Advanced Monte Carlo Methods: Quasi-Monte Carlo

Prof. Mike Giles

mike.giles@maths.ox.ac.uk

Oxford University Mathematical Institute

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 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!)

Low Discrepancy Sequences

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

The star discrepancy $D_N^*(x^{(1)}, \ldots x^{(N)})$ 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 defined (for sufficiently differentiable f) as 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

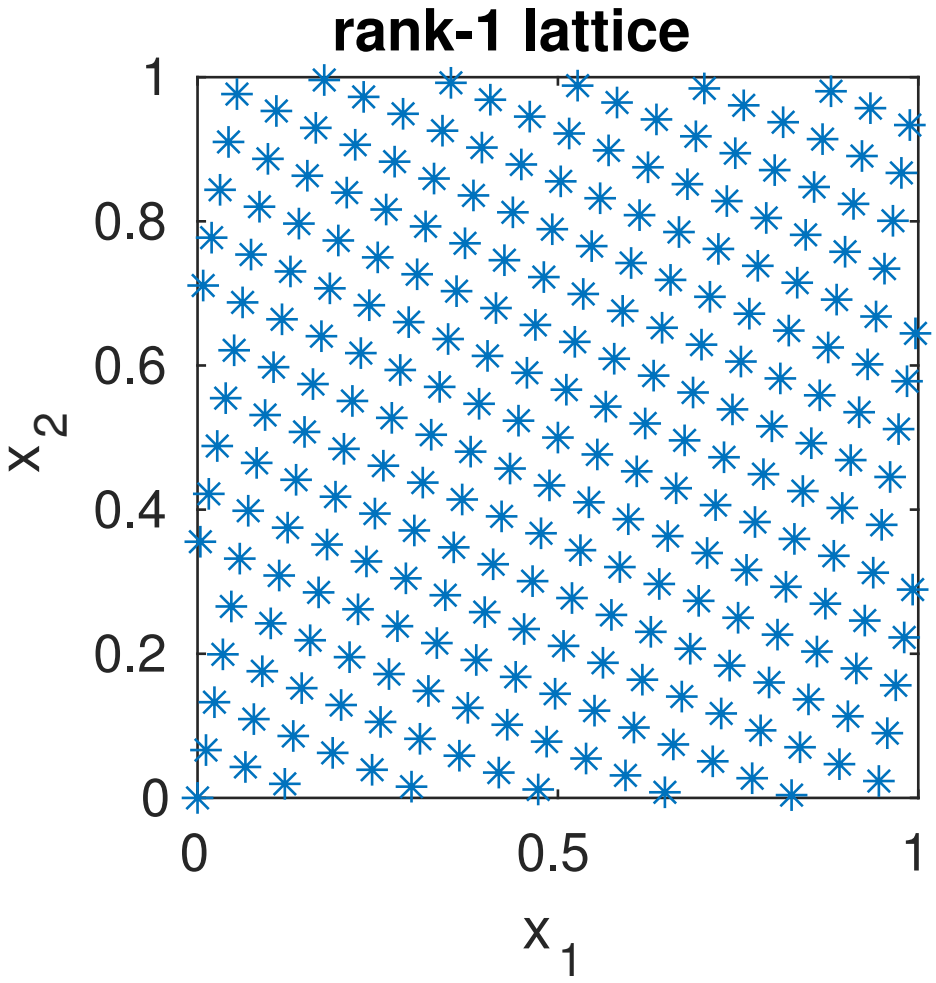
$$x^{(i)} = \frac{i}{N} z \mod 1$$

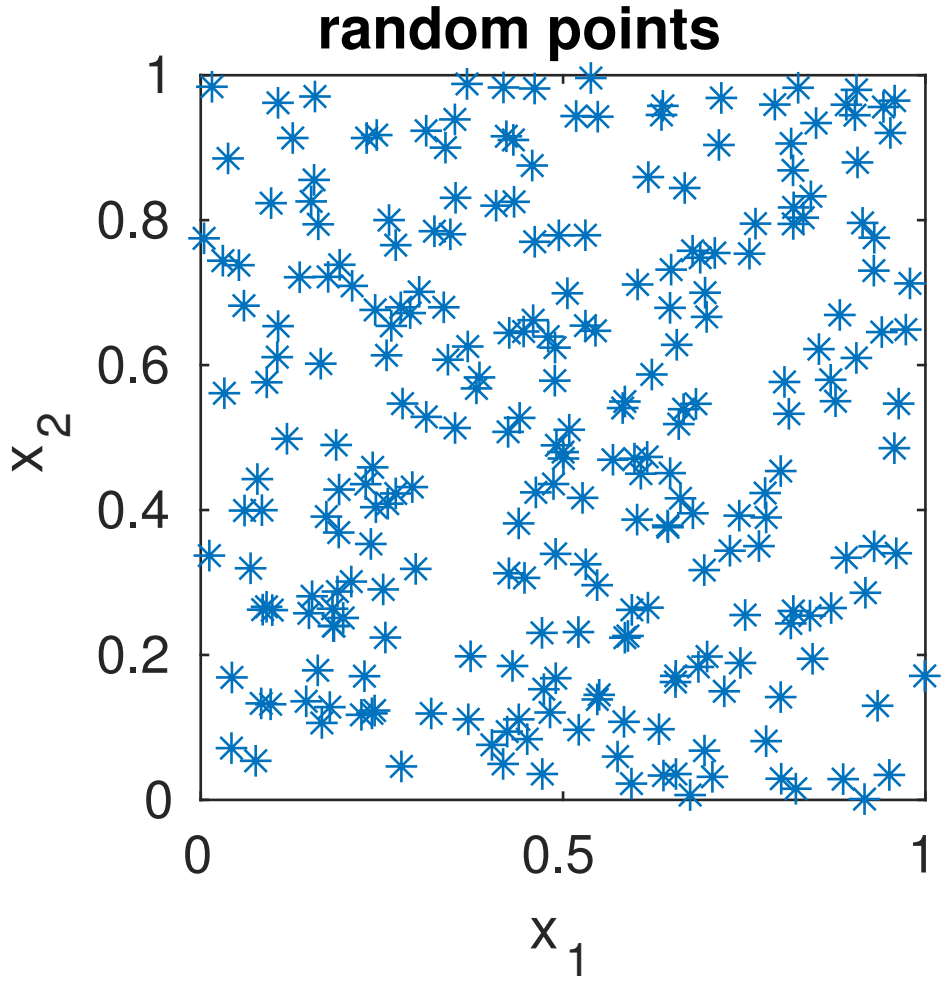
where z is a d-dimensional "generating vector",
and $r \mod 1$ means dropping the integer part of r

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





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.

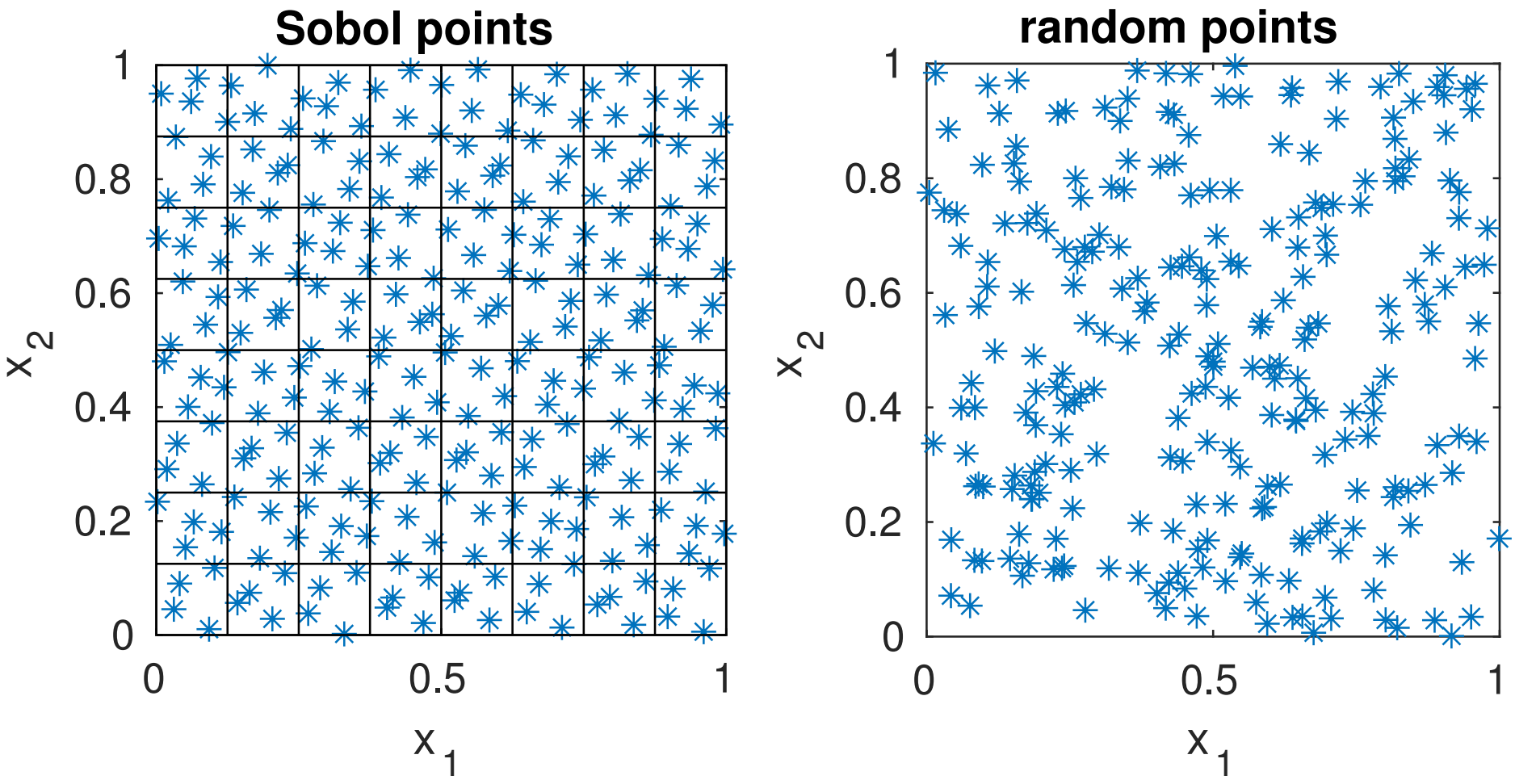
Sobol Sequences

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



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.

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

$$\begin{aligned} & 0.1010011 \\ \underline{\vee} \quad & 0.0110110 \\ = \quad & 0.1100101 \end{aligned}$$

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

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.

Ideally, we'd like to use a change of variables, so the function we're integrating depends only on the first coordinate.

Dominant Dimensions

Suppose we have an European option, based on d multiple underlying assets with

$$\log S_i(T) = \log S_i(0) + \left(r - \tfrac{1}{2}\sigma_i^2\right) T + \sigma_i W_i(T)$$

and the $\log S_i(T)$ have covariance matrix Σ.

If U is a d-dimensional QMC point, can produce uncorrelated quasi-random Normals using

$$X_i = \Phi^{-1} U_i$$

but how do we generate correlated quasi-Normals?

Dominant Dimensions

Previously, have generated correlated Normals through

$$Y = L\,X$$

with X i.i.d. $N(0,1)$ Normals, and L is any matrix such that $L\,L^T = \Sigma$.

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 in diagonal matrix Λ (and associated eigenvectors U) arranged in descending order, from largest ($\Longrightarrow$ most important?) to smallest.

Path-dependent options

Same ingredients as simple European options:

- Sobol or lattice rule quasi-uniform generators
- PCA to best use QMC inputs for multi-dimensional applications
- randomised QMC to regain confidence interval

New ingredient:

- how best to use QMC inputs to generate Brownian increments

Quasi-Monte Carlo

When using standard Normal random inputs for MC simulation, can express expectation as a multi-dimensional integral with respect to inputs

$$V = \mathbb{E}[\widehat{f}(\widehat{S})] = \int \widehat{f}(\widehat{S})\ \phi(Z)\ \mathrm{d}Z$$

where $\phi(Z)$ is multi-dimensional standard Normal p.d.f.

Putting $Z_n = \Phi^{-1}U_n$ turns this into an integral over a M-dimensional hypercube

$$V = \mathbb{E}[\widehat{f}(\widehat{S})] = \int \widehat{f}(\widehat{S})\ \mathrm{d}U$$

Quasi-Monte Carlo

This is then approximated as

$$N^{-1} \sum_{n} \widehat{f}(\widehat{S}^{(n)})$$

and each path calculation involves the computations

$$U \;\rightarrow\; Z \;\rightarrow\; \Delta W \;\rightarrow\; \widehat{S} \;\rightarrow\; \widehat{f}$$

The key step here is the second, how best to convert the vector Z into the vector ΔW. With standard Monte Carlo, as long as ΔW has the correct distribution, how it is generated is irrelevant, but with QMC it does matter.

Quasi-Monte Carlo

For a scalar Brownian motion $W(t)$ with $W(0)=0$, defining $W_n=W(nh)$, each W_n is Normally distributed and for $j \geq k$

$$\mathbb{E}[W_j\, W_k] = \mathbb{E}[W_k^2] + \mathbb{E}[(W_j - W_k)\, W_k] = t_k$$

since $W_j - W_k$ is independent of W_k.

Hence, the covariance matrix for W is Ω with elements

$$\Omega_{j,k} = \min(t_j, t_k)$$

Quasi-Monte Carlo

The task now is to find a matrix L such that

$$L\ L^T = \Omega = h \begin{pmatrix} 1 & 1 & \dots & 1 & 1 \\ 1 & 2 & \dots & 2 & 2 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & 2 & \dots & M-1 & M-1 \\ 1 & 2 & \dots & M-1 & M \end{pmatrix}$$

We will consider 3 possibilities:

- Cholesky factorisation
- PCA
- Brownian Bridge treatment

Cholesky factorisation

The Cholesky factorisation gives

$$L = \sqrt{h} \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ 1 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & 1 & \dots & 1 & 0 \\ 1 & 1 & \dots & 1 & 1 \end{pmatrix}$$

and hence

$$W_n = \sum_{m=1}^{n} \sqrt{h}\, Z_m \quad \Longrightarrow \quad \Delta W_n = W_n - W_{n-1} = \sqrt{h}\, Z_n$$

i.e. standard MC approach

PCA construction

The PCA construction uses

$$L = U\,\Lambda^{1/2} = \left(\; U_1 \,\middle|\, U_2 \,\middle|\, \ldots \; \right) \begin{pmatrix} \lambda_1^{1/2} & & \\ & \lambda_2^{1/2} & \\ & & \ldots \end{pmatrix}$$

with the eigenvalues λ_n and eigenvectors U_n arranged in descending order, from largest to smallest.

Numerical computation of the eigenvalues and eigenvectors is costly for large numbers of timesteps, so instead use theory due to Åkesson and Lehoczky (1998)

PCA construction

It is easily verified that

$$\Omega^{-1} = h^{-1} \begin{pmatrix} 2 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & \cdots & \cdots & \cdots & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & -1 \\ & & & & & -1 & 1 \end{pmatrix}.$$

This looks like the finite difference operator approximating a second derivative, and so the eigenvectors are Fourier modes.

PCA construction

The eigenvectors of both Ω^{-1} and Ω are

$$(U_m)_n = \frac{2}{\sqrt{2M+1}} \sin\left(\frac{(2m-1)\, n\pi}{2M+1}\right)$$

and the eigenvalues of Ω are the reciprocal of those of Ω^{-1},

$$\lambda_m = \frac{h}{4}\left(\sin\left(\frac{(2m-1)\,\pi}{2\,(2M+1)}\right)\right)^{-2}$$

Because the eigenvectors are Fourier modes, an efficient FFT transform can be used (Scheicher, 2006) to compute

$$L\,Z \;=\; U\,\left(\Lambda^{1/2}\,Z\right) \;= \sum_m (\sqrt{\lambda_m}\;Z_m)\,U_m$$

Brownian Bridge construction

The Brownian Bridge construction uses the theory from a previous lecture.

The final Brownian value is constructed using Z_1:

$$W_M = \sqrt{T}\, Z_1$$

Conditional on this, the midpoint value $W_{M/2}$ is Normally distributed with mean $\frac{1}{2}W_M$ and variance $T/4$, and so can be constructed as

$$W_{M/2} = \tfrac{1}{2}W_M + \sqrt{T/4}\, Z_2$$

Brownian Bridge construction

The quarter and three-quarters points can then be constructed as

$$\begin{aligned} W_{M/4} &= \tfrac{1}{2}W_{M/2} + \sqrt{T/8}\, Z_3 \\ W_{3M/4} &= \tfrac{1}{2}(W_{M/2} + W_M) + \sqrt{T/8}\, Z_4 \end{aligned}$$

and the procedure continued recursively until all Brownian values are defined.

(This assumes M is a power of 2 – if not, the implementation is slightly more complex)

I have a slight preference for this method because it is particularly effective for European option for which $S(T)$ is very strongly dependent on $W(T)$.

Multi-dimensional Brownian motion

The preceding discussion concerns the construction of a single, scalar Brownian motion.

Suppose now that we have to generate a P-dimensional Brownian motion with correlation matrix Σ between the different components.

What do we do?

Multi-dimensional Brownian motion

First, using either PCA or BB to construct P uncorrelated Brownian paths using

- $Z_1, Z_{1+P}, Z_{1+2P}, Z_{1+3P}, \ldots$ for first path
- $Z_2, Z_{2+P}, Z_{2+2P}, Z_{2+3P}, \ldots$ for second path
- $Z_3, Z_{3+P}, Z_{3+2P}, Z_{3+3P}, \ldots$ for third path
- etc.

This uses the "best" dimensions of Z for the overall behaviour of all of the paths.

Multi-dimensional Brownian motion

Second, define

$$W_n^{corr} = L_\Sigma \, W_n^{uncorr} \quad \Longrightarrow \quad \Delta W_n^{corr} = L_\Sigma \, \Delta W_n^{uncorr}$$

where W_n^{uncorr} is the uncorrelated sequence,
W_n^{corr} is the correlated sequence, and

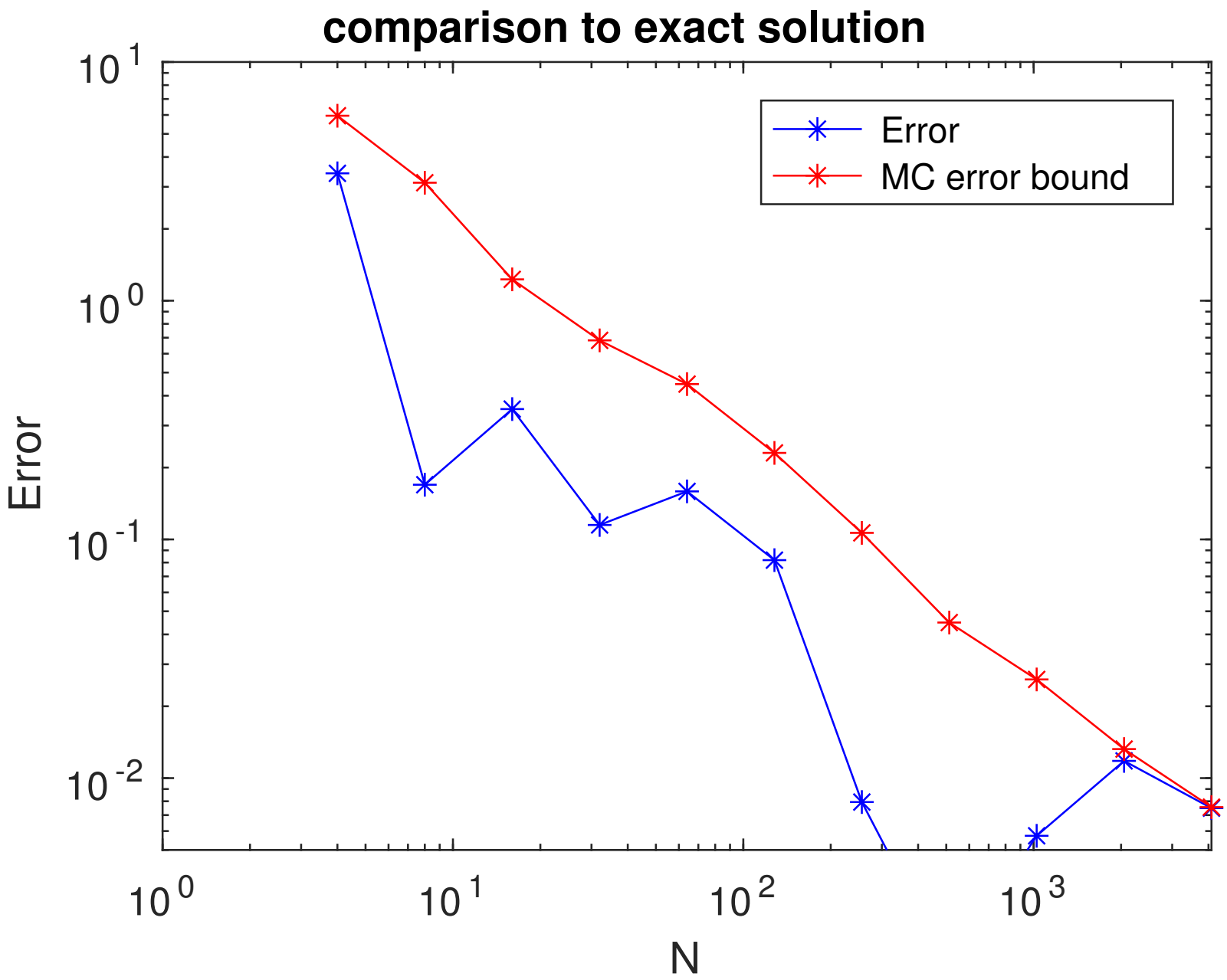
$$L_\Sigma \, L_\Sigma^T = \Sigma$$

Numerical results

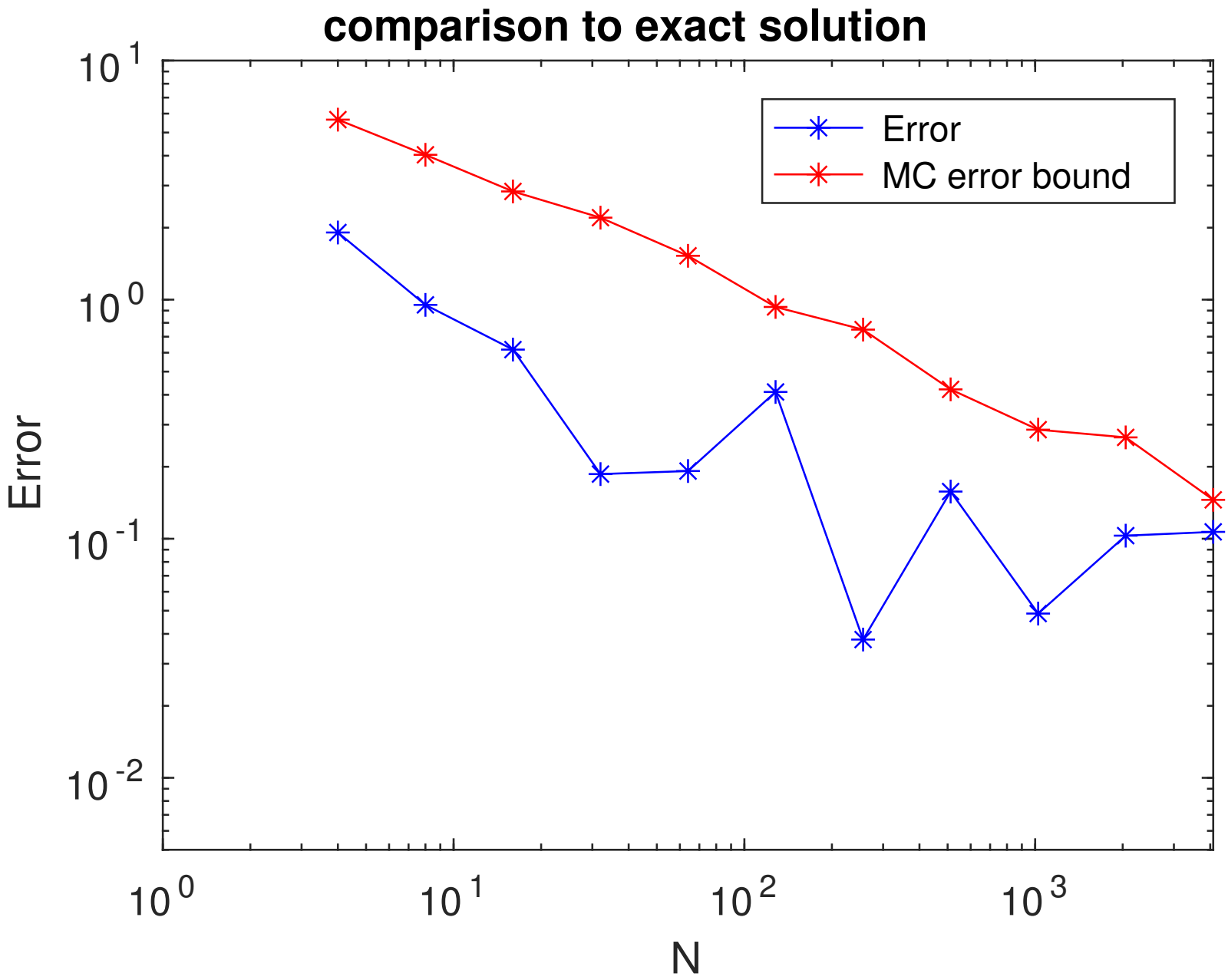
Usual European call test case based on geometric Brownian motion:

- 128 timesteps so weak error is negligible
- comparison between
  - QMC using Brownian Bridge
  - QMC without Brownian Bridge
  - standard MC
- QMC calculations use Sobol generator
- all calculations use 64 “sets” of points – for QMC calcs, each has a different random offset
- plots show error and 3 s.d. error bound

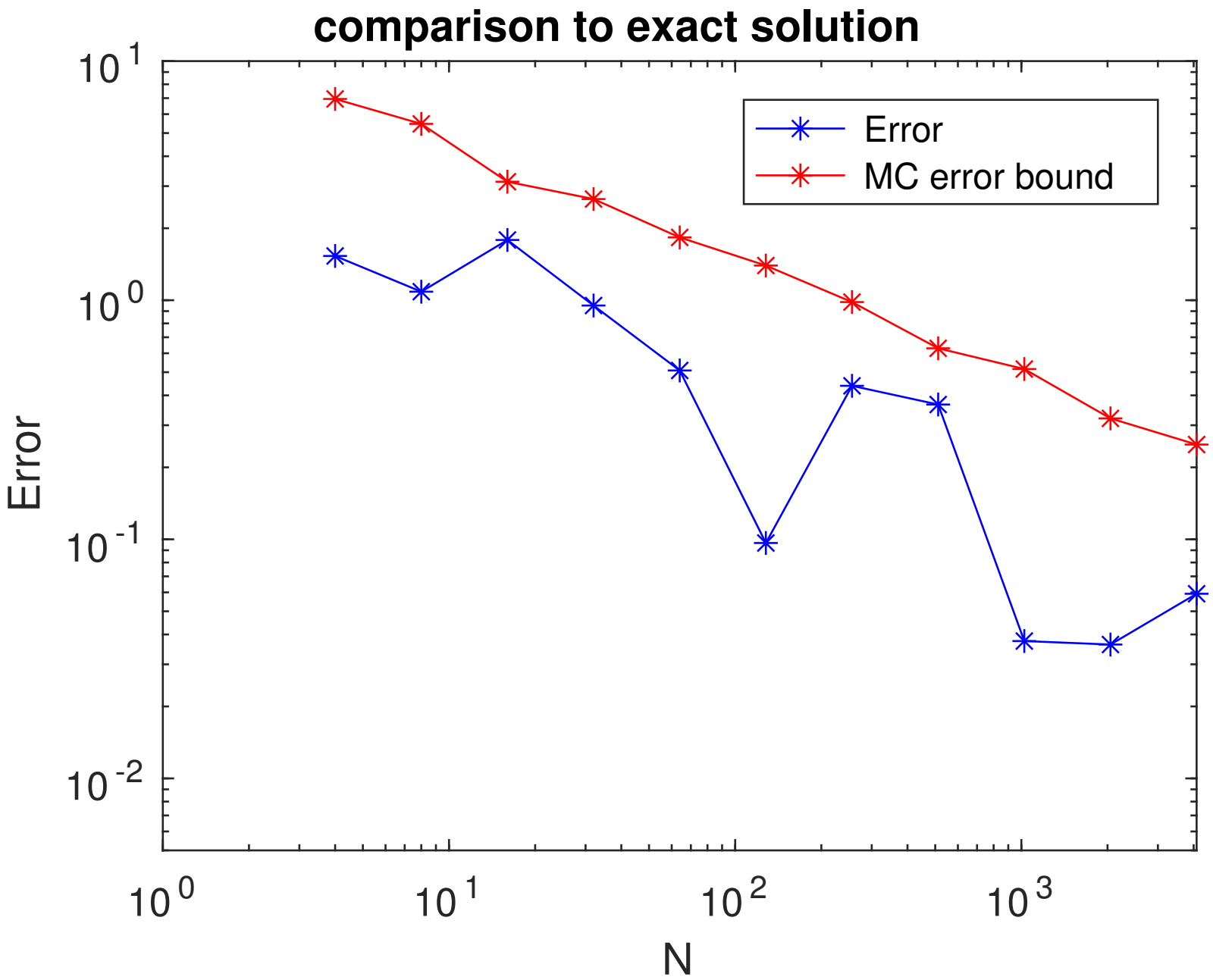
QMC with Brownian Bridge



QMC without Brownian Bridge



Standard Monte Carlo



Final words

- QMC offers large computational savings over the standard Monte Carlo approach
- best to use randomised QMC to regain confidence intervals, at the cost of slightly poorer accuracy
- very important to use PCA or Brownian Bridge construction to create discrete Brownian increments – much better than “standard” approach which is equivalent to Cholesky factorisation of covariance matrix