called on to resolve a dispute

Early History of Probability Theory

- Probability was first used to understand games of chance
  1. Antoine Gombaud, chevalier de Méré, a French nobleman called on Blaise Pascal and Pierre de Fermat were called on to resolve a dispute
  2. Correspondence between Pascal and Fermat led to Huygens writing a text on “Probability"

Early History of Probability Theory

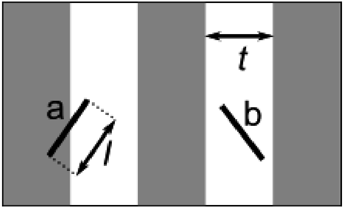
- Probability was first used to understand games of chance
    1. Antoine Gombaud, chevalier de Méré, a French nobleman called on Blaise Pascal and Pierre de Fermat were called on to resolve a dispute
    2. Correspondence between Pascal and Fermat led to Huygens writing a text on “Probability"
    3. Jacob Bernoulli, Abraham de Moivre, and Pierre-Simon, marquis de Laplace, led development of modern “Probability"

Early History of Probability Theory

- Probability was first used to understand games of chance
  1. Antoine Gombaud, chevalier de Méré, a French nobleman called on Blaise Pascal and Pierre de Fermat were called on to resolve a dispute
  2. Correspondence between Pascal and Fermat led to Huygens writing a text on “Probability"
  3. Jacob Bernoulli, Abraham de Moivre, and Pierre-Simon, marquis de Laplace, led development of modern “Probability"
  4. 1812: Laplace, *Théorie Analytique des Probabilités*

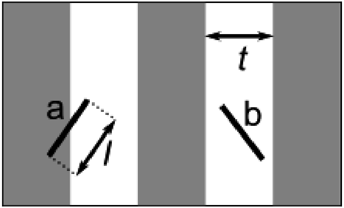
Early History of Monte Carlo: Before Los Alamos

- Buffon Needle Problem: Early Monte Carlo (experimental mathematics)



Early History of Monte Carlo: Before Los Alamos

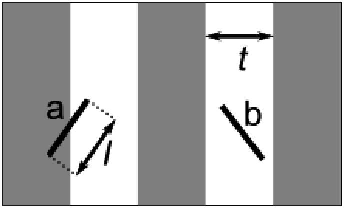
- Buffon Needle Problem: Early Monte Carlo (experimental mathematics)



1. Problem was first stated in 1777 by Georges-Louis Leclerc, comte de Buffon

Early History of Monte Carlo: Before Los Alamos

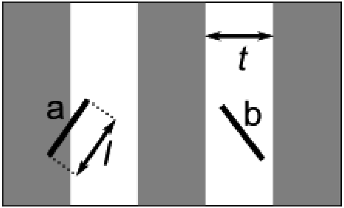
- Buffon Needle Problem: Early Monte Carlo (experimental mathematics)



1. Problem was first stated in 1777 by Georges-Louis Leclerc, comte de Buffon
2. Involves dropping a needle on a lined surface and can be used to estimate π

Early History of Monte Carlo: Before Los Alamos

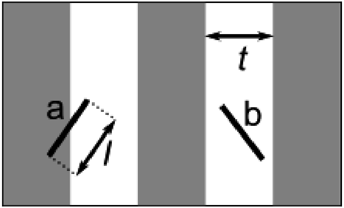
- Buffon Needle Problem: Early Monte Carlo (experimental mathematics)



1. Problem was first stated in 1777 by Georges-Louis Leclerc, comte de Buffon
2. Involves dropping a needle on a lined surface and can be used to estimate π
3. Note: Union Capt. Fox did this while in a CSA prison camp, and produced good results that later turned out to be "fudged"

Early History of Monte Carlo: Before Los Alamos

- Buffon Needle Problem: Early Monte Carlo (experimental mathematics)



1. Problem was first stated in 1777 by Georges-Louis Leclerc, comte de Buffon
2. Involves dropping a needle on a lined surface and can be used to estimate π
3. Note: Union Capt. Fox did this while in a CSA prison camp, and produced good results that later turned out to be “fudged”

- In the 1930’s, Fermi used sampling methods to estimate quantities involved in controlled fission

The Stars Align at Los Alamos

- Los Alamos brought together many interesting factors to give birth to modern Monte Carlo algorithms

The Stars Align at Los Alamos

- Los Alamos brought together many interesting factors to give birth to modern Monte Carlo algorithms
  1. The Problems: Simulation of neutron histories (neutronics), hydrodynamics, thermonuclear detonation

The Stars Align at Los Alamos

- Los Alamos brought together many interesting factors to give birth to modern Monte Carlo algorithms
    1. The Problems: Simulation of neutron histories (neutronics), hydrodynamics, thermonuclear detonation
    2. The People: Enrico Fermi, Stan Ulam, John von Neumann, Nick Metropolis, Edward Teller, ...

The Stars Align at Los Alamos

- Los Alamos brought together many interesting factors to give birth to modern Monte Carlo algorithms
  1. The Problems: Simulation of neutron histories (neutronics), hydrodynamics, thermonuclear detonation
  2. The People: Enrico Fermi, Stan Ulam, John von Neumann, Nick Metropolis, Edward Teller, ...
  3. The Technology: Massive human computers using hand calculators, the Fermiac, access to early digital computers

The Stars Align at Los Alamos

- Los Alamos brought together many interesting factors to give birth to modern Monte Carlo algorithms
  1. The Problems: Simulation of neutron histories (neutronics), hydrodynamics, thermonuclear detonation
  2. The People: Enrico Fermi, Stan Ulam, John von Neumann, Nick Metropolis, Edward Teller, ...
  3. The Technology: Massive human computers using hand calculators, the Fermiac, access to early digital computers
- The Name: Ulam's uncle would borrow money from the family by saying that “I just have to go to Monte Carlo”

The Problems

- Simulation of neutron histories (neutronics)

The Problems

- Simulation of neutron histories (neutronics)
    1. Given neutron positions/momenta, geometry

The Problems

- Simulation of neutron histories (neutronics)
  1. Given neutron positions/momenta, geometry
  2. Compute flux, criticality, fission yield

The Problems

- Simulation of neutron histories (neutronics)
  1. Given neutron positions/momenta, geometry
  2. Compute flux, criticality, fission yield
- Hydrodynamics due to nuclear implosion

The Problems

- Simulation of neutron histories (neutronics)
  1. Given neutron positions/momenta, geometry
  2. Compute flux, criticality, fission yield
- Hydrodynamics due to nuclear implosion
- Simulation of thermonuclear reactions: ignition, overall yield

The Problems

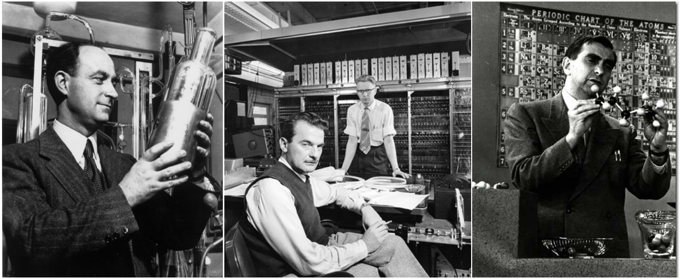
- Simulation of neutron histories (neutronics)
  1. Given neutron positions/momenta, geometry
  2. Compute flux, criticality, fission yield
- Hydrodynamics due to nuclear implosion
- Simulation of thermonuclear reactions: ignition, overall yield
  1. All these problems were more easily solved using Monte Carlo/Lagrangian methods

The Problems

- Simulation of neutron histories (neutronics)
  1. Given neutron positions/momenta, geometry
  2. Compute flux, criticality, fission yield
- Hydrodynamics due to nuclear implosion
- Simulation of thermonuclear reactions: ignition, overall yield
  1. All these problems were more easily solved using Monte Carlo/Lagrangian methods
  2. Geometry is problematic for deterministic methods but not for MC

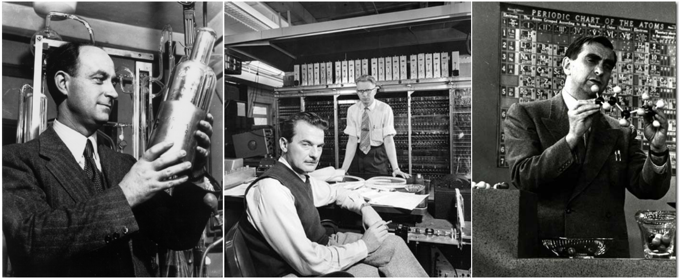
The People

- Los Alamos brought together many interesting people to work on the fission problem:



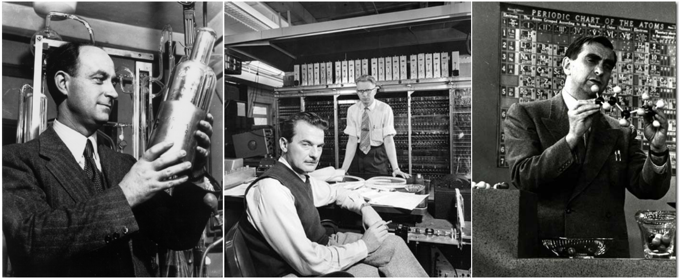
The People

- Los Alamos brought together many interesting people to work on the fission problem:
- The Physicists



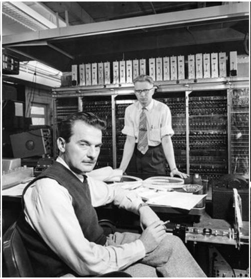
The People

- Los Alamos brought together many interesting people to work on the fission problem:
- The Physicists
    1. Enrico Fermi: experimental Nuclear Physics and computational approaches



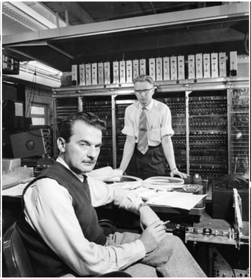
The People

- Los Alamos brought together many interesting people to work on the fission problem:
- The Physicists
  1. Enrico Fermi: experimental Nuclear Physics and computational approaches
  2. Nick Metropolis: one of the first “computer programmers” for these problems



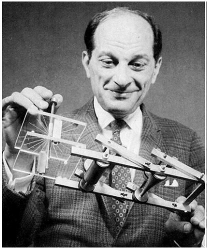
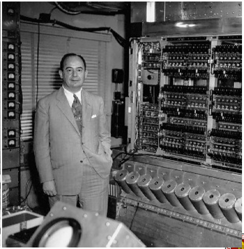
The People

- Los Alamos brought together many interesting people to work on the fission problem:
- The Physicists
  1. Enrico Fermi: experimental Nuclear Physics and computational approaches
  2. Nick Metropolis: one of the first “computer programmers” for these problems
  3. Edward Teller: more interested in the “super”

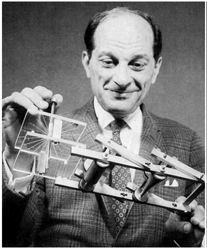

The People

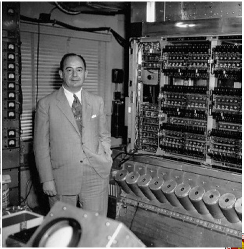
- The Mathematicians

The People

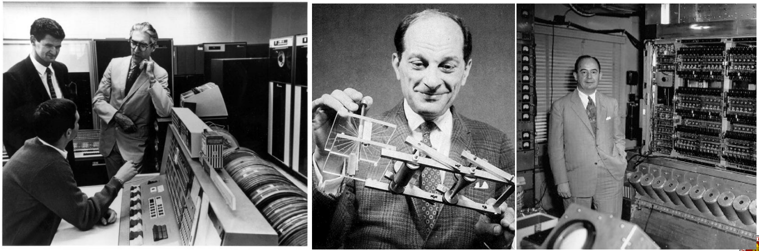
- The Mathematicians
  1. Robert Richtmyer: ran the numerical analysis activities at Los Alamos





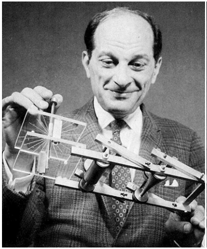
The People

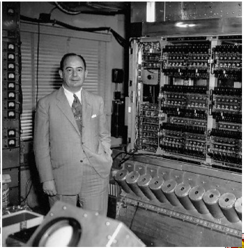
- The Mathematicians
  1. Robert Richtmyer: ran the numerical analysis activities at Los Alamos
  2. Stanislaw (Stan) Ulam: became interested in using “statistical sampling” for many problems



The People

- The Mathematicians
    1. Robert Richtmyer: ran the numerical analysis activities at Los Alamos
    2. Stanislaw (Stan) Ulam: became interested in using “statistical sampling” for many problems
    3. John von Neumann: devised Monte Carlo algorithms and helped develop digital computers





The Technology

- Simulation via computation was necessary to make progress at Los Alamos

The Technology

- Simulation via computation was necessary to make progress at Los Alamos
- Many different computational techniques were in used

The Technology

- Simulation via computation was necessary to make progress at Los Alamos
- Many different computational techniques were in used
  1. Traditional digital computation: hand calculators used by efficient technicians

The Technology

- Simulation via computation was necessary to make progress at Los Alamos
- Many different computational techniques were in used
    1. Traditional digital computation: hand calculators used by efficient technicians
    2. Analog computers including the Fermiac (picture to follow)

The Technology

- Simulation via computation was necessary to make progress at Los Alamos
- Many different computational techniques were in used
  1. Traditional digital computation: hand calculators used by efficient technicians
  2. Analog computers including the Fermiac (picture to follow)
  3. Shortly after the war, access to digital computers: ENIAC at Penn/Army Ballistics Research Laboratory (BRL)

The Technology

- Simulation via computation was necessary to make progress at Los Alamos
- Many different computational techniques were in used
  1. Traditional digital computation: hand calculators used by efficient technicians
  2. Analog computers including the Fermiac (picture to follow)
  3. Shortly after the war, access to digital computers: ENIAC at Penn/Army Ballistics Research Laboratory (BRL)
  4. Continued development and acquisition of digital computers by Metropolis including the MANIAC

An Analog Monte Carlo Computer: The Fermiac

- Neutronics required simulating exponentially distributed flights based on material cross-sections

An Analog Monte Carlo Computer: The Fermiac

- Neutronics required simulating exponentially distributed flights based on material cross-sections
- Many neutron histories are required to get statistics

An Analog Monte Carlo Computer: The Fermiac

- Neutronics required simulating exponentially distributed flights based on material cross-sections
- Many neutron histories are required to get statistics
- Fermiac allows simulation of exponential flights inputting the cross-section manually

An Analog Monte Carlo Computer: The Fermiac

- Neutronics required simulating exponentially distributed flights based on material cross-sections
- Many neutron histories are required to get statistics
- Fermiac allows simulation of exponential flights inputting the cross-section manually
- Fermiac is used on a large piece of paper with the geometry drawn for neutronics simulations

An Analog Monte Carlo Computer: The Fermiac

- Neutronics required simulating exponentially distributed flights based on material cross-sections
- Many neutron histories are required to get statistics
- Fermiac allows simulation of exponential flights inputting the cross-section manually
- Fermiac is used on a large piece of paper with the geometry drawn for neutronics simulations
- Fermiac allows an efficient graphical simulation of neutronics

An Analog Monte Carlo Computer: The Fermiac

- Neutronics required simulating exponentially distributed flights based on material cross-sections
- Many neutron histories are required to get statistics
- Fermiac allows simulation of exponential flights inputting the cross-section manually
- Fermiac is used on a large piece of paper with the geometry drawn for neutronics simulations
- Fermiac allows an efficient graphical simulation of neutronics
- Parallelism is achievable with the Fermiac

An Analog Monte Carlo Computer: The Fermiac

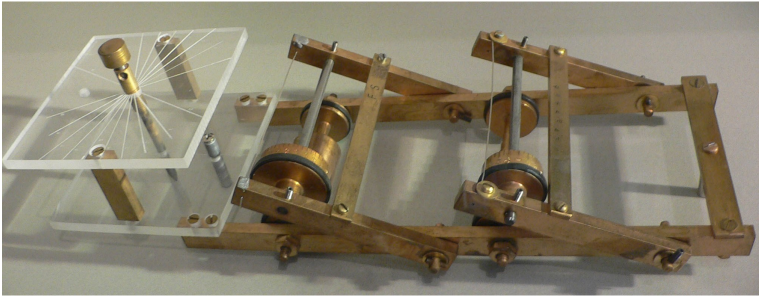


Figure: Enrico Fermi's Fermiac at the Bradbury Museum in Los Alamos

An Analog Monte Carlo Computer: The Fermiac

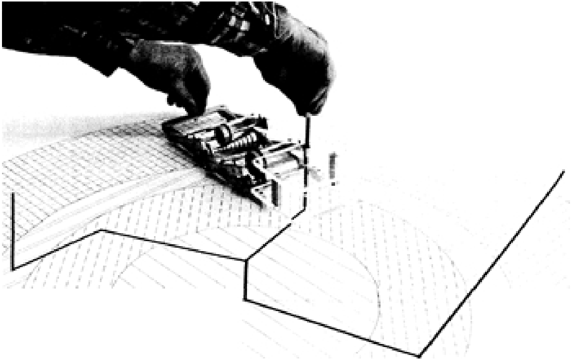


Figure: The Fermiac in Action

An Early Digital Computer: The ENIAC

- ENIAC: Electronic Numerical Integrator And Computer

An Early Digital Computer: The ENIAC

- ENIAC: Electronic Numerical Integrator And Computer
- Funded by US Army with contract signed on June 5, 1943

An Early Digital Computer: The ENIAC

- ENIAC: Electronic Numerical Integrator And Computer
- Funded by US Army with contract signed on June 5, 1943
- Built in secret by the University of Pennsylvania's Moore School of Electrical Engineering

An Early Digital Computer: The ENIAC

- ENIAC: Electronic Numerical Integrator And Computer
- Funded by US Army with contract signed on June 5, 1943
- Built in secret by the University of Pennsylvania's Moore School of Electrical Engineering
- Completed February 14, 1946 in Philadelphia and used until November 9, 1946

An Early Digital Computer: The ENIAC

- ENIAC: Electronic Numerical Integrator And Computer
- Funded by US Army with contract signed on June 5, 1943
- Built in secret by the University of Pennsylvania's Moore School of Electrical Engineering
- Completed February 14, 1946 in Philadelphia and used until November 9, 1946
- Moved (with upgrade) to Aberdeen Proving Grounds and began operations July 29, 1947

An Early Digital Computer: The ENIAC

- ENIAC: Electronic Numerical Integrator And Computer
- Funded by US Army with contract signed on June 5, 1943
- Built in secret by the University of Pennsylvania's Moore School of Electrical Engineering
- Completed February 14, 1946 in Philadelphia and used until November 9, 1946
- Moved (with upgrade) to Aberdeen Proving Grounds and began operations July 29, 1947
- Remained in continuous operation at the Army BRL until 1955

An Early Digital Computer: The ENIAC

- ENIAC is a completely programmable computer using first a plug panel

An Early Digital Computer: The ENIAC

- ENIAC is a completely programmable computer using first a plug panel
- ENIAC first contained (military rejects!)

An Early Digital Computer: The ENIAC

- ENIAC is a completely programmable computer using first a plug panel
- ENIAC first contained (military rejects!)
  1. 17,468 vacuum tubes

An Early Digital Computer: The ENIAC

- ENIAC is a completely programmable computer using first a plug panel
- ENIAC first contained (military rejects!)
  1. 17,468 vacuum tubes
  2. 7,200 crystal diodes

An Early Digital Computer: The ENIAC

- ENIAC is a completely programmable computer using first a plug panel
- ENIAC first contained (military rejects!)
  1. 17,468 vacuum tubes
  2. 7,200 crystal diodes
  3. 1,500 relays, 70,000 resistors

An Early Digital Computer: The ENIAC

- ENIAC is a completely programmable computer using first a plug panel
- ENIAC first contained (military rejects!)
  1. 17,468 vacuum tubes
  2. 7,200 crystal diodes
  3. 1,500 relays, 70,000 resistors
  4. 10,000 capacitors

An Early Digital Computer: The ENIAC

- ENIAC is a completely programmable computer using first a plug panel
- ENIAC first contained (military rejects!)
  1. 17,468 vacuum tubes
  2. 7,200 crystal diodes
  3. 1,500 relays, 70,000 resistors
  4. 10,000 capacitors
  5. about 5 million hand-soldered joints

An Early Digital Computer: The ENIAC

- ENIAC is a completely programmable computer using first a plug panel
- ENIAC first contained (military rejects!)
  1. 17,468 vacuum tubes
  2. 7,200 crystal diodes
  3. 1,500 relays, 70,000 resistors
  4. 10,000 capacitors
  5. about 5 million hand-soldered joints
- Clock was 5KHz

An Early Digital Computer: The ENIAC

- ENIAC is a completely programmable computer using first a plug panel
- ENIAC first contained (military rejects!)
  1. 17,468 vacuum tubes
  2. 7,200 crystal diodes
  3. 1,500 relays, 70,000 resistors
  4. 10,000 capacitors
  5. about 5 million hand-soldered joints
- Clock was 5KHz
- Ended up with a 100-word core memory

An Early Digital Computer: The ENIAC

- ENIAC is a completely programmable computer using first a plug panel
- ENIAC first contained (military rejects!)
  1. 17,468 vacuum tubes
  2. 7,200 crystal diodes
  3. 1,500 relays, 70,000 resistors
  4. 10,000 capacitors
  5. about 5 million hand-soldered joints
- Clock was 5KHz
- Ended up with a 100-word core memory
- Metropolis would go to BRL to work on the “Los Alamos” problem on the ENIAC

An Early Digital Computer: The ENIAC

Figure: The ENIAC at the University of Pennsylvania

An Early Digital Computer: The ENIAC

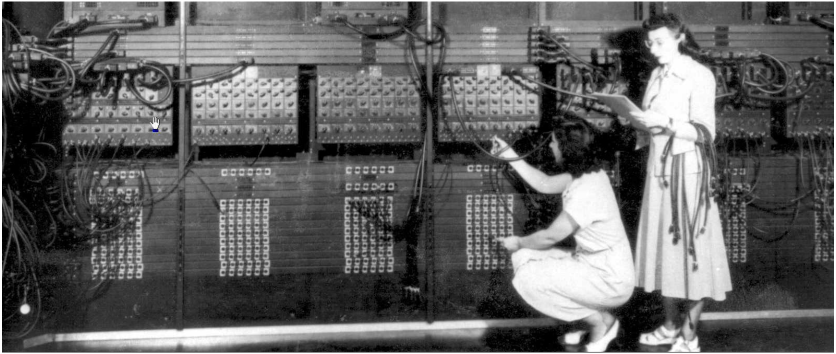


Figure: Programming the ENIAC

An Early Digital Computer: The ENIAC

Figure: Tubes from the ENIAC

The Birth of Monte Carlo Methods

- After the was digital computer was perfect for “statistical sampling”

The Birth of Monte Carlo Methods

- After the was digital computer was perfect for “statistical sampling”
  1. Individual samples were often very simple to program

The Birth of Monte Carlo Methods

- After the was digital computer was perfect for “statistical sampling”
  1. Individual samples were often very simple to program
  2. Small memory was not a big constraint for these methods

The Birth of Monte Carlo Methods

- After the was digital computer was perfect for “statistical sampling”
  1. Individual samples were often very simple to program
  2. Small memory was not a big constraint for these methods
  3. A much better use for digital vs. human computers

The Birth of Monte Carlo Methods

- After the was digital computer was perfect for “statistical sampling”
  1. Individual samples were often very simple to program
  2. Small memory was not a big constraint for these methods
  3. A much better use for digital vs. human computers
- Early Monte Carlo Meetings

The Birth of Monte Carlo Methods

- After the was digital computer was perfect for “statistical sampling”
  1. Individual samples were often very simple to program
  2. Small memory was not a big constraint for these methods
  3. A much better use for digital vs. human computers
- Early Monte Carlo Meetings
  1. 1952, Los Angeles: RAND Corp., National Bureau of Standards (NBS now NIST), Oak Ridge

The Birth of Monte Carlo Methods

- After the was digital computer was perfect for “statistical sampling”
  1. Individual samples were often very simple to program
  2. Small memory was not a big constraint for these methods
  3. A much better use for digital vs. human computers
- Early Monte Carlo Meetings
  1. 1952, Los Angeles: RAND Corp., National Bureau of Standards (NBS now NIST), Oak Ridge
  2. 1954, Gainesville, FL: University of Florida Statistical Lab

Integration: The Classic Monte Carlo Application

1. Consider computing $I = \int_0^1 f(x)\, dx$

Integration: The Classic Monte Carlo Application

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

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

Integration: The Classic Monte Carlo Application

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

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

- *Rectangle*: $w_i = \frac{1}{N}$, $x_i = \frac{i}{N}$

Integration: The Classic Monte Carlo Application

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

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

- *Rectangle*: $w_i = \frac{1}{N}$, $x_i = \frac{i}{N}$
- *Trapezoidal*: $w_i = \frac{2}{N}$, $w_1 = w_N = \frac{1}{N}$, $x_i = \frac{i}{N}$

Integration: The Classic Monte Carlo Application

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

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

   - *Rectangle*: $w_i = \frac{1}{N}$, $x_i = \frac{i}{N}$
   - *Trapezoidal*: $w_i = \frac{2}{N}$, $w_1 = w_N = \frac{1}{N}$, $x_i = \frac{i}{N}$
3. Monte Carlo method has two parts to estimate a numerical quantity of interest, I

Integration: The Classic Monte Carlo Application

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

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

   - *Rectangle*: $w_i = \frac{1}{N}$, $x_i = \frac{i}{N}$
   - *Trapezoidal*: $w_i = \frac{2}{N}$, $w_1 = w_N = \frac{1}{N}$, $x_i = \frac{i}{N}$
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.

Integration: The Classic Monte Carlo Application

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

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

   - *Rectangle*: $w_i = \frac{1}{N}$, $x_i = \frac{i}{N}$
   - *Trapezoidal*: $w_i = \frac{2}{N}$, $w_1 = w_N = \frac{1}{N}$, $x_i = \frac{i}{N}$
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 score: $f(x_i)$

Integration: The Classic Monte Carlo Application

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

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

   - *Rectangle*: $w_i = \frac{1}{N}$, $x_i = \frac{i}{N}$
   - *Trapezoidal*: $w_i = \frac{2}{N}$, $w_1 = w_N = \frac{1}{N}$, $x_i = \frac{i}{N}$

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 score: $f(x_i)$
   - One averages and uses a confidence interval for an error bound

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

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

Other Early Monte Carlo Applications

- Numerical linear algebra based on sums: $S = \sum_{i=1}^{N} a_i$

Other Early Monte Carlo Applications

- Numerical linear algebra based on sums: $S = \sum_{i=1}^{N} a_i$
    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 $a_i \neq 0$

Other Early Monte Carlo Applications

- Numerical linear algebra based on sums: $S = \sum_{i=1}^{N} a_i$
  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 $E[a_i/p_i] = \sum_{i=1}^{M} \left(\frac{a_i}{p_i}\right) p_i = S$

Other Early Monte Carlo Applications

- Numerical linear algebra based on sums: $S = \sum_{i=1}^{N} a_i$
  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 $x = Hx + b$

Other Early Monte Carlo Applications

- Numerical linear algebra based on sums: $S = \sum_{i=1}^{N} a_i$
  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:

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

Other Early Monte Carlo Applications

- Numerical linear algebra based on sums: $S = \sum_{i=1}^{N} a_i$
  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:

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

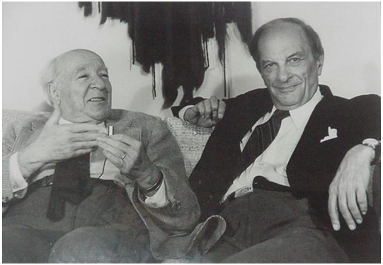
and in particular we have $x^k = \sum_{i=0}^{k-1} H^i b$, and similarly the Neumann series converges:

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

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

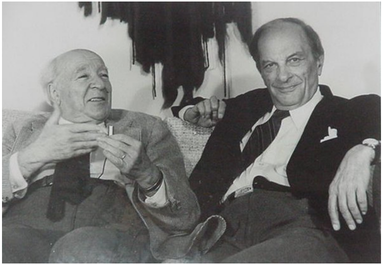
Other Early Monte Carlo Applications

- Methods for partial differential and integral equations



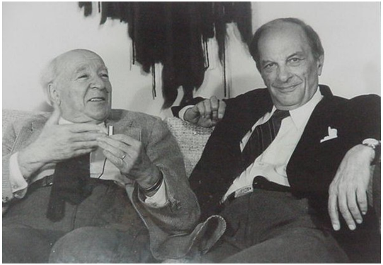
Other Early Monte Carlo Applications

- Methods for partial differential and integral equations
  1. Integral equation methods are similar in construction to the linear system methods



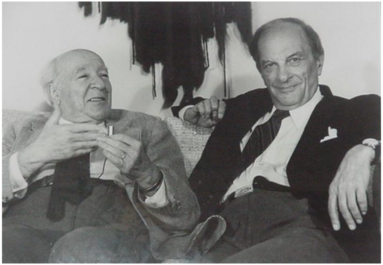
Other Early Monte Carlo Applications

- Methods for partial differential and integral equations
    1. Integral equation methods are similar in construction to the linear system methods
    2. PDEs can be solved by using the Feynman-Kac formula



Other Early Monte Carlo Applications

- Methods for partial differential and integral equations
  1. Integral equation methods are similar in construction to the linear system methods
  2. PDEs can be solved by using the Feynman-Kac formula
  3. Note Kac and Ulam both were trained in Lwów



Monte Carlo Methods: Numerical Experimental that Use Random Numbers

- A Monte Carlo method is any process that consumes random numbers

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

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

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

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

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

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

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

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

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

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)

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?

Early Random Number Generators on Digital Computers

- Middle-Square method: von Neumann

Early Random Number Generators on Digital Computers

- Middle-Square method: von Neumann
  1. 10 digit numbers: $x_{n+1} = \lfloor \frac{x_n^2}{10^5} \rfloor \pmod{10^{10}}$

Early Random Number Generators on Digital Computers

- Middle-Square method: von Neumann
  1. 10 digit numbers: $x_{n+1} = \lfloor \frac{x_n^2}{10^5} \rfloor \pmod{10^{10}}$
  2. Multiplication leads to good mixing

Early Random Number Generators on Digital Computers

- Middle-Square method: von Neumann
  1. 10 digit numbers: $x_{n+1} = \lfloor \frac{x_n^2}{10^5} \rfloor \pmod{10^{10}}$
  2. Multiplication leads to good mixing
  3. Zeros in lead to short periods and cycle collapse

Early Random Number Generators on Digital Computers

- Middle-Square method: von Neumann
  1. 10 digit numbers: $x_{n+1} = \lfloor \frac{x_n^2}{10^5} \rfloor \pmod{10^{10}}$
  2. Multiplication leads to good mixing
  3. Zeros in lead to short periods and cycle collapse
- Linear congruential method: D. H. Lehmer

Early Random Number Generators on Digital Computers

- Middle-Square method: von Neumann
  1. 10 digit numbers: $x_{n+1} = \lfloor \frac{x_n^2}{10^5} \rfloor \pmod{10^{10}}$
  2. Multiplication leads to good mixing
  3. Zeros in lead to short periods and cycle collapse
- Linear congruential method: D. H. Lehmer
- $x_{n+1} = ax_n + c \pmod{m}$

Early Random Number Generators on Digital Computers

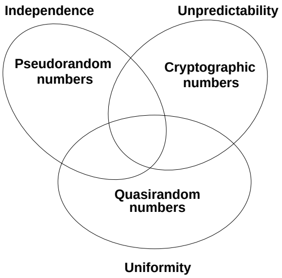
- Middle-Square method: von Neumann
  1. 10 digit numbers: $x_{n+1} = \lfloor \frac{x_n^2}{10^5} \rfloor \pmod{10^{10}}$
  2. Multiplication leads to good mixing
  3. Zeros in lead to short periods and cycle collapse
- Linear congruential method: D. H. Lehmer
- $x_{n+1} = ax_n + c \pmod{m}$
- Good properties with good parameters

Early Random Number Generators on Digital Computers

- Middle-Square method: von Neumann
  1. 10 digit numbers: $x_{n+1} = \lfloor \frac{x_n^2}{10^5} \rfloor \pmod{10^{10}}$
  2. Multiplication leads to good mixing
  3. Zeros in lead to short periods and cycle collapse
- Linear congruential method: D. H. Lehmer
- $x_{n+1} = ax_n + c \pmod{m}$
- Good properties with good parameters
- Has become very popular

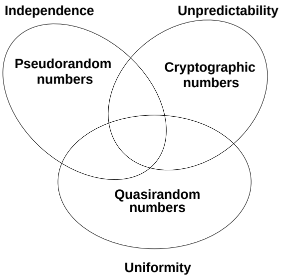
What are Random Numbers Used For?

- There are many types of random numbers



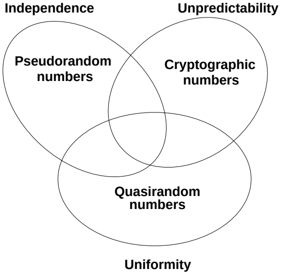
What are Random Numbers Used For?

- There are many types of random numbers
  1. “Real" random numbers: a mathematical idealization



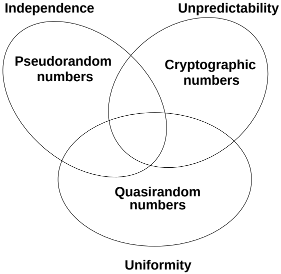
What are Random Numbers Used For?

- There are many types of random numbers
  1. “Real" random numbers: a mathematical idealization
  2. Random numbers based on a “physical source” of randomness



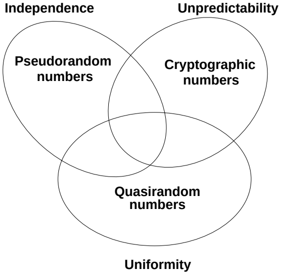
What are Random Numbers Used For?

- There are many types of random numbers
  1. “Real" random numbers: a mathematical idealization
  2. Random numbers based on a “physical source” of randomness
  3. Computational Random numbers



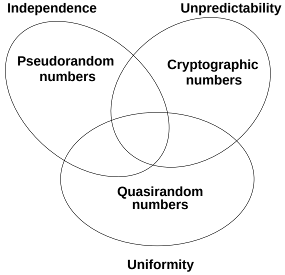
What are Random Numbers Used For?

- There are many types of random numbers
  1. “Real" random numbers: a mathematical idealization
  2. Random numbers based on a “physical source” of randomness
  3. Computational Random numbers
  1. Pseudorandom numbers: deterministic sequence that passes tests of randomness



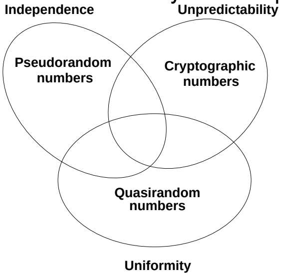
What are Random Numbers Used For?

- There are many types of random numbers
  1. “Real" random numbers: a mathematical idealization
  2. Random numbers based on a “physical source” of randomness
  3. Computational Random numbers
  1. Pseudorandom numbers: deterministic sequence that passes tests of randomness
  2. Cryptographic numbers: totally unpredictable



What are Random Numbers Used For?

- There are many types of random numbers
  1. “Real" random numbers: a mathematical idealization
  2. Random numbers based on a “physical source” of randomness
  3. Computational Random numbers
  1. Pseudorandom numbers: deterministic sequence that passes tests of randomness
  2. Cryptographic numbers: totally unpredictable
  3. Quasirandom numbers: very uniform points



Future Work on Random Numbers

1. Support for new architectures

Future Work on Random Numbers

1. Support for new architectures
   - Multicore processors: OpenMP-SPRNG

Future Work on Random Numbers

1. Support for new architectures
   - Multicore processors: OpenMP-SPRNG
   - GPGPU support: LCGs and FLGs

Future Work on Random Numbers

1. Support for new architectures
   - Multicore processors: OpenMP-SPRNG
   - GPGPU support: LCGs and FLGs
2. Testing Random Numbers

Future Work on Random Numbers

1. Support for new architectures
   - Multicore processors: OpenMP-SPRNG
   - GPGPU support: LCGs and FLGs
2. Testing Random Numbers
   - Hardware random numbers

Future Work on Random Numbers

1. Support for new architectures
   - Multicore processors: OpenMP-SPRNG
   - GPGPU support: LCGs and FLGs
2. Testing Random Numbers
   - Hardware random numbers
   - Cryptographic test suites

Future Work on Random Numbers

1. Support for new architectures
   - Multicore processors: OpenMP-SPRNG
   - GPGPU support: LCGs and FLGs
2. Testing Random Numbers
   - Hardware random numbers
   - Cryptographic test suites
   - Support for new RNG suites

Future Work on Random Numbers

1. Support for new architectures
   - Multicore processors: OpenMP-SPRNG
   - GPGPU support: LCGs and FLGs
2. Testing Random Numbers
   - Hardware random numbers
   - Cryptographic test suites
   - Support for new RNG suites
3. Applications

Future Work on Random Numbers

1. Support for new architectures
   - Multicore processors: OpenMP-SPRNG
   - GPGPU support: LCGs and FLGs
2. Testing Random Numbers
   - Hardware random numbers
   - Cryptographic test suites
   - Support for new RNG suites
3. Applications
   - Monte Carlo on new architectures

Future Work on Random Numbers

1. Support for new architectures
   - Multicore processors: OpenMP-SPRNG
   - GPGPU support: LCGs and FLGs
2. Testing Random Numbers
   - Hardware random numbers
   - Cryptographic test suites
   - Support for new RNG suites
3. Applications
   - Monte Carlo on new architectures
   - Reproducibility and system integrity

Future Work on Random Numbers

1. Support for new architectures
   - Multicore processors: OpenMP-SPRNG
   - GPGPU support: LCGs and FLGs
2. Testing Random Numbers
   - Hardware random numbers
   - Cryptographic test suites
   - Support for new RNG suites
3. Applications
   - Monte Carlo on new architectures
   - Reproducibility and system integrity
   - Computational resilience and fault tolerance

Future Work on Random Numbers

1. Support for new architectures
   - Multicore processors: OpenMP-SPRNG
   - GPGPU support: LCGs and FLGs
2. Testing Random Numbers
   - Hardware random numbers
   - Cryptographic test suites
   - Support for new RNG suites
3. Applications
   - Monte Carlo on new architectures
   - Reproducibility and system integrity
   - Computational resilience and fault tolerance
4. Commercialization of SPRNG

References

[M. Mascagni, T. Anderson, H. Yu and Y. Qiu (2014)]
Papers on SPRNG generators for Multicore and GPGPU
One submitted and three in preparation

References

[M. Mascagni, T. Anderson, H. Yu and Y. Qiu (2014)]
Papers on `SPRNG` generators for Multicore and GPGPU
One submitted and three in preparation

[M. Mascagni and H. Chi (2004)]
Parallel Linear Congruential Generators with Sophie-Germain Moduli,
Parallel Computing, **30**: 1217–1231.

References

[M. Mascagni, T. Anderson, H. Yu and Y. Qiu (2014)]
Papers on SPRNG generators for Multicore and GPGPU
One submitted and three in preparation

[M. Mascagni and H. Chi (2004)]
Parallel Linear Congruential Generators with Sophie-Germain Moduli,
Parallel Computing, **30**: 1217–1231.

[M. Mascagni and A. Srinivasan (2004)]
Parameterizing Parallel Multiplicative Lagged-Fibonacci Generators,
Parallel Computing, **30**: 899–916.

References

[M. Mascagni, T. Anderson, H. Yu and Y. Qiu (2014)]
Papers on `SPRNG` generators for Multicore and GPGPU
One submitted and three in preparation

[M. Mascagni and H. Chi (2004)]
Parallel Linear Congruential Generators with Sophie-Germain Moduli,
Parallel Computing, **30**: 1217–1231.

[M. Mascagni and A. Srinivasan (2004)]
Parameterizing Parallel Multiplicative Lagged-Fibonacci Generators,
Parallel Computing, **30**: 899–916.

[M. Mascagni and A. Srinivasan (2000)]
Algorithm 806: SPRNG: A Scalable Library for Pseudorandom Number Generation,
ACM Transactions on Mathematical Software, **26**: 436–461.

Questions?