Monte Carlo Methods for Uncertainty Quantification

Mike Giles
Mathematical Institute, University of Oxford

KU Leuven Summer School on Uncertainty Quantification
May 30–31, 2013

Lecture outline

Lecture 1: Monte Carlo basics
- random number generation
- Monte Carlo estimation
- Law of Large Numbers and confidence interval
- basic mean/variance manipulations
- antithetic sampling
- control variate

Lecture 2: Variance reduction
- importance sampling
- stratified sampling
- Latin Hypercube
- randomised quasi-Monte Carlo

Lecture 3: financial applications
- financial models
- approximating SDEs
- weak and strong convergence
- mean square error decomposition
- multilevel Monte Carlo

Lecture 4: PDE applications
- PDEs with uncertainty
- examples
- multilevel Monte Carlo

Random Number Generation

Monte Carlo simulation starts with random number generation, usually split into 2 stages:
- generation of independent uniform (0, 1) random variables
- conversion into random variables with a particular distribution (e.g. Normal)

Very important: never write your own generator, always use a well validated generator from a reputable source
- Matlab
- NAG
- Intel MKL
- AMD ACML
Uniform Random Variables

Pseudo-random number generators use a deterministic (i.e. repeatable) algorithm to generate a sequence of (apparently) random numbers on (0, 1) interval.

What defines a good generator?

- a long period – how long it takes before the sequence repeats itself. $2^{32}$ is not enough – need at least $2^{40}$.
- various statistical tests to measure “randomness”. Well validated software will have gone through these checks.
- trivially-parallel Monte Carlo simulation on a compute cluster requires the ability to “skip-ahead” to an arbitrary starting point in the sequence.

First computer gets first $10^6$ numbers, second computer gets second $10^6$ numbers, etc.

Normal Random Variables

$N(0, 1)$ Normal random variables (mean 0, variance 1) have the probability distribution

$$p(x) = \phi(x) \equiv \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right)$$

The Box-Muller method takes two independent uniform (0, 1) random numbers $y_1, y_2$, and defines

$$x_1 = \sqrt{-2\log(y_1)} \cos(2\pi y_2)$$
$$x_2 = \sqrt{-2\log(y_1)} \sin(2\pi y_2)$$

It can be proved that $x_1$ and $x_2$ are $N(0, 1)$ random variables, and independent:

$$p_{\text{joint}}(x_1, x_2) = p(x_1) p(x_2)$$

Inverse CDF

A more flexible alternative uses the cumulative distribution function $CDF(x)$ for a random variable $X$, defined as

$$CDF(x) = \mathbb{P}(X < x)$$

If $Y$ is a uniform (0, 1) random variable, then can define $X$ by

$$X = CDF^{-1}(Y).$$

For $N(0, 1)$ Normal random variables,

$$CDF(x) = \Phi(x) \equiv \int_{-\infty}^{x} \phi(s) \, ds = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{1}{2}s^2\right) \, ds$$

$\Phi^{-1}(y)$ is approximated in software in a very similar way to the implementation of $\cos, \sin, \log$.
Normal Random Variables

The Normal CDF \( \Phi(x) \) is related to the error function \( \text{erf}(x) \) through

\[
\Phi(x) = \frac{1}{2} + \frac{1}{2} \text{erf}(x/\sqrt{2}) \quad \Rightarrow \quad \Phi^{-1}(y) = \sqrt{2} \text{erf}^{-1}(2y-1)
\]

This is the function I use in Matlab:

```matlab
% x = ncfinv(y)
% inverse Normal CDF
function x = ncfinv(y)
x = sqrt(2)*erfinv(2*y-1);
```

Correlated Normal Random Variables

We often need a vector \( y \) of Normally distributed variables with a prescribed covariance matrix, so that \( E[y y^T] = \Sigma \).

Suppose \( x \) is a vector of independent \( N(0,1) \) variables, and define \( y = Lx \).

Each element of \( y \) is Normally distributed, \( E[y] = L E[x] = 0 \), and

\[
E[y y^T] = E[L x x^T L^T] = L E[x x^T] L^T = LL^T
\]

since \( E[x x^T] = I \) because

- elements of \( x \) are independent \( \Rightarrow E[x_i x_j] = 0 \) for \( i \neq j \)
- elements of \( x \) have unit variance \( \Rightarrow E[x_i^2] = 1 \)

Hence choose \( L \) so that \( LL^T = \Sigma \).
Correlated Normal Random Variables

One choice is a Cholesky factorisation in which $L$ is lower-triangular.

Alternatively, if $\Sigma$ has eigenvalues $\lambda_i \geq 0$, and orthonormal eigenvectors $u_i$, so that

$$\Sigma u_i = \lambda_i u_i, \quad \Rightarrow \quad \Sigma U = U \Lambda$$

then

$$\Sigma = U \Lambda U^T = L L^T$$

where

$$L = U \Lambda^{1/2}.$$ 

This is the PCA decomposition; it is no better than the Cholesky decomposition for standard Monte Carlo simulation, but is often better for stratified sampling and quasi-Monte Carlo methods.

Expectation and Integration

GetRapid assumes that $X$ is a random variable uniformly distributed on $[0,1]$. Then the expectation of a function $f(X)$ is equal to its integral:

$$\mathbb{E}[f(X)] = \int_0^1 f(x) \, dx.$$ 

The generalisation to a $d$-dimensional “cube” $I^d = [0,1]^d$, is

$$\mathbb{E}[f(X)] = \int_{I^d} f(x) \, dx.$$ 

Thus the problem of finding expectations is directly connected to the problem of numerical quadrature (integration), often in very large dimensions.

In general, define

-error $\varepsilon_N(f) = I[f] - I_N[f]$

bias $= \mathbb{E}[\varepsilon_N(f)]$

RMSE, “root-mean-square-error” $= \sqrt{\mathbb{E}[(\varepsilon_N(f))^2]}$

The Central Limit Theorem proves (roughly speaking) that for large $N$

$$\varepsilon_N(f) \sim \sigma \sqrt{N^{-1/2}} Z$$

with $Z$ a $N(0,1)$ random variable and $\sigma^2$ the variance of $f$:

$$\sigma^2 = \mathbb{E}[(f - \mathbb{E}[f])^2] = \int_{I^d} (f(x) - \mathbb{E}[f])^2 \, dx.$$
More precisely, provided $\sigma$ is finite, then as $N \to \infty$,

$$
\text{CDF}(N^{1/2} \sigma^{-1} \varepsilon_N) \to \text{CDF}(Z)
$$

so that

$$
P\left[N^{1/2} \sigma^{-1} \varepsilon_N < s\right] \to P[Z < s] = \Phi(s)
$$

and

$$
P\left[N^{1/2} \sigma^{-1} \varepsilon_N \geq s\right] \to P[|Z| > s] = 2 \Phi(-s)
$$

$$
P\left[N^{1/2} \sigma^{-1} \varepsilon_N < s\right] \to P[|Z| < s] = 1 - 2 \Phi(-s)
$$

Given $N$ samples, the empirical variance is

$$
\tilde{\sigma}^2 = N^{-1} \sum_{n=1}^{N} (f(x_n) - I_N)^2 = I_N^{(2)} - (I_N)^2
$$

where

$$
I_N = N^{-1} \sum_{n=1}^{N} f(x_n), \quad I_N^{(2)} = N^{-1} \sum_{n=1}^{N} (f(x_n))^2
$$

$\tilde{\sigma}^2$ is a slightly biased estimator for $\sigma^2$; an unbiased estimator is

$$
\hat{\sigma}^2 = (N-1)^{-1} \sum_{n=1}^{N} (f(x_n) - I_N)^2 = \frac{N}{N-1} \left( I_N^{(2)} - (I_N)^2 \right)
$$

How many samples do we need for an accuracy of $\varepsilon$ with probability $c$?

Since

$$
P\left[N^{1/2} \sigma^{-1} |c| < s\right] \approx 1 - 2 \Phi(-s),
$$

define $s$ so

$$
1 - 2 \Phi(-s) = c \iff s = -\Phi^{-1}((1-c)/2)
$$

<table>
<thead>
<tr>
<th>$c$</th>
<th>$0.683$</th>
<th>$0.9545$</th>
<th>$0.9973$</th>
<th>$0.99994$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s$</td>
<td>$1.0$</td>
<td>$2.0$</td>
<td>$3.0$</td>
<td>$4.0$</td>
</tr>
</tbody>
</table>

Then $|c| < N^{-1/2} \sigma s$ with probability $c$, so to get $|c| < \varepsilon$ we can put

$$
N^{-1/2} \hat{s}(c) = \varepsilon \implies N = \left( \frac{\hat{s}(c)}{\varepsilon} \right)^2.
$$

Note: twice as much accuracy requires 4 times as many samples.
Finance Applications

Geometric Brownian motion for a single asset:

\[ S_T = S_0 \exp \left( (r - \frac{1}{2}\sigma^2)T + \sigma W_T \right) \]

\( W_T \) is \( N(0, T) \) random variable, so can put

\[ W_T = \sqrt{T} \, Y = \sqrt{T} \, \Phi^{-1}(U) \]

where \( Y \) is a \( N(0, 1) \) r.v. and \( U \) is a uniform \((0, 1)\) r.v.

We are then interested in the price of financial options which can be expressed as

\[ V = \mathbb{E}[f(S(T))] = \int_{0}^{1} f(S(T)) \, dU, \]

for some “payoff” function \( f(S) \).

Finance Applications

For the European call option,

\[ f(S) = \exp(-rT) \, (S - K)^+ \]

while for the European put option

\[ f(S) = \exp(-rT) \, (K - S)^+ \]

where \( K \) is the strike price, and \((y)^+ \equiv \max(0, y)\).

For numerical experiments we will consider a European call with \( r=0.05, \, \sigma = 0.2, \, T = 1, \, S_0 = 110, \, K = 100.\)

The analytic value is known for comparison.

Finance Applications

MC calculation with up to \( 10^6 \) paths; true value = 17.663
The upper and lower bounds are given by

$$\text{Mean} \pm \frac{3 \tilde{\sigma}}{\sqrt{N}},$$

so more than a 99.7% probability that the true value lies within these bounds.

MATLAB code:

```matlab
r=0.05; sig=0.2; T=1; S0=110; K=100;
N = 1:1000000;
U = rand(1,max(N)); % uniform random variable
Y = ncfinv(U); % inverts Normal cum. fn.
S = S0*exp((r-sig^2/2)*T + sig*sqrt(T)*Y);
F = exp(-r*T)*max(0,S-K);
sum1 = cumsum(F); % cumulative summation of
sum2 = cumsum(F.^2); % payoff and its square
val = sum1./N;
rms = sqrt(sum2./N - val.^2);
```

err = european_call(r,sig,T,S0,K,'value') - val;
plot(N,err, ...
N,err-3*rms./sqrt(N), ...
axis([0 length(N) -1 1])
xlabel('N'); ylabel('Error')
legend('MC error','lower bound','upper bound')
The payoff is
\[ f = \exp(-rT) \left( \frac{1}{M} \sum_i S_i - K \right)^+ \]
and so the expectation can be written as the \( M \)-dimensional integral
\[ \int_M f(U) \, dU. \]

This is a good example for Monte Carlo simulation – cost scales linearly with the number of stocks, whereas it would be exponential for grid-based numerical integration.

MATLAB code:
```matlab
r=0.05; sig=0.2; T=1; S0=110; K=100;
Sigma = sig^2*T*( eye(5) + 0.1*(ones(5)-eye(5)));
L = chol(Sigma,'lower');
N = 1:1000000;
U = rand(5,max(N)); % uniform random variable
Y = ncfinv(U); % inverts Normal cum. fn.
S = S0*exp((r-sig^2/2)*T + L*Y);
F = exp(-r*T)*max(0,sum(S,1)/5-K);
sum1 = cumsum(F); % cumulative summation of
sum2 = cumsum(F.^2); % payoff and its square
val = sum1./N;
rms = sqrt(sum2./N - val.^2);
```

Summary so far

- Monte Carlo quadrature is straightforward and robust
- confidence bounds can be obtained as part of the calculation
- can calculate the number of samples \( N \) needed for chosen accuracy
- much more efficient than grid-based methods for high dimensions
- accuracy = \( O(N^{-1/2}) \), CPU time = \( O(N) \)
  \[ \implies \text{accuracy} = O(\text{CPU time}^{-1/2}) \]
  \[ \implies \text{CPU time} = O(\text{accuracy}^{-2}) \]
- the key now is to reduce number of samples required by reducing the variance – antithetic variables and control variates in this lecture
Elementary Manipulations

If \( X_1 \) and \( X_2 \) are independent continuous random variables, then
\[
p_{\text{joint}}(x_1, x_2) = p_1(x_1) p_2(x_2)
\]
and hence
\[
E[f_1(X_1) f_2(X_2)] = \int \int f_1(x_1) f_2(x_2) p_{\text{joint}}(x_1, x_2) \, dx_1 \, dx_2
= \int \int f_1(x_1) f_2(x_2) p_1(x_1) p_2(x_2) \, dx_1 \, dx_2
= \left( \int f_1(x_1) p_1(x_1) \, dx_1 \right) \left( \int f_2(x_2) p_2(x_2) \, dx_2 \right)
= E[f_1(X_1)] E[f_2(X_2)]
\]
and in particular
\[
\text{Cov}[X_1, X_2] = E \left[ (X_1 - E[X_1]) (X_2 - E[X_2]) \right]
= E[X_1 - E[X_1]] E[X_2 - E[X_2]] = 0
\]

Elementary Manipulations

In addition,
\[
\text{V}[a + \mu] = \text{V}[a] + 2 \text{Cov}[a, b] + \text{V}[b]
\]
where
\[
\text{Cov}[a, b] \equiv E[(a - E[a]) (b - E[b])]
\]

Since
\[
|\text{Cov}[a, b]| \leq \sqrt{\text{V}[a]} \sqrt{\text{V}[b]}
\]
it follows that
\[
\text{V}[a + b] \leq \left( \sqrt{\text{V}[a]} + \sqrt{\text{V}[b]} \right)^2
\]
\[
\implies \sqrt{\text{V}[a + b]} \leq \sqrt{\text{V}[a]} + \sqrt{\text{V}[b]}
\]

If \( a, b \) are independent then \( \text{V}[a + b] = \text{V}[a] + \text{V}[b] \), and more generally the variance of a sum of independents is equal to the sum of their variances.

Antithetic variables

The simple estimator for \( E[f(X)] \) from the last lecture has the form
\[
N^{-1} \sum_i f(X(i))
\]
where \( X(i) \) is the \( i^{th} \) independent sample of the random variable \( X \).

If \( X \) has a symmetric probability distribution, \(-X\) is just as likely. Antithetic estimator replaces \( f(X(i)) \) by
\[
\overline{T}(i) = \frac{1}{2} \left( f(X(i)) + f(-X(i)) \right)
\]
Clearly still unbiased since
\[
E[\overline{T}] = \frac{1}{2} \left( E[f(X)] + E[f(-X)] \right) = E[f(X)]
\]
**Antithetic variables**

The variance is given by

\[ \Var[f] = \frac{1}{4} \left( \Var[f(X)] + 2 \Cov[f(X), f(-X)] + \Var[f(-X)] \right) \]

The variance is always reduced, but the cost is almost doubled, so net benefit only if \( \Cov[f(X), f(-X)] < 0 \).

Two extremes:
- A linear payoff, \( f = a + bX \), is integrated exactly since \( f = a \) and \( \Cov[f(X), f(-X)] = -\Var[f] \)
- A symmetric payoff \( f(X) = f(-X) \) is the worst case since \( \Cov[f(X), f(-X)] = \Var[f] \)

General assessment – usually not very helpful, but can be good in particular cases where the payoff is nearly linear.

**Control Variates**

Suppose we want to estimate \( \mathbb{E}[f(X)] \), and there is another function \( g(X) \) for which we know \( \mathbb{E}[g(X)] \).

We can use this by defining a new estimator

\[ \hat{f} = f - \lambda (g - \mathbb{E}[g]) \]

Again unbiased since \( \mathbb{E}[\hat{f}] = \mathbb{E}[f] = \mathbb{E}[f] \)

The resulting variance is

\[ N^{-1} \Var[f] \left( 1 - \frac{(\Cov[f,g])^2}{\Var[f] \Var[g]} \right) = N^{-1} \Var[f] (1 - \rho^2) \]

where \(-1 < \rho < 1\) is the correlation between \( f \) and \( g \).

The challenge is to choose a good \( g \) which is well correlated with \( f \). The covariance, and hence the optimal \( \lambda \), can be estimated numerically.