Monte Carlo Methods, I

M. D. Jones, Ph.D.

Center for Computational Research
University at Buffalo
State University of New York

High Performance Computing I, 2013

What's in a Name?

The name **Monte Carlo** refers, of course, to the Mediterranean resort (capital of Monaco), but also to a mathematical method based on sampling of random variates:

- Older than you might guess - made mainstream by work on the Manhattan project, first known use by Comte de Buffon in 1777
- Usually MC methods are reserved for **stochastic processes**, i.e. ones in which the evolution of states is determined by random events
- Nicolas Metropolis coined the term in its modern usage during the Manhattan Project
- (in)Famous for MC estimation of integrals ...

π in the Sky

Consider a circle inscribed within a square. The ratio of the area of the circle to that of the square is $\pi/4$. If we were to throw "darts" randomly into the square, the number of darts lying inside the circle relative to the total number thrown will also be $\pi/4$. It is quite easy to code this up on a computer, and in effect what we are doing is evaluating the integral,

$$\int_0^1 \int_0^{\sqrt{1-x^2}} dxdy.$$

of course we could also solve such an integral without resorting to dart-throwing ...

but in contrast to the simple estimation of the integral expression for π, suppose that we had to evaluate a partition function in statistical physics,

$$\mathcal{Z} = \int d^3\mathbf{r}_1 \ldots d^3\mathbf{r}_{N_p} e^{-\beta \sum_{i<j} v(r_{ij})},$$

which is for N_p classical particles interacting through pair potential $v(r_{ij})$.

- This integral has $3N_p$ dimensions ...
- How are you going to evaluate this integral analytically, or by finite differences (elements)?

MC Versus Deterministic Methods

So, our working description of MC methods might be:

- Solution using probabilistic methods of non-probabilistic problems
- What about radiation transport? Essentially a stochastic process in its own right, but can also be modeled very well using MC ...
- Can view it either way - "natural simulation," or solution of, say, Bethe-Bloch equation using MC methods
- So, in practice, we will use the term, "Monte Carlo" to apply to both situations, under the general umbrella of a probabilistic method
- One should note, however, that this does not preclude one from using a proper mathematical formulation - this is a key to doing MC more **efficiently**

Practical MC

- Quantum Field Theory (QCD)
- Statistical Mechanics
- Radiation Transport
  - Cancer Therapy (Dosimetry)
  - Stellar Evolution & Modeling
- Circuit Design (VLSI)
- Resource Exploration (Oil & Gas)
- Traffic Flow
- Econometrics (Derivatives Valuation & Forecasting)
- Nuclear Reactor Design

MC Advantages

Advantages of Monte Carlo:

- No timestep issues (compare with MD).
- Natural constant temperature formulation.
- Ergodicity - unlike MD, you can invent new transition rules to improve convergence.
- Discrete system - degrees of freedom need not be continuous.
- Parallelization - MC is particularly well suited to parallel computers in all their forms.

Basic Probability Recap

Given a set of random **events**, $\{E_k\}$, associated with probability p_k, $(0 \leq p_k \leq 1)$

$$P(E_k) = p_k$$

- $P\{E_i \text{ and/or } E_j\} \leq p_i + p_j$
- E_i and E_j are **mutually exclusive**:

$$P\{E_i \text{ and } E_j\} = 0$$

$$P\{E_i \text{ or } E_j\} = p_i + p_j$$

Consider compound experiment with two possible events, $\{E_i\}$ with probability p_{1i}, and $\{F_j\}$ with p_{2j}. The outcome is (E_i, F_j)

- $P\{E_i, F_j\} = p_{ij}$, the **joint probability** of E_i and F_j.
- E_i and F_j are **independent** if

$$p_{ij} = p_{1i} \cdot p_{2j}$$

- E_i and F_j are not **independent** - joint probability is

$$p_{ij} = \left(\sum_k p_{ik} \right) \left[\frac{p_{ij}}{\sum_k p_{ik}} \right] = p(i) \left[\frac{p_{ij}}{\sum_k p_{ik}} \right],$$

where $p(i)$ is the (marginal) probability for event E_i regardless of the outcome of the second event, $\sum_i p(i) = 1$ and $p(i) = p_{1i}$.

- **Conditional probability**: probability of F_j given that E_i has occurred:

$$p(j|i) = \frac{p_{ij}}{\sum_k p_{ik}}$$

  and $\sum_j p(j|i) = 1$.
- All joint probabilities can be factored into a marginal distribution and a conditional probability (can also be generalized to more than two events)

Random Variables

Properties of random variables:

- Every elementary outcome E_i is associated with a real result x_i
- Expectation (stochastic mean value):

$$\langle x \rangle = E(x) = \sum_i p_i x_i$$

- For real-valued function $g(x_i)$,

$$\langle g(x) \rangle = E(g(x)) = \sum_i p_i g(x_i)$$

- Constants λ_1, λ_2, and functions g_1, g_2:

$$\langle \lambda_1 g_1(x) + \lambda_2 g_2(x) \rangle = \lambda_1 \langle g_1 \rangle + \lambda_2 \langle g_2 \rangle$$

- n-th moment of x:

$$\langle x^n \rangle = \sum_i p_i x_i^n,$$

- **central moments** of x:

$$\langle (x-\mu)^n \rangle = \sum_i p_i (x_i - \mu)^n,$$

where $\mu = \langle x \rangle$

- The **variance** of x, $\mathrm{var}\{x\}$, is simply the second central moment of x:

$$\begin{aligned} \langle (x-\mu)^2 \rangle &= \sum_i p_i (x_i - \mu)^2, \\ &= \sum_i p_i x_i^2 - \langle x \rangle^2, \\ &= \langle x^2 \rangle - \langle x \rangle^2 \end{aligned}$$

Covariance and Correlation

- Two random variables x and y are independent if:

$$\langle xy \rangle = \langle x \rangle \langle y \rangle$$

- **Covariance**, the degree of independence of x and y

$$\mathrm{cov}\{x, y\} = \langle xy \rangle - \langle x \rangle \langle y \rangle$$

  which will be zero if x and y are independent (note also that $\mathrm{cov}\{x, x\} = \mathrm{var}\{x\}$, and that covariance can be positive or negative)

- **Correlation coefficient**, $\rho(x, y)$

$$\rho(x, y) = \frac{\mathrm{cov\{x,y\}}}{[\mathrm{var}\{x\}\mathrm{var}\{y\}]^{1/2}}$$

  and $-1 \leq \rho \leq 1$. Negative correlation can be used in MC for variance reduction.

Continuous Random Variables

Variables are not always discreet, of course, in the continuous case we get **probability distribution functions** or cumulative distribution functions,

$$F(x) = P(\text{random selection of X gives value less than } x),$$

for intervals in which $F(x)$ is differentiable we get a **probability density function**, or **pdf**,

$$f(x) = dF(x)/dx \geq 0,$$

if F is not continuous, the discontinuous discrete values are singled out (formally using Dirac delta functions). Multivariate distributions,

$$F(x, y) = P(X \leq x, Y \leq y),$$
$$f(x, y) = \partial^2 F(x, y)/\partial x \partial y,$$

and properties in the continuous formulation are a logical extension of the discrete case:

$$\langle x \rangle = E(x) = \int_{-\infty}^{\infty} x f(x) dx,$$

$$\langle g(x) \rangle = E(g(x)) = \int_{-\infty}^{\infty} g(x) f(x) dx,$$

$$f(y|x) = f(x,y) / \int f(x,y) dy,$$

$$E(y|x) = \int y f(y|x) dy = \int y f(x,y) dy / \left(\int f(x,y') dy' \right).$$

Monte Carlo Quadrature

We have taken the long road to approach MC integration, no suppose that we draw random variables x_i from pdf $f(x)$, and then we construct the "estimator," G,

$$G = \frac{1}{N}\sum_{n=1}^{N} g_n(x_n),$$

with expectation and variance:

$$\begin{aligned} E(G) &= E\left(\frac{1}{N}\sum_{n=1}^{N} g_n(x_n)\right), \\ &= \frac{1}{N}\sum_{n=1}^{N}\langle g(x)\rangle, \\ &= \langle g(x)\rangle, \\ var\{G\} &= \frac{1}{N} var\{g(x)\}. \end{aligned}$$

Hence the expectation of our estimator has the same mean as $g(x)$, and the variance of this mean decreases like $1/N$,

$$\begin{aligned}\langle g(x)\rangle &= \int_{-\infty}^{\infty} g(x)f(x)dx \\ &= E\left(\frac{1}{N}\sum_{n=1}^{N} g(x_n)\right),\end{aligned}$$

where $\{x_n\}$ represents a series of random variables drawn from pdf $f(x)$.

How Random is Random?

Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.

John Von Neumann, 1951

- Digital computers do not (one would hope!) produce truly "random" numbers.
- A "good" random number generator (RNG) should produce a known probability distribution function (pdf) (usually uniform over a finite interval), with no trends, biases, short periodicities, or undesired correlations between subsequent random numbers.

How Random is Random? (cont'd)

Somewhat more amusing insights into random number generation:

http://www.random.org/analysis

http://xkcd.com/221

We should be able to settle on a more quantifiable definition of "random."

RNG Vocabulary

Are random numbers, in fact, truly random?

- **Random** - canonical example is radioactive decay. “Truly random.”
- **Pseudo-Random** - “appears” to be random, but is, in fact, repeatable and predictable.
- **Quasi-Random** - fills interval sequentially (i.e. 0,1,2,3,...100 or 100,99,98,...0). A preset level of granularity.

Desirable RNG Properties

Properties of a “good” random number generator:

- **Uncorrelated** - sequences of RNs should be independent of one another. A common failing of poor RNGs is that *n*-tuples of individual RNs are correlated.
- **Very Long Periodicity** - the repeat length of a RNG should be as long as possible (another common failing - the repeat length is much shorter than that of the algorithm used in the RNG).
- **Efficient** - should not consume a sizable fraction of the compute time (few percent at most). Low overhead and minimal interprocessor communication.

Evaluating RNGs

Methods for evaluation/testing of RNGs:

- **Knuth's Empirical Tests**, from volume 2 of *The Art of Computer Programming*, by Donald E. Knuth (Addison, Reading, 1981). A battery of statistical tests.
- **Application Dependent**, use two very different RNGs in your application, if the results differ, the RNGs are suspect.

Linear Congruential RNG

Description of the linear congruential generators:

- Definition of the Linear Congruential Generator (LCG),

$$X_{i+1} = f(X_i) = (a * X_i + c) \mod m$$
$$a, c, m, X \in \mathcal{Z}$$

- Characterized: LCG(a, c, m, X_0), using X_0 as the **seed**.
- Period: m.
- Uniform RNs on the unit interval:

$$R_i = \frac{X_i}{m-1}, R_i \in [0, 1]$$

Downside of the LCG RNG

- Advantages:
  - Simple to program,
  - Efficient,
- Disadvantages:
  - Low-order bits are not very random,
  - k-tuples lie on $(k-1)$ dimensional (hyper)planes, with **at most** $m^{1/k}$ such planes. An improper choice of the parameters a, b, c, m leads to **many** fewer planes.

LCG Example 1

- Consider LCG$(6, 0, 13)$, which produces the following sequence

i	X_i	X_{i+1}
0	1	7
1	7	3
2	3	5
3	5	4
4	4	11
5	11	1

  which has a period considerably shorter than $m = 13$.

LCG Example 1 (cont'd)

- LCG$(5, 0, 13)$ is even worse,

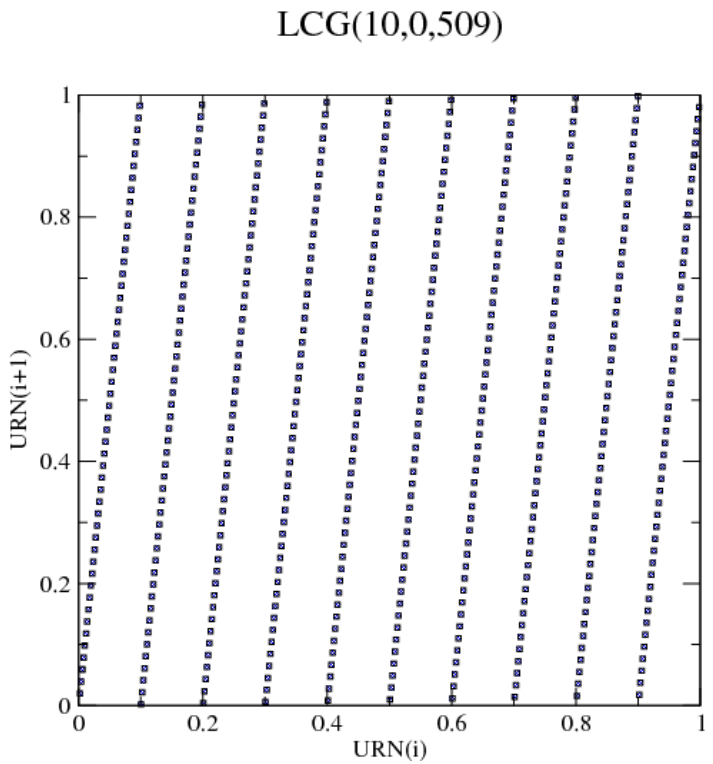
i	X_i	X_{i+1}
0	1	8
1	8	1

which has a period of only 2.

LCG Granularity

Granularity problems in the LCG generators:

- The n-tuple problem: n-tuples lie on $(n-1)$ (hyper)planes, with a maximum of $m^{1/n}$ such planes.
- Consider LCG$(10, 0, 509)$ and 2-tuples.



Minimal Standard LCG

- "Minimal standard" LCG, proposed by Park & Miller

  [S. K. Park and K. W. Miller, Comm. of ACM **31**, 1192 (1988)]

- LCG$(16807, 0, 2^{31} - 1)$
- Easily implemented using Schrage's algorithm to prevent integer overflow:

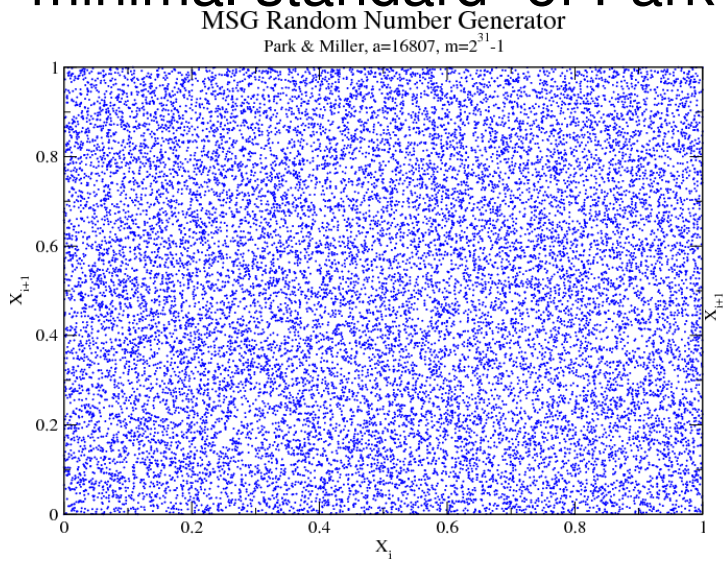
$$(a * X) \bmod m = \begin{cases} a(X \bmod q) - r[X/q] & \text{if it is} \geq 0, \\ a(X \bmod q) - r[X/q] + m & \text{otherwise.} \end{cases}$$

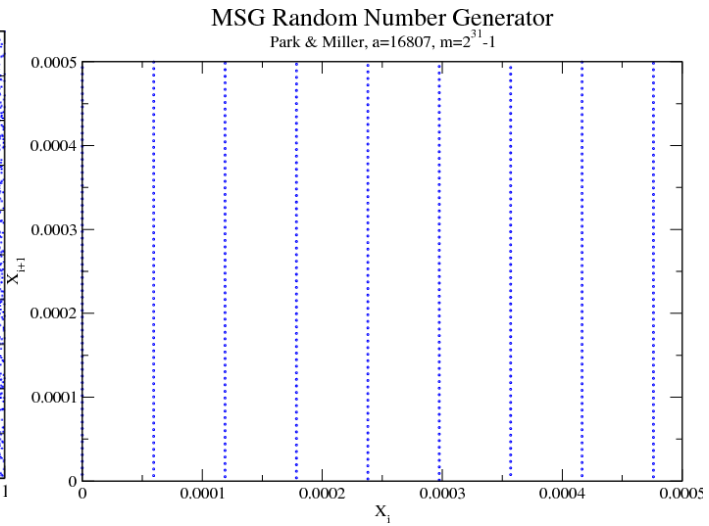
  where $q = m/a$, $r = m\%a$.

- OK for "small" simulations, but limited to 32-bit period.

MS-LCG Granularity

- Granularity is an unfortunate part of any LCG; consider even the "minimal standard" of Park & Miller.





An (In)famous Example of an LCG

The RANDU LCG was **widely** used in 1960s and 1970s:

- LCG(65539,0,2^{31},X_0)
- $X_{k+2} = (2^{16} + 3)^2 X_k = 6X_{k+1} - 9X_k$
- Result is that 3-tuples fall on 15 planes (c.f. G. Marsaglia, Proc. Nat. Acad. Sci. **61**, 25-28 (1968)).

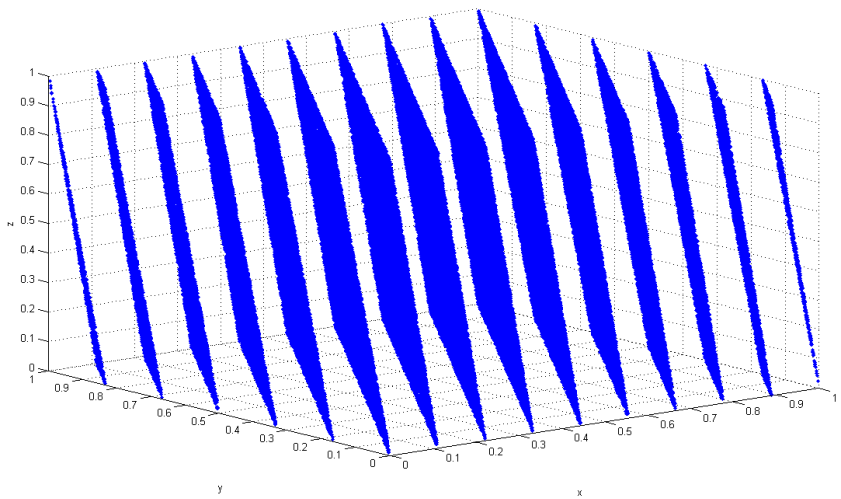


Image courtesy Wikipedia commons.

Lagged Fibonacci RNG

Another popular choice for RNGs is the lagged Fibonacci:

- Definition of the Lagged Fibonacci Generator (LFG),

$$X_i = (X_{i-l} + X_{i-k}) \mod m,$$
$$l > k > 0 \qquad l, k, m \in \mathcal{Z}$$

- Recall classic Fibonacci sequence, $X_n = X_{n-1} + X_{n-2}$.
- Characterized: LFG(l, k, M), where in most applications $m = 2^M$. $X_0, X_1, \ldots, X_{l-1}$ needed to obtain next element in the sequence.
- Period: $(2^l - 1)2^{M-1}$.

LFG in Practice

- LFGs have recently become popular; tests by Marsaglia (1985) reveal only a weakness in the "Birthday Spacings Test" for low k and l values.

  [G. Marsaglia, in *Computer Science and Statistics: The Interface*, ed. by L. Billard (Elsevier, North Holland, 1985).]

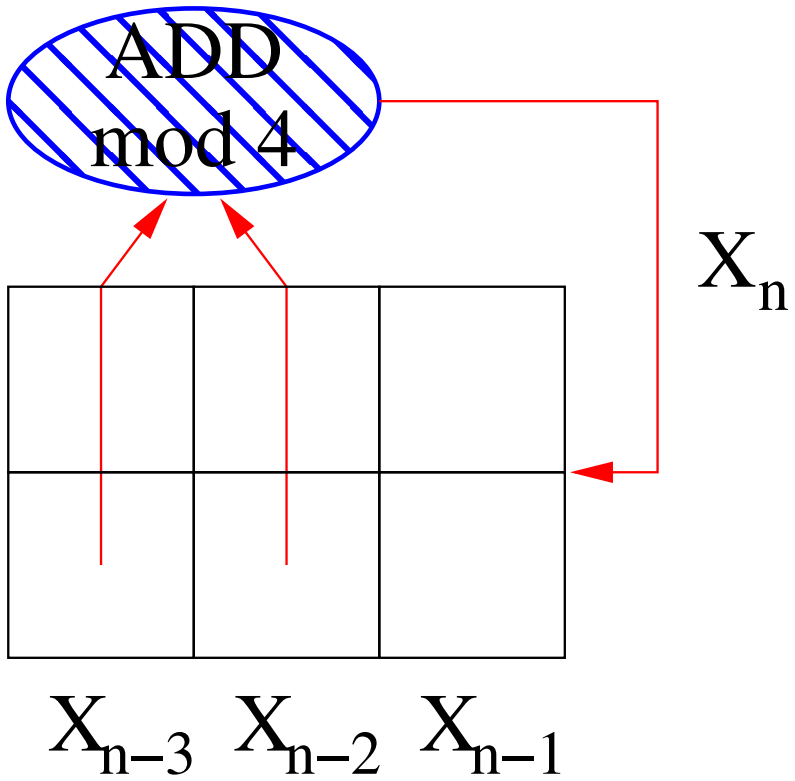
- Computationally Simple: integer add, logical AND (for $m = 2^M$), decrement two array pointers.
- Drawback: Have to maintain l words of memory, rather than a single integer in the case of a LCG.

LFG in Theory

- LFG(I,k,M) acts like a linear shift register (LSR), with M bits.
- Period, P, of the $M \times I$ rectangular register bit pattern is $(2^I - 1)2^{M-1}$, but there are $2^{M \times I} >> P$ such patterns. Result is many **independent P-long cycles**. How many?
- $2^{(I-1)(M-1)}$ independent P-cycles.
- Lends itself to many parallel RN streams!

LFG Example

- Consider LFG(3,2,2).
  - Acts like a linear shift register (LSR) with 2-bit words.
  - Period $P = 14$, 8 independent cycles.



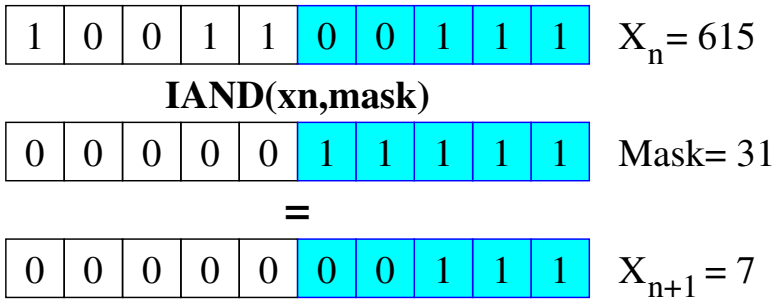
LFG Comments

- Steadily becoming more popular (with the exception of MT ...)
- Somewhat more complex than LCGs, but offer many desirable properties.
- Initialization of an LFG is tricky, and the RNG quality is sensitive to the initial state
- True test is in the simulations, which are ongoing.

Bit Masks for Mod Arithmetic

Bit masks for mod arithmetic:

- Useful for both LCGs and LFGs (but most useful for LFGs).
- Performing Modulo 2^M for ($M = 5$),



- Much faster than integer division.

Mersenne Twister

For something rather different, we have the Mersenne Twister:

- M. Matsumoto and T. Nishimura, ACM Trans. Model. Comput. Simul. **8**, 3 (1998)
- based on matrix linear-recurrence, period chosen to be a Mersenne prime (hence the name - recall that a Mersenne prime is a prime that is $2^m - 1$)
- Classed as a twisted generalized feedback shift register
- Most common form is `MT19937` and its 64-bit variant, `MT19937-64`, which has period $2^{19937} - 1$.
- Excellent statistical test record, reasonably fast, many implementations available
- Some disadvantages - complex to implement relative to other types of pRNGs

General Advice on RNGs

Some general advice for RNGs:

- Use a “trusted” RNG with a long period that produces a reproducible stream (**NOT** the one built into your compiler, OS, etc.).
- There is no golden bullet for RNGs. A reasonable LCG is the “minimal standard.” (period is likely too short for large MC simulation problems)

  [S. K. Park and K. W. Miller, Comm. of ACM **31**, 1192 (1988)]

(Not Quite) Last Word on RNGs

Some last words on RNGs:

- Can use statistical tests for RNG evaluation & performance (exemplified by full battery in Knuth, NIST suite, or `DIEHARD`)
- Simulation test - use two very different RNGs in your application, and make sure that you get a consistent answer!
- "Trust, but verify ..." the old cold-war axiom actually applies equally well (except perhaps for the trust part) to pseudo-RNGs:

`http://www.random.org/analysis/`

> *One of us recalls producing a "random" plot with only 11 planes, and being told by his computer center's programming consultant that he had misused the random number generator: "We guarantee that each number is random individually, but we don't guarantee that more than one of them is random."*

Numerical Recipes in C; the Art of Scientific Computing, W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling (Cambridge, New York, 1988), p. 207.

GNU Scientific Library

The *GNU Scientific Library* (GSL) supports a range of RNG algorithms:

www.gnu.org/software/gsl

(in addition to other numerical routines, of course). Also includes functions for sample random variates from specific distributions (e.g. Gaussian, Poisson, etc.)

GSL RNG Algorithms

Brief listing of recommended RNGs in the GSL (publication references for all contained within GSL):

Algorithm	Comment
CMRG	combined multiple recursive of L'Ecuyer
GSFR4	LFG of length 4, by Ziff
MRG	5th order multiple recursive of L'Ecuyer, Blouin, and Coutre
MT19937	"Mersenne Twister" of Matsumoto and Nishimura
RANLUX	*ranlux* by Lüscher, LFG with skipping ("luxury" RNs)
RANLUX389	as above, but with all 24 bits decorrelated
RANLXS0	2nd generation of *ranlux* by Lüscher, single (24 bit) output
RANLXS1	as above, increasing order of strength
RANLXS2	as above, increasing order of strength
RANLXD1	as *RANLXS1*, modified for 48 bit (double) output
RANLXD2	as above, increasing order of strength
TAUS	combined Tausworth generator by L'Ecuyer

The GSL also provides a number of RNGs for backwards compatibility (not that you would want to use them, but for testing older code before moving on to RNGs with more modern established "randomness" credentials):

Routine	Comment
RANF	48 bit LCG from old CRAY MATHLIB
MINSTD	32 bit "Minimal Standard" of Park and Miller
RANDU	infamous IBM LCG, period of 2^{29}, exemplifies poor RNG
VAX	DEC VAX 32 bit LCG *MTH$RANDOM*

(above is only a small sampling of available deprecated RNGs)

MATLAB reportedly uses the minimal standard LCG of Park and Miller.

Parallel RNGs

There was an ASCI project specifically devoted to parallel RNGs.

- `SPRNG` - the Scalable Parallel Random Number Generators Library.

  `http://sprng.cs.fsu.edu/`

- Includes LCGs, LFGs, Others (combined multiple recursive, modified/multiplicative LFGs)
- `FORTRAN, C` and `C++` Interfaces.
- Explicit support for independent concurrent streams of RNs (and established lack of correlation between them)

I Canna Give Ye Any More Speed, Captain!

Speed of RNGs can be an issue - you don't want the RNG to become the bottleneck in a MC simulation. Some representative timings for the `GSL` and `SPRNG` RNGs:

RNG	Precision	MRS (10^6 RN/s)
`SPRNG-LCG64`	double	59
	single	48
	integer	167
`SPRNG-MLFG`	double	71
	single	48
	integer	83
`GSL-RANLUX`	integer	8.5
`GSL-RANLXD2`	integer	2.3
`GSL-MT19937`	integer	65

More RNG References

Some good starting points for testing RNGs:

- `DIEHARD` random number test suite of G. Marsaglia,

  http://stat.fsu.edu/pub/diehard

- `NIST` test suite,

  http://csrc.nist.gov/rng

- Donald E. Knuth, *The Art of Computer Programming: Seminumerical Algorithms*, Vol 2, 3rd Ed. (Addison-Wesley, New York, 1997) ISBN 0201896842.

The Knuth reference is simply excellent overall.

Transformation Method

What if we need RNs that are not uniform in the interval $[0, 1]$ (as provided by most pRNGs)?
Suppose that $x \in \mathrm{URN}[0, 1]$ is a uniform random variate, and we want to sample a function of x, say $y(x)$,

$$p(y) = p(x) \left| \frac{dx}{dy} \right|,$$

For $p(y) = f(y)$ positive and normalized this is simple to transform,

$$y(x) = F^{-1}(x)$$

where F is the indefinite integral of f. This approach is only practical when one can find F^{-1}.

Example 1: Simple Transformation

Suppose that u is a uniform random variate on the unit interval, and we want our sampling to be distributed like $\alpha e^{-\alpha x}$. So $f(x) = \alpha e^{-\alpha x}$, and the inversion is particularly simple in this case:

- $F = 1 - e^{-\alpha x}$ (indefinite integral, chosen to be non-negative and normalized)
- Solve $F(x) = u$ for x, $x = -\ln(1-u)/\alpha$
- (Note that $1 - u$ is also a uniform random variate on $(0, 1]$, so we could equally well take samples as $x = -\ln(u)/\alpha$)

Example: Box-Muller Method for Normal Deviates

The transformation method can be applied in multiple dimensions, of course - one of the most common cases is that used to produce normal (i.e. Gaussian) random variates (here u_1 and u_2 are uniform random numbers on the unit interval):

$$p(y)dy = \frac{1}{\sqrt{2\pi}} e^{-y^2/2} dy,$$

Note that p is not easily inverted, so the Box-Muller method uses the two dimensional equivalent (which is invertible by switching to polar coordinates) to produce pairs of Gaussian random variates,

$$y_1 = \sqrt{-2 \ln u_1} \cos(2\pi u_1), \qquad u_1 = \exp[-(y_1^2 + y_2^2)/2],$$

$$y_2 = \sqrt{-2 \ln u_2} \cos(2\pi u_2), \qquad u_2 = \frac{\tan^{-1}(y_2/y_1)}{2\pi},$$

Rejection Method

The rejection method is quite general and does not require us to know how to compute the cumulative distribution function, $\int p(x)dx$, only that $p(x)dx$ is known and computable.

- Choose known distribution $q(x) > f(x)$ that bounds $f(x)$
- Sample x from $q(x)$ and a uniform deviate u
- If $u < p(x)/q(x)$ accept x as representative of $p(x)$, otherwise reject and repeat

Useful for sampling binomial, Poisson, other distributions, and forms the basis for the very general Metropolis method (discussed later).

Normal Deviates by Rejection

Rejection can be used to produce Gaussian random numbers - consider the "polar" form of Marsaglia (possibly older):

- Generate uniform random numbers v_1, v_2 in the unit interval
- Form $R^2 = v_1^2 + v_2^2$, if $R = 0$ or $R > 1$ reject and sample again
- Resulting accepted R values used for u_1 in usual Box-Muller, while angle of (v_1, v_2) with respect to the v_1 axis serves for u_2 (so $\cos(2\pi u_2) = v_1/R$ and $\sin(2\pi u_2) = v_2/R$
- Cost: 2 multiplies, 1 add, 1 logarithm, 1 division, and 1 square root (compared with 3 multiplies, 1 logarithm, 1 trigonometric, and 1 square root)