Quasi-Monte Carlo for finance applications

M. B. Giles[1] F. Y. Kuo[2] I. H. Sloan[3]
B. J. Waterhouse[4]

(Received 14 August 2008; revised 24 October 2008)

Abstract

Monte Carlo methods are used extensively in computational finance to estimate the price of financial derivative options. We review the use of quasi-Monte Carlo methods to obtain the same accuracy at a much lower computational cost, and focus on three key ingredients: the generation of Sobol' and lattice points, reduction of effective dimension using the principal component analysis approach at full potential, and randomization by shifting or digital shifting to give an unbiased estimator with a confidence interval. Our aim is to provide a starting point for finance practitioners new to quasi-Monte Carlo methods.

Contents

http://anziamj.austms.org.au/ojs/index.php/ANZIAMJ/article/view/1440 gives this article, © Austral. Mathematical Soc. 2008. Published November 12, 2008. ISSN 1446-8735. (Print two pages per sheet of paper.)

1 Introduction

Many problems from mathematical finance are expressed as the expected value of some payoff function depending on quantities, such as stock prices, which satisfy stochastic differential equations driven by underlying Brownian motion. The expectation is a very high dimensional integral, with the dimension being the product of the number of Brownian motions and the number of time steps in the discretization. For example, the valuation of a parcel of mortgage backed securities for a 30 year loan with monthly repayment opportunities yields a 360-dimensional integral; while daily monitoring of 80 stocks on trading days for a period of five years leads to an integral with more than one million variables.

At present practitioners mostly tackle high dimensional problems with complex payoffs by a *Monte Carlo* (MC) approach: a series of Brownian paths are generated randomly to simulate the evolution of stock prices over time, and the expected value is then estimated by averaging over all sample paths. The law of large numbers guarantees the convergence of the estimate as the number of paths N is increased, but this convergence is slow, with the

root mean square (r.m.s.) error being $\mathcal{O}(N^{-1/2})$.

We review the alternative *quasi-Monte Carlo* (QMC) approach, in which the randomly generated paths are replaced by carefully constructed paths in order to improve the rate of convergence. The idea of considering QMC methods in computational finance is far from new [6, 8, e.g.], but in this article we discuss the key ingredients which are essential for achieving the full potential of these methods.

2 Elements of quasi-Monte Carlo

MC and QMC methods both approximate an integral over the unit cube

$$\int_{[0,1]^d} f(\boldsymbol{x})\,d\boldsymbol{x} \quad \text{by} \quad \frac{1}{N}\sum_{i=0}^{N-1} f(\boldsymbol{x}^{(i)})\,.$$

Their difference is in the choice of the points $\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N-1)}$. The MC points are independent and identically distributed uniform random vectors from the unit cube, whereas the QMC points are chosen deterministically to be 'more uniform than random'. While the r.m.s. MC error is $\mathcal{O}(N^{-1/2})$, QMC errors can have a faster rate of convergence, often close to $\mathcal{O}(N^{-1})$ and sometimes better.

QMC error bounds typically take the form of a *discrepancy*, which measures the quality of the points, times a measure of the variation of f. Such bounds separate the dependence on the point sets from the dependence on the integrand. In general we have no control over the integrand, and so we choose the QMC points to make the discrepancy as small as possible. QMC point sets with discrepancy $\mathcal{O}(N^{-1}(\log N)^d)$ or better are collectively known as *low-discrepancy point sets*. Recent research has focussed on two different strategies for achieving high uniformity of the points in the unit cube.

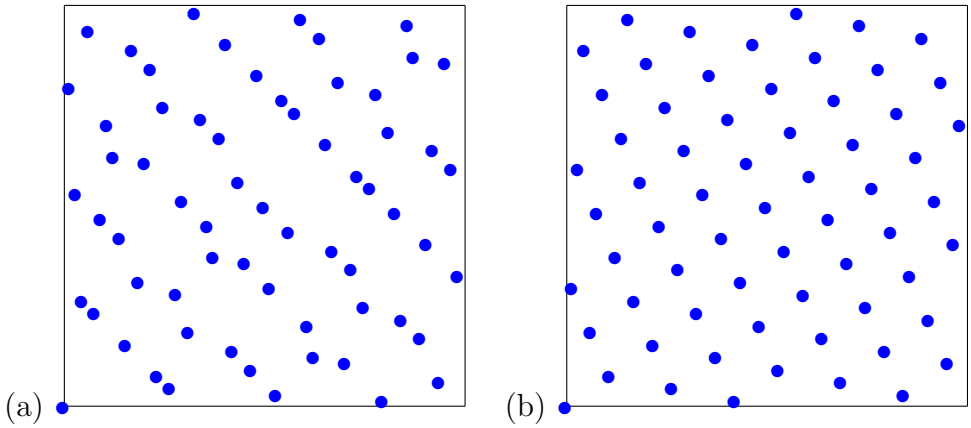

FIGURE 1: (a) Sobol' points; (b) Lattice points.

Digital nets such as Sobol' points The concept behind digital nets is to have the right number of points in various sub-divisions of the unit cube [12, e.g.]. Figure 1(a) shows the first 64 points of a two-dimensional Sobol' sequence, which is an example of a digital $(0, 6, 2)$-net in base 2. If we divide the unit square into 64 rectangles of the same shape and size, by successively bisecting m times in one direction and $6-m$ times in the other, then each rectangle will include exactly one point, for any value of m.

The construction of Sobol' points requires distinct *primitive polynomials* and so-called *direction numbers*[1] [9, e.g.]. Once the direction numbers $v_{j,1}, v_{j,2}, \ldots$ for dimension j are obtained, the Sobol' points in dimension j in the Gray code order are obtained by taking $x_j^{(0)} = 0$,

$$x_j^{(i+1)} = x_j^{(i)} \oplus v_{j,c_i}\,, \quad i = 0, 1, \ldots\,,$$

where c_i is the index of the first 0 bit from the right in the binary representation of $i = (\cdots i_3 i_2 i_1)_2$, and $\oplus$ denotes the bit-wise exclusive-or operation.

[1] http://www.maths.unsw.edu.au/~fkuo/sobol/

Lattice rules One way to visualize a lattice point set is to think of a sheared product grid where the axes have been stretched and rotated under certain constraints. A lattice rule (of rank 1) is uniquely specified by its *generating vector* $\boldsymbol{z} = (z_1, z_2, \ldots, z_d)^T$, which is an integer vector having no factor in common with N. The ith lattice point is simply

$$\boldsymbol{x}^{(i)} = \text{frac}\left(\frac{i}{N}\boldsymbol{z}\right), \quad i = 0, 1, \ldots, N-1,$$

where frac$(\cdot)$ means to replace each component of a vector by its fractional part, for example, $\text{frac}(3.7, 1.2)^T = (0.7, 0.2)^T$. Figure 1(b) shows the two-dimensional lattice rule with $N = 64$ and $\boldsymbol{z} = (1, 19)^T$. The above formula gives a lattice point set of a fixed size N. To obtain an infinite sequence of lattice points, we take

$$\boldsymbol{x}^{(i)} = \text{frac}\left(\psi(i)\,\boldsymbol{z}\right), \quad i = 0, 1, \ldots,$$

where $\psi(i)$ is obtained by mirroring the bits of i at the binary point, for example, if $i = 6 = 110_2$ then $\psi(i) = 0.011_2 = 0.375$.

Good lattice rule generating vectors can be constructed *component by component*[2] [4, 5, 11, e.g.]. These constructions require some input parameters known as *weights* [15, e.g.], which are chosen to model the dimension structure of the integrands. With the wisdom of hindsight, it now seems that the lattice rules used for many past experiments were poorly chosen.

3 Basic finance problem formulation

We now demonstrate how to formulate a standard finance problem as an integral over the unit cube, to which we apply QMC methods.

[2] http://www.maths.unsw.edu.au/~fkuo/lattice/

Single asset path dependent We assume the Black–Scholes model that the price of a stock follows a geometric Brownian motion with d equally spaced time steps in the time interval $[0, T]$. Thus the stock price at time $t_j = jT/d$ is

$$S_j = S_0 \exp\left[(r - \tfrac{1}{2}\sigma^2)\, t_j + \sigma w_j\right],$$

where S_0 is the stock price at time 0, r is the risk-free interest rate, σ is the volatility, and the vector $\boldsymbol{w} = (w_1, w_2, \ldots, w_d)^T$ corresponds to a Brownian path. Since $\boldsymbol{w}$ is normally distributed with mean zero and covariance matrix $\boldsymbol{\Sigma} = [\min(t_i, t_j)]_{i,j=1}^d$ which is symmetric and positive definite (SPD), the expected value of some payoff function $g(\boldsymbol{w})$ is the d-dimensional integral

$$\begin{aligned}
\mathbb{E}(\text{Payoff}) &= \int_{\mathbb{R}^d} g(\boldsymbol{w}) \frac{\exp(-\frac{1}{2}\boldsymbol{w}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{w})}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma})}}\, d\boldsymbol{w} \\
&= \int_{\mathbb{R}^d} g(\mathbf{A}\boldsymbol{z}) \frac{\exp(-\frac{1}{2}\boldsymbol{z}^T\boldsymbol{z})}{\sqrt{(2\pi)^d}}\, d\boldsymbol{z} \\
&= \int_{[0,1]^d} g[\mathbf{A}\Phi^{-1}(\boldsymbol{x})]\, d\boldsymbol{x},
\end{aligned}$$

where the matrix $\mathbf{A}$ comes from the factorization $\boldsymbol{\Sigma} = \mathbf{A}\mathbf{A}^T$, and $\Phi^{-1}(\cdot)$ denotes the inverse normal cumulative distribution function which is to be applied component-wise to a vector or matrix. In effect, we use the substitution $\boldsymbol{w} = \mathbf{A}\boldsymbol{z}$ to get independent standard normal variables $\boldsymbol{z}$, and the change of variable $\boldsymbol{z} = \Phi^{-1}(\boldsymbol{x})$ then maps the resulting integral into the unit cube $[0, 1]^d$. QMC methods approximate this integral by

$$\frac{1}{N}\sum_{i=0}^{N-1} g\left[\mathbf{A}\Phi^{-1}(\boldsymbol{x}^{(i)})\right],$$

with $\boldsymbol{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)})$ for $i = 0, 1, \ldots, N-1$. The factorization of $\boldsymbol{\Sigma} = \mathbf{A}\mathbf{A}^T$ is not unique. In the next section we discuss ways of carrying out the factorization.

Multi asset path dependent Assume now that there are s stocks with constant volatilities σ_i and correlation $\rho_{i,j}$, $i,j = 1,2,\ldots,s$. The covariance between these s stocks is given by the SPD matrix $\boldsymbol{\Lambda} = [\sigma_i \rho_{i,j} \sigma_j]_{i,j=1}^{s}$. Letting $\boldsymbol{w}$ be a ds-dimensional vector containing the Brownian motions of all stocks, the expected value of some payoff function $g(\boldsymbol{w})$ is the ds-dimensional integral

$$\begin{aligned}
\mathbb{E}(\text{Payoff}) &= \int_{\mathbb{R}^{ds}} g(\boldsymbol{w}) \frac{\exp[-\frac{1}{2}\boldsymbol{w}^T(\boldsymbol{\Sigma}\otimes\boldsymbol{\Lambda})^{-1}\boldsymbol{w}]}{\sqrt{(2\pi)^{ds}\det(\boldsymbol{\Sigma}\otimes\boldsymbol{\Lambda})}}\,d\boldsymbol{w} \\
&= \int_{\mathbb{R}^{ds}} g[(\mathbf{A}\otimes\mathbf{B})\boldsymbol{z}] \frac{\exp(-\frac{1}{2}\boldsymbol{z}^T\boldsymbol{z})}{\sqrt{(2\pi)^{ds}}}\,d\boldsymbol{z} \\
&= \int_{[0,1]^{ds}} g[(\mathbf{A}\otimes\mathbf{B})\Phi^{-1}(\boldsymbol{x})]\,d\boldsymbol{x}\,,
\end{aligned}$$

where $\boldsymbol{\Sigma} = \mathbf{A}\mathbf{A}^T$ and $\boldsymbol{\Lambda} = \mathbf{B}\mathbf{B}^T$, and $\mathbf{A}\otimes\mathbf{B}$ denotes the *Kronecker product* of the $d \times d$ matrix $\mathbf{A}$ and the $s \times s$ matrix $\mathbf{B}$,

$$\mathbf{A}\otimes\mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1d}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{d1}\mathbf{B} & \cdots & a_{dd}\mathbf{B} \end{bmatrix}.$$

QMC methods approximate this integral by

$$\frac{1}{N}\sum_{i=0}^{N-1} g\left[(\mathbf{A}\otimes\mathbf{B})\Phi^{-1}(\boldsymbol{x}^{(i)})\right],$$

with $\boldsymbol{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_{ds}^{(i)})$ being a ds-dimensional point.

4 Dimension reduction through principal component analysis

Contemporary wisdom has it that QMC works well for many problems in finance because such problems have *low effective dimension* [3, e.g.], meaning either that the integrand can be well approximated by a sum of terms each depending only on a small number of variables, or that the integrand mainly depends on a few leading variables. This property rests critically on how the variables are chosen, and in particular, on how we factorize the covariance matrices, since the quality of all QMC methods is better for earlier variables.

Single asset path dependent The most straightforward factorization of the SPD covariance matrix is to take $\mathbf{A}$ to be the Cholesky factor of $\mathbf{\Sigma}$; this corresponds to the standard approach which generates the Brownian path incrementally. The *Brownian Bridge* approach [3, e.g.] constructs the end point of the Brownian path first and then successively refines the midpoint of each interval; this leads to a different matrix $\mathbf{A}$ which usually reduces the effective dimension. In the *Principal Component Analysis* (PCA) approach [1, e.g.], we take

$$\mathbf{A} = \begin{bmatrix} \sqrt{\lambda_1}\boldsymbol{\eta}_1 & \cdots & \sqrt{\lambda_d}\boldsymbol{\eta}_d \end{bmatrix} ,$$

that is, the jth column of $\mathbf{A}$ is $\sqrt{\lambda_j}\boldsymbol{\eta}_j$, where $(\lambda_j, \boldsymbol{\eta}_j)_{j=1}^d$ denotes the eigenpairs of $\mathbf{\Sigma}$, with ordered eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$ and unit-length column eigenvectors $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_d$.

For many finance problems the PCA approach often leads to the optimal factorization. However, there is a common misconception that the PCA approach is much more costly than the standard and Brownian Bridge approaches because

1. a general matrix-vector product $\mathbf{A}\Phi^{-1}(\boldsymbol{x})$ requires $\mathcal{O}(d^2)$ operations

whereas the other two approaches can be computed directly in $\mathcal{O}(d)$ operations without the need to perform matrix-vector multiplication,

2. in general the computation of the eigenpairs requires $\mathcal{O}(d^3)$ operations.

Fortunately the eigenpairs for this covariance matrix $\boldsymbol{\Sigma} = [\min(t_i, t_j)]_{i,j=1}^{d}$ with equally spaced time steps are known explicitly [8, e.g.]. In particular, the eigenvectors are sine functions and thus the matrix-vector product $\mathbf{A}\Phi^{-1}(\boldsymbol{x})$ can be evaluated using *Discrete Sine Transform* (DST) in $\mathcal{O}(d \log d)$ operations [14], which is beneficial when d is large.

Multi asset path dependent We factorize the time covariance matrix $\boldsymbol{\Sigma} = \mathbf{A}\mathbf{A}^\intercal$ following the PCA approach as described above, and do an analogous factorization for the asset covariance matrix $\boldsymbol{\Lambda} = \mathbf{B}\mathbf{B}^\intercal$, that is,

$$\mathbf{B} = \begin{bmatrix} \sqrt{\mu_1}\xi_1 & \cdots & \sqrt{\mu_s}\xi_s \end{bmatrix},$$

where $(\mu_j, \xi_j)_{j=1}^{s}$ denotes the eigenpairs of $\boldsymbol{\Lambda}$ with ordered eigenvalues and unit-length column eigenvectors.

We see that the naive multiplication of $(\mathbf{A}\otimes\mathbf{B})\Phi^{-1}(\boldsymbol{x})$ requires $\mathcal{O}(d^2 s^2)$ operations which would be prohibitive for large d and s. Re-arranging the elements of $\boldsymbol{x}$ as a $d \times s$ matrix $\mathbf{X}$ (using row major), we correspondingly interpret

$$(\mathbf{A} \otimes \mathbf{B})\Phi^{-1}(\boldsymbol{x}) \quad \text{as} \quad \mathbf{A}\Phi^{-1}(\mathbf{X})\,\mathbf{B}^\intercal .$$

Multiplication with $\mathbf{A}$ can be done in $\mathcal{O}(ds \log d)$ operations using DST as discussed above. Multiplication with $\mathbf{B}$ requires the usual $\mathcal{O}(ds^2)$ operations, but this is usually acceptable since s is typically much smaller than d.

To achieve the full benefit from the PCA ordering, we need to arrange the variables in the $d \times s$ matrix $\mathbf{X}$ with respect to the ordering of the products

$$\{\lambda_i \mu_j\}_{i=1,\ldots,d,\, j=1,\ldots,s},$$

which are the eigenvalues of the full $ds \times ds$ matrix $\mathbf{\Sigma} \otimes \mathbf{\Lambda}$. We sort the products $\lambda_i \mu_j$ and apply a priori the corresponding 'permutation' to the components of the *generating vector* for lattice rules, or to the *primitive polynomials* and *direction numbers* for Sobol' points. Apart from the setup cost, this permutation does not introduce additional cost to each evaluation of the payoff function, and we have the optimal ordering of the ds variables. The idea of applying a permutation a priori for obtaining the optimal PCA ordering is new in this article.

5 Error estimation via randomization

The r.m.s. MC error is easily obtained by estimating the variance of the function (see Table 1). On the other hand, QMC methods, although having a faster rate of convergence, lack a practical error estimate. 'Randomized' QMC methods [6, e.g.] combine the best of both worlds.

Shifting 'Shifting' is the simplest form of randomization, and it preserves the lattice structure. The idea is to move all the points in the same direction by the same amount, and if any point falls outside the unit cube then it is 'wrapped' back into the cube from the opposite side.

More precisely, given a real vector $\mathbf{\Delta}$ in the unit cube, known as the *shift*, the $\mathbf{\Delta}$-shift of the QMC points $\mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(n-1)}$ consists of

$$\operatorname{frac}(\mathbf{x}^{(i)} + \mathbf{\Delta}), \quad i = 0, 1, \ldots, n-1\,.$$

We generate a number of independent random shifts $\mathbf{\Delta}_0, \mathbf{\Delta}_1, \ldots, \mathbf{\Delta}_{q-1}$ and form the approximations $Q_0, Q_1, \ldots, Q_{q-1}$, where Q_k is the approximation of the integral using a $\mathbf{\Delta}_k$-shift of the original QMC rule. Then we take the average $\bar{Q} = (Q_0 + Q_1 + \cdots + Q_{q-1})/q$ as our final approximation to the integral. An unbiased estimate for the standard error of $\bar{Q}$ is given in Table 1.

TABLE 1: MC versus QMC with $N = qn$ function evaluations.

MC	Randomized QMC
$\bar{M} = \frac{1}{N}\sum_{i=0}^{N-1} f(\boldsymbol{x}^{(i)})$	$\bar{Q} = \frac{1}{q}\sum_{k=0}^{q-1} Q_k$
	$Q_k = \frac{1}{n}\sum_{i=0}^{n-1} f[\mathrm{frac}(\boldsymbol{x}^{(i)} + \boldsymbol{\Delta}_k)]$
r.m.s. error $\overset{\text{Theory}}{=} \frac{\sqrt{\text{variance of } f}}{\sqrt{N}}$	r.m.s. error $\overset{\text{Theory}}{\leq} \frac{c_\delta \, \lVert f \rVert}{\sqrt{q}\, n^{1-\delta}}$
r.m.s. error $\overset{\text{Practice}}{\approx}$ $\sqrt{\frac{1}{N(N-1)}\sum_{i=0}^{N-1}\left(f(\boldsymbol{x}^{(i)}) - \bar{M}\right)^2}$	r.m.s. error $\overset{\text{Practice}}{\approx}$ $\sqrt{\frac{1}{q(q-1)}\sum_{k=0}^{q-1}\left(Q_k - \bar{Q}\right)^2}$

The theoretical QMC error bound assumes that f belongs to some *weighted* Sobolev space with mixed first derivatives [15, e.g.]. Typically we take n in the thousands or more while keeping q small around 10 or 20.

Scrambling such as digital shift 'Scrambling' is another popular but more complicated randomization method; it preserves the digital net structure. There is a simple form of scrambling called *digital shift*, which is similar to shifting: we just need to replace $\mathrm{frac}(\boldsymbol{x}^{(i)} + \boldsymbol{\Delta})$ by

$$\boldsymbol{x}^{(i)} \oplus \boldsymbol{\Delta}, \quad i = 0, 1, \ldots, n-1,$$

where $\oplus$ denotes the exclusive-or operator as before.

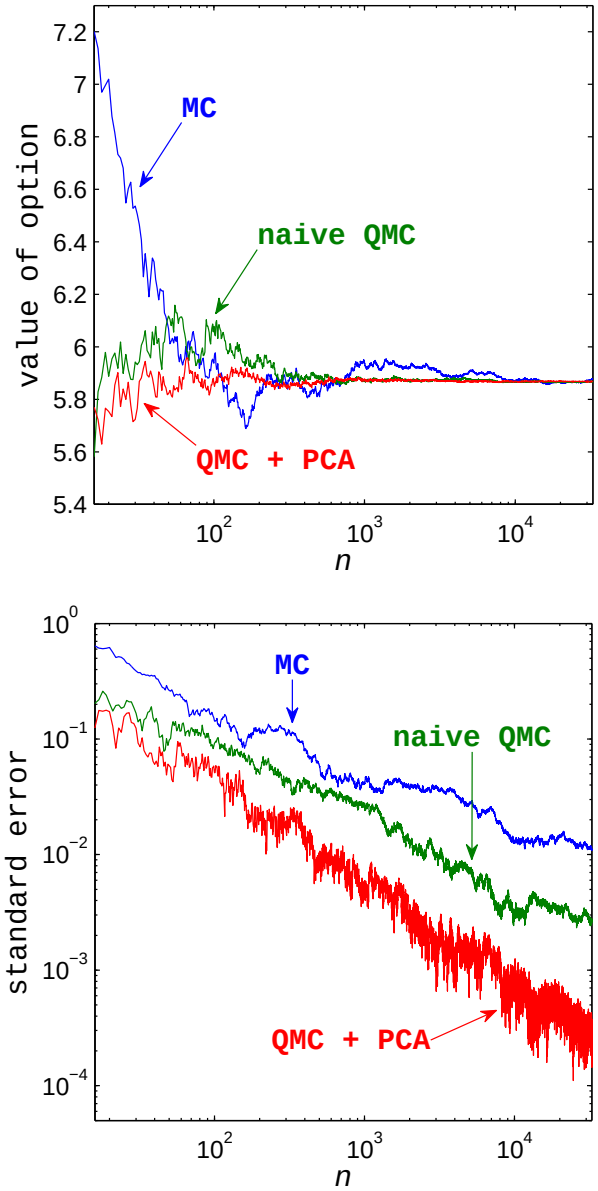

FIGURE 2: Value of arithmetic Asian option and standard error with $N = 10n$.

6 Discussion

To provide a glimpse of how these techniques improve upon the efficiency of MC, we present in Figure 2 the convergence of MC versus QMC for an arithmetic Asian option with $s = 5$ stocks, $d = 256$ time discretizations and $N = 327,680$ function evaluations. The QMC method is a rank 1 lattice rule with $n = 32,768$ points and $q = 10$ random shifts. The figure shows the value of option and the estimated standard error as n increases from 16 to $32,768$.

The take home message is that a lot more can be gained from QMC methods if the points are not naively substituted, but rather insightfully deployed. Although the PCA approach appears to be the optimal strategy in many cases, we stress that it is not the optimal approach for all finance problems [16]. We add that variance reduction techniques for MC are also beneficial for QMC.

QMC for finance applications is an active area of research; the latest advances include multilevel QMC [7], adaptive QMC [13], QMC for jump diffusion models [2], and fast PCA with unequal time steps [10].

Acknowledgements The support of the Engineering and Physical Sciences Research Council and the Australian Research Council is gratefully acknowledged. We thank Dirk Nuyens and Stephen Joe for valuable comments.

References

[1] P. Acworth, M. Broadie, and P. Glasserman, A comparison of some Monte Carlo and quasi-Monte Carlo techniques for option pricing, in: *Monte Carlo and quasi-Monte Carlo methods 1996* (P. Hellekalek, G.

Larcher, H. Niederreiter, and P. Zinterhof, eds.), Springer Verlag, Berlin, 1–18 (1998). C315

[2] J. Baldeaux, QMC for finance beyond Black-Scholes, submitted to *ANZIAM J. Proc. CTAC 2008*. C320

[3] R. E. Caflisch, W. Morokoff, and A. B. Owen, Valuation of mortgage-backed securities using Brownian bridges to reduce effective dimension, *J. Comp. Finance* **1**, 27–46 (1997). http://www.thejournalofcomputationalfinance.com/public/showPage.html?page=919 C315

[4] R. Cools, F. Y. Kuo, and D. Nuyens, Constructing embedded lattice rules for multivariate integration, *SIAM J. Sci. Comput.* **28**, 2162–2188 (2006). doi:10.1137/06065074X C312

[5] J. Dick, F. Pillichshammer, and B. J. Waterhouse, The construction of good extensible rank-1 lattices, *Math. Comp.* **77**, 2345–2373 (2008). doi:10.1090/S0025-5718-08-02009-7 C312

[6] P. L'Ecuyer, Quasi-Monte Carlo methods in finance, in: *Proceedings of the 2004 Winter Simulation Conference* (R. G. Ingalls, M.D. Rossetti, J. S. Smith,and B. A. Peters, eds.), IEEE Computer Society Press, Los Alamitos, 1645–1655 (2004). doi:10.1109/WSC.2004.1371512 C310, C317

[7] M. B. Giles and B. J. Waterhouse, Multilevel quasi-Monte Carlo path simulation, in preparation. C320

[8] P. Glasserman, *Monte Carlo methods in financial engineering*, Springer–Verlag, New York, 2004. C310, C316

[9] S. Joe and F. Y. Kuo, Constructing Sobol' sequences with better two-dimensional projections, *SIAM J. Sci. Comput.* **30**, 2635–2654 (2008). doi:10.1137/070709359 C311

[10] J. Keiner and B. J. Waterhouse, Fast implementation of the PCA method for finance problems with unequal time steps, in preparation. C320

[11] F. Y. Kuo and I. H. Sloan, Lifting the curse of dimensionality, *Notices Amer. Math. Soc.* **52**, 1320–1329 (2005). http://www.ams.org/notices/200511/index.html C312

[12] H. Niederreiter, *Random number generation and quasi-Monte Carlo methods*, SIAM, Philadelphia, 1992. C311

[13] D. Nuyens and B. J. Waterhouse, Adaptive quasi-Monte Carlo in finance, in preparation. C320

[14] K. Scheicher, Complexity and effective dimension of discrete Lévy areas, *J. Complexity* **23**, 152–168 (2007). doi:10.1016/j.jco.2006.12.006 C316

[15] I. H. Sloan, X. Wang, and H. Woźniakowski, Finite-order weights imply tractability of multivariate integration, *J. Complexity* **20**, 46–74 (2004). doi:10.1016/j.jco.2003.11.003 C312, C318

[16] X. Wang and I. H. Sloan, Quasi-Monte Carlo methods in financial enginnering: an equivalence principle and dimension reduction, in preparation. C320

Author addresses

1. **M. B. Giles**, Oxford University Mathematical Institute, 24–29 St Giles, Oxford, OX1 3LB, ENGLAND.
mailto:mike.giles@maths.ox.ac.uk

2. **F. Y. Kuo**, School of Mathematics and Statistics, University of New South Wales, Sydney, NSW 2052, AUSTRALIA.
mailto:f.kuo@unsw.edu.au

3. **I. H. Sloan**, School of Mathematics and Statistics, University of New South Wales, Sydney, NSW 2052, Australia.
mailto:i.sloan@unsw.edu.au

4. **B. J. Waterhouse**, School of Mathematics and Statistics, University of New South Wales, Sydney, NSW 2052, Australia.
mailto:benjw@maths.unsw.edu.au