QUASI-MONTE CARLO METHODS: WHERE RANDOMNESS AND DETERMINISM COLLIDE

MICHAEL MASCAGNI: mascagni@cs.fsu.edu
HOME PAGE: http://www.cs.fsu.edu/~mascagni

Department of Computer Science
203 Love Building
Florida State University
Tallahassee, FL 32306-4530

Supported by ARO, DOE/ASCI, and NSF.

Typeset by AMS-TEX
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?
What are Random Numbers used for?

1. Random numbers are used extensively in simulation, statistics, and in *Monte Carlo* computations
   - Simulation: use random numbers to “randomly pick” event outcomes based on statistical or experiential data
   - Statistics: use random numbers to generate data with a particular distribution to calculate statistical properties (when analytic techniques fail)

2. There are many Monte Carlo applications of great interest
   - Numerical Quadrature (see below)
   - Quantum mechanics: Solving Schrödinger’s equation with Green’s function Monte Carlo via random walks
What are Random Numbers used for? (Cont.)

▶ Mathematics: Using the Feynman-Kac/path integral methods to solve partial differential equations with random walks
▶ Defense: neutronics, nuclear weapons design
▶ Finance: derivatives, mortgage-backed securities

3. There are many types of random numbers
▶ “Real” random numbers: uses a “physical source” of randomness
▶ Pseudorandom numbers: deterministic sequence that passes tests of randomness
▶ Quasirandom numbers: well distributed (low discrepancy) points
Qualitative Aspects of Randomness

• Randomness has (at least) three different aspects which are \textit{not} mutually exclusive:

  ▶ Independence: new numbers know nothing about previous numbers
  ▶ Uniformity: ensembles of numbers spread out evenly
  ▶ Unpredictability: knowing previous values gives you no edge in predicting future values

• These three faces of randomness have motivated different kinds of computational random numbers:

  ▶ Pseudorandom: Passes statistical tests, which are essentially tests of independence
  ▶ Quasirandom: Very highly uniformly distributed
  ▶ Cryptological: Totally unpredictable to defeat cryptographers
The Classic Monte Carlo Application: Numerical Integration

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

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

A. Rectangle: $w_i = \frac{1}{N}, x_i = \frac{i}{N}$

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

1. Monte Carlo quadrature:

$$I \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i), \quad x_i \text{ is } U[0, 1], \ i.i.d.$$  

2. Big advantage seen in multidimensional integration, consider (s-dimensions):

$$I = \int_{[0,1]^s} f(x_1, \ldots, x_s) \, dx_1 \ldots dx_s$$
The Classic Monte Carlo Application: Numerical Integration (Cont.)

3. Errors are significantly different, with $N$ function evaluations we see the curse of dimensionality:

A. Product trapezoidal rule: Error $= O(N^{-2/s})$

B. Monte Carlo: Error $= O(N^{-1/2})$ (indep. of $s$!!)

4. Note: the errors are deterministic for the trapezoidal rule whereas the MCM error is a variance bound

5. For $s = 1$, $E[f(x_i)] = I$ when $x_i$ is $U[0, 1]$, so $E[\frac{1}{N} \sum_{i=1}^{N} f(x_i)] = I$, and $Var[\frac{1}{N} \sum_{i=1}^{N} f(x_i)] = Var[f(x_i)]/N$. $Var[f(x_i)] = \int_0^1 (f(x) - I)^2 dx$
Pseudorandom Numbers

- Pseudorandom numbers mimic the properties of “real” random numbers
  A. Pass statistical tests
  B. Reduce error is $O(N^{-\frac{1}{2}})$ in Monte Carlo

- Some common pseudorandom number generators:
  1. Linear congruential: $x_n = ax_{n-1} + c \pmod{m}$
  2. Shift register: $y_n = y_{n-s} + y_{n-r} \pmod{2}$, $r > s$
  3. Additive lagged-Fibonacci:
     $$z_n = z_{n-s} + z_{n-r} \pmod{2^k}, \ r > s$$
  4. Combined: $w_n = y_n + z_n \pmod{p}$
  5. Multiplicative lagged-Fibonacci:
     $$x_n = x_{n-s} \times x_{n-r} \pmod{2^k}, \ r > s$$
  6. Implicit inversive congruential:
     $$x_n = a\overline{x_{n-1}} + c \pmod{p}$$
  7. Explicit inversive congruential: $x_n = a\overline{n} + c \pmod{p}$
Pseudorandom Numbers (Cont.)

- Some properties of pseudorandom number generators, integers: \( \{x_n\} \) from modulo \( m \) recursion, and \( U[0, 1], z_n = \frac{x_n}{m} \)

A. Should be a purely period sequence (e.g.: DES and IDEA are not provably periodic)

B. Period length: \( Per(x_n) \) should be large

C. Cost per bit should be moderate (not cryptography)

D. Should be based on theoretically solid and empirically tested recursions

E. Should be a totally reproducible sequence
Pseudorandom Numbers (Cont.)

• Some common facts about pseudorandom number generators:

1. Recursions modulo a power-of-two are cheap, but have simple structure

2. Recursions modulo a prime are more costly, but have higher quality: use Mersenne primes: \(2^p - 1\), where \(p\) is prime, too

3. Shift-registers (Mersenne Twisters) are efficient and have good quality

4. Lagged-Fibonacci generators are efficient, but have some flaws

5. Combining generators is “provably good”

6. Modular inversion is very costly
Quasirandom Numbers

- Can abandon hope of a computational sequence that is sufficiently “random” by using a deterministic sequence that has a small star discrepancy yet may be quite nonrandom: so-called “quasirandom” numbers

- **Definition**: The star discrepancy $D_N^*$ of $x_1, \ldots, x_N \in J$ (measure of uniformity):

$$D_N^* = D_N^*(x_1, \ldots, x_N) = \sup_{0 \leq u \leq 1} \left| \frac{1}{N} \sum_{n=1}^{N} \chi_{[0,u)}(x_n) - u \right|,$$

where $\chi$ is the characteristic function

- **Theorem** (Koksma, 1942): if $f(x)$ has bounded variation $V(f)$ on $[0, 1]$ and $x_1, \ldots, x_N \in [0, 1]$ with star discrepancy $D_N^*$, then:

$$\left| \frac{1}{N} \sum_{n=1}^{N} f(x_n) - \int_{0}^{1} f(x) \, dx \right| \leq V(f)D_N^*$$
Discrepancy Facts

◊ Real random numbers have (the law of the iterated logarithm):

\[ D_N^* = O(N^{-1/2}(\log \log N)^{-1/2}) \]

◊ Klaus F. Roth (Fields medalist in 1958) proved the following lower bound in 1954 for the star discrepancy of \( N \) points in \( s \) dimensions:

\[ D_N^* \geq O(N^{-1}(\log N)^{s-1/2}) \]

◊ Sequences (indefinite length) and point sets have different “best discrepancies” at present:

- Sequence: \( D_N^* \leq O(N^{-1}(\log N)^{s-1}) \)
- Point set: \( D_N^* \leq O(N^{-1}(\log N)^{s-2}) \)
Some Types of Quasirandom Numbers

diamond Must choose point sets (finite #) or sequences (infinite #) with small $D_N^*$

diamond Often used is the van der Corput sequence in base $b$: $x_n = \Phi_b(n - 1), n = 1, 2, \ldots$, where for $b \in \mathbb{Z}, b \geq 2$:

$$\Phi_b \left( \sum_{j=0}^{\infty} a_j b^j \right) = \sum_{j=0}^{\infty} a_j b^{-j-1} \quad \text{with}$$

$$a_j \in \{0, 1, \ldots, b - 1\}$$

For v.d. Corput sequence $N D_N^* \leq \frac{\log N}{3 \log 2} + O(1)$

diamond With $b = 2$, we get $\left\{ \frac{1}{2}, \frac{1}{4}, \frac{3}{8}, \frac{5}{8}, \frac{3}{8}, \frac{7}{8}, \ldots \right\}$

diamond With $b = 3$, we get $\left\{ \frac{1}{3}, \frac{2}{3}, \frac{1}{9}, \frac{4}{9}, \frac{7}{9}, \frac{2}{9}, \frac{5}{9}, \frac{8}{9}, \ldots \right\}$
Some Types of QRNs (Cont.)

1. Another interpretation of the v.d. Corput sequence:
   ◦ Define the $i$th $\ell$-bit "direction number" as: $v_i = 2^i$
     (think of this as a bit vector)
   ◦ Think of $n-1$ via its base-2 representation $n-1 = b_{\ell-1}b_{\ell-2}\ldots b_1b_0$
   ◦ Thus we have
     $$\Phi_2(n-1) = 2^{-\ell} \bigoplus_{i=0, b_i=1}^{i=\ell-1} v_i$$

2. The Sobol’ sequence works the same!!
   ◦ Use recursions with a primitive binary polynomial define the (dense) $v_i$
   ◦ The Sobol’ sequence is defined as:
     $$s_n = 2^{-\ell} \bigoplus_{i=0, b_i=1}^{i=\ell-1} v_i$$
   ◦ For speed of implementation, we use Gray-code ordering
Some Types of Quasirandom Numbers (Cont.)

- Other small $D_N^*$ points sets and sequences:
  1. Halton sequence: $x_n = (\Phi_{b_1}(n - 1), \ldots, \Phi_{b_s}(n - 1))$, $n = 1, 2, \ldots$, $D_N^* = O\left(N^{-1}(\log N)^s\right)$ if $b_1, \ldots, b_s$ pairwise relatively prime
  2. Hammersley point set: $x_n =\left(\frac{n-1}{N}, \Phi_{b_1}(n - 1), \ldots, \Phi_{b_{s-1}}(n - 1)\right)$, $n = 1, 2, \ldots, N$, $D_N^* = O\left(N^{-1}(\log N)^{s-1}\right)$ if $b_1, \ldots, b_{s-1}$ are pairwise relatively prime
  3. Ergodic dynamics: $x_n = \{n\alpha\}$, where $\alpha = (\alpha_1, \ldots, \alpha_s)$ is irrational and $\alpha_1, \ldots, \alpha_s$ are linearly independent over the rationals then for almost all $\alpha \in \mathbb{R}^s$, $D_N^* = O(N^{-1}(\log N)^{s+1+\epsilon})$ for all $\epsilon > 0$
Some Types of Quasirandom Numbers (Cont.)

- $(t, m, s)$-nets and $(t, s)$-sequences and generalized Niederreiter sequences

1. Let $b \geq 2$, $s > 1$ and $0 \leq t \leq m \in \mathbb{Z}$ then a $b$-ary box, $J \subset [0, 1)^s$, is given by

\[
J = \prod_{i=1}^{s} \left[ \frac{a_i}{b^{d_i}}, \frac{a_i + 1}{b^{d_i}} \right]
\]

where $d_i \geq 0$ and the $a_i$ are $b$-ary digits, note that $|J| = b^{-\sum_{i=1}^{s} d_i}$

2. A set of $b^m$ points is a $(t, m, s)$-net if each $b$-ary box of volume $b^{t-m}$ has exactly $b^t$ points in it

3. Such $(t, m, s)$-nets can be obtained via Generalized Niederreiter sequences, in dimension $j$ of $s$: $y_{i}^{(j)}(n) = C^{(j)}a_i(n)$, where $n$ has the $b$-ary representation $n = \sum_{k=0}^{\infty} a_k(n)b^k$ and $x_{i}^{(j)}(n) = \sum_{k=1}^{m} y_{k}^{(j)}(n)q^{-k}$
A Picture is Worth a Thousand Words:
4096 Pseudorandom Pairs
SPRNG Sequence

4096 Points of SPRNG Sequence
A Picture is Worth a Thousand Words:
4096 Quasirandom Pairs

2–D Projection of Sobol’ Sequence

4096 Points of Sobol Sequence
Projects, Collaborators, and Invitations

1. Project #1: *ASCI-Hardened SPRNG*, Parallel and Distributed Pseudorandom Number Generation
   - New generators with new properties
   - New parallelizations
   - Random number testing on the web
   - Applications based testing
   - Web-based Monte Carlo integration engine
   - SPRNG support

2. Project #2: *Parallel and Distributed Quasirandom Numbers*
   - Fast implementations
   - Full periods of linear congruential generators
   - Scrambling
   - Web-based quasi-Monte Carlo integration engine
Projects, Collaborators, and Invitations

3. Project #3: *Quasi-Monte Carlo for Markov Chain Problems*
   - Modification of the Koksma-Hlawka inequality
   - Optimal number streams for Markov Chains
   - Solving linear algebra problems: systems and eigenvalues
   - Solving PDE boundary value problems
   - Best practices
   - Discrete constructions

4. Project #3: *First- and Last-Passage Monte Carlo Methods*
   - Algorithms for the Feynman-Kac formula
   - Applications to materials, biology, and chemistry
Projects, Collaborators
and Invitations (Cont.)

3. Collaborators

- David Ceperley, Lubos Mitas, Faisal Saied: NCSA
- Ashok Srinivasan, Aneta Karaivanova: FSU
- Miron Livny: Univ. of Wisconsin–Madison
- James Rathkopf, Malvin Kalos: LLNL
- Forrest Brown, Todd Urbatsch: LANL
- Roger Blomquist: Argonne NL
- Karl Sabelfeld: Weierstrass Institute, Berlin
- Nikolai Simonov: Siberian Branch of the Russian Academy of Sciences, Novosibirsk
- Alain Dubos, Olivier Smidts: Free University of Brussels
- Makoto Matsumoto: Kyoto University, Japan
- Karl Entacher, Wolfgang Schmid: University of Salzburg, Austria
- Ivan Dimov, Emanouil Atanassov: Bulgarian Academy of Sciences, Sofia
Projects, Collaborators
and Invitations (Cont.)

4. Students Wanted for These Research Areas
   ▶ Monte Carlo & Quasi-Monte Carlo methods
   ▶ Pseudo- & quasirandom Number generation
   ▶ Computational number theory & algebra
   ▶ High-performance computing
   ▶ Parallel & distributed computing, web-base computing, grid computing
   ▶ Computer security
   ▶ Applications of Monte Carlo to materials, biophysics, biochemistry, finance
   ▶ Tests of randomness