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 2: Variance reduction

- importance sampling
- stratified sampling
- Latin Hypercube
- randomised quasi-Monte Carlo

Importance Sampling

Importance sampling involves a change of probability measure.

Instead of taking X from a distribution with p.d.f. $p_1(X)$, we instead take it from a different distribution with p.d.f. $p_2(X)$.

$$\begin{aligned}\mathbb{E}_1[f(X)] &= \int f(X)\, p_1(X)\, \mathrm{d}X \\ &= \int f(X)\, \frac{p_1(X)}{p_2(X)}\, p_2(X)\, \mathrm{d}X \\ &= \mathbb{E}_2[f(X)\, R(X)]\end{aligned}$$

where $R(X) = p_1(X)/p_2(X)$ is the Radon-Nikodym derivative.

Importance Sampling

We want the new variance $\mathbb{V}_2[f(X)\, R(X)]$ to be smaller than the old variance $\mathbb{V}_1[f(X)]$.

How do we achieve this? Ideal is to make $f(X)R(X)$ constant, so its variance is zero.

More practically, make $R(X)$ small where $f(X)$ is large, and make $R(X)$ large where $f(X)$ is small.

Small $R(X) \Longleftrightarrow$ large $p_2(X)$ relative to $p_1(X)$, so more random samples in region where $f(X)$ is large.

Particularly important for rare event simulation where $f(X)$ is zero almost everywhere.

Importance Sampling

Simple example: want to estimate $\mathbb{E}[X^8]$ when X is a $N(0,1)$ Normal random variable.

Here

$$p_1(x) = \phi(x) \equiv \frac{1}{\sqrt{2\pi}} \exp(-\tfrac{1}{2}x^2)$$

but what should we choose for $p_2(x)$?

Want more samples in extreme "tails", so instead take samples from a $N(0,\sigma^2)$ distribution with $\sigma > 1$:

$$p_2(x) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp(-\tfrac{1}{2}x^2/\sigma^2)$$

Importance Sampling

The Radon-Nikodym derivative is

$$\begin{aligned} R(X) &= \exp\left(-\frac{x^2}{2}\right) \;/\; \frac{1}{\sigma_2}\exp\left(-\frac{x^2}{2\,\sigma_2^2}\right) \\ &= \sigma_2 \exp\left(-\frac{x^2(\sigma_2^2-1)}{2\,\sigma_2^2}\right) \\ &> 1 \text{ for small } |x| \\ &\ll 1 \text{ for large } |x| \end{aligned}$$

This is good for applications where both tails are important. If only one is important then it might be better to shift the mean towards that end.

Importance Sampling

Note that

$$\begin{aligned} \mathbb{V}_2[f(X)\;R(X)] &= \mathbb{E}_2[f(X)^2\;R(X)^2] - (\mathbb{E}_2[f(X)\;R(X)])^2 \\ &= \mathbb{E}_1[f(X)^2 R(X)] - (\mathbb{E}_1[f(X)])^2 \end{aligned}$$

so to minimise the variance we can try to minimise $\mathbb{E}_1[f(X)^2 R(X)]$.

If the new distribution is defined parametrically (e.g. a Normal distribution $N(\mu,\sigma^2)$ with mean μ and variance σ^2) then we have an optimisation problem:

$$\text{Find } \mu,\sigma \text{ to minimise } \mathbb{E}_1[f(X)^2 R(X)]$$

Can use a few samples to estimate $\mathbb{E}_1[f(X)^2 R(X)]$ and do the optimisation, then use those values of μ,σ to construct the real estimate for $\mathbb{E}_2[f(X)\;R(X)]$.

Stratified Sampling

The key idea is to achieve a more regular sampling of the most "important" dimension in the uncertainty.

Start by considering a one-dimensional problem:

$$I = \int_0^1 f(U)\,\mathrm{d}U.$$

Instead of taking N samples, drawn from uniform distribution on $[0,1]$, instead break the interval into M strata of equal width and take L samples from each.

Stratified Sampling

Define U_{ij} to be the value of i^{th} sample from strata j,

$$\begin{aligned}\overline{F}_j &= L^{-1}\sum_i f(U_{ij}) = \text{average from strata } j,\\ \overline{F} &= M^{-1}\sum_j \overline{F}_j = \text{overall average}\end{aligned}$$

and similarly let

$$\begin{aligned}\mu_j &= \mathbb{E}[f(U)\,|\,U\in\text{strata } j],\\ \sigma_j^2 &= \mathbb{V}[f(U)\,|\,U\in\text{strata } j],\\ \mu &= \mathbb{E}[f],\\ \sigma^2 &= \mathbb{V}[f].\end{aligned}$$

Stratified Sampling

With stratified sampling,

$$\mathbb{E}[\overline{F}] = M^{-1}\sum_j \mathbb{E}[\overline{F}_j] = M^{-1}\sum_j \mu_j = \mu$$

so it is unbiased.

The variance is

$$\begin{aligned}\mathbb{V}[\overline{F}] = M^{-2}\sum_j \mathbb{V}[\overline{F}_j] &= M^{-2}L^{-1}\sum_j \sigma_j^2\\ &= N^{-1}M^{-1}\sum_j \sigma_j^2\end{aligned}$$

where $N = LM$ is the total number of samples.

Stratified Sampling

Without stratified sampling, $\mathbb{V}[\overline{F}] = N^{-1}\sigma^2$ with

$$\begin{aligned}\sigma^2 &= \mathbb{E}[f^2] - \mu^2\\ &= M^{-1}\sum_j \mathbb{E}[f(U)^2\,|\,U\in\text{ strata } j] \;-\; \mu^2\\ &= M^{-1}\sum_j (\mu_j^2+\sigma_j^2) \;-\; \mu^2\\ &= M^{-1}\sum_j \left((\mu_j-\mu)^2+\sigma_j^2\right)\\ &\geq M^{-1}\sum_j \sigma_j^2\end{aligned}$$

Thus stratified sampling reduces the variance.

Stratified Sampling

How do we use this for MC simulations?

For a one-dimensional application:

- Break $[0,1]$ into M strata
- For each stratum, take L samples U with uniform probability distribution
- Compute average within each stratum, and overall average.

Stratified Sampling

Test case: European call
$r = 0.05,\ \sigma = 0.5,\ T = 1,\ S_0 = 110,\ K = 100,\ N = 10^4$ samples

M	L	MC error bound
1	10000	1.39
10	1000	0.55
100	100	0.21
1000	10	0.07

Application

MATLAB code:

```
for M = [1 10 100 1000]
  L = N/M;  ave=0; var=0;
  for m = 1:M
    U = (m-1+rand(1,L))/M;
    Y = ncfinv(U);
    S = S0*exp((r-sig^2/2)*T + sig*sqrt(T)*Y);
    F = exp(-r*T)*max(0,S-K);
    ave1 =  sum(F)/L;
    var1 = (sum(F.^2)/L - ave1^2)/(L-1);
    ave = ave + ave1/M;
    var = var + var1/M^2;
  end
end
```


Stratified Sampling

Sub-dividing a stratum always reduces the variance, so the optimum choice is to use 1 sample per stratum

However, need multiple samples in each stratum to estimate the variance and obtain a confidence interval.

This tradeoff between efficiency and confidence/reliability happens also with quasi-Monte Carlo sampling

Despite this, worth noting that when using just 1 sample per stratum, the variance of the overall estimator is $O(N^{-3})$, much better than the usual $O(N^{-1})$.

Stratified Sampling

For a multivariate application, one approach is to:

- Break $[0,1]$ into M strata
- For each stratum, take L samples U with uniform probability distribution
- Define $X_1 = \Phi^{-1}(U)$
- Simulate other elements of X using standard Normal random number generation
- Multiply X by matrix C to get $Y = C\,X$ with desired covariance
- Compute average within each stratum, and overall average

Stratified Sampling

The effectiveness of this depends on a good choice of C.

Ideally, want the function $f(Y)$ to depend solely on the value of X_1 so it reduces to a one-dimensional application.

Not easy in practice, requires good insight or a complex optimisation, so instead generalise stratified sampling approach to multiple dimensions.

Stratified Sampling

For a d-dimensional application, can split each dimension of the $[0,1]^d$ hypercube into M strata producing M^d sub-cubes.

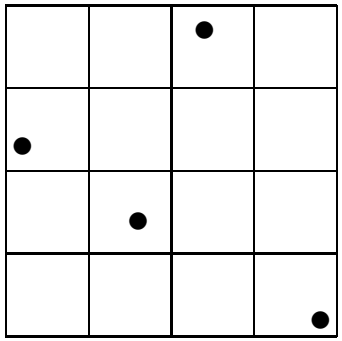
One generalisation of stratified sampling is to generate L points in each of these hypercubes

However, the total number of points is $L\,M^d$ which for large d would force M to be very small in practice.

Instead, use a method called Latin Hypercube sampling

Latin Hypercube

Generate M points, dimension-by-dimension, using 1D stratified sampling with 1 value per stratum, assigning them randomly to the M points to give precisely one point in each stratum



Latin Hypercube

This gives one set of M points, with average

$$\overline{f} = M^{-1} \sum_{m=1}^{M} f(U_m)$$

Since each of the points U_m is uniformly distributed over the hypercube,

$$\mathbb{E}[\overline{f}] = \mathbb{E}[f]$$

The fact that the points are not independently generated does not affect the expectation, only the (reduced) variance

Latin Hypercube

We now take L independently-generated set of points, each giving an average $\overline{f}_l$.

Averaging these

$$L^{-1} \sum_{l=1}^{L} \overline{f}_l$$

gives an unbiased estimate for $\mathbb{E}[f]$, and the empirical variance for $\overline{f}_l$ gives a confidence interval in the usual way.

Latin Hypercube

Note: in the special case in which the function $f(U)$ is a sum of one-dimensional functions:

$$f(U) = \sum_i f_i(U_i)$$

where U_i is the i^{th} component of U, then Latin Hypercube sampling reduces to 1D stratified sampling in each dimension.

In this case, potential for very large variance reduction by using large sample size M.

Much harder to analyse in general case.

Quasi Monte Carlo

Standard Monte Carlo approximates high-dimensional hypercube integral

$$\int_{[0,1]^d} f(x)\, \mathrm{d}x$$

by

$$\frac{1}{N} \sum_{i=1}^{N} f(x^{(i)})$$

with points chosen randomly, giving

- r.m.s. error proportional to $N^{-1/2}$
- confidence interval

Quasi Monte Carlo

Standard quasi Monte Carlo uses the same equal-weight estimator

$$\frac{1}{N} \sum_{i=1}^{N} f(x^{(i)})$$

but chooses the points systematically so that

- error roughly proportional to N^{-1}
- no confidence interval

(We'll get the confidence interval back later by adding in some randomisation!)

Quasi-Monte Carlo

The key is to use points which are fairly uniformly spread within the hypercube, not clustered anywhere.

There is theory to prove that for certain point constructions, and certain function classes,

$$\text{Error} < C\,\frac{(\log N)^d}{N}$$

- for small dimension d, ($d<10$?) this is much better than $N^{-1/2}$ r.m.s. error for standard MC
- for large dimension d, $(\log N)^d$ could be enormous, so not clear there is any benefit

Rank-1 Lattice Rule

A rank-1 lattice rule has the simple construction

$$x^{(i)} = \frac{i}{N}\,z \mod 1$$

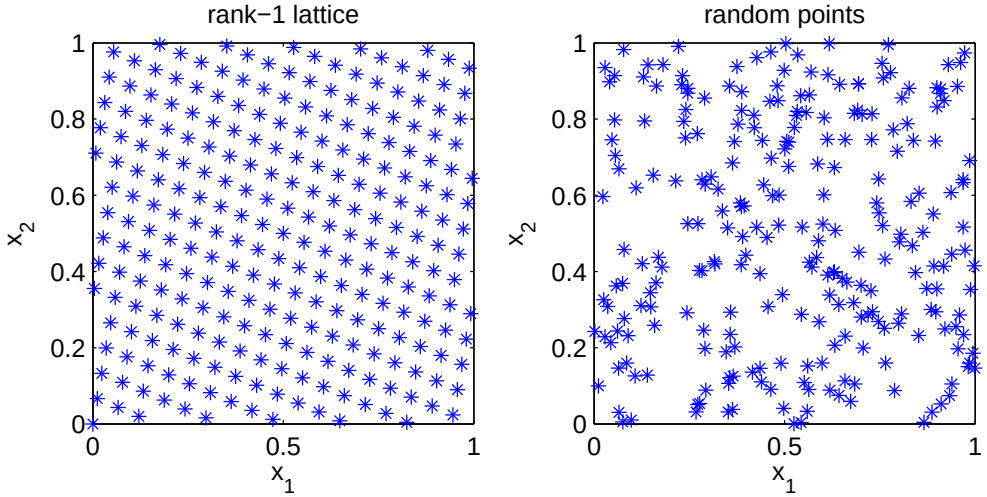
where z is a special d-dimensional "generating vector" with integer components co-prime with N (i.e. GCF is 1) and $r \mod 1$ means dropping the integer part of r

In each dimension k, the values $x_k^{(i)}$ are a permutation of the equally spaced points $0, 1/N, 2/N \ldots (N-1)/N$ which is great for integrands f which vary only in one dimension.

Also very good if $f(x) = \sum_k f_k(x_k)$.

Rank-1 Lattice Rule

Two dimensions: 256 points



Sobol Sequences

Sobol sequences $x^{(i)}$ have the property that for small dimensions $d<40$ the subsequence $2^m \le i < 2^{m+1}$ has precisely 2^{m-d} points in each sub-unit formed by d bisections of the original hypercube.

For example:

- cutting it into halves in any dimension, each has 2^{m-1} points
- cutting it into quarters in any dimension, each has 2^{m-2} points
- cutting it into halves in one direction, then halves in another direction, each quarter has 2^{m-2} points
- etc.

The generation of these sequences is a bit complicated, but it is fast and plenty of software is available to do it. MATLAB has `sobolset` as part of the Statistics toolbox.

Randomised QMC

In the best cases, QMC error is $O(N^{-1})$ instead of $O(N^{-1/2})$ but without a confidence interval.

To get a confidence interval using a rank-1 lattice rule, we use several sets of QMC points, with the N points in set m defined by

$$x^{(i,m)} = \left(\frac{i}{N} z \; + \; X^{(m)}\right) \mod 1$$

where $X^{(m)}$ is a random offset vector, uniformly distributed in $[0,1]^d$

Randomised QMC

For each m, let

$$\overline{f}_m = \frac{1}{N}\sum_{i=1}^{N} f(x^{(i,m)})$$

This is a random variable, and since $\mathbb{E}[f(x^{(i,m)})] = \mathbb{E}[f]$ it follows that $\mathbb{E}[\overline{f}_m] = \mathbb{E}[f]$

By using multiple sets, we can estimate $\mathbb{V}[\overline{f}]$ in the usual way and so get a confidence interval

More sets $\Longrightarrow$ better variance estimate, but poorer error.

Some people use as few as 10 sets, but I prefer 32.

Randomised QMC

For Sobol sequences, randomisation is achieved through digital scrambling:

$$x^{(i,m)} = x^{(i)} \underline{\vee} \; X^{(m)}$$

where the exclusive-or operation $\underline{\vee}$ is applied bitwise so that

$$\begin{array}{rl} & 0.1010011 \\ \underline{\vee} & 0.0110110 \\ = & 0.1100101 \end{array}$$

The benefit of the digital scrambling is that it maintains the special properties of the Sobol sequence.

MATLAB's `sobolset` supports digital scrambling.

Dominant Dimensions

QMC points have the property that the points are more uniformly distributed through the lowest dimensions. Consequently, important to think about how the dimensions are allocated to the problem.

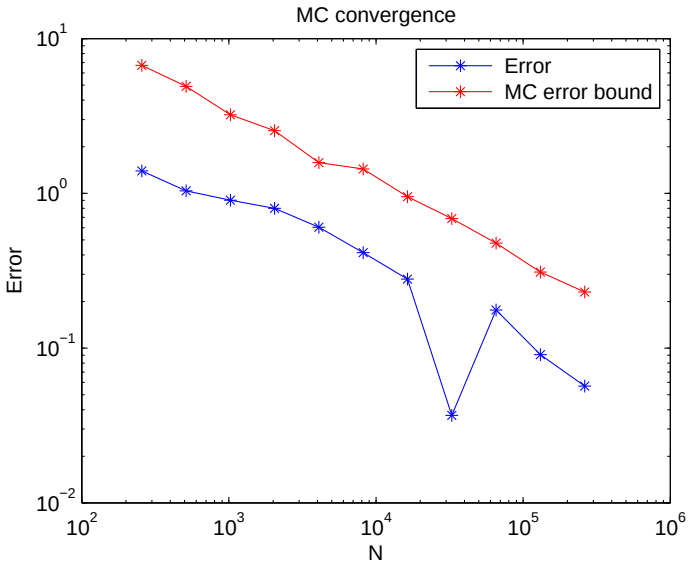
Previously, have generated correlated Normals through $Y = L\,X$ with X i.i.d. $N(0,1)$ Normals.

For Monte Carlo, Y's have same distribution for any L such that $LL^T = \Sigma$, but for QMC different L's are equivalent to a change of coordinates and it can make a big difference.

Usually best to use a PCA construction $L = U\,\Lambda^{1/2}$ with eigenvalues arranged in descending order, from largest ($\Longrightarrow$ most important?) to smallest.

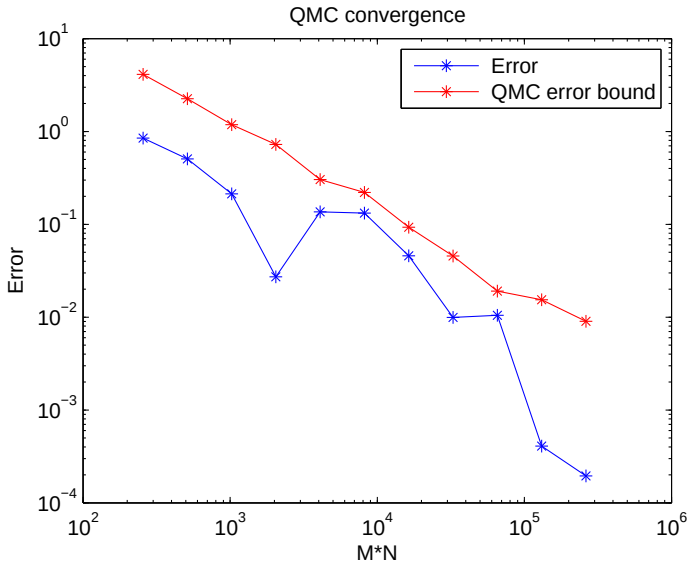
Finance Applications

1D call option – Monte Carlo convergence



Finance Applications

1D call option – Sobol QMC convergence



Application

Main piece of MATLAB code:

```
M = 2^p;                  % number of points in each set
N = 64;                   % number of sets of points

for n = 1:N
  Ps = sobolset(1);       % dimension 1
  Ps = scramble(Ps,'MatousekAffineOwen');
  U  = net(Ps,M)';
  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);
  Fave(n) = sum(F)/M;
end

V  = sum(Fave)/N;
sd = sqrt((sum(Fave.^2)/N - (sum(Fave)/N)^2)/(N-1));
```


Final comments

- Control variates can sometimes be very useful – needs good insight to find a suitable control variate
- Importance sampling is very useful when the main contribution to the expectation comes from rare extreme events
- Stratified sampling is very effective in 1D, but not so clear how to use it in multiple dimensions
- Latin Hypercube is one generalisation – particularly effective when function can be almost decomposed into a sum of 1D functions

Final words

- quasi-Monte Carlo can give a much lower error than standard MC; $O(N^{-1})$ in best cases, instead of $O(N^{-1/2})$
- randomised QMC is important to regain confidence interval
- correct selection of dominant dimensions can also be important
- Hard to predict which variance reduction approach will be most effective
- Advice: when facing a new class of applications, try each one, and don't forget you can sometimes combine different techniques (e.g. stratified sampling with antithetic variables, or Latin Hypercube with importance sampling)