Proceedings of the 2019 Winter Simulation Conference
N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, eds.

ARRAY-RQMC FOR OPTION PRICING UNDER STOCHASTIC VOLATILITY MODELS

Amal Ben Abdellah
Pierre L'Ecuyer
Florian Puchhammer

Département d'Informatique et de Recherche Opérationnelle
Pavillon Aisenstadt, Université de Montréal
C.P. 6128, Succ. Centre-Ville
Montréal (Québec), H3C 3J7, CANADA

ABSTRACT

Array-RQMC has been proposed as a way to effectively apply randomized quasi-Monte Carlo (RQMC) when simulating a Markov chain over a large number of steps to estimate an expected cost or reward. The method can be very effective when the state of the chain has low dimension. For pricing an Asian option under an ordinary geometric Brownian motion model, for example, Array-RQMC can reduce the variance by factors in the millions. In this paper, we show how to apply this method and we study its effectiveness in case the underlying process has stochastic volatility. We show that Array-RQMC can also work very well for these models, even if it requires RQMC points in larger dimension. We examine in particular the variance-gamma, Heston, and Ornstein-Uhlenbeck stochastic volatility models, and we provide numerical results.

1 INTRODUCTION

Quasi-Monte Carlo (QMC) and randomized QMC (RQMC) methods can improve efficiency significantly when estimating an integral in a moderate number of dimensions, but their use for simulating Markov chains over a large number of steps has been limited so far. The array-RQMC method, developed for that purpose, has been shown to work well for some chains having a low-dimensional state. It simulates an array of n copies of the Markov chain so that each chain follows its exact distribution, but the copies are not independent, and the empirical distribution of the states at any given step of the chain is a "low-discrepancy" approximation of the exact distribution. At each step, the n chains (or states) are matched one-to-one to a set of n RQMC points whose dimension is the dimension of the state plus the number of uniform random numbers required to advance the chain by one more step. The first coordinates of the points are used to match the states to the points and the other coordinates provide the random numbers needed to determine the next state. When the chains have a large-dimensional state, the dimension used for the match can be reduced via a mapping to a lower-dimensional space. Then the matching is performed by sorting both the points and the chains. When the dimension of the state exceeds 1, this matching is done via a multivariate sort. The main idea is to evolve the array of chains in a way that from step to step, the empirical distribution of the states keeps its low discrepancy. For further details on the methodology, sorting strategies, convergence analysis, applications, and empirical results, we refer the reader to Lécot and Tuffin (2004), Demers et al. (2005), L'Ecuyer et al. (2006), L'Ecuyer et al. (2007), L'Ecuyer et al. (2008), El Haddad et al. (2008), L'Ecuyer et al. (2009), El Haddad et al. (2010), Dion and L'Ecuyer (2010), L'Ecuyer and Sanvido (2010), Gerber and Chopin (2015), L'Ecuyer et al. (2018), and the other references given there.

The aim of this paper is to examine how Array-RQMC can be applied for option pricing under a stochastic volatility process such as the variance gamma, Heston, and Ornstein-Uhlenbeck models. We explain and compare various implementation alternatives, and report empirical experiments to assess the (possible) gain in efficiency and convergence rate. A second objective is for the WSC community to become better aware of this method, which can have numerous other applications.

Array-RQMC has already been applied for pricing Asian options when the underlying process evolves as a geometric Brownian motion (GBM) with fixed volatility (L'Ecuyer et al. 2009; L'Ecuyer et al. 2018). In that case, the state is two-dimensional (it contains the current value of the GBM and its running average) and a single random number is needed at each step, so the required RQMC points are three-dimensional. In their experiments, L'Ecuyer et al. (2018) observed an empirical variance of the average payoff that decreased approximately as $\mathscr{O}(n^{-2})$ for Array-RQMC, in a range of reasonable values of n, compared with $\mathscr{O}(n^{-1})$ for independent random points (Monte Carlo). For $n = 2^{20}$ (about one million chains), the variance ratio between Monte Carlo and Array-RQMC was around 2 to 4 millions.

In view of this spectacular success, one wonders how well the method would perform when the underlying process is more involved, e.g., when it has stochastic volatility. This is relevant because stochastic volatility models are more realistic than the plain GBM model (Madan and Seneta 1990; Madan et al. 1998). Success is not guaranteed because the dimension of the required RQMC points is larger. For the Heston model, for example, the RQMC points must be five-dimensional instead of three-dimensional, because the state has three dimensions and we need two uniform random numbers at each step. It is unclear a priori if there will be any significant variance reduction for reasonable values of n.

The remainder is organized as follows. In Section 2, we state our general Markov chain model and provide background on the Array-RQMC algorithm, including matching and sorting strategies. In Section 3, we describe our experimental setting, and the types of RQMC point sets that we consider. Then we study the application of Array-RQMC under the variance-gamma model in Section 4, the Heston model in Section 5, and the Ornstein-Uhlenbeck model in Section 6. We end with a conclusion.

2 BACKGROUND: MARKOV CHAIN MODEL, RQMC, AND ARRAY-RQMC

The option pricing models considered in this paper fit the following framework, which we use to summarize the Array-RQMC algorithm. We have a discrete-time *Markov chain* $\{X_j,\, j \geq 0\}$ defined by a stochastic recurrence over a measurable state space $\mathscr{X}$:

$$X_0 = x_0, \qquad \text{and} \qquad X_j = \varphi_j(X_{j-1}, \mathbf{U}_j), \quad j = 1, \ldots, \tau,$$

where $x_0 \in \mathscr{X}$ is a deterministic initial state, $\mathbf{U}_1, \mathbf{U}_2, \ldots$ are independent random vectors uniformly distributed over the d-dimensional unit cube $(0,1)^d$, the functions $\varphi_j : \mathscr{X} \times (0,1)^d \to \mathscr{X}$ are measurable, and τ is a fixed positive integer (the time horizon). The goal is:

$$\text{Estimate} \quad \mu_{\mathrm{y}} = \mathbb{E}[Y], \qquad \text{where} \quad Y = g(X_\tau)$$

and $g : \mathscr{X} \to \mathbb{R}$ is a *cost* (or reward) function. Here we have a cost only at the last step but in general there can be a cost function for each step and Y would be the sum of these costs (L'Ecuyer et al. 2008).

Crude Monte Carlo estimates μ by the average $\bar{Y}_n = \frac{1}{n}\sum_{i=0}^{n-1} Y_i$, where $Y_0, \ldots, Y_{n-1}$ are n independent realizations of Y. One has $\mathbb{E}[\bar{Y}_n] = \mu_{\mathrm{y}}$ and $\mathrm{Var}[\bar{Y}_n] = \mathrm{Var}[Y]/n$, assuming that $\mathbb{E}[Y^2] = \sigma_{\mathrm{y}}^2 < \infty$. Note that the simulation of each realization of Y requires a vector $\mathbf{V} = (\mathbf{U}_1, \ldots, \mathbf{U}_\tau)$ of $d\tau$ independent uniform random variables over $(0,1)$, and crude Monte Carlo produces n independent replicates of this random vector.

Randomized quasi-Monte Carlo (RQMC) replaces the n independent realizations of $\mathbf{V}$ by n *dependent* realizations, which form an RQMC point set in $d\tau$ dimensions. That is, each $\mathbf{V}_i$ has the uniform distribution over $[0,1)^{d\tau}$, and the point set $P_n = \{V_0, \ldots, V_{n-1}\}$ covers $[0,1)^{d\tau}$ more evenly than typical independent random points. With RQMC, $\bar{Y}_n$ remains an unbiased estimator of μ, but its variance can be much smaller, and can converge faster than $\mathscr{O}(1/n)$ under certain conditions. For more details, see Dick and Pillichshammer (2010), L'Ecuyer and Lemieux (2000), L'Ecuyer (2009), L'Ecuyer (2018), for example. However, when $d\tau$ is large, standard RQMC typically becomes ineffective, in the sense that it does not bring much variance reduction unless the problem has special structure.

Array-RQMC is an alternative approach developed specifically for Markov chains (L'Ecuyer et al. 2006; L'Ecuyer et al. 2008; L'Ecuyer et al. 2018). To explain how it works, let us first suppose for simplicity (we will relax it later) that there is a mapping $h : \mathscr{X} \to \mathbb{R}$, that assigns to each state a *value* (or score) which summarizes in a single real number the most important information that we should retain from that state (like the value function in stochastic dynamic programming). This h is called the *sorting function*. The algorithm simulates n (dependent) realizations of the chain "in parallel". Let $X_{i,j}$ denote the state of chain i at step j, for $i = 0,\dots,n-1$ and $j = 0,\dots,\tau$. At step j, the n chains are sorted by increasing order of their values of $h(X_{i,j-1})$, the n points of an RQMC point set in $d+1$ dimensions are sorted by their first coordinate, and each point is matched to the chain having the same position in this ordering. Each chain i is then moved forward by one step, from state $X_{i,j-1}$ to state $X_{i,j}$, using the d other coordinates of its assigned RQMC point. Then we move on to the next step, the chains are sorted again, and so on.

The sorting function can in fact be more general and have the form $h : \mathscr{X} \to \mathbb{R}^c$ for some small integer $c \geq 1$. Then the mapping between the chains and the points must be realized in a c-dimensional space, i.e., via some kind of c-dimensional multivariate sort. The RQMC points then have $c+d$ coordinates, and are sorted with the same c-dimensional multivariate sort based on their first c coordinates, and mapped to the corresponding chains. The other d coordinates are used to move the chains ahead by one step. In practice, the first c coordinates of the RQMC points do not have to be randomized at each step; they are usually fixed and the points are already sorted in the correct order based on these coordinates.

Some multivariate sorts are described and compared by El Haddad et al. (2008), L'Ecuyer et al. (2009), L'Ecuyer (2018). For example, in a *multivariate batch sort*, we select positive integers $n_1,\dots,n_c$ such that $n = n_1 \dots n_c$. The states are first sorted by their first coordinate in n_1 packets of size n/n_1, then each packet is sorted by the second coordinate into n_2 packets of size n/n_1n_2, and so on. The RQMC points are sorted in exactly the same way, based on their first c coordinates. In the *multivariate split sort*, we assume that $n = 2^e$ and we take $n_1 = n_2 = \cdots = n_e = 2$. That is, we first split the points in 2 packets based on the first coordinate, then split each packet in two by the second coordinate, and so on. If $e > c$, after c splits we get back to the first coordinate and continue.

Examples of heuristic sorting functions $h : \mathscr{X} \to \mathbb{R}$ are given in (L'Ecuyer et al. 2008; L'Ecuyer et al. 2018). Wächter and Keller (2008) and Gerber and Chopin (2015) suggested to first map the c-dimensional states to $[0,1]^c$ and then use a space filling curve in $[0,1]^c$ to map them to $[0,1]$, which provides a total order. Gerber and Chopin (2015) proposed to map the states to $[0,1]^c$ via a component-wise rescaled logistic transformation, then order them with a Hilbert space-filling curve. See L'Ecuyer et al. (2018) for a more detailed discussion. Under smoothness conditions, they proved that the resulting unbiased Array-RQMC estimator has $o(1/n)$ variance, which beats the $\mathscr{O}(1/n)$ Monte Carlo rate.

Algorithm 1 states the Array-RQMC procedure in our setting. Indentation delimits the scope of the "**for**" loops. For any choice of sorting function h, the average $\hat{\mu}_{\text{arqmc},n} = \bar{Y}_n$ returned by this algorithm is always an *unbiased* estimator of μ. An unbiased estimator of $\text{Var}[\bar{Y}_n]$ can be obtained by making m independent realizations of $\hat{\mu}_{\text{arqmc},n}$ and computing their *empirical variance*.

Algorithm 1 : Array-RQMC Algorithm for Our Setting

```
for $i = 0,\dots,n-1$ do $X_{i,0} \leftarrow x_0$;
for $j = 1,2,\dots,\tau$ do
    Sorting: Compute an appropriate permutation $\pi_j$ of the $n$ chains, based on
        the $h(X_{i,j-1})$, to match the $n$ states with the RQMC points;
    Randomize afresh the RQMC points $\{\mathbf{U}_{0,j},\dots,\mathbf{U}_{n-1,j}\}$;
    for $i = 0,\dots,n-1$ do $X_{i,j} = \varphi_j(X_{\pi_j(i),j-1}, \mathbf{U}_{i,j})$;
return the average $\hat{\mu}_{\text{arqmc},n} = \bar{Y}_n = (1/n)\sum_{i=0}^{n-1} g(X_{i,\tau})$ as an estimate of $\mu_{\text{y}}$.
```

3 EXPERIMENTAL SETTING

For all the option pricing examples in this paper, we have an asset price that evolves as a stochastic process $\{S(t), t \geq 0\}$ and a payoff that depends on the values of this process at fixed observation times $0 = t_0 < t_1 < t_2 < ... < t_c = T$. More specifically, for given constants r (the interest rate) and K (the strike price), we consider an *European option* whose payoff is

$$Y = Y_{\mathrm{e}} = g(S(T)) = e^{-rT} \max(S(T) - K, 0)$$

and a discretely-observed *Asian option* whose payoff is

$$Y = Y_{\mathrm{a}} = g(\bar{S}) = e^{-rT} \max(\bar{S} - K, 0)$$

where $\bar{S} = (1/c)\sum_{j=1}^{c} S(t_j)$. In this second case, the running average $\bar{S}_j = (1/j)\sum_{\ell=1}^{j} S(t_\ell)$ must be kept in the state of the Markov chain. The information required for the evolution of $S(t)$ depends on the model and is given for each model in forthcoming sections. It must be maintained in the state. For the case where S is a plain GBM, the state of the Markov chain at step j can be taken as $X_j = (S(t_j), \bar{S}_j)$, a two-dimensional state, as was done in L'Ecuyer et al. (2009) and L'Ecuyer et al. (2018).

In our examples, the states are always multidimensional. To match them with the RQMC points, we will use a split sort, a batch sort, and a Hilbert-curve sort, and compare these alternatives. The Hilbert sort requires a transformation of the ℓ-dimensional states to the unit hypercube $[0,1]^\ell$. For this, we use a *logistic transformation* defined by $\psi(x) = (\psi_1(x_1), ..., \psi_\ell(x_\ell)) \in [0,1]^\ell$ for all $x = (x_1, \dots, x_\ell) \in \mathscr{X}$, where

$$\psi_j(x_j) = \left[1 + \exp\left(-\frac{x_j - \underline{x}_j}{\bar{x}_j - \underline{x}_j}\right)\right]^{-1}, \quad j = 1, ..., \ell,$$

with constants $\bar{x}_j = \mu_j + 2\sigma_j$ and $\underline{x}_j = \mu_j - 2\sigma_j$ in which μ_j and σ_j are estimates of the mean and the variance of the distribution of the jth coordinate of the state. In Section 4, we will also consider just taking a linear combination of the two coordinates, to map a two-dimensional state to one dimension.

For RQMC, we consider

(1) Independent points, which corresponds to crude Monte Carlo (MC);
(2) Stratified sampling over the unit hypercube (Stratif);
(3) Sobol' points with a random linear matrix scrambling and a digital random shift (Sobol'+LMS);
(4) Sobol' points with nested uniform scrambling (Sobol'+NUS);
(5) A rank-1 lattice rule with a random shift modulo 1 followed by a baker's transformation (Lattice+baker).

The first two are not really RQMC points, but we use them for comparison. For stratified sampling, we divide the unit hypercube into $n = k^{\ell+d}$ congruent subcubes for some integer $k > 1$, and we draw one point randomly in each subcube. For a given target n, we take k as the integer for which $k^{\ell+d}$ is closest to this target n. For the Sobol' points, we took the default direction numbers in SSJ, which are from Lemieux et al. (2004). The LMS and NUS randomizations are explained in Owen (2003) and L'Ecuyer (2009). For the rank-1 lattice rules, we used generating vectors found by Lattice Builder (L'Ecuyer and Munger 2016), using the $\mathscr{P}_2$ criterion with order-dependent weights $(0.8)^k$ for projections of order k.

For each example, each sorting method, each type of point set, and each selected value of n, we ran simulations to estimate $\mathrm{Var}[\bar{Y}_n]$. For the stratified and RQMC points, this variance was estimated by replicating the RQMC scheme $m = 100$ times independently. For a fair comparison with the MC variance $\sigma_{\mathrm{y}}^2 = \mathrm{Var}[Y]$, for these point sets we used the *variance per run*, defined as $n\mathrm{Var}[\bar{Y}_n]$. We define the *variance reduction factor* (VRF) for a given method compared with MC by $\sigma_{\mathrm{y}}^2/(n\mathrm{Var}[\bar{Y}_n])$. In each case, we fitted a linear regression model for the variance per run as a function of n, in log-log scale. We denote by $\hat{\beta}$ the regression slope estimated by this linear model.

In the remaining sections, we explain how the process $\{S(t), t \geq 0\}$ is defined in each case, how it is simulated. We show how we can apply Array-RQMC and we provide numerical results. All the experiments were done in Java using the SSJ library (L'Ecuyer and Buist 2005; L'Ecuyer 2016).

4 OPTION PRICING UNDER A VARIANCE-GAMMA PROCESS

The *variance-gamma* (VG) model was proposed for option pricing by Madan and Seneta (1990) and Madan, Carr, and Chang (1998), and further studied by Fu et al. (1998), Avramidis et al. (2003), Avramidis and L'Ecuyer (2006), for example. A VG process is essentially a Brownian process for which the time clock runs at random and time-varying speed driven by a gamma process. The VG process with parameters (θ, σ^2, ν) is defined as $Y = \{Y(t) = X(G(t)), t \geq 0\}$ where $X = \{X(t), t \geq 0\}$ is a Brownian motion with drift and variance parameters θ and σ^2, and $G = \{G(t), t \geq 0\}$ is a gamma process with drift and volatility parameters 1 and ν, independent of X. This means that $X(0) = 0$, $G(0) = 0$, both B and G have independent increments, and for all $t \geq 0$ and $\delta > 0$, we have $X(t+\delta) - X(t) \sim \text{Normal}(\delta\theta, \delta\sigma^2)$, a normal random variable with mean $\delta\theta$ and variance $\delta\sigma^2$, and $G(t+\delta) - G(t) \sim \text{Gamma}(\delta/\nu, \nu)$, a gamma random variable with mean δ and variance $\delta\nu$. The gamma process is always non-decreasing, which ensures that the time clock never goes backward. In the VG model for option pricing, the asset value follows the *geometric variance-gamma* (GVG) process $S = \{S(t), t \geq 0\}$ defined by

$$S(t) = S(0) \exp\left[(r+\omega)t + X(G(t))\right],$$

where $\omega = \ln(1 - \theta\nu - \sigma^2\nu/2)/\nu$.

To generate realizations of $\bar{S}$ for this process, we must generate $S(t_1), \ldots, S(t_\tau)$, and there are many ways of doing this. With Array-RQMC, we want to do it via a Markov chain with a low-dimensional state. The running average $\bar{S}_j$ must be part of the state, as well as sufficient information to generate the future of the path. A simple procedure for generating the path is to sample sequentially $G(t_1)$, then $Y(t_1) = X(G(t_1))$ conditional on $G(t_1)$, then $G(t_2)$ conditional on $G(t_1)$, then $Y(t_2) = X(G(t_2))$ conditional on $(G(t_1), G(t_2), Y(t_1))$, and so on. We can then compute any $S(t_j)$ directly from $Y(t_j)$.

It is convenient to view the sampling of $(G(t_j), Y(t_j))$ conditional on $(G(t_{j-1}), Y(t_{j-1}))$ as one step (step j) of the Markov chain. The state of the chain at step $j-1$ can be taken as $X_{j-1} = (G(t_{j-1}), Y(t_{j-1}), \bar{S}_{j-1})$, so we have a three-dimensional state, and we need two independent uniform random numbers at each step, one to generate $G(t_j)$ and the other to generate $Y(t_j) = X(G(t_j))$ given $(G(t_{j-1}), G(t_j), Y(t_{j-1}))$, both by inversion. Applying Array-RQMC with this setting would require a five-dimensional RQMC point set at each step, unless we can map the state to a lower-dimensional representation.

However, a key observation here is that the distribution of the increment $\Delta Y_j = Y(t_j) - Y(t_{j-1})$ depends only on the increment $\Delta_j = G(t_j) - G(t_{j-1})$ and not on $G(t_{j-1})$. This means that there is no need to memorize the latter in the state! Thus, we can define the state at step j as the two-dimensional vector $X_j = (Y(t_j), \bar{S}_j)$, or equivalently $X_j = (S(t_j), \bar{S}_j)$, and apply Array-RQMC with a four-dimensional RQMC point set if we use a two-dimensional sort for the states, and a three-dimensional RQMC point set if we map the states to a one-dimensional representation (using a Hilbert curve or a linear combination of the coordinates, for example). At step j, we generate $\Delta_j \sim \text{Gamma}((t_j - t_{j-1})/\nu, \nu)$ by inversion using a uniform random variate $U_{j,1}$, i.e., via $\Delta_j = F_j^{-1}(U_{j,1})$ where F_j is the cdf of the $\text{Gamma}((t_j - t_{j-1})/\nu, \nu)$ distribution, then ΔY_j by inversion from the normal distribution with mean $\theta\Delta_j$ and variance $\sigma^2\Delta_j$, using a uniform random variate $U_{j,2}$. Algorithm 2 summarizes this procedure. The symbol Φ denotes the standard normal cdf. We have

$$X_j = (Y(t_j), \bar{S}_j) = \varphi_j(Y(t_{j-1}), \bar{S}_{j-1}, U_{j,1}, U_{j,2})$$

where φ_j is defined by the algorithm. The payoff function is $g(X_c) = \bar{S}_c = \bar{S}$.

With this two-dimensional state representation, if we use a split sort or batch, we need four-dimensional RQMC points. With the Hilbert-curve sort, we only need three-dimensional RQMC points. We also tried a simple linear mapping $h_j : \mathbb{R}^2 \to \mathbb{R}$ defined by $h_j(S(t_j), \bar{S}_j) = b_j\bar{S}_j + (1-b_j)S(t_j)$ where $b_j = (j-1)/(\tau-1)$.

Algorithm 2 Computing $X_j = (Y(t_j), \bar{S}_j)$ given $(Y(t_{j-1}), \bar{S}_{j-1})$, for $1 \le j \le \tau$.

```
Generate $U_{j,1}, U_{j,2} \sim$ Uniform$(0,1)$, independent;
$\Delta_j = F_j^{-1}(U_{j,1}) \sim$ Gamma$((t_j - t_{j-1})/\nu, \nu)$;
$Z_j = \Phi^{-1}(U_{j,2}) \sim$ Normal$(0,1)$;
$Y(t_j) \leftarrow Y(t_{j-1}) + \theta\Delta_j + \sigma\sqrt{\Delta_j} Z_j$;
$S(t_j) \leftarrow S(0)\exp[(r+\omega)t_j + Y(t_j)]$;
$\bar{S}_j = [(j-1)\bar{S}_{j-1} + S(t_j)]/j$;
```

At each step j, this h_j maps the state X_j to a real number $h_j(X_j)$, and we sort the states by increasing order of their value of $h_j(X_j)$. It uses a convex linear combination of $S(t_j)$ and $\bar{S}_j$ whose coefficients depend on j. The rationale for the (heuristic) choice of b_j is that in the late steps (when j is near τ), the current average $\bar{S}_j$ is more important (has more predictive power for the final payoff) than the current $S(t_j)$, whereas in the early steps, the opposite is true.

We made an experiment with the following model parameters, taken from Avramidis and L'Ecuyer (2006): $\theta = -0.1436$, $\sigma = 0.12136$, $\nu = 0.3$, $r = 0.1$, $T = 240/365$, $\tau = 10$, $t_j = 24j/365$ for $j = 1, \ldots, \tau$, $K = 100$, and $S(0) = 100$. The time unit is one year, the horizon is 240 days, and there is an observation time every 24 days. The exact value of the expected payoff for the Asian option is $\mu \approx 8.36$, and the MC variance per run is $\sigma_y^2 = \mathrm{Var}[Y_a] \approx 59.40$.

Table 1: Regression slopes $\hat{\beta}$ for $\log_2 \mathrm{Var}[\hat{\mu}_n^{\mathrm{arqmc}}]$ vs $\log_2(n)$, and VRF compared with MC for $n = 2^{20}$, denoted VRF20, for the Asian option under the VG model.

Sort	Point sets	$\hat{\beta}$	VRF20
Split sort	MC	-1	1
	Stratif	-1.17	42
	Sobol'+LMS	-1.77	91,550
	Sobol'+NUS	-1.80	106,965
	Lattice+baker	-1.83	32,812
Batch sort ($n_1 = n_2$)	MC	-1	1
	Stratif	-1	42
	Sobol'+LMS	-1.71	100,104
	Sobol'+NUS	-1.54	90,168
	Lattice+baker	-1.95	58,737
Hilbert sort (with logistic map)	MC	-1	1
	Stratif	-1.43	204
	Sobol'+LMS	-1.59	68,297
	Sobol'+NUS	-1.67	79,869
	Lattice+baker	-1.55	45,854
Linear map sort	MC	-1	1
	Stratif	-1.35	192
	Sobol'+LMS	-1.64	115,216
	Sobol'+NUS	-1.75	166,541
	Lattice+baker	-1.72	68,739

Table 1 summarizes the results. For each selected sorting method and point set, we report the estimated slope $\hat{\beta}$ for the linear regression model of $\log_2 \mathrm{Var}[\hat{\mu}_n^{\mathrm{arqmc}}]$ as a function of $\log_2(n)$ obtained from $m = 100$ independent replications with $n = 2^e$ for $e = 16, \ldots, 20$, as well as the variance reduction factors (VRF) observed for $n = 2^{20}$ (about one million samples), denoted VRF20. For MC, the exact slope (or convergence rate) β is known to be $\beta = -1$. We see from the table that Array-RQMC provides much better convergence rates (at least empirically), and reduces the variance by very large factors for $n = 2^{20}$. Interestingly, the largest factors are obtained with the Sobol' points combined with our heuristic linear map sort, although

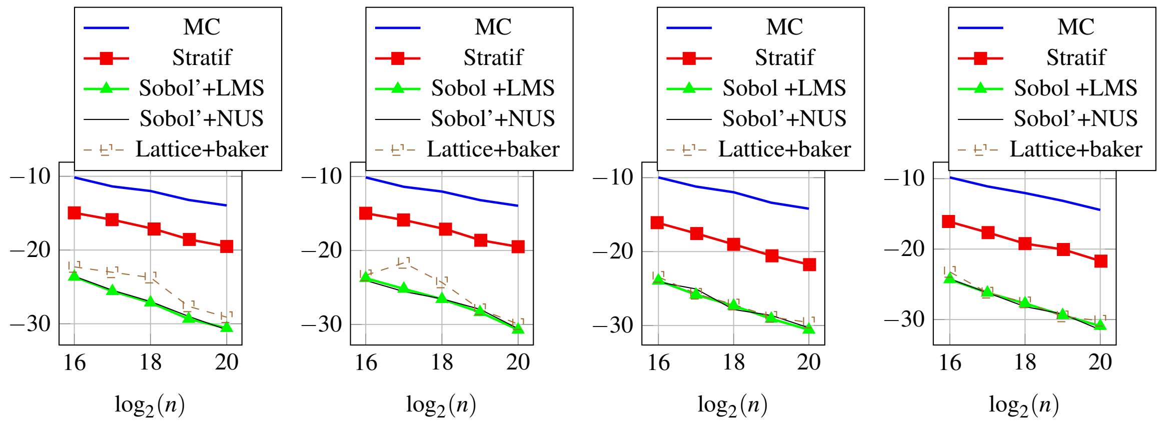

Figure 1: Plots of empirical $\log_2 \mathrm{Var}[\hat{\mu}_n^{\mathrm{arqmc}}]$ vs $\log_2(n)$ for various sorts and point sets, based on $m = 100$ independent replications. Left to right: split sort, batch sort, Hilbert sort, linear map sort.

the other sorts are also doing quite well. Figure 1 shows plots of $\log_2 \mathrm{Var}[\hat{\mu}_n^{\mathrm{arqmc}}]$ vs $\log_2(n)$ for selected sorts. It gives an idea of how well the linear model fits in each case.

There are other ways of defining the steps of the Markov chain for this example. For example, one can have one step for each Uniform$(0,1)$ random number that is generated. This would double the number of steps, from c to $2c$. We generate Δ_1 in the first step, $Y(t_1)$ in the second step, Δ_2 in the third step, $Y(t_2)$ in the fourth step, and so on. Generating a single uniform per step instead of two reduces by 1 the dimension of the required RQMC point set. At odd step numbers, when we generate a Δ_j, the state can still be taken as $(Y(t_{j-1}), \bar{S}_{j-1})$ and we only need three-dimensional RQMC points, so we save one dimension. But at even step numbers, we need Δ_j to generate $Y(t_j)$, so we need a three-dimensional state $(Y(t_{j-1}), \Delta_j, \bar{S}_{j-1})$ and four-dimensional RQMC points. We tried this approach and it did not perform better than the one described earlier, with two uniforms per step. It is also more complicated to implement.

Avramidis et al. (2003), Avramidis and L'Ecuyer (2006) describe other ways of simulating the VG process, for instance Brownian and gamma bridge sampling (BGBS) and difference of gammas bridge sampling (DGBS). BGBS generates first $G(t_c)$ then $Y(t_c)$, then conditional on this it generates $G(t_{c/2})$ then $Y(t_{c/2})$ (assuming that c is even), and so on. DGBS writes the VG process Y as a difference of two independent gamma processes and simulate both using the bridge idea just described: first generate the values of the two gamma processes at t_c, then at $t_{c/2}$, etc. When using classical RQMC, these sampling methods brings an important variance reduction compared with the sequential one we use here for our Markov chain. Combining them with Array-RQMC is impractical, however, because the dimension of the state (the number of values that we need to remember and use for the sorting) grows up to about c, which is much to high, and the implementation is much more complicated. Also, these methods are effective when c is a power or 2 and $t_j = jT/c$, because then the conditional sampling for the gamma process is always from a symmetrical beta distribution and there is an efficient inversion method for that (L'Ecuyer and Simard 2006), but they are less effective otherwise. For comparison, we made an experiment using classical RQMC with these methods for the same numerical example as given here, but with $c = 8$ and $c = 16$ instead of $c = 10$, to have powers of 2, and $t_j = jT/c$. For Sobol'+LMS with $n = 2^{20}$, for $d = 16$, the values of VRF20 for BGSS, BGBS, and DGBS were 85, 895, 550, respectively. For $d = 8$, these values were 183, 1258, and 3405. The VRF20 values for Sobol'+LMS in table 1 and are much larger that these, showing that Array-RQMC can provide much larger variance reductions.

For this VG model, we do not report results on the European option with Array-RQMC, because the Markov chain would have only one step: We can generate directly $G(t_c)$ and then $Y(t_c)$. For this, ordinary RQMC works well enough (L'Ecuyer 2018).

5 OPTION PRICING UNDER THE HESTON VOLATILITY MODEL

The Heston volatility model is defined by the following two-dimensional stochastic differential equation:

$$\begin{aligned} \mathrm{d}S(t) &= rS(t)\mathrm{d}t + V(t)^{1/2}S(t)\mathrm{d}B_1(t), \\ \mathrm{d}V(t) &= \lambda(\sigma^2 - V(t))\mathrm{d}t + \xi V(t)^{1/2}\mathrm{d}B_2(t), \end{aligned}$$

for $t \geq 0$, where (B_1, B_2) is a pair of standard Brownian motions with correlation ρ between them, r is the risk-free rate, σ^2 is the long-term average variance parameter, λ is the rate of return to the mean for the variance, and ξ is a volatility parameter for the variance. The processes $S = \{S(t), t \geq 0\}$ and $V = \{V(t), t \geq 0\}$ represent the asset price and the volatility, respectively, as a function of time. We will examine how to estimate the price of European and Asian options with Array-RQMC under this model. Since we do not know how to generate $(S(t+\delta), V(t+\delta))$ exactly from its conditional distribution given $(S(t), V(t))$ in this case, we have to discretize the time. For this, we use the Euler method with τ time steps of length $\delta = T/\tau$ to generate a skeleton of the process at times $w_j = j\delta$ for $j = 1, \ldots, \tau$, over $[0, T]$. For the Asian option, we assume for simplicity that the observation times $t_1, \ldots, t_c$ used for the payoff are all multiples of δ, so each of them is equal to some w_j.

Table 2: Regression slopes $\hat{\beta}$ for $\log_2 \mathrm{Var}[\hat{\mu}_n^{\mathrm{arqmc}}]$ vs $\log_2(n)$, and VRF compared with MC for $n = 2^{20}$, denoted VRF20, for the Asian option under the Heston model.

		European		Asian	
Sort	Point sets	$\hat{\beta}$	VRF20	$\hat{\beta}$	VRF20
Split sort	MC	-1	1	-1	1
	Stratif	-1.26	103	-1.29	38
	Sobol'+LMS	-1.59	44,188	-1.48	6,684
	Sobol'+NUS	-1.46	30,616	-1.46	5,755
	Lattice+baker	-1.50	26,772	-1.55	5,140
Batch sort	MC	-1	1	-1	1
	Stratif	-1.24	91	-1.25	33
	Sobol'+LMS	-1.66	22,873	-1.23	815
	Sobol'+NUS	-1.72	30,832	-1.38	1,022
	Lattice+baker	-1.75	12,562	-1.22	762
Hilbert sort (with logistic map)	MC	-1	1	-1	1
	Stratif	-1.26	43	-1.05	29
	Sobol'+LMS	-1.14	368	-0.87	39
	Sobol'+NUS	-1.06	277	-1.11	49
	Lattice+baker	-1.12	250	-0.89	42

Following Giles (2008), to reduce the bias due to the discretization, we make the change of variable $W(t) = e^{\lambda t}(V(t) - \sigma^2)$, with $\mathrm{d}W(t) = e^{\lambda t}\xi V(t)^{1/2}\mathrm{d}B_2(t)$, and apply the Euler method to (S, W) instead of (S, V). The Euler approximation scheme with step size δ applied to W gives

$$\widetilde{W}(j\delta) = \widetilde{W}((j-1)\delta) + e^{\lambda(j-1)\delta}\xi(\widetilde{V}((j-1)\delta)\delta)^{1/2}Z_{j,2}.$$

Rewriting it in terms of V by using the reverse identity $V(t) = \sigma^2 + e^{-\lambda t}W(t)$, and after some manipulations, we obtain the following discrete-time stochastic recurrence, which we will simulate by Array-RQMC:

$$\begin{aligned} \widetilde{V}(j\delta) &= \max\left[0, \sigma^2 + e^{-\lambda\delta}\left(\widetilde{V}((j-1)\delta) - \sigma^2 + \xi(\tilde{V}((j-1)\delta)\delta)^{1/2}Z_{j,2}\right)\right], \\ \widetilde{S}(j\delta) &= (1 + r\delta)\widetilde{S}((j-1)\delta) + (\widetilde{V}((j-1)\delta)\delta)^{1/2}\widetilde{S}((j-1)\delta)Z_{j,1}, \end{aligned}$$

where $(Z_{j,1}, Z_{j,2})$ is a pair of standard normals with correlation ρ. We generate this pair from a pair $(U_{j,1}, U_{j,2})$ of independent Uniform$(0,1)$ variables via $Z_{j,1} = \Phi^{-1}(U_{j,1})$ and $Z_{j,2} = \rho Z_{j,1} + \sqrt{1-\rho^2}\,\Phi^{-1}(U_{j,2})$. We

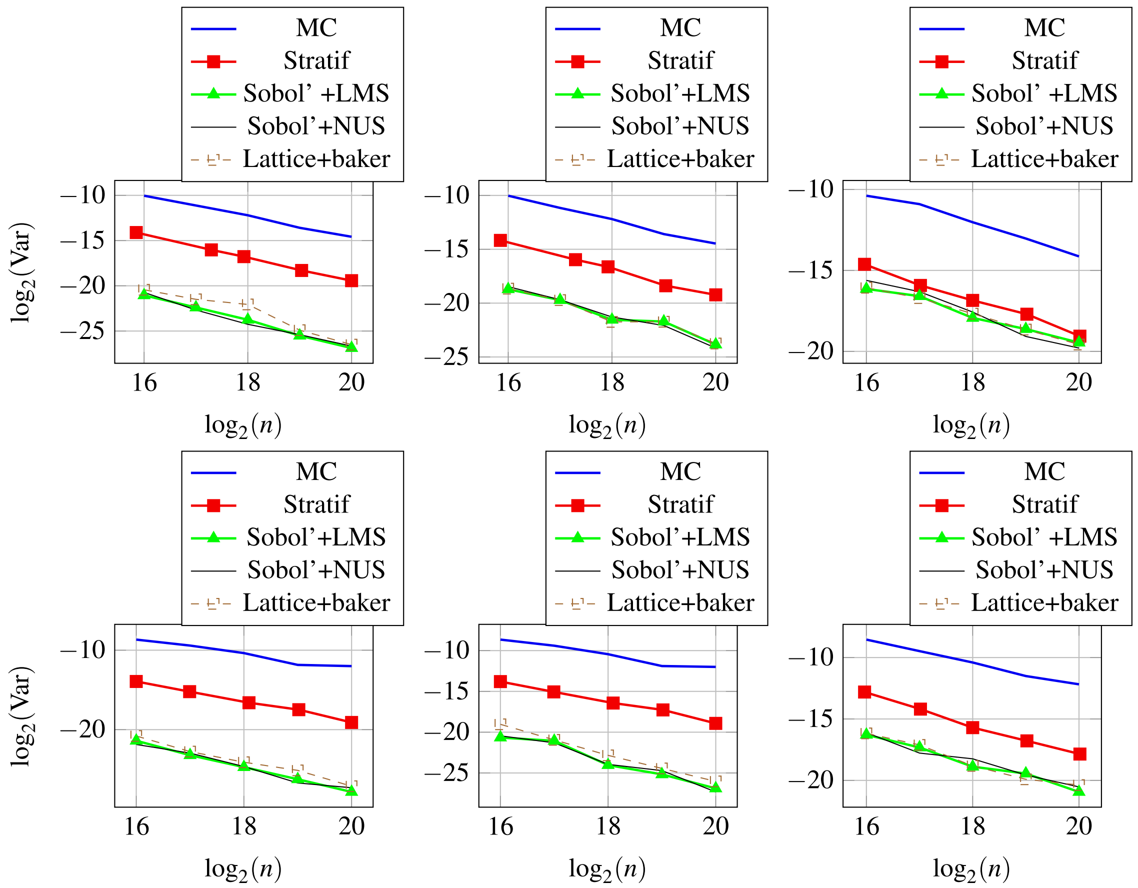

Figure 2: Plots of empirical $\log_2 \mathrm{Var}[\hat{\mu}_n^{\mathrm{arqmc}}]$ vs $\log_2(n)$ for various sorts and point sets, based on $m = 100$ independent replications, for the Heston model. Asian option (above) and European option (below), with split sort (left), batch sort (middle), and Hilbert sort (right).

then approximate each $S(j\delta)$ by $\widetilde{S}(j\delta)$. The running average $\bar{S}_j$ at step j must be the average of the $S(t_k)$ at the observation times $t_k \leq w_j = j\delta$. If we denote $N_j = \sum_{k=1}^{c} \mathbb{I}[t_k \leq j\delta]$, we have $\bar{S}_j = (1/N_j)\sum_{k=1}^{N_j} S(t_k)$, which we approximate by $\bar{\widetilde{S}}_j = (1/N_j)\sum_{k=1}^{N_j} \widetilde{S}(t_k)$. Here, the state of the chain is $X_j = (\widetilde{S}(j\delta), \widetilde{V}(j\delta))$ when pricing the European option and $X_j = (\widetilde{S}(j\delta), \widetilde{V}(j\delta), \bar{\widetilde{S}}_j)$ when pricing the Asian option. And two uniform random numbers, $(U_{j,1}, U_{j,2})$, are required at each step of the chain. We thus need four-dimensional RQMC point sets for the European option and five-dimensional RQMC point sets for the Asian option, if we do not map the state to a lower-dimensional representation. If we map the state to one dimension, as in the Hilbert curve sort, then we only need three-dimensional RQMC points for both option types.

We tried an alternative Markov chain definition in which the chain advances by one step each time a uniform random number is used, as in the VG example, to reduce the dimension of the RQMC points, but this gave no improvement.

We ran experiments with $T = 1$ (one year), $K = 100$, $S(0) = 100$, $V(0) = 0.04$, $r = 0.05$, $\sigma = 0.2$, $\lambda = 5$, $\xi = 0.25$, $\rho = -0.5$, and $c = \tau = 16$. This gives $\delta = 1/16$, so the time discretization for Euler is very coarse, but a smaller δ gives similar results in terms of variance reduction by Array-RQMC. For example, we ran experiments with $\tau = 256$ instead of $\tau = 16$ and the VRF20's had approximately the same sizes. Table 2 reports the estimated slopes $\hat{\beta}$ and VRF20, as in Table 1. Again, we observe large variance reductions and improved convergence rates from Array-RQMC. The best results are obtained with the split sort. Figure 2 shows plots of $\log_2 \mathrm{Var}[\hat{\mu}_n^{\mathrm{arqmc}}]$ vs $\log_2(n)$ for selected sorts.

6 OPTION PRICING UNDER THE ORNSTEIN-UHLENBECK VOLATILITY MODEL

The Ornstein-Uhlenbeck volatility model is defined by the following stochastic differential equations:

$$\begin{aligned} \mathrm{d}S(t) &= rS(t)\mathrm{d}t + e^{V(t)}S(t)\mathrm{d}B_1(t), \\ \mathrm{d}V(t) &= \alpha(b - V(t))\mathrm{d}t + \sigma \mathrm{d}B_2(t), \end{aligned}$$

for $t \geq 0$, where (B_1, B_2) is a pair of standard Brownian motions with correlation ρ between them, r is the risk-free rate, b is the long-term average volatility, α is the rate of return to the average volatility, and is σ a variance parameter for the volatility process. The processes $S = \{S(t), t \geq 0\}$ and $V = \{V(t), t \geq 0\}$ represent the asset price and the volatility process. We simulate these processes using Euler's method with τ time steps of length δ, as we did for the Heston model, but without a change of variable. The discrete-time approximation of the stochastic recurrence is

$$\begin{aligned} \widetilde{S}(j\delta) &= \widetilde{S}((j-1)\delta) + r\delta\widetilde{S}((j-1)\delta) + \exp\left[\widetilde{V}((j-1)\delta)\right]\sqrt{\delta}Z_{j,1}, \\ \widetilde{V}(j\delta) &= \alpha\delta b + (1-\alpha\delta)\widetilde{V}((j-1)\delta) + \sigma\sqrt{\delta}Z_{j,2}, \end{aligned}$$

where $(Z_{j,1}, Z_{j,2})$ is a pair of standard normals with correlation ρ. To generate this pair, we generate independent Uniform$(0,1)$ variables $(U_{j,1}, U_{j,2})$, and put $Z_{j,1} = \Phi^{-1}(U_{j,1})$ and $Z_{j,2} = \rho Z_{j,1} + \sqrt{1-\rho^2}\,\Phi^{-1}(U_{j,2})$. For either the European or Asian option, the state of the Markov chain and the dimension of the RQMC points are the same as for the Heston model.

We ran a numerical experiment with $T = 1$, $K = 100$, $S(0) = 100$, $V(0) = 0.04$, $r = 0.05$, $b = 0.4$, $\alpha = 5$, $\sigma = 0.2$, $\rho = -0.5$, and $c = \tau = 16$ (so $\delta = 1/16$). Table 3 reports the estimated regression slopes $\hat{\beta}$ and VRF2. With $\tau = 256$ instead of $\tau = 16$, the VRF20's have about the same sizes.

Table 3: Regression slopes $\hat{\beta}$ for $\log_2 \mathrm{Var}[\hat{\mu}_n^{\mathrm{arqmc}}]$ vs $\log_2(n)$, and VRF compared with MC for $n = 2^{20}$, denoted VRF20, for the European and Asian options under the Ornstein-Uhlenbeck model.

		European		Asian	
Sort	Point sets	$\hat{\beta}$	VRF20	$\hat{\beta}$	VRF20
Batch sort	MC	-1	1	-1	1
	Stratif	-1.28	111	-1.23	29.
	Sobol'+LMS	-1.35	61,516	-1.22	4,558
	Sobol'+NUS	-1.31	56,235	-1.22	5,789
	Lattice+baker	-1.37	61,318	-1.20	5,511
Hilbert sort (with logistic map)	MC	-1	1	-1	1
	Stratif	-1.40	440	-1.37	250
	Sobol'+LMS	-1.52	194,895	-1.40	41,100
	Sobol'+NUS	-1.68	191,516	-1.37	39,861
	Lattice+baker	-1.59	165,351	-1.47	37,185

CONCLUSION

We have shown how Array-RQMC can be applied for pricing options under stochastic volatility models, and gave detailed examples with the VG, Heston, and Ornstein-Uhlenbeck models. With the models, the method requires higher-dimensional RQMC points than with the simpler GBM model studied previously, and when time has to be discretized to apply Euler's method, the number of steps of the Markov chain is much larger. For these reasons, it was not clear a priori if Array-RQMC would be effective. Our empirical results show that it brings very significant variance reductions compared with crude Monte Carlo.

ACKNOWLEDGMENTS

This work has been supported by a discovery grant from NSERC-Canada, a Canada Research Chair, and a Grant from the IVADO Fundamental Research Program, to P. L'Ecuyer.

REFERENCES

Avramidis, A. N., and P. L'Ecuyer. 2006. "Efficient Monte Carlo and Quasi-Monte Carlo Option Pricing Under the Variance-Gamma Model". *Management Science* 52(12):1930–1944.

Avramidis, A. N., P. L'Ecuyer, and P.-A. Tremblay. 2003. "Efficient Simulation of Gamma and Variance-Gamma Processes". In *Proceedings of the 2003 Winter Simulation Conference*, edited by S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, 319–326. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Demers, V., P. L'Ecuyer, and B. Tuffin. 2005. "A Combination of Randomized Quasi-Monte Carlo with Splitting for Rare-Event Simulation". In *Proceedings of the 2005 European Simulation and Modeling Conference*, 25–32. Ghent, Belgium: EUROSIS.

Dick, J., and F. Pillichshammer. 2010. *Digital Nets and Sequences: Discrepancy Theory and Quasi-Monte Carlo Integration*. Cambridge, U.K.: Cambridge University Press.

Dion, M., and P. L'Ecuyer. 2010. "American Option Pricing with Randomized Quasi-Monte Carlo Simulations". In *Proceedings of the 2010 Winter Simulation Conference*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, 2705–2720. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

El Haddad, R., C. Lécot, and P. L'Ecuyer. 2008. "Quasi-Monte Carlo Simulation of Discrete-Time Markov Chains on Multidimensional State Spaces". In *Monte Carlo and Quasi-Monte Carlo Methods 2006*, edited by A. Keller, S. Heinrich, and H. Niederreiter, 413–429. Berlin: Springer-Verlag.

El Haddad, R., C. Lécot, P. L'Ecuyer, and N. Nassif. 2010. "Quasi-Monte Carlo Methods for Markov Chains with Continuous Multidimensional State Space". *Mathematics and Computers in Simulation* 81:560–567.

Fu, M. C., D. B. Madan, and T. Wang. 1998. "Pricing Continuous Asian Options: A Comparison of Monte Carlo and Laplace Transform Inversion Methods". *Journal of Computational Finance* 2:49–74.

Gerber, M., and N. Chopin. 2015. "Sequential Quasi-Monte Carlo". *Journal of the Royal Statistical Society, Series B* 77(Part 3):509–579.

Giles, M. B. 2008. "Multilevel Monte Carlo path simulation". *Operations Research* 56(3):607–617.

Lécot, C., and B. Tuffin. 2004. "Quasi-Monte Carlo Methods for Estimating Transient Measures of Discrete Time Markov Chains". In *Monte Carlo and Quasi-Monte Carlo Methods 2002*, edited by H. Niederreiter, 329–343. Berlin: Springer-Verlag.

L'Ecuyer, P. 2009. "Quasi-Monte Carlo Methods with Applications in Finance". *Finance and Stochastics* 13(3):307–349.

L'Ecuyer, P. 2016. "SSJ: Stochastic Simulation in Java". http://simul.iro.umontreal.ca/ssj/, accessed 27th July 2019.

L'Ecuyer, P. 2018. "Randomized Quasi-Monte Carlo: An Introduction for Practitioners". In *Monte Carlo and Quasi-Monte Carlo Methods: MCQMC 2016*, edited by P. W. Glynn and A. B. Owen, 29–52. Berlin: Springer.

L'Ecuyer, P., and E. Buist. 2005. "Simulation in Java with SSJ". In *Proceedings of the 2005 Winter Simulation Conference*, edited by M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 611–620. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

L'Ecuyer, P., V. Demers, and B. Tuffin. 2007. "Rare-Events, Splitting, and Quasi-Monte Carlo". *ACM Transactions on Modeling and Computer Simulation* 17(2):Article 9.

L'Ecuyer, P., C. Lécot, and A. L'Archevêque-Gaudet. 2009. "On Array-RQMC for Markov Chains: Mapping Alternatives and Convergence Rates". In *Monte Carlo and Quasi-Monte Carlo Methods 2008*, edited by P. L'Ecuyer and A. B. Owen, 485–500. Berlin: Springer-Verlag.

L'Ecuyer, P., C. Lécot, and B. Tuffin. 2006. "Randomized Quasi-Monte Carlo Simulation of Markov Chains with an Ordered State Space". In *Monte Carlo and Quasi-Monte Carlo Methods 2004*, edited by H. Niederreiter and D. Talay, 331–342. Berlin: Springer-Verlag.

L'Ecuyer, P., C. Lécot, and B. Tuffin. 2008. "A Randomized Quasi-Monte Carlo Simulation Method for Markov Chains". *Operations Research* 56(4):958–975.

L'Ecuyer, P., and C. Lemieux. 2000. "Variance Reduction via Lattice Rules". *Management Science* 46(9):1214–1235.

L'Ecuyer, P., and D. Munger. 2016. "Algorithm 958: Lattice Builder: A General Software Tool for Constructing Rank-1 Lattice Rules". *ACM Transactions on Mathematical Software* 42(2):Article 15.

L'Ecuyer, P., D. Munger, C. Lécot, and B. Tuffin. 2018. "Sorting Methods and Convergence Rates for Array-RQMC: Some Empirical Comparisons". *Mathematics and Computers in Simulation* 143:191–201.

L'Ecuyer, P., and C. Sanvido. 2010. "Coupling from the Past with Randomized Quasi-Monte Carlo". *Mathematics and Computers in Simulation* 81(3):476–489.

L'Ecuyer, P., and R. Simard. 2006. "Inverting the Symmetrical Beta Distribution". *ACM Transactions on Mathematical Software* 32(4):509–520.

Lemieux, C., M. Cieslak, and K. Luttmer. 2004. *RandQMC User's Guide: A Package for Randomized Quasi-Monte Carlo Methods in C*. Software user's guide, http://www.math.uwaterloo.ca/~clemieux/randqmc.html, accessed 27th July 2019.

Madan, D. B., P. P. Carr, and E. C. Chang. 1998. "The Variance Gamma Process and Option Pricing". *European Finance Review* 2:79–105.

Madan, D. B., and E. Seneta. 1990. "The Variance Gamma (V.G.) Model for Share Market Returns". *Journal of Business* 63:511–524.

Owen, A. B. 2003. "Variance with Alternative Scramblings of Digital Nets". *ACM Transactions on Modeling and Computer Simulation* 13(4):363–378.

Wächter, C., and A. Keller. 2008. "Efficient Simultaneous Simulation of Markov Chains". In *Monte Carlo and Quasi-Monte Carlo Methods 2006*, edited by A. Keller, S. Heinrich, and H. Niederreiter, 669–684. Berlin: Springer-Verlag.

AUTHOR BIOGRAPHIES

AMAL BEN ABDELLAH is a PhD student in computer science at the Université de Montréal, Canada. Her main research interests are randomized quasi-Monte Carlo methods, Array-RQMC, density estimation and stochastic simulation in general. She is currently working on quasi-Monte Carlo methods for the simulation of Markov chains and density estimation. Her email address is amal.ben.abdellah@umontreal.ca.

PIERRE L'ECUYER is a Professor in the Departement d'Informatique et de Recherche Opérationnelle, at the Université de Montréal, Canada. He is a member of the CIRRELT and GERAD research centers. His main research interests are random number generation, quasi-Monte Carlo methods, efficiency improvement via variance reduction, sensitivity analysis and optimization of discrete-event stochastic systems, and discrete-event simulation in general. He has published over 270 scientific articles, and has developed software libraries and systems for random number generation and stochastic simulation (SSJ, TestU01, RngStreams, Lattice Builder, etc.). He has been a referee for 153 different scientific journals. More information can be found on his web page: http://www.iro.umontreal.ca/~lecuyer. Email: lecuyer@iro.umontreal.ca.

FLORIAN PUCHHAMMER is a postdoctoral fellow at the Université de Montréal, Canada. His main research interests are discrepancy theory, quasi-Monte Carlo methods, randomized quasi-Monte Carlo methods, uniform distribution of sequences, information based complexity. He is currently working on the quasi-Monte Carlo Methods for the simulation of Markov Chains and for density estimation. His email address is florian.puchhammer@umontreal.ca.