Stochastic Simulation: Lecture 7

Christoph Reisinger

Oxford University Mathematical Institute

Modified from earlier slides by Prof. Mike Giles.

Quasi Monte Carlo

- low discrepancy sequences
- Koksma-Hlawka inequality
- rank-1 lattice rules and Sobol sequences
- randomised QMC
- identification of dominant dimension

Quasi Monte Carlo

Standard Monte Carlo approximates the 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

- unbiased estimator
- 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

- the estimate is biased
- error roughly proportional to N^{-1}
- no confidence interval

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

Low Discrepancy Sequences

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

The star discrepancy $D_N^*(x^{(1)}, \dots x^{(N)})$ of a set of N points is defined as

$$D_N^* = \sup_{B \in J} \left| \frac{A(B)}{N} - \lambda(B) \right|$$

where J is the set of all hyper-rectangles of the form

$$\prod [u_i^-, u_i^+], \quad u_i^\pm \in [0, 1],$$

$A(B)$ is the number of points in B, and $\lambda(B)$ is the volume (or measure) of B.

Low Discrepancy Sequences

There are sequences for which

$$D_N^* \leq C \frac{(\log N)^d}{N}$$

where d is the dimension of the problem.

This is important because of the Koksma–Hlawka inequality.

Koksma–Hlawka Inequality

$$\left| \frac{1}{N} \sum_{i=1}^{N} f(x^{(i)}) - \int_{[0,1]^d} f(x) \, \mathrm{d}x \right| \leq V(f) \; D_N^*(x^{(1)}, \ldots x^{(N)})$$

where $V(f)$ is the Hardy–Krause variation of f (for sufficiently differentiable f) is a sum of terms of the form

$$\int_{[0,1]^k} \left| \frac{\partial^k f}{\partial x_{i_1} \ldots \partial x_{i_k}} \right|_{x_j=1, j \neq i_1, \ldots, i_k} \mathrm{d}x$$

with $i_1 < i_2 < \ldots < i_k$ for $k \leq d$.

Problem: not a useful error bound

- in finance applications f often isn't even bounded
- even when it is, it's not sufficiently differentiable and estimating $V(f)$ is computationally demanding

Koksma-Hlawka Inequality

However, still useful because of what it tells us about the asymptotic behaviour:

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

- 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

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

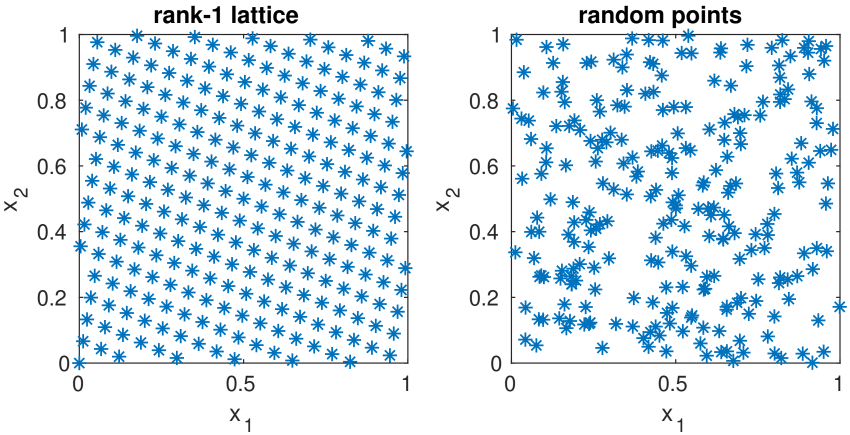
where z is a special d-dimensional "generating vector"
with integer components co-prime with N (i.e. GCF is 1)
and $r \mod 1$ means dropping the integer part of r

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 $\quad f(x) = \sum_k f_k(x_k).$

Rank-1 Lattice Rule

Two dimensions: 256 points



Sobol Sequences

The most popular QMC approach uses Sobol sequences $x^{(i)}$ which have the property that for small dimensions $d < 40$ the subsequence

$$2^m \leq i < 2^{m+1}$$

of length 2^m has precisely 2^{m-d} points in each of the little cubes of volume 2^{-d} formed by bisecting the unit hypercube in each dimension, and similar properties hold with other pieces.

Sobol Sequences

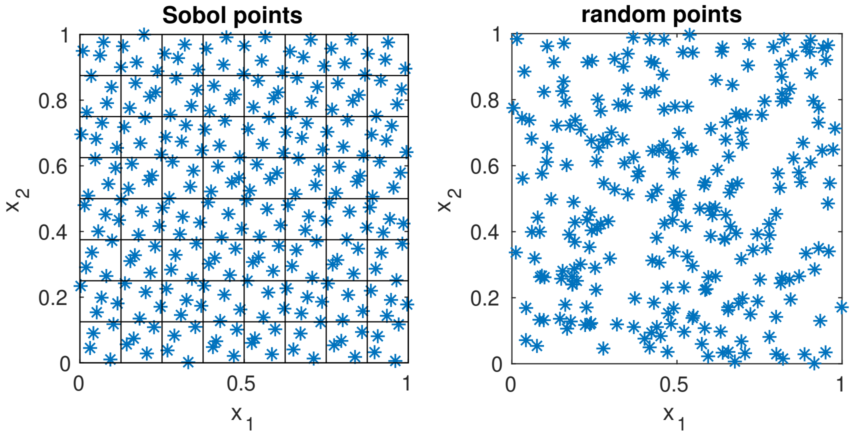
For example:

- 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.

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 a bias and without a confidence interval.

To eliminate the bias and 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 (or digital shifting):

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

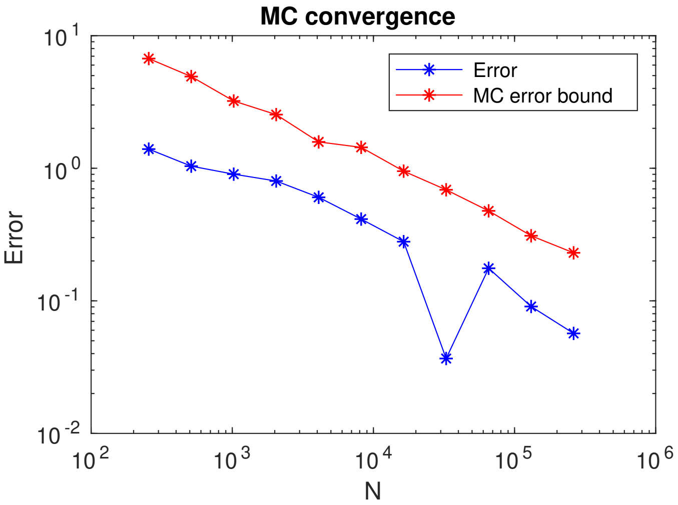
where the exclusive-or operation $\veebar$ is applied bitwise so that

$$\begin{array}{cc} & 0.1010011 \\ \veebar & 0.0110110 \\ = & 0.1100101 \end{array}$$

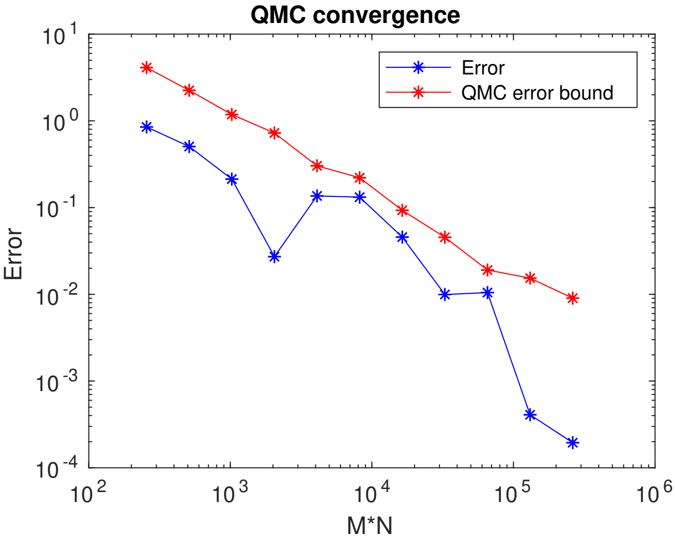
Note: the bit-wise exclusive-or operator is $\wedge$ in C/C++ and Python.

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

Call option: MC



Call option: Sobol QMC



Dominant Dimensions

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

Previously, have generated correlated Normals through

$$Y = LX$$

with X i.i.d. $N(0,1)$ Normals.

For Monte Carlo, Y's have same distribution for any L such that $LL^T = \Sigma$.

Dominant Dimensions

However, 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.

Basket call option

- 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

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

Basket call option

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 words

- QMC can give a much lower error than standard MC; $O(N^{-1})$ in best cases, instead of $O(N^{-1/2})$
- supporting theory is not particularly useful
- randomised QMC is important to eliminate the bias and regain a confidence interval
- correct selection of dominant dimensions can also be important
- Sobol sequences are most used in industry, and often available in software libraries