Monte Carlo Methods for Uncertainty Quantification

Mike Giles

Mathematical Institute, University of Oxford

Contemporary Numerical Techniques

Lecture outline

Lecture 2: Variance reduction

- control variate
- Latin Hypercube
- randomised quasi-Monte Carlo

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 averaging N samples of a new estimator

$$\widehat{f} = f - \lambda\,(g - \mathbb{E}[g])$$

Again unbiased since $\mathbb{E}[\widehat{f}] \;=\; \mathbb{E}[f] - \lambda\,\mathbb{E}[\,g - \mathbb{E}[g]\,] \;=\; \mathbb{E}[f]$

Control Variates

For a single sample,

$$\begin{aligned}\mathbb{V}[f-\lambda\,(g-\mathbb{E}[g])] &= \mathbb{V}[f-\lambda\,g] \\ &= \mathbb{V}[f]-2\,\lambda\,\mathrm{Cov}[f,g]+\lambda^2\,\mathbb{V}[g]\end{aligned}$$

For an average of N samples,

$$\mathbb{V}[\overline{f}-\lambda\,(\overline{g}-\mathbb{E}[g])] = N^{-1}\left(\mathbb{V}[f]-2\,\lambda\,\mathrm{Cov}[f,g]+\lambda^2\,\mathbb{V}[g]\right)$$

To minimise this, the optimum value for λ is

$$\lambda = \frac{\mathrm{Cov}[f,g]}{\mathbb{V}[g]}$$

Control Variates

The resulting variance is

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

where $-1\le\rho\le 1$ 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 λ, can be estimated numerically.

Latin Hypercube

The central idea is to achieve a more regular sampling of the unit hypercube $[0,1]^d$ when trying to estimate

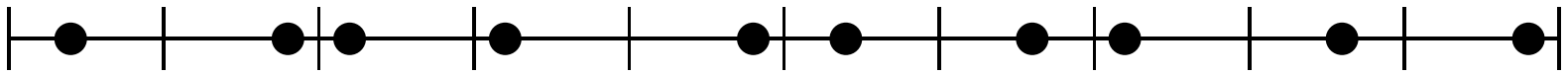
$$\int_{[0,1]^d} f(U)\,\mathrm{d}U.$$

We 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 $[0,1]$, break the interval into N strata (or sub-intervals) of width $1/N$ and take 1 sample from each, with a uniform random distribution within the stratum.

Stratified Sampling



For j^{th} stratum, if $f(U)$ is differentiable then

$$f(U) \approx f(U_j) + f'(U_j)\,(U - U_j)$$

where U_j is midpoint of stratum, and hence

$$\begin{aligned} \mathbb{V}[f(U) \,|\, U \in \text{stratum } j] &\approx \left(f'(U_j)\right)^2 \; \mathbb{V}[U - U_j \,|\, U \in \text{ stratum } j] \\ &= \frac{1}{12N^2} \left(f'(U_j)\right)^2 \end{aligned}$$

since the stratum has width $1/N$ so

$$\mathbb{V}[U - U_j \,|\, U \in \text{ stratum } j] \;=\; \int_{-1/(2N)}^{1/(2N)} U^2 \; N \,\mathrm{d}U$$

Stratified Sampling

Summing all of the variances (due to independence) and dividing by N^2 (due to averaging) the variance of the average over all strata is then

$$\frac{1}{12N^4}\sum_j \left(f'(U_j)\right)^2 \approx \frac{1}{12N^3}\int \left(f'(U)\right)^2 \, \mathrm{d}U$$

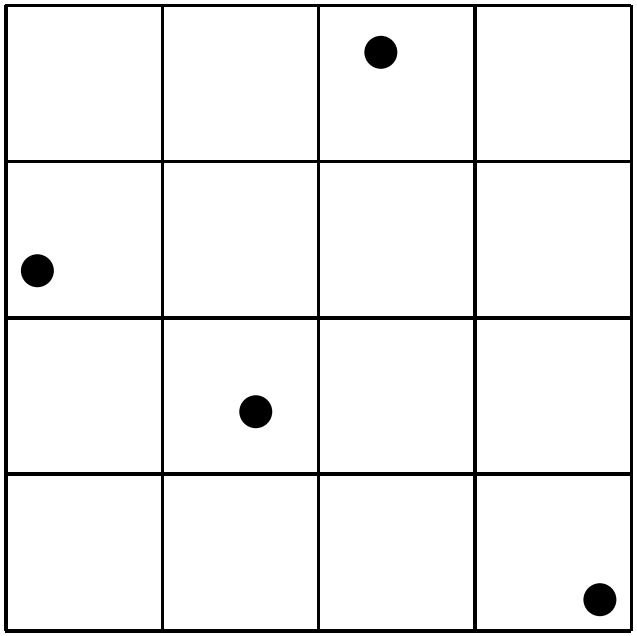
so the r.m.s. error is $O(N^{-3/2})$, provided $f'(U)$ is square integrable.

This is much better than the usual $O(N^{-1/2})$ r.m.s. error
– shows how powerful stratified sampling can be.

Latin Hypercube

Latin Hypercube gemeralises this idea to multiple dimensions.

Cut each dimension into L strata, and generate L points assigning them randomly to the L^d cubes to give precisely one point in each stratum



Latin Hypercube

This gives one set of L points, with average

$$\overline{f} = L^{-1} \sum_{\ell=1}^{L} f(U_\ell)$$

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 M independently-generated set of points, each giving an average $\overline{f}_m$.

Averaging these

$$M^{-1} \sum_{m=1}^{M} \overline{f}_m$$

gives an unbiased estimate for $\mathbb{E}[f]$, and the empirical variance for $\overline{f}_m$ 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 L.

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 $N^{-1/2}$
- an unbiased estimator
- 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}
- a biased estimator
- no confidence interval

(We'll fix the bias and 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

Sobol Sequences

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

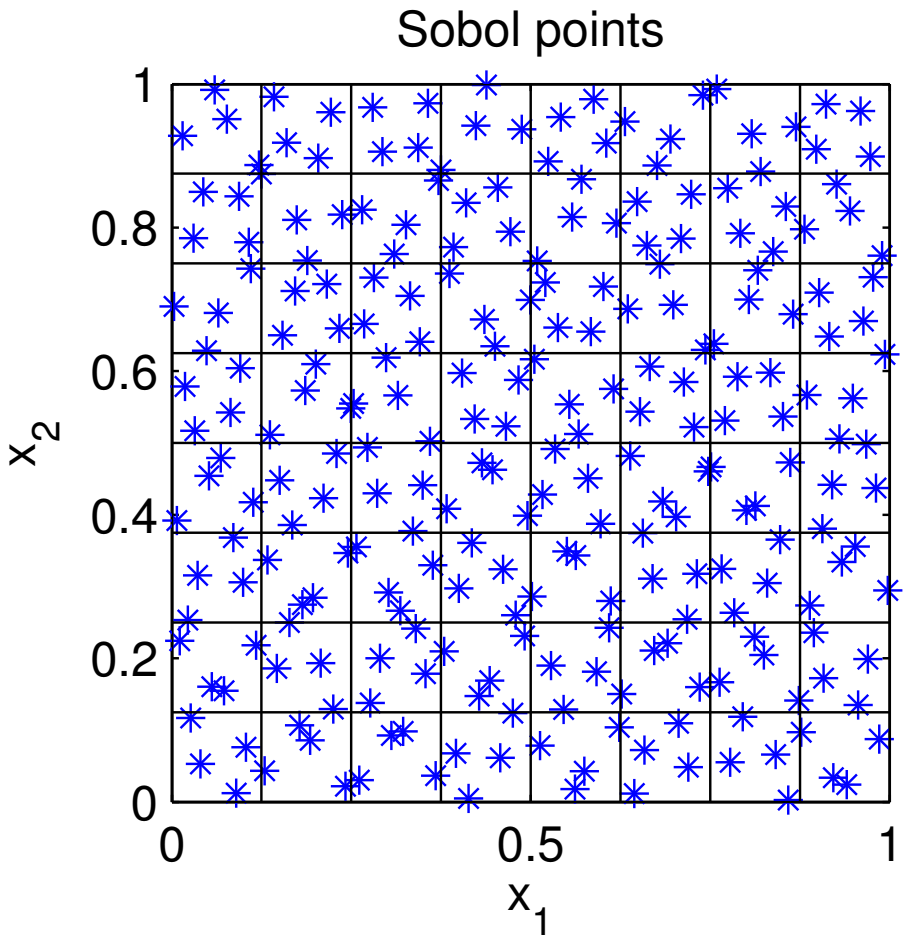
For example:

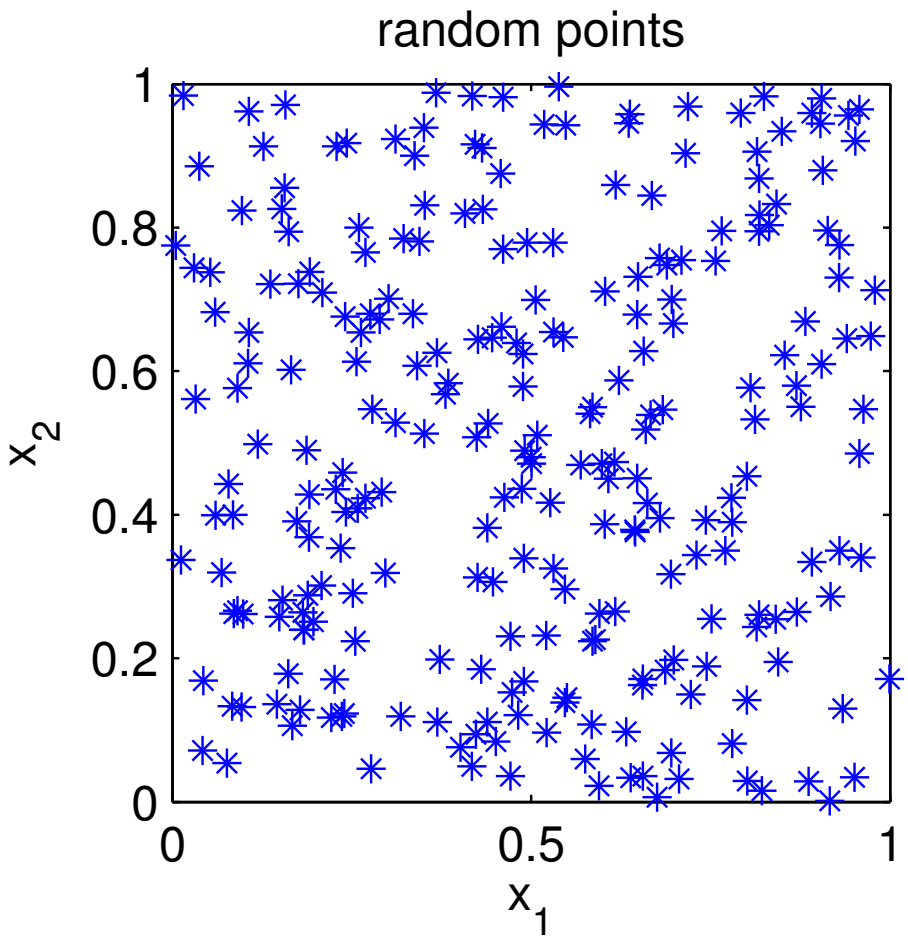
- cutting it into halves in any dimension, each has 2^{m-1} points
- cutting it into quarters in any dimension, each has 2^{m-2} points
- cutting it into halves in one direction, then halves in another direction, each quarter has 2^{m-2} points
- etc.

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.

Sobol Sequences

Two dimensions: 256 points





Randomised QMC

In the best cases, QMC error is $O(N^{-1})$ instead of $O(N^{-1/2})$ but with bias and no confidence interval.

To fix this, we introduce randomisation through a "digital scrambling" which maintains the special properties of the Sobol sequence.

For the i^{th} point in the m^{th} set of points, we define

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

where $X^{(m)}$ is a uniformly-distributed random point in $[0,1)^d$, and the exclusive-or operation $\underline{\vee}$ is applied elementwise and bitwise so that

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

MATLAB's `sobolset` supports this digital scrambling.

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.

Finance Application

In the basket call option example, the asset simulation can be turned into

$$S_i(T) = S_i(0) \ \exp\left((r - \tfrac{1}{2}\sigma_i^2)\, T + (L\, Y)_i \right)$$

where Y is a vector of 5 independent unit normals and

$$L\, L^T = \Sigma$$

with $\Sigma_{ij} = \sigma_i\, \sigma_j\, \rho_{ij}$.

There are two standard ways of generating L:

- Cholesky factorisation (so L is lower-triagular)
- PCA factorisation ($L = U\Lambda^{1/2}$, where Λ is diagonal matrix of eigenvalues, and U is orthonormal matrix of eigenvectors)

Financial Application

- 5 underlying assets starting at $S_0 = 100$, with call option on arithmetic mean with strike $K = 100$
- Geometric Brownian Motion model, $r = 0.05$, $T = 1$
- volatility $\sigma = 0.2$ and covariance matrix

$$\Sigma = \sigma^2 \begin{pmatrix} 1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 1 \end{pmatrix}$$

Financial Application

Numerical results using $2^{20} \approx 10^6$ samples in total, comparing MC, Latin Hypercube and Sobol QMC, each with either Cholesky or PCA factorisation of Σ.

	Cholesky		PCA	
	Val	Err Bnd	Val	Err Bnd
Monte Carlo	7.0193	0.0239	7.0250	0.0239
Latin Hypercube	7.0244	0.0081	7.0220	0.0015
Sobol QMC	7.0228	0.0007	7.0228	0.0001

Final comments

- Control variates can sometimes be very useful – needs good insight to find a suitable control variate
- Latin Hypercube achieves a more uniform spread of sampling points – particularly effective when function can be almost decomposed into a sum of 1D functions
- 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 and eliminate bias