A Randomized Quasi-Monte Carlo Simulation Method for Markov Chains

Pierre L'Ecuyer
GERAD and Département d'Informatique et de Recherche Opérationnelle
Université de Montréal, C.P. 6128, Succ. Centre-Ville, Montréal (Québec), H3C 3J7, Canada
lecuyer@iro.umontreal.ca

Christian Lécot
Laboratoire de Mathématiques, Université de Savoie, 73376 Le Bourget-du-Lac Cedex, France,
Christian.Lecot@univ-savoie.fr

Bruno Tuffin
IRISA-INRIA, Campus Universitaire de Beaulieu, 35042 Rennes Cedex, France
Bruno.Tuffin@irisa.fr

This version: August 23, 2006

Abstract. We introduce and study a randomized quasi-Monte Carlo method for estimating the state distribution at each step of a Markov chain. The number of steps in the chain can be random and unbounded. The method simulates n copies of the chain in parallel, using a $(d+1)$-dimensional highly-uniform point set of cardinality n, randomized independently at each step, where d is the number of uniform random numbers required at each transition of the Markov chain. This technique is effective in particular to obtain a low-variance unbiased estimator of the expected total cost up to some random stopping time, when state-dependent costs are paid at each step. It is generally more effective when the state space has a natural order related to the cost function.

We provide numerical illustrations where the variance reduction with respect to standard Monte Carlo is substantial. The variance can be reduced by factors of several thousands in some cases. We prove bounds on the convergence rate of the worst-case error and variance for special situations. In line with what is typically observed in randomized quasi-Monte Carlo contexts, our empirical results indicate much better convergence than what these bounds guarantee.

Subject classifications: Simulation: efficiency. Probability: Markov processes.
Area of review: simulation.
Keywords: Variance reduction, randomized quasi-Monte Carlo, Markov chain, random walk, regenerative process, digital nets, lattice rules.

1 Introduction

A wide variety of real-life systems can be modeled as Markov chains with a large state space. In most interesting situations, analytic formulas are not available for these Markov chains and matrix-based numerical methods require too much time, so Monte Carlo simulation becomes the standard way of estimating performance measures for these systems.

In this paper, we propose a randomized quasi-Monte Carlo (RQMC) algorithm, based on deterministic methods introduced by Lécot and Tuffin (2004) and Lécot and Ogawa (2002), to improve simulation efficiency for discrete-time Markov chains. The algorithm simulates n copies of the chain in parallel and induces negative dependence between the corresponding sample paths by using some form of generalized antithetic variates (Wilson, 1983; Ben-Ameur et al., 2004). Our aim is that the empirical distribution of the states of these n chains, at any given step j, is a better approximation of the corresponding theoretical distribution than if the n chains were simulated independently. As a result, performance measure estimators obtained by taking an average across the n copies of the chain will typically have much smaller variance.

Markov Chain Setting. We consider a Markov chain $\{X_j,\, j \geq 0\}$ with state space $\mathcal{X}$, that evolves according to the stochastic recurrence:

$$X_0 = x_0, \qquad X_j = \varphi_j(X_{j-1}, \mathbf{U}_j),\ j \geq 1, \tag{1}$$

where the $\mathbf{U}_j$ are independent random vectors uniformly distributed over the d-dimensional unit cube $[0,1)^d$ (henceforth denoted $\mathbf{U}_j \sim \mathrm{U}[0,1)^d$). Each $\mathbf{U}_j$ represents the d uniform random numbers required to simulate step j of the chain. Every discrete-time Markov chain that can be simulated on a computer fits this framework. It is also always possible to take $d = 1$ by defining the chain so that each newly generated uniform random number in the simulation corresponds to one step of the chain, although this representation is not always natural. Here, for more flexibility, we simply assume that d is a finite constant.

We want to estimate the expected total cost $\mu = E[Y]$ where

$$Y = \sum_{j=0}^{\tau} c_j(X_j), \tag{2}$$

each $c_j : \mathcal{X} \to \mathbb{R}$ is a *cost function*, τ is a stopping time with respect to the filtration $\{\mathcal{F}_j,\, j \geq 0\}$ generated by $\{(j, X_j),\, j \geq 0\}$, $E[\tau] < \infty$, and we assume that the functions φ_j and c_j are easy to

evaluate at any given point, for each j. We also assume (implicitly) that $\mathcal{X}$, the φ_j's, and the c_j's satisfy the appropriate measure-theoretic requirements so that all objects of interest in this paper are well-defined.

Estimation by Monte Carlo. The *standard Monte Carlo* (MC) method simulates the random variable Y as follows. For $j = 1, \ldots, \tau$, generate $\mathbf{U}_j \sim \mathrm{U}[0,1)^d$, compute $X_j = \varphi_j(X_{j-1}, \mathbf{U}_j)$, and add the realization of $c_j(X_j)$ to an accumulator, which at the end will contain the realization of Y. This can be replicated n times independently, and the sample mean and variance of the n realizations of Y can be taken as unbiased estimators of the exact mean and variance of Y. From this, one can compute a confidence interval on μ, e.g., via the central-limit theorem.

Let $s = \sup_\omega \tau d$, where the supremum is taken over all possible sample paths ω, so $s = \infty$ if τ is unbounded. In this setting, the random variable Y can be written as a function of a sequence of s independent $\mathrm{U}(0,1)$ random variables, say $Y = f(U_1, \ldots, U_s)$, for some complicated function f, where $\mathbf{U}_j = (U_{(j-1)d+1}, \ldots, U_{jd})$ for each j. The MC method generates n independent random points $\mathbf{V}_i = (U_{i,1}, \ldots, U_{i,s})$, $i = 1, \ldots, n$, uniformly distributed in the s-dimensional unit cube $[0,1)^s$, evaluates the function f at each of these points, and takes the average $\bar{Y}_n$ of these n evaluations as an estimator of μ.

Quasi-Monte Carlo (QMC). The classical QMC method replaces the n independent random points $\mathbf{V}_j$ by a *deterministic* set of distinct points $P_n = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ that cover the unit cube $[0,1)^s$ more uniformly than typical random points. There are many ways of measuring the uniformity of P_n. One of them is the star discrepancy, defined as

$$D_n^*(P_n) = \sup_{\mathbf{x} \in [0,1)^s} |\mathrm{card}(P_n \cap [\mathbf{0}, \mathbf{x}))/n - \mathrm{vol}([\mathbf{0}, \mathbf{x}))|$$

where $\mathrm{card}(P)$ means the cardinality of the set P, $[\mathbf{0}, \mathbf{x})$ is the s-dimensional rectangular box with opposite corners $\mathbf{0}$ and $\mathbf{x}$, and vol denotes its volume. A classical worst-case error bound for numerical integration by QMC is given by the Koksma-Hlawka inequality

$$\|\bar{Y}_n - \mu\| \leq V(f) D_n^*(P_n), \tag{3}$$

where $\bar{Y}_n$ represents the average value of f over the n points, μ is its theoretical average over the unit cube, and $V(f)$ is the Hardy-Krause *total variation* of the integrand f (see Niederreiter, 1992,

for the details). For a one-dimensional function $f : [0,1] \to \mathbb{R}$, we have

$$V(f) = \sup_{L \geq 1,\, 0=x_0<\cdots<x_L=1} \sum_{j=1}^{L} |f(x_j) - f(x_{j-1})| = \int_0^1 |df(x)|. \tag{4}$$

For $s > 1$, it is much more complicated; we just point out that $V(f)$ is infinite whenever f has a discontinuity not aligned with one of the axes (Owen, 2005). This happens frequently. Moreover, in most practical situations, the bound (3) is useless because it is extremely loose and also much too difficult to compute. Hlawka (1971) derives another error bound in which $V(f)$ is replaced by a different notion of variation, called the mean oscillation of f. In fact, a large family of bounds can be derived by adopting different definitions of discrepancy together with corresponding definitions of function variation (Hickernell, 1998; L'Ecuyer and Lemieux, 2002). These bounds can be used to get asymptotic convergence rates but are rarely convenient for practical error assessment.

Randomized Quasi-Monte Carlo (RQMC). The usual RQMC method (we call it *classical RQMC*, to distinguish it from our new algorithm) turns QMC into a variance reduction method by carefully randomizing P_n. This can be seen as an application of the *generalized antithetic variates* (GAV) principle (Wilson, 1983), as explained in Ben-Ameur et al. (2004). The idea is to induce "negative dependence" between the points $\mathbf{V}_i$ (for example, recalling that each $\mathrm{Cov}(\mathbf{V}_i, \mathbf{V}_j)$ is a diagonal matrix, we may ask for all diagonal elements of the diagonal matrix $\sum_{(i,j):i\neq j} \mathrm{Cov}(\mathbf{V}_i, \mathbf{V}_j)$ to be negative) by generating them in a way that

(a) each point $\mathbf{V}_i$ has the uniform distribution over $[0,1)^s$, and

(b) the point set $\{\mathbf{V}_1, \ldots, \mathbf{V}_n\}$ covers $[0,1)^s$ more uniformly (in some sense) than a set of independent random points.

We call a point set that satisfies these two conditions an *RQMC point set*. This definition is incomplete: we still need to select a specific measure of uniformity so that "more uniformly" has a well-defined meaning. We leave this choice open for now, because we want to keep the generality. Specific measures and conditions are always adopted when proving bounds on the convergence speed of the error and variance for particular RQMC methods, and we will do the same when we prove such results for our method, but the RQMC method in general and the algorithm proposed in this paper are defined independently of any measure of uniformity. Examples of RQMC point sets include randomly shifted lattice rules, scrambled digital nets, digital nets with a random digital shift, a

Latin hypercube sample or a stratified sample followed by a random permutation of the points, and so forth (Owen, 1998; L'Ecuyer and Lemieux, 2002; Glasserman, 2004). For the stratified sample, we partition the unit cube into n boxes of equal volume and generate one point randomly and uniformly in each box. Then we permute the n points randomly, so that for each fixed i, $\mathbf{V}_i$ has probability $1/n$ of being in any given box. The same permutation trick applies to Latin hypercube sampling as well.

The estimator $\bar{Y}_n$ is computed by averaging the values of f over the n points in the same way as for MC. These n values are not independent, but we can estimate the variance of $\bar{Y}_n$ by replicating this scheme m times, with independent randomizations of the same point set. Under the above conditions on the randomization, the sample mean and sample variance of these m averages are unbiased estimators of the exact mean and variance of $\bar{Y}_n$. Further details on this *classical RQMC* approach (and variants of it for high-dimensional contexts) can be found in Owen (1998); L'Ecuyer and Lemieux (2000, 2002) and other references given there. This approach is typically more efficient than MC when s is small (e.g., less than 20 or so) or if the function f has *low effective dimension* in some sense, as explained in Owen (1998) and L'Ecuyer and Lemieux (2002).

Introductory overviews of QMC and RQMC methods can be found in Kollig and Keller (2002); Owen (2003a); Glasserman (2004); L'Ecuyer (2004a), and the user's guide of the package `hups` in SSJ (L'Ecuyer, 2004b), for example. For more advanced material, see Niederreiter (1992); Owen (1998); L'Ecuyer and Lemieux (2002) and the proceedings of the biennial MCQMC Conference.

Array-RQMC. The RQMC method proposed in this paper, called *array-RQMC*, operates differently. We simulate n copies of the chain in parallel. To simulate step j for all copies, we use a randomized $(d+1)$-dimensional highly-uniform (or "low-discrepancy") point set $P'_{n,j}$ of cardinality n, as explained in the next section, where $d \ll s$ typically. These point sets are randomized independently at the different steps, in a way that the sample path of any given copy of the chain obeys the correct probability law (the same as with the MC method). As a result, we have an unbiased estimator $\bar{Y}_n$ for the average cost μ. The aim of the proposed method is to induce dependence across the n copies so that the empirical distribution of the n realizations of X_j (at step j) gives a much better approximation of the distribution F_j of the random variable X_j than if the chains were simulated independently.

The original deterministic method of Lécot and Tuffin (2004) was designed to approximate

transient measures over a *fixed* number of steps, for discrete-time and discrete-state Markov chains with a totally ordered state space. That method uses a $(0, 2)$-sequence in base 2 (Niederreiter, 1992). At step j of the chain, it reorders the n copies according to their current states and "simulates" the transitions (next states) for the n copies by employing the elements nj to $nj + n - 1$ of the $(0, 2)$-sequence in place of uniform random numbers to drive the simulation. It assumes that simulating each transition of the chain requires a single uniform random variable, i.e., $d = 1$ in equation (1). Convergence to the correct value was proved by Lécot and Tuffin (2004) under a condition on the structure of the transition probability matrix of the Markov chain. In contrast, our method is a randomized algorithm that provides an unbiased estimator. It also applies to Markov chains with a more general state space, with a random and unbounded number of steps, and the number d of uniform random variates that are required to generate the next state in one step of the Markov chain can be larger than 1. It thus covers a much broader range of applications.

We have theoretical results on the convergence rate of the variance of the mean estimator (as $n \to \infty$) only for simplified special cases of the algorithm. For the case where $\mathcal{X} \subseteq (-\infty, \infty]$ and $d = 1$, if each $P'_{n,j}$ is constructed by stratified sampling in the unit square, and under mild additional assumptions, we prove that the variance converges as $O(n^{-3/2})$. For a less restrictive assumption than stratified sampling (see Assumption 1), still with $\mathcal{X} \subseteq (-\infty, \infty]$ and $d = 1$, we show that the *worst-case error* converges as $O(n^{-1/2})$. We conduct empirical experiments with other types of RQMC point sets as well, with a variety of examples. The results indicate that the variance goes down much faster (as a function of n) with the proposed method than for standard MC and faster than classical RQMC in many situations. In our experiments, popular RQMC point sets such as randomized lattice rules and Sobol' nets also perform much better than the stratification. This gap between theoretical and empirical results (we often get much better variance improvements in practice than what we can prove) is common with QMC and RQMC methods (Tuffin, 1996; Tezuka, 2002), where the theoretical bounds have limited applicability in general and the surprisingly good empirical results are often only heuristically justified. The theoretical results are nevertheless a first step toward a better understanding of the proposed algorithm.

2 The Array-RQMC Algorithm

For the remainder of the paper, we assume that $\mathcal{X} \subseteq \mathbb{R}^\ell \cup \{\infty\}$ and that there is a function $h : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ that assigns a real number to each state of the chain, with $h(\infty) = \infty$. This h is called the *sorting function* and will be used to order the states: we say that state x_1 is *smaller* than state x_2 if $h(x_1) < h(x_2)$ and that the two states are *h-equivalent* if $h(x_1) = h(x_2)$. When sorting the states, h-equivalent states can be placed in arbitrary order. A similar type of function h is used in the *splitting methodology* for rare-event simulation, where it is called the *importance function* (Garvels et al., 2002; Glasserman et al., 1999). We use a different name, to avoid possible confusion in case the two methods are combined. Just like for splitting, a good choice of h is crucial for the performance of the algorithm especially when $\mathcal{X}$ has more than one dimension. The state ∞ is an *absorbing state* used to indicate that we have already reached the stopping time τ at the previous step or earlier; i.e., $X_j = \infty$ for $j > \tau$. We also have $c_j(\infty) = 0$ although the cost at the stopping time τ can be nonzero.

The *array-RQMC* algorithm works by simulating n copies of the chain in parallel as follows. In *step 1*, we take an RQMC point set $P_{n,1} = \{\mathbf{u}_{0,1}, \ldots, \mathbf{u}_{n-1,1}\}$ in $[0,1)^d$, define

$$X_{i,1} = \varphi_1(x_0, \mathbf{u}_{i,1}) \qquad \text{for } i = 0, \ldots, n-1,$$

and estimate the distribution F_1 of X_1 by the empirical distribution $\hat{F}_1$ of $X_{0,1}, \ldots, X_{n-1,1}$. This gives

$$\begin{aligned} F_1(x) &= P[X_1 \le x] \\ &= \int_{[0,1)^d} I(\varphi_1(x_0, \mathbf{u}) \le x)\, d\mathbf{u} \qquad (5) \\ &\approx \frac{1}{n} \sum_{i=0}^{n-1} I(\varphi_1(x_0, \mathbf{u}_{i,1}) \le x) \qquad (6) \\ &= \frac{1}{n} \sum_{i=0}^{n-1} I(X_{i,1} \le x) \stackrel{\text{def}}{=} \hat{F}_1(x), \end{aligned}$$

where I denotes the indicator function. The approximation in (6) amounts to estimating the integral (5) by RQMC. If the function φ_1 is not too badly behaved, this should give better accuracy than using standard MC.

In *step* j, we start with the empirical distribution $\hat{F}_{j-1}$ of $X_{0,j-1}, \ldots, X_{n-1,j-1}$ as an approximation of the distribution F_{j-1} of X_{j-1}, and want to compute a good approximation $\hat{F}_j$ of F_j. We

can write

$$
\begin{aligned}
F_j(x) &= P[X_j \le x] \;=\; E[I(\varphi_j(X_{j-1}, \mathbf{U}_j) \le x)] \\
&= \int_{\mathcal{X}} \int_{[0,1)^d} I(\varphi_j(y, \mathbf{u}) \le x)\; d\mathbf{u}\; dF_{j-1}(y) \qquad (7) \\
&\approx \int_{\mathcal{X}} \int_{[0,1)^d} I(\varphi_j(y, \mathbf{u}) \le x)\; d\mathbf{u}\; d\hat{F}_{j-1}(y) \qquad (8) \\
&= \frac{1}{n} \sum_{i=0}^{n-1} E[I(\varphi_j(X_{i,j-1}, \mathbf{U}_{i,j}) \le x)], \qquad (9)
\end{aligned}
$$

where the approximation in (8) is obtained by replacing F_{j-1} in (7) with its approximation $\hat{F}_{j-1}$. When the n copies of the Markov chain are simulated independently via standard MC, (9) is estimated by its realization (the expectation is removed), where the $\mathbf{U}_{i,j}$'s are independent and uniformly distributed over $[0,1)^d$. Our aim here is to estimate the $(d+1)$-dimensional integral (8) by RQMC instead.

To do that, we introduce a $(d+1)$-dimensional *modified RQMC point set* $P'_{n,j}$ defined as $P'_{n,j} = \{\mathbf{u}'_{i,j} = ((i+0.5)/n, \mathbf{u}_{i,j}),\ 0 \le i < n\}$ where $P_{n,j} = \{\mathbf{u}_{0,j}, \ldots, \mathbf{u}_{n-1,j}\}$ is an RQMC point set in $[0,1)^d$, and with the following two properties:

(a) $\mathbf{u}_{i,j}$ is a random vector uniformly distributed over $[0,1)^d$ for each i and

(b) $P'_{n,j}$ is "highly uniform" in $[0,1)^{d+1}$, in a sense that we leave open for now (as in our definition of RQMC point set).

This $P'_{n,j}$ is not quite an RQMC point set, because the first coordinate is not a random variable with the uniform distribution over $[0,1)$. A typical way of defining $P'_{n,j}$ is to take a $(d+1)$-dimensional RQMC point set, sort the points by order of their first coordinate, and then replace the first coordinate of the ith point by $(i+0.5)/n$ for each i. Item (b) is not needed for the proofs of Propositions 1 and 2 later in this section, but is crucial to obtain a variance reduction in the array-RQMC algorithm.

To explain how the integral (8) is approximated by RQMC with the point set $P'_{n,j}$, we first consider the special case where $\ell = 1$ (so $\mathcal{X} \subseteq (-\infty, \infty]$) and $h(x) = x$. Let $X_{(0),j-1}, \ldots, X_{(n-1),j-1}$ be the states at step $j-1$ sorted by increasing order. Define the inverse of $\hat{F}_{j-1}$ by $\hat{F}^{-1}_{j-1}(v) = \inf\{x \in$

$\mathcal{X} : \hat{F}_{j-1}(x) \geq v\}$ for all $v \in [0,1]$. We approximate (8) as follows:

$$\begin{aligned}
\int_{\mathcal{X}} \int_{[0,1)^d} I(\varphi_j(y,\mathbf{u}) \leq x)\, d\mathbf{u}\, d\hat{F}_{j-1}(y) &= \int_{[0,1)^{d+1}} I(\varphi_j(\hat{F}_{j-1}^{-1}(v),\mathbf{u}) \leq x)\, d\mathbf{u}\, dv && (10)\\
&\approx \frac{1}{n}\sum_{i=0}^{n-1} I(\varphi_j(\hat{F}_{j-1}^{-1}((i+0.5)/n),\mathbf{u}_{i,j}) \leq x) && (11)\\
&= \frac{1}{n}\sum_{i=0}^{n-1} I(\varphi_j(X_{(i),j-1},\mathbf{u}_{i,j}) \leq x) && (12)\\
&= \frac{1}{n}\sum_{i=0}^{n-1} I(X_{i,j} \leq x) \stackrel{\text{def}}{=} \hat{F}_j(x).
\end{aligned}$$

In (11), we approximate the integral in (10) by RQMC over $[0,1)^{d+1}$ with the point set $P'_{n,j}$. The sorting at step $j-1$ is needed for (12) to be true; i.e., the ith point of $P'_{n,j}$ must be assigned to the chain whose state is $X_{(i),j-1}$. Observe that this point set gives a perfect stratification of the distribution $\hat{F}_{j-1}$, with exactly one observation per stratum (the strata are the jumps of $\hat{F}_{j-1}$). On the other hand, these observations are generally not independent across the strata. The distribution F_j is estimated by the empirical distribution $\hat{F}_j$ of the realizations $X_{i,j} = \varphi_j(X_{(i),j-1},\mathbf{u}_{i,j})$, $i = 0,\ldots,n-1$. This RQMC approximation in (11) is expected to be more accurate than using standard MC. The approximation error in (8) depends on how well F_{j-1} is approximated by $\hat{F}_{j-1}$, i.e., on an accumulation of integration errors in (11) over the previous stages. The rationale of the array-RQMC method is to reduce this integration error by using RQMC at each stage.

In the case where $\ell > 1$, the inverse function $\hat{F}_{j-1}^{-1}$ is not well-defined, so (10) no longer stands. But suppose that the function h is selected so that whenever two states $X_{1,j}$ and $X_{2,j}$ are h-equivalent, the distribution of the future costs $c_{j'}(X_{j'})$ for $j' > j$ conditional on $X_{i,j}$ is the same for $i = 1$ and $i = 2$. In this idealistic setting, knowing the value of $Z_j = h(X_j)$ at step j provides as much relevant information as knowing the state X_j, so it suffices to approximate the (univariate) distribution function G_j of Z_j instead of approximating F_j. Assuming that $Z_j = \tilde{\varphi}_j(Z_{j-1},\mathbf{U}_j)$ for some functions $\tilde{\varphi}_j$, denoting $Z_{i,j-1} = h(X_{i,j-1})$, and denoting $\hat{G}_j$ the empirical distribution of the sorted values

$Z_{(0),j} \le \cdots \le Z_{(n-1),j}$, we can use the same argument as above with G instead of F:

$$\begin{aligned}
G_j(z) &= P[Z_j \le z] \\
&= \int_{-\infty}^{\infty} \int_{[0,1)^d} I(\tilde{\varphi}_j(y, \mathbf{u}) \le z)\, d\mathbf{u}\, dG_{j-1}(y) \\
&\approx \int_{-\infty}^{\infty} \int_{[0,1)^d} I(\tilde{\varphi}_j(y, \mathbf{u}) \le z)\, d\mathbf{u}\, d\hat{G}_{j-1}(y) \\
&= \int_{[0,1)^{d+1}} I(\tilde{\varphi}_j(\hat{G}_{j-1}^{-1}(v), \mathbf{u}) \le z)\, d\mathbf{u}\, dv \\
&\approx \frac{1}{n} \sum_{i=0}^{n-1} I(\tilde{\varphi}_j(\hat{G}_{j-1}^{-1}((i+0.5)/n), \mathbf{u}_{i,j}) \le z) \\
&= \frac{1}{n} \sum_{i=0}^{n-1} I(\tilde{\varphi}_j(Z_{(i),j-1}, \mathbf{u}_{i,j}) \le z) \\
&= \frac{1}{n} \sum_{i=0}^{n-1} I(Z_{i,j} \le z) \;=\; \hat{G}_j(z).
\end{aligned}$$

In practice, it is usually not possible (or too difficult) to select h so that h-equivalent states X are exactly equivalent in the sense that they give the same distributions for the future costs. But a good choice of h should try to approximate this. The performance of the method depends on this choice. With a bad choice of h, the variance may not be reduced at all or may even increase, but reasonable heuristic choices may suffice to reduce the variance, as our examples will illustrate. The following algorithm and the Propositions 1 and 2 that follow are valid regardless of h.

The array-RQMC algorithm can be summarized as follows. We first select a d-dimensional QMC point set $\tilde{P}_n = (\tilde{\mathbf{u}}_0, \ldots, \tilde{\mathbf{u}}_{n-1})$ and a randomization method for $\tilde{P}_n$ such that if $P_n = (\mathbf{u}_0, \ldots, \mathbf{u}_{n-1})$ denotes a realization of the randomization, then $P'_n = \{((i+0.5)/n,\, \mathbf{u}_i),\, 0 \le i < n\}$ is a modified RQMC point set. Then we simulate in parallel n copies of the chain, numbered $0, \ldots, n-1$, as follows (the braces delimit the scope of the "while" loops):

Array-RQMC algorithm:

Let $X_{0,0} \leftarrow x_0$, $\ldots$, $X_{n-1,0} \leftarrow x_0$, and $j \leftarrow 1$;

While $X_{0,j-1} < \infty$ do {

Randomize $\tilde{P}_n$ afresh into $P_{n,j} = \{\mathbf{u}_{0,j}, \ldots, \mathbf{u}_{n-1,j}\}$;

Let $i \leftarrow 0$;

While ($i < n$ and $X_{i,j-1} < \infty$) do {

$X_{i,j} \leftarrow \varphi_j(X_{i,j-1}, \mathbf{u}_{i,j})$; $\quad i \leftarrow i+1$;

}

Sort the states $X_{0,j}, \ldots, X_{n-1,j}$ by increasing order of their values of $h(X_{i,j})$,

and renumber them in this order, i.e., so that $h(X_{0,j}) \leq \cdots \leq h(X_{n-1,j})$;

$j \leftarrow j + 1$;

}

Return the average $\bar{Y}_n$ of the n realizations of Y as an estimator of μ.

Let $\bar{Y}_{n,j}$ denote the average cost at step j, across the n copies. We have $\bar{Y}_n = \sum_{j=1}^{\infty} \bar{Y}_{n,j}$. The randomness of τ may of course affect the performance of the algorithm, because the number of copies of the chain that remain alive decreases with the number of steps, so it could happen that there remains just a few copies for several steps near the end, in which case only a few points from P_n are used in these steps. Two of our numerical examples, in Sections 4.3 and 4.6, have a random τ.

Proposition 1 *The averages $\bar{Y}_{n,j}$ and $\bar{Y}_n$ are* unbiased *estimators of $E[c_j(X_j)]$ and μ, respectively.*

Proof. The successive steps of the chain use independent randomizations. Therefore, for each chain, from the assumption made in Item (a) of the definition of a modified RQMC point set, the vectors that take place of the $\mathbf{U}_j$'s in the recurrence (1) to generate the successive steps j of the chain are independent random variables uniformly distributed over $[0,1)^d$. Thus, any given copy of the chain obeys the correct probabilistic model defined by (1) and (2), so $c_j(X_j)$ and Y have the correct expectations, $E[c_j(X_j)]$ and μ, and their averages over the n copies as well. □

To estimate the variance of $\bar{Y}_n$ and compute a confidence interval on μ, we can replicate this entire procedure independently m times. That is, across the m replications, all randomizations are independent. With this, we have:

Proposition 2 *The empirical variance of the m copies of $\bar{Y}_n$ is an unbiased estimator of* $\mathrm{Var}[\bar{Y}_n]$.

Proof. This follows from the fact that the m copies of $\bar{Y}_n$ are i.i.d. unbiased estimators of μ. These m copies are independent because randomized points from different copies at any given step are independent. □

This proposition implies that the variance of the overall average converges as $O(1/m)$ when $m \to \infty$. In the next section, we examine the convergence rate as a function of n when $n \to \infty$, for simplified cases.

3 Convergence

We want theoretical results on how fast $\bar{Y}_{n,j}$ and $\bar{Y}_n$ converge to their expectations. A first idea would be to bound the integration error represented by the approximation signs in (6), (8), and (11), via the Koksma-Hlawka inequality (3). This inequality is ineffective since the integrand $I(\varphi_j(\hat{F}_{j-1}^{-1}(v), \mathbf{u}) \leq x)$ in (10) may have infinite Hardy-Krause variation: It is equal to 1 in part of the unit cube, 0 elsewhere, and the shape and complexity of the boundary between these two regions depends on $\varphi_1, \ldots, \varphi_j$. This boundary (on which f is discontinuous) is often not aligned with one of the axes.

In what follows, we prove bounds on the convergence rate directly from first principles, for special cases for which $\ell = d = 1$ (so $\mathcal{X} \subseteq \mathbb{R} \cup \{\infty\}$ and the distribution function F_j is defined over $\mathbb{R}$) and $P'_{n,j}$ has special properties. We will end up bounding $|\bar{Y}_{n,j} - E[c_j(X_j)]|$ by the product of the Kolmogorov distance between $\hat{F}_j$ and F_j, defined as

$$\Delta_j = \sup_{x \in \mathcal{X}} |\hat{F}_j(x) - F_j(x)| = \sup_{x \in \mathbb{R}} |\hat{F}_j(x) - F_j(x)|,$$

and the total variation of the cost function c_j. This is Proposition 8. To bound this product, we will obtain bounds on Δ_j by using notions of histogram and integral discrepancies whose properties are examined in the next subsection.

Related results have been obtained by Lécot (1996) for general real-valued functions defined over the unit cube and having bounded variation, in a deterministic setting, under the stronger assumption that P_n is a (t, m, s)-net, and with different methods of proof.

3.1 Histogram and Integral Discrepancies

A *histogram with L intervals* over the unit square is defined as the surface under a step function over the interval $[0, 1]$: Partition the unit interval $[0, 1)$ at the bottom of the square into L subintervals, say of lengths $q_1, \ldots, q_L$ where $q_1 + \cdots + q_L = 1$. Over the ith interval, put a rectangle of height h_i, where $0 \leq h_i \leq 1$, and with the same width as the interval. The histogram H is the union of these rectangles. We say that the histogram is *monotone* (increasing or decreasing) if $h_1 \leq \cdots \leq h_L$ or $h_1 \geq \cdots \geq h_L$.

Let $\mathcal{H}(L)$ be the family of all histograms with L intervals over the unit square, and $\mathcal{H}^+(L)$ the subfamily of all monotone histograms. The *L-histogram discrepancy* of a point set P_n in the unit

square is defined as

$$D_{\mathrm{h}}(L, P_n) = \sup_{H \in \mathcal{H}(L)} |\mathrm{card}(P_n \cap H)/n - \mathrm{area}(H)|$$

where $\mathrm{area}(H) = \sum_{i=1}^{L} q_i h_i$ denotes the area of H. If $\mathcal{H}(L)$ is replaced by $\mathcal{H}^+(L)$, we get the *L-staircase discrepancy* of P_n, denoted $D_{\mathrm{h}}^+(L, P_n)$. The following lemma is rather straightforward:

Lemma 3 *Let P_n denote the first n points of a two-dimensional low-discrepancy sequence whose star discrepancy satisfies $D_n^*(P_n) = O(n^{-1} \log n)$. Then there is a constant K such that for all L,*

$$D_{\mathrm{h}}^+(L, P_n) \le D_{\mathrm{h}}(L, P_n) \le LKn^{-1} \log n.$$

Proof. In every histogram H, each rectangle can be written as a difference of two rectangular boxes anchored at the origin. Thus, H can be written as a sum and difference of $2L$ such boxes. But we know that the star discrepancy of P_n is in $O(n^{-1} \log n)$ and the last inequality follows. The first inequality is obvious. □

Several two-dimensional sequences that satisfy this requirement are readily available (Niederreiter, 1992); for instance, one can take the two-dimensional Sobol' sequence. However, the bound in Lemma 3 is linear in L so it is not very useful for Markov chains with large state spaces (assuming that each state of the chain is associated with one of the L subintervals and vice-versa). The next lemma provides a bound that does not depend on L. It is based on the forthcoming Assumption 1 and a notion of integral discrepancy obtained when $L \to \infty$.

Let $\mathcal{F}(v)$ denote the set of functions $f : [0,1] \to [0,1]$ such that $V(f) \le v$, where $V(f)$ is the variation of f, defined in (4). It is well known that a function of bounded variation over a given interval is Riemann-integrable over that interval. For $f : [0,1] \to [0,1]$, let $H(f) = \{(x, y) \in [0,1]^2 : 0 \le y \le f(x)\}$ be the surface under f. For a point set P_n, we define the *integral discrepancy at variation v* by

$$D_{\mathrm{i}}(P_n, v) = \sup_{f \in \mathcal{F}(v)} |\mathrm{card}(P_n \cap H(f))/n - \mathrm{area}(H(f))|\,. \tag{13}$$

(An equivalent quantity is defined in Niederreiter (1992), page 17.) If f has bounded variation, $H(f)$ can be approximated arbitrarily closely by an histogram H having L rectangles of heights $h_1 = f(x_1), \ldots, h_L = f(x_L)$ where $0 < x_1 < \cdots < x_L = 1$ for some large L. If $V(f) \le v$, the step function f_{H} that corresponds to this H has total variation

$$V(f_{\mathrm{H}}) = \sum_{i=2}^{L} |h_i - h_{i-1}| \le V(f) \le v.$$

Hence, we have that

$$D_{\mathrm{i}}(P_n, v) = \sup_{L\geq 1,\, H\in\mathcal{H}(L),\, V(f_{\mathrm{H}})\leq v} \left|\mathrm{card}(P_n \cap H)/n - \mathrm{area}(H)\right|.$$

Assumption 1 *Suppose that n is a square number, so $\sqrt{n}$ is an integer, and that if we partition the unit square into n subsquares of size $n^{-1/2} \times n^{-1/2}$, each of those subsquares contains exactly one point from P_n.*

Several well-known RQMC point sets satisfy this assumption, for example a digitally shifted version of the two-dimensional Hammersley point set (or Sobol' net, or Faure net) in prime base b, or an affine matrix scramble of it, when n is a power of b (Matoušek, 1999; Owen, 2003b), or even a stratified sample of n points in the n subsquares (we will return to this one in Section 3.4).

Lemma 4 *Under Assumption 1, for any $v \geq 0$, we have $D_{\mathrm{i}}(P_n, v) \leq (v+1)n^{-1/2}$.*

Proof. Consider a function $f : [0,1] \to [0,1]$ with $V(f) \leq v$. We define the *extended graph* of f, denoted $G(f)$, as the boundary between $H(f)$ and $[0,1]^2 \setminus H(f)$. This is the graph of f, $\{(x, f(x)) : 0 \leq x \leq 1\}$, to which we add vertical lines that link the graph pieces where f is discontinuous. The idea of the proof is to bound the number of subsquares that intersect $G(f)$ and then bound the error in terms of this number. The n squares are partitioned into $\sqrt{n}$ columns that correspond to $\sqrt{n}$ intervals on the horizontal axis. Suppose that $G(f)$ goes into ℓ_j different subsquares in column j. Clearly, these ℓ_j subsquares must form a connected rectangle (see Figure 1). For any of these subsquares S, we have an *overestimation* of $\mathrm{area}(H(f) \cap S)$ if the point of P_n lying in subsquare S is in $H(f)$ and *underestimation* otherwise. The total error in any given column is the total amount of overestimation minus the total amount of underestimation. Let $S_{\mathrm{t},j}$ and $S_{\mathrm{b},j}$ denote the top and bottom subsquares from this rectangle, respectively. They may be the same subsquare, if $\ell_j = 1$.

Suppose that we are overestimating in $S_{\mathrm{t},j}$ (as in Figure 1). If we are underestimating in $S_{\mathrm{b},j}$, or if the top and bottom subsquares are the same, the combined error in the top and bottom subsquares cannot exceed $1/n$, so the error in column j is at most $(\ell_j - 1)/n$. Otherwise, i.e., if we are also overestimating in $S_{\mathrm{b},j}$ and $\ell_j \geq 2$ (as in Figure 1), then the error in the bottom subsquare is the surface of this subsquare which is above $G(f)$. This surface cannot exceed $V_{\mathrm{b},j} n^{-1/2}$ where $V_{\mathrm{b},j}$ is the variation of f in this subsquare. Then in this second case, the error in column

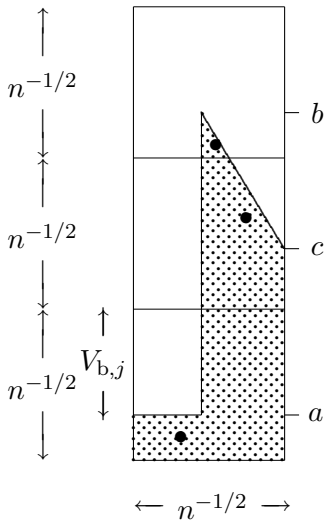

Figure 1: Illustration of the proof of Lemma 4. The shaded surface is $H(f)$ for a given column j. Here, $\ell_j = 3$, we overestimate in each of those three subsquares, and $V_j = (b-a) + (b-c)$.

j is at most $(\ell_j - 1)/n + V_{\mathrm{b},j} n^{-1/2}$. But in both cases, the total variation V_j in column j satisfies $V_j \geq (\ell_j - 2)n^{-1/2} + V_{\mathrm{b},j}$, so the error in column j cannot exceed $V_j n^{-1/2} + n^{-1}$. This same error bound can be obtained by a symmetrical argument in the case where we are underestimating in $S_{\mathrm{t},j}$.

By adding these inequalities over all columns, we obtain that the total error cannot exceed $V(f)n^{-1/2} + n^{1/2}n^{-1} = (V(f)+1)n^{-1/2} \leq (v+1)n^{-1/2}$. By taking the sup over f as in (13), the result follows. □

Let $\mathcal{F}^+$ be the set of *monotone* functions $f : [0,1] \to [0,1]$ and define

$$D_{\mathrm{i}}^+(P_n) = \sup_{f \in \mathcal{F}^+} \left| \mathrm{card}(P_n \cap H(f))/n - \mathrm{area}(H(f)) \right|.$$

For $f \in \mathcal{F}^+$, $V(f)$ cannot exceed 1. This gives:

Corollary 5 *Under Assumption 1, we have* $D_{\mathrm{i}}^+(P_n) \leq 2n^{-1/2}$.

To see that this discrepancy bound is tight, consider the constant function $f(x) = n^{-1/2}(1+\epsilon)$ for a small $\epsilon > 0$ and suppose that each column has two points in $H(f)$. Then $V(f) = 0$ and the error is $(1-\epsilon)n^{-1/2}$, which can be arbitrarily close to the bound $n^{-1/2}$. Theorem 1 of Hlawka (1971) yields an alternative bound for this situation: if we apply it with $s = 2$ and a two-dimensional function that equals 1 in the histogram and 0 elsewhere, we get the looser bound $D_{\mathrm{i}}^+(P_n) \leq 18n^{-1/2}$.

Instead of asking only for the points of P_n to be evenly distributed among the n subsquares, we could have the stronger requirement that they form a $(0, k, 2)$-*net in base 2*, assuming that $n = 2^k$ for

some integer k. This means that for every partition of the unit square into n rectangles of width 2^{-q} and height 2^{-k+q} for some $q = 0, 1, \ldots, k$, every rectangle contains exactly one point. Consider the function $f(x) = x$. Proposition 5 of Lécot (1996) shows that for this example, there is a $(0, k, 2)$-net (namely the Hammersley point set in base 2) for which the error is $n^{-1/2}/2$. In other words, the rate of the bound of Lemma 4 is tight even under this stronger assumption.

3.2 Error Bounds on the State Distribution

Define $\Delta_j(z) = \hat{F}_j(z) - F_j(z)$ and

$$\Delta_j = \sup_{z \in \mathbb{R}} |\Delta_j(z)|,$$

the distance between $\hat{F}_j$ and F_j, with the convention that $\Delta_0 = 0$. Here we derive bounds on Δ_j, under the following assumption, which implies that each φ_j is nondecreasing with respect to its second argument.

Assumption 2 *The Markov chain has a one-dimensional state space $\mathcal{X} \subseteq \mathbb{R}$, so $\ell = 1$, and at each step j, we use inversion from a single uniform random variable to generate the next state X_j from its conditional distribution given X_{j-1}.*

At step j of the Markov chain, for $x \in \mathcal{X}$ and $z \in \mathbb{R}$, let

$$\begin{aligned} F_j(z) &= P[X_j \le z], \\ F_j(z \mid x) &= P[X_j \le z \mid X_{j-1} = x], \\ \Lambda_j(z) &= V(F_j(z \mid \cdot)) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} |dF_j(z \mid x)|, \quad \text{and} \\ \Lambda_j &= \sup_{z \in \mathbb{R}} \Lambda_j(z), \end{aligned}$$

where the differential $dF_j(z \mid x)$ in the definition of $\Lambda_j(z)$ is with respect to x. Thus, $\Lambda_j(z)$ is the total variation of the function $F_j(z \mid \cdot)$. If the Markov chain is stochastically monotone, which means that $F_j(z \mid y) = P[X_j \ge x \mid X_{j-1} = y]$ is monotone in y for each j, then Λ_j cannot exceed 1.

Let $\hat{F}_j$ be the empirical distribution of the states of the n copies of the chain at step j and

$$\tilde{F}_j(z) = \int_{-\infty}^{\infty} F_j(z \mid x) d\hat{F}_{j-1}(x) = \frac{1}{n} \sum_{i=0}^{n-1} F_j(z \mid X_{(i),j-1}),$$

so $\tilde{F}_j$ is the conditional distribution function of X_j when X_{j-1} is generated from $\hat{F}_{j-1}$. The value of $\tilde{F}_j(z)$ is equal to the area of the histogram $H_{j,z}$ whose height over the interval $[i/n, (i+1)/n)$ is

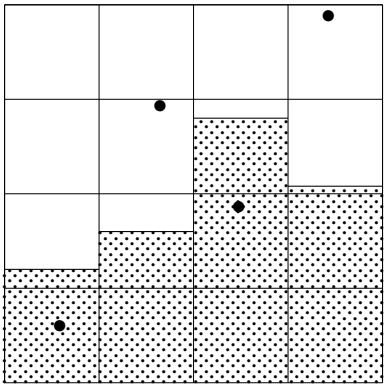

Figure 2: An example of an histogram $H_{j,z}$ for $n = 4$. The histogram is the shaded area, with height $F_j(z \mid X_{(i),j-1})$, and $\hat{F}_j(z)$ is the fraction of $P'_{n,j}$ that falls in $H_{j,z}$.

$F_j(z \mid X_{(i),j-1})$, for $i = 0, \ldots, n-1$. This histogram $H_{j,z}$ is inside the unit square $[0,1)^2$ (see Figure 2 for an illustration). We also have that

$$V[f_{H_{j,z}}] = \sum_{i=1}^{n-1} \left| F_j(z \mid X_{(i-1),j-1}) - F_j(z \mid X_{(i),j-1}) \right| \leq \Lambda_j(z) \leq \Lambda_j, \tag{14}$$

where we have exploited the fact that the states are sorted by increasing order.

The following proposition provides bounds on Δ_j.

Proposition 6 *Let assumption 2 hold.*

(a) Suppose that the Markov chain has a finite state space $\mathcal{X} = \{1, \ldots, L\}$ and that the star discrepancy of each $P'_{n,j}$ satisfies $D_n^*(P'_{n,j}) = O(n^{-1} \log n)$ w.p.1 (this can easily be achieved for example by constructing $P'_{n,j}$ as follows: take the first n points of a $(0,2)$-sequence in some base b, either randomized or deterministic, sort them by their first coordinate, and replace the first coordinate of point i by $(i + 0.5)/n$ for each i). Then

$$|\hat{F}_j(z) - \tilde{F}_j(z)| \leq LKn^{-1} \log n$$

for some constant K. If this holds for all $j \geq 1$ and $z \in \mathbb{R}$, then

$$\Delta_j \leq LKn^{-1} \log n \sum_{k=1}^{j} \prod_{i=k+1}^{j} \Lambda_i. \tag{15}$$

for all j.

(b) If $P'_{n,j}$ satisfies Assumption 1, then for all $j \geq 1$ and $z \in \mathbb{R}$,

$$|\hat{F}_j(z) - \tilde{F}_j(z)| \leq (\Lambda_j(z) + 1)n^{-1/2}$$

and

$$\Delta_j \;\leq\; n^{-1/2}\sum_{k=1}^{j}(\Lambda_k+1)\prod_{i=k+1}^{j}\Lambda_i. \tag{16}$$

Proof. At step j, we have

$$|\Delta_j(z)| = |\hat{F}_j(z) - F_j(z)| \leq |\hat{F}_j(z) - \tilde{F}_j(z)| + |\tilde{F}_j(z) - F_j(z)|. \tag{17}$$

To bound Δ_j we will bound the two quantities on the right of (17). We have

$$\begin{aligned}\tilde{F}_j(z) - F_j(z) &= \int_{-\infty}^{\infty} F_j(z \mid x)d\hat{F}_{j-1}(x) - \int_{-\infty}^{\infty} F_j(z \mid x)dF_{j-1}(x)\\ &= \int_{-\infty}^{\infty}(F_{j-1}(x) - \hat{F}_{j-1}(x))dF_j(z \mid x)\end{aligned} \tag{18}$$

where the second equality is obtained via integration by parts. Therefore,

$$|\tilde{F}_j(z) - F_j(z)| \;\leq\; \int_{-\infty}^{\infty}|\hat{F}_{j-1}(x) - F_{j-1}(x)||dF_j(z \mid x)| \;\leq\; \Lambda_j(z)\Delta_{j-1} \;\leq\; \Lambda_j\Delta_{j-1}. \tag{19}$$

From (19) and (17), we have

$$\Delta_j = \sup_{z\in\mathbb{R}}|\hat{F}_j(z) - F_j(z)| \leq \Lambda_j\Delta_{j-1} + \sup_{z\in\mathbb{R}}|\hat{F}_j(z) - \tilde{F}_j(z)|.$$

Together with the fact that $\Delta_0 = 0$, this gives

$$\Delta_j \leq \sum_{k=1}^{j}\sup_{z\in\mathbb{R}}|\hat{F}_k(z) - \tilde{F}_k(z)|\prod_{i=k+1}^{j}\Lambda_i, \tag{20}$$

where an empty product is assumed to be 1.

We recall that $\tilde{F}_j(z)$ is the area of the histogram $H_{j,z}$ and observe that $\hat{F}_j(z)$ is the fraction of the points of $P'_{n,j}$ that fall in $H_{j,z}$ (Figure 2). Therefore,

$$\hat{F}_j(z) - \tilde{F}_j(z) = \mathrm{card}(P'_{n,j} \cap H_{j,z})/n - \mathrm{area}(H_{j,z}), \tag{21}$$

which implies, using (13), that

$$|\hat{F}_j(z) - \tilde{F}_j(z)| \leq D_{\mathrm{i}}(P'_{n,j}, V(H_{j,z})) \leq D_{\mathrm{i}}(P'_{n,j}, \Lambda_j(z)). \tag{22}$$

This, together with Lemmas 3 and 4, proves the proposition. □

Corollary 7 *If the Markov chain is also* stochastically increasing, *i.e., $P[X_j \geq x \mid X_{j-1} = y]$ is non-decreasing in y for each j, the bound (16) becomes*

$$\Delta_j \leq 2jn^{-1/2}.$$

Proof. Recall that in that case, $F_j(z \mid y)$ is non-decreasing in y, so $\Lambda_j \leq 1$ for each j. □

3.3 Worst-Case Error Bounds on the Expected Average Cost

The next step is to bound the error on the expected cost at step j. Let

$$V(c_j) = \int_{-\infty}^{\infty} |dc_j(x)|,$$

the total variation of the cost function c_j. Our bounds on Δ_j proved in the preceding section readily provide bounds on the error $\left|\bar{Y}_{n,j} - E[c_j(X_j)]\right|$ as follows, in the case where c_j has bounded variation.

Proposition 8 *We have*

$$\left|\bar{Y}_{n,j} - E[c_j(X_j)]\right| \;\leq\; \Delta_j V(c_j).$$

Proof. Using integration by parts for the third equality, we get

$$\begin{aligned}
\left|\bar{Y}_{n,j} - E[c_j(X_j)]\right| &= \left|\frac{1}{n}\sum_{i=0}^{n-1} c_j(X_{i,j}) - E[c_j(X_j)]\right| \\
&= \left|\int_{-\infty}^{\infty} c_j(z)d\hat{F}_j(z) - \int_{-\infty}^{\infty} c_j(z)dF_j(z)\right| \\
&= \left|\int_{-\infty}^{\infty} (\hat{F}_j(z) - F_j(z))dc_j(z)\right| \\
&\leq \int_{-\infty}^{\infty} |\Delta_j(z)||dc_j(z)| \\
&\leq \Delta_j V(c_j).
\end{aligned}$$

□

This proposition tells us that the square error of $\bar{Y}_{n,j}$ converges at worst at the same rate as Δ_j^2 when $n \to \infty$. If τ is bounded, this implies that $|\bar{Y}_n - \mu|^2$ converges as $O(\sum_{j=1}^{\tau} \Delta_j^2) = O(1/n)$ in the worst case.

3.4 Variance Bounds for array-RQMC

Proposition 6(b) gives a *worst-case deterministic* bound of $O(1/n)$ for the square error of $\bar{Y}_{n,j}$, in contrast with an *expected* square error of $O(1/n)$ for ordinary Monte Carlo. In what follows, we obtain a better bound on the *convergence rate of the variance* by exploiting randomization. The proof is given in a setting where the following holds:

Assumption 3 *Let Assumption 1 hold and let the random variables $v_i = \mathbf{u}_i \bmod n^{-1/2}$ be pairwise independent and uniformly distributed over $[0, n^{-1/2})$ (the $\mathbf{u}_i$'s are one-dimensional in this case).*

The second part of this assumption simply means that the second coordinate of each point of $P'_{n,j}$ is uniformly distributed over the (vertical) interval determined by the box that contains this point, *independently* of the position of any other point. With this independence assumption, it is much easier to bound the variance of the array-RQMC estimator. A point set that satisfies this assumption is easily obtained by *stratified sampling* over the unit square: Generate one point in each subsquare of the partition, uniformly over the subsquare and independently across the different subsquares, sort these points by the first coordinate, and then replace the first coordinate of the ith point by $(i + 0.5)/n$. Any point set that satisfies Assumption 3 is also equivalent to this stratified sampling construction in the array-RQMC algorithm.

Proving the result without the independence assumption is more difficult and we leave it for future work. Intuitively, with RQMC point sets that have more uniformity and negative dependence than a stratified sample (e.g., good lattice rules and nets), we would expect an even better variance reduction. This is indeed what we have observed in all our numerical experiments (see Section 4).

Define

$$\Gamma_j = \frac{1}{4n^{3/2}} \sum_{k=1}^{j} (\Lambda_k + 1) \prod_{i=k+1}^{j} \Lambda_i^2. \tag{23}$$

Proposition 9 *Under Assumptions 1, 2, and 3, for each j and z, we have*

$$\mathrm{Var}[\hat{F}_j(z) - \tilde{F}_j(z)] \;\le\; (\Lambda_j + 1) n^{-3/2}/4, \tag{24}$$

$$\mathrm{Var}[\hat{F}_j(z) - F_j(z)] \;\le\; \Gamma_j, \tag{25}$$

and

$$E\left[\left(\bar{Y}_{n,j} - E[c_j(X_j)]\right)^2\right] \le \Gamma_j V(c_j)^2. \tag{26}$$

Proof. To prove (24), we first recall that $\tilde{F}_j(z)$ is the area of the histogram $H_{j,z}$ whereas $\hat{F}_j(z)$ is the fraction of $P'_{n,j}$ that falls in this histogram. We enumerate the n subsquares with $i = 0, \ldots, n-1$. Let S_i be the ith subsquare and $\delta_j(z, i) = \mathrm{card}(P'_{n,j} \cap H_{j,z} \cap S_i) - n\ \mathrm{area}(H_{j,z} \cap S_i)$. We have

$$\hat{F}_j(z) - \tilde{F}_j(z) = \mathrm{card}(P'_{n,j} \cap H_{j,z})/n - \mathrm{area}(H_{j,z}) = \sum_{i=0}^{n-1} \delta_j(z, i)/n.$$

For any given j and each i, $\delta_j(z, i)$ is a Bernoulli random variable minus its mean, so $E[\delta_j(z, i)] = 0$ and $\mathrm{Var}[\delta_j(z, i)] \le 1/4$. These Bernoulli random variables have nonzero variance only for the subsquares that intersect the histogram boundary (that separates $H_{j,z}$ from $[0, 1]^2 \setminus H_{j,z}$), because for the other subsquares they are constant.

Here we consider a fixed j and drop the subscript j for a moment. Let $S_{\mathrm{b},c}$ and $S_{\mathrm{t},c}$ denote the lowest and highest subsquares that intersect the histogram boundary in column c, and let $V_{\mathrm{b},c}$ and $V_{\mathrm{t},c}$ be $n^{1/2}$ times the variation of the histogram in these two subsquares. Let p_c be n times the area of $S_{\mathrm{t},c}$ contained in the histogram and q_c be n times the area of $S_{\mathrm{b},c}$ *not* contained in the histogram. We suppose, until indicated otherwise, that $S_{\mathrm{b},c}$ and $S_{\mathrm{t},c}$ are not the same subsquare. Then the Bernoulli variables $\delta_j(z,i)$ that correspond to these two subsquares are independent and have variances bounded by $p_c(1-p_c)$ and $q_c(1-q_c)$ (which cannot exceed $1/4$), respectively. We will now prove the following bound on the sum of their variances:

$$p_c(1-p_c) + q_c(1-q_c) \le \frac{V_{\mathrm{b},c} + V_{\mathrm{t},c} + 1}{4}. \tag{27}$$

Denote $v = V_{\mathrm{b},c} + V_{\mathrm{t},c}$. If $v \ge 1$, (27) holds trivially, so let us assume that $v \le 1$. At any given point on the horizontal axis, the histogram boundary cannot be in $S_{\mathrm{b},c}$ and $S_{\mathrm{t},c}$ at the same time. Let ρ be the fraction of the horizontal interval in column c where the histogram is in $S_{\mathrm{t},c}$. Then we have $p_c \le \rho V_{\mathrm{t},c}$ and $q_c \le (1-\rho)V_{\mathrm{b},c}$. We now observe that either $\rho V_{\mathrm{t},c} \le v/4$ or $(1-\rho)V_{\mathrm{b},c} \le v/4$. To see this, consider a rectangle of width 1 and height v, with bottom left corner at point $(0,0)$, divided into four subrectangles by a vertical line at ρ and an horizontal line at $V_{\mathrm{t},c}$. The quantities $\rho V_{\mathrm{t},c}$ and $(1-\rho)V_{\mathrm{b},c}$ are the surfaces of two opposite subrectangles of this rectangle, so their surfaces cannot be *both* larger than a quarter of the rectangle's surface v. Indeed, suppose that $\rho V_{\mathrm{t},c} > v/4$. Then, $V_{\mathrm{t},c} > v/(4\rho)$ and therefore $(1-\rho)(v - V_{\mathrm{t},c}) > v/4$ would imply that $(1-\rho)(v - v/(4\rho)) > v/4$, i.e., $0 < (1-\rho)(4\rho-1) - \rho = -(2\rho-1)^2$, which is a contradiction. Therefore, we have either $p_c(1-p_c) \le v/4$ or $q_c(1-q_c) \le v/4$, and since these two quantities never exceed $1/4$, the bound (27) follows. If $S_{\mathrm{b},c}$ and $S_{\mathrm{t},c}$ are the same subsquare, the variance of $\delta_j(z,i)$ in this subsquare cannot exceed $1/4$.

If other subsquares intersect the histogram in column c, between $S_{\mathrm{b},c}$ and $S_{\mathrm{t},c}$, then in each of these subsquares the histogram variation is at least $n^{-1/2}$ and the variance of the corresponding $\delta_j(z,i)$ is at most $1/4$. By adding the above inequalities over all the columns, we obtain that the sum (over i) of variances of all Bernoulli variables $\delta_j(z,i)$ is bounded by $(\Lambda_j + 1)n^{1/2}/4$.

Since these $\delta_j(z,i)$'s are pairwise independent across the different values of i, we obtain

$$\mathrm{Var}\left[\hat{F}_j(z) - \tilde{F}_j(z)\right] \;=\; \sum_{i=0}^{n-1} \mathrm{Var}[\delta_j(z,i)/n] \;\le\; n^{1/2}(\Lambda_j+1)/(4n^2) \;=\; (\Lambda_j+1)n^{-3/2}/4,$$

which proves (24).

We now prove (25) by induction on j. It obviously holds for $j = 0$, because $\hat{F}_0 = F_0$. Suppose it holds for $j-1$, for all z. Observe that from the proof of Proposition 1, $E[\hat{F}_j(x)] = P[X_j \le x] = F_j(x)$, so $E[\Delta_j(x)] = 0$. Then,

$$E[\Delta_{j-1}(x)\Delta_{j-1}(y)] \;=\; \mathrm{Cov}[\Delta_{j-1}(x),\, \Delta_{j-1}(y)] \;\le\; \sup_{z\in\mathbb{R}} \mathrm{Var}[\Delta_{j-1}(z)] \;\le\; \Gamma_{j-1}$$

for all states x, y. Therefore, using (18) for the first equality and assuming that we can interchange the expectation and integral in the third equality,

$$\begin{aligned}
& E[(\tilde{F}_j(z) - F_j(z))^2] \\
= \;& E\left[\left(\int_{-\infty}^{\infty} (\hat{F}_{j-1}(x) - F_{j-1}(x)) dF_j(z \mid x)\right)^2\right] \\
= \;& E\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (\Delta_{j-1}(x)\Delta_{j-1}(y)) dF_j(z \mid x) dF_j(z \mid y)\right] \\
= \;& \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} E[\Delta_{j-1}(x)\Delta_{j-1}(y)] dF_j(z \mid x) dF_j(z \mid y) \\
\le \;& \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Gamma_{j-1} dF_j(z \mid x) dF_j(z \mid y) \\
\le \;& \Lambda_j^2(z)\Gamma_{j-1}.
\end{aligned}$$

Combining this with (24) and (17), and observing that $\hat{F}_j(z) - \tilde{F}_j(z)$ has mean 0 and is uncorrelated with $\tilde{F}_j(z) - F_j(z)$, we obtain that

$$\begin{aligned}
E[(\hat{F}_j(z) - F_j(z))^2] \;&=\; \mathrm{Var}[(\hat{F}_j(z) - F_j(z))^2] \\
&=\; \mathrm{Var}[(\hat{F}_j(z) - \tilde{F}_j(z))^2] + \mathrm{Var}[(\tilde{F}_j(z) - F_j(z))^2] \\
&=\; E[(\hat{F}_j(z) - \tilde{F}_j(z))^2] + E[(\tilde{F}_j(z) - F_j(z))^2] \\
&\le\; \Lambda_j^2(z)\Gamma_{j-1} + (\Lambda_j + 1)n^{-3/2}/4 \\
&\le\; \Gamma_j
\end{aligned}$$

and this completes the induction. To prove the last part, we have

$$\begin{aligned}
& E\left[\left|\frac{1}{n}\sum_{i=0}^{n-1} c_j(X_{i,j}) - E[c_j(X_j)]\right|^2\right] \\
= \;& E\left[\left|\int_{-\infty}^{\infty} c_j(z)d\hat{F}_j(z) - \int_{-\infty}^{\infty} c_j(z)dF_j(z)\right|^2\right] \\
= \;& E\left[\left|\int_{-\infty}^{\infty} (\hat{F}_j(z) - F_j(z))dc_j(z)\right|^2\right] \\
\leq \;& \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} E(\Delta_j(x)\Delta_j(y))|dc_j(x)||dc_j(y)| \\
\leq \;& \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Gamma_j|dc_j(x)||dc_j(y)| \\
\leq \;& (V(c_j))^2\Gamma_j.
\end{aligned}$$

□

From (23) and (26), we thus have an $O(n^{-3/2})$ convergence rate for the variance of the cost at step j if $\ell = d = 1$, the Λ_j's are bounded, and the cost function has bounded variation. When the chain is stochastically increasing, we have $\Lambda_j \leq 1$ for all j and the variance bound becomes $(n^{-3/2}/2)j(V(c_j))^2$.

This could be generalized in principle to higher-dimensional settings, although we are quickly hit by the curse of dimensionality when ℓ or d increases. A potential extension would need a counterpart of Lemma 4. To illustrate the idea, we sketch how this can be done for $d = 2$ and $\ell = 1$ (the state space is still one-dimensional but we need two uniforms per step), for a stochastically increasing Markov chain, so that we have a two-dimensional increasing histogram in the unit cube. Partition the cube into n subcubes by partitioning each axis into $n^{1/3}$ equal parts (assuming that $n^{1/3}$ is an integer). The histogram boundary is now a surface. If we fix one of the two horizontal coordinates to a multiple of $n^{-1/3}$, this determines a vertical plane and the intersection of this plane with the histogram boundary can be covered by at most $2n^{1/3}$ subcubes in a similar manner as in the proof of Lemma 4. We can do this for each multiple of $n^{-1/3}$, and repeat in each direction. We find that the histogram boundary can be covered by at most $Kn^{2/3}$ subcubes for some constant K. Press et al. (1992), page 314, sketch a similar argument in a setting where the integral of an indicator function over the three-dimensional unit cube is approximated by averaging over the (deterministic) points of a Sobol' point set. However, their argument is highly heuristic; it would only stand if the locations of the points in the n subcubes where independent uniform random variables instead of being deterministic.

In general, for a $d+1$-dimensional cube, we conjecture that the histogram boundary can be covered by $Kn^{-d/(d+1)}$ subcubes for some constant K that may depend on d but not on n. This can be turned into a variance bound of $O(n^{2-d/(d+1)}) = O(n^{1+1/(d+1)})$.

The result could also be generalized to the case where c_j has infinite variation (e.g., unbounded state spaces and cost functions) if we assume that large-variation areas have low probability.

In our numerical experiments of Section 4, instead of generating the points independently in the different squares as in the assumptions of Proposition 9, we will generate them according to RQMC schemes that provide more uniformity, with the aim of inducing a larger amount of negative dependence (i.e., more variance reduction) than with straightforward stratification. Some of these RQMC schemes (e.g., the two-dimensional Sobol' nets) have one point per subsquare as in Proposition 9, but none of them really satisfies the assumptions of Proposition 9 because the locations of the points in two different squares are not independent. These RQMC schemes turn out to work better in practice, but so far we have no counterpart of Proposition 9 for them.

4 Numerical Illustrations

We compare MC, classical RQMC, and array-RQMC on a set of examples. We first describe the different RQMC point sets used in our examples, then we explain our experimental setting.

For the RQMC methods, we use Korobov lattice rules and Sobol' point sets. A Korobov rule is defined by two parameters $0 < a < n$ and the corresponding s-dimensional point set is

$$P_n = \{\mathbf{v}_i = (i/n,\, (ia \bmod n)/n, \ldots, (ia^{s-1} \bmod n)/n),\ i = 0, \ldots, n-1\}$$

(Niederreiter, 1992; Sloan and Joe, 1994; L'Ecuyer and Lemieux, 2000). Given n and a, there is no limit on s, so P_n can be viewed as an infinite-dimensional point set (L'Ecuyer and Lemieux, 2000). This P_n is randomized by applying a random shift modulo 1 to all the points, simultaneously. We also consider applying the baker's transformation, which transforms each coordinate u to $2u$ if $u < 1/2$ and to $2(1-u)$ if $u \geq 1/2$, after the random shift. This transformation has a "locally antithetic" effect and provably improves the convergence rate when integrating a smooth function with a randomly shifted lattice rule (Hickernell, 2002). For the parameters, we take n equal to the largest prime number smaller than 2^k for $k = 10, 12, \ldots, 20$, a equal to the odd integer nearest to $n/1.61803399$ when $s = 2$ (so a/n is near the golden ratio; the rule always performs well in the

two-dimensional spectral test with this choice), and a from Table 1 of L'Ecuyer and Lemieux (2000) for point sets in dimensions $s > 2$ (which is always the case for classical RQMC).

Our second type of point set is a Sobol' net with $n = 2^k$ points for $k = 10, 12, \ldots, 20$ (Bratley and Fox, 1988), randomized by a left (upper triangular) matrix scrambling followed by a random digital shift (Matoušek, 1999; L'Ecuyer and Lemieux, 2002; Owen, 2003b; Glasserman, 2004), as implemented in the SSJ software (L'Ecuyer, 2004b). Available implementations of Sobol' nets have an upper bound on the dimension s of the point set, so we cannot use these point sets when the dimension is large or unbounded.

With classical RQMC, we consider the following point sets (in each case, the name in parentheses will refer to the corresponding combination): a randomly-shifted Korobov rule (*Classical-Korobov*); a randomly-shifted Korobov rule with the baker's transformation (*Classical-Korobov-Baker*); and a randomized Sobol' point set (*Classical-Sobol*) when the dimension s was small.

For array-RQMC, the randomization is not applied to the first coordinate, but only to the subsequent coordinates, as explained when we gave our conditions on $P'_{n,j}$. The first coordinate is skipped and is only used to enumerate the points. The Korobov points are always enumerated by order of their (skipped) first coordinate, which takes the values $0, 1/n, \ldots, (n-1)/n$ in succession (in the implicit definition of P'_n, we add $0.5/n$ to these values). For instance, if $d = 1$, the ith point of P_n before the shift is $(ia \bmod n)/n$. The points of a Sobol' net are usually enumerated by order of their Gray code, because this is a bit faster than enumerating them in their natural order (see, e.g., Antonov and Saleev, 1979; Bratley and Fox, 1988; L'Ecuyer, 2004b). Enumerating the points of P_n by their Gray code is equivalent to applying a permutation to the second and further coordinates of the points of P'_n, i.e., it performs an additional scrambling for these coordinates. It is also possible to enumerate the points by order of their first coordinate. We tried both. (Note that for classical RQMC, the order of enumeration is irrelevant.)

In summary, for array-RQMC, we have the following types of d-dimensional RQMC point sets for P_n: a $(d+1)$-dimensional Korobov lattice rule with its first coordinate skipped, randomly shifted (*Array-Korobov*); the same Korobov rule with the random shift followed by a baker's transformation (*Array-Korobov-Baker*); the first n points of a randomized Sobol' sequence where the points are enumerated by order of their Gray code (*Array-Sobol*); the same randomized Sobol' point set but with the points enumerated in their natural order (*Array-Sobol-NoGray*); and for $d = 1$, in the first example, we also tried a stratified sample that satisfies Assumption 3 (*Array-Stratification*).

For each combination and each value of n considered, we estimate the *variance reduction factor* (VRF) compared with standard MC, defined as $\mathrm{Var}[Y]/(n\ \mathrm{Var}[\bar{Y}_n])$ where $\bar{Y}_n$ is the estimator with the RQMC method considered. We estimate $\mathrm{Var}[\bar{Y}_n]$ by the sample variance of $m = 100$ independent copies of $\bar{Y}_n$ and $\mathrm{Var}[Y]$ by the empirical variance of a very large number of independent replicates of Y (up to several millions). Thus, $\mathrm{Var}[Y]$ is very accurately estimated, but $\mathrm{Var}[\bar{Y}_n]$ is not. As a result, the VRF estimates are noisy; as a crude assessment, they have about 10 to 20% relative accuracy. They are all rounded to the nearest integer. The simulations were performed in Java, using the SSJ simulation library (L'Ecuyer, 2004b).

If we measure the work by the number of simulated copies of the chain, then the VRF as we have defined it also measures the *efficiency improvement* of the RQMC method compared with MC. Another possibility would be to measure the work in terms of CPU time. In fact, the CPU times to simulate the chains (excluding the sorting) are about the same for all the methods, with one exception: there is an overhead of 10 to 40 percent for the baker's transformation, presumably because we have implemented it as an additional layer that applies to any point set and involves more method calls. On the other hand, sorting the chains at each step of the array-RQMC algorithm brings significant additional overhead. It requires an $O(n \log n)$ effort as $n \to \infty$, whereas the simulation itself requires $O(n)$ time. In our experiments, we just used the general-purpose sorting procedures available in the standard Java libraries of JDK 1.5 from SUN. When the state space was one-dimensional (in the examples of Sections 4.1, 4.2, 4.3, an 4.6), we stored the states of the chains in an array of real numbers and just sorted that array at each step of array-RQMC. Then, the additional CPU time for sorting the chains ranged approximately from 50% (for $n = 2^{10}$) to 90% (for $n = 2^{20}$), so the overall CPU time was multiplied by a factor between 1.5 and 1.9 with array-RQMC. For higher-dimensional state spaces (the examples of Sections 4.4 and 4.5), the chains were stored as an array of Java *objects*, and in that case the sorting algorithm invokes a *comparison* method each time it compares two objects. This is very general and flexible, but certainly not very efficient. In this case, using array-RQMC multiplied the CPU time by a factor from 2 to 5 with $n = 2^{10}$ and from 8 to 14 with $n = 2^{20}$ (depending on the example). These factors could be reduced by implementing specialized sorting procedures with much fewer method invocations, but we did not do that and made no serious attempt at optimizing our programs for speed. Just as a quick test after completing our experiments, we tried replacing the sorting procedure of JDK 1.5 by another general-purpose procedure taken from the Colt library (Hoschek, 2004) (which also invokes a comparison method but

uses a different sorting algorithm). For array-RQMC with $n = 2^{20}$, on the same computer, the CPU times were approximately halved. To obtain efficiency improvement factors in terms of CPU times, the VRFs reported in the tables would have to be divided by the factors just mentioned. We did not divide by these factors in the tables because they are highly dependent on the Java libraries and Java virtual machine that are used, the choice of programming language, the size of cache memory on the computer, etc., and they can probably be reduced. The VRFs given in the tables are independent of programming implementations.

For array-RQMC with the stratification, the overhead is even more important because the randomization is more expensive and we also have to sort the points explicitly at each step of the chain. In our implementation (for the first example only), the stratified version takes about the same time as the other array-RQMC versions for $n = 2^{10}$ and about twice the time for $n = 2^{20}$.

4.1 An $M/M/1$ Queue with $d = 1$

Consider a single-server queue with i.i.d. exponential interarrival times A_j with mean 1 and i.i.d. exponential service times S_j with mean $\rho < 1$. This ρ is also the utilization factor of the server. We want to estimate the expected average waiting time of the first t customers, denoted μ. We could compute μ numerically without simulation; we just use this simple academic example to illustrate our method.

Let W_j denote the waiting time of customer j in this system, where the first customer (who arrives to the empty system) has number 0. These W_j's satisfy the *Lindley recurrence*: $W_0 = 0$ and $W_j = \max(0, W_{j-1} + S_{j-1} - A_j)$ for $j \geq 1$ (Kleinrock, 1975). We estimate μ by the sample average $Y = (W_0 + \cdots + W_{t-1})/t$. To compute Y, we need to generate the $2(t-1)$ random variates $S_0, A_1, \ldots, S_{t-1}, A_t$. This estimator Y is unbounded (so the Koksma-Hlawka inequality gives an infinite bound for it), but it has bounded variance.

We define a Markov chain that moves by one step each time one of these random variates is generated. That is, $X_0 = W_0$, $X_1 = W_0 + S_0$, $X_2 = W_1$, $X_3 = W_1 + S_1$, and so on. In this case, $d = 1$ and $s = 2(t-1)$. Later, we will consider the case where the chain moves by one step every $d/2$ customers (where d is even), so $X_j = W_{jd/2}$ and $s = 2(t-1)/d$. In all cases, this Markov chain is stochastically increasing.

Our results are for $t = 100$, with $\rho = 0.2$, 0.5, and 0.8. The MC variance per run σ^2 was estimated

by making 100×2^{18} independent simulation runs and our best estimates of μ were obtained via array-RQMC with $n = 2^{20}$. These estimates are (with an accuracy up to the given digits): $\mu = 0.04922$ and $\sigma^2 = 0.0005393$ for $\rho = 0.2$, $\mu = 0.48000$ and $\sigma^2 = 0.06307$ for $\rho = 0.5$, and $\mu = 2.48004$ and $\sigma^2 = 3.1544$ for $\rho = 0.8$.

Table 1: Empirical VRFs of RQMC with respect to MC, for the average waiting time of 100 customers in an $M/M/1$ queue with utilization factor ρ, with $n \approx 2^k$ points.

ρ		$k = 10$	$k = 12$	$k = 14$	$k = 16$	$k = 18$	$k = 20$
	a for classical Korobov-Baker	306	1397	5693	944	118068	802275
	a for array-Korobov	633	2531	10125	40503	162013	648055
0.2	Classical-Korobov-Baker	5	8	15	16	59	117
	Classical-Sobol	1	1	3	1	13	28
	Array-Korobov	18	55	49	292	850	2169
	Array-Korobov-Baker	43	159	306	991	3168	10590
	Array-Sobol	87	282	836	3705	10640	47850
	Array-Sobol-NoGray	46	112	276	874	2914	7429
	Array-Stratification	12	17	31	57	104	183
0.5	Classical-Korobov-Baker	10	7	13	6	14	14
	Classical-Sobol	2	1	4	5	9	10
	Array-Korobov	14	46	33	231	686	2034
	Array-Korobov-Baker	44	200	241	1155	3540	15650
	Array-Sobol	123	504	1083	5651	13830	55160
	Array-Sobol-NoGray	55	130	302	1188	3507	11260
	Array-Stratification	14	23	40	81	142	281
0.8	Classical-Korobov-Baker	11	2	15	17	21	26
	Classical-Sobol	3	2	4	6	10	11
	Array-Korobov	15	85	33	337	727	5119
	Array-Korobov-Baker	70	463	287	2225	10080	75920
	Array-Sobol	370	1281	3240	19730	57290	233100
	Array-Sobol-NoGray	117	288	996	4580	13210	48660
	Array-Stratification	21	40	77	153	246	535

Table 1 reports the empirical VRFs for this example. The array-RQMC methods clearly outperform both MC and classical RQMC, even though classical RQMC is already significantly more efficient than MC (up to 100 times more efficient in one case). The improvement factor is larger when the queue has more traffic (i.e., for larger ρ, which is also when the variance is larger) and larger for the Sobol' nets than for the Korobov rules. For the Korobov rules, adding the baker's transform really makes a difference for both the classical and array methods. This shows that the transformation can be very effective even when the integrand is not smooth (as in the array-RQMC case). We have seen that in all the examples we tried. For this reason, we do not report results for

Korobov rules without the baker's transform in what follows (with one exception).

For the Sobol' nets, the results are better when the points are enumerated in Gray code order. The corresponding scrambling appears helpful, yet we do not have a clear explanation of why.

The stratification provides a significant improvement compared with MC and is even better than classical RQMC, but not quite competitive with the other array-RQMC implementations. We implemented it only for the present example. An interesting fact is that the VRF with Array-Stratification is roughly multiplied by 2 when n is multiplied by 4 (i.e., from one column to the next), so it seems to be proportional to $\sqrt{n}$. This means that the variance appears to be proportional to $n^{-2/3}$, which corresponds exactly to the upper bound proved in Proposition 9.

For classical RQMC, the improvement is much better for the Korobov rules with the baker's transform than for the Sobol' nets. Without the baker's transform (not show in the table) the Korobov rules are just slightly better than the Sobol' nets.

4.2 Increasing d

Suppose that at each step of the Markov chain, we generate d random variates to compute the waiting times of $d/2$ customers ($d = 1$ represents the case examined in the previous subsection). Then the integral (10) approximated by array-RQMC at each step is a $(d+1)$-dimensional integral and we anticipate that the integration error will increase with d. This is confirmed by the following results.

Table 2 shows the empirical VRFs for various values of d, with $n \approx 2^{18}$. For classical RQMC, the exact VRF does not depend on d and the variation observed in the table is only statistical noise; it gives an idea of the accuracy of our VRF estimators. For Array-Sobol, the VRF decreases with d, but not so fast. Moreover, the "NoGray" version becomes comparable to the regular one for $d > 2$. The VRFs are still substantial even for $d = 8$, where the RQMC method approximates 9-dimensional integrals at each step of the Markov chain.

4.3 Random dimension: a Regenerative System

So far in this example, s was fixed at $2(t-1)$. We now modify the example so that $s = \infty$ (variable stopping time). Recall that the $M/M/1$ queue is a *regenerative* system that regenerates whenever a customer arrives to an empty system. Each regenerative cycle contains a random and unbounded number of customers. Suppose we want to estimate $\mu = E[Y]$, where we take the following two

Table 2: Estimated VRFs of classical RQMC and d-dimensional array-RQMC with respect to MC, for the mean waiting time of 100 customers in an $M/M/1$ queue with utilization factor ρ, for selected values of d and $n \approx 2^{18}$.

ρ		$d=1$	$d=2$	$d=4$	$d=8$
0.2	Classical-Korobov-Baker	59	70	73	78
	Classical-Sobol	13	13	12	12
	Array-Korobov-Baker	3168	571	283	137
	Array-Sobol	10640	4329	2247	352
	Array-Sobol-NoGray	2914	5294	2476	403
0.5	Classical-Korobov-Baker	14	22	16	18
	Classical-Sobol	9	6	9	7
	Array-Korobov-Baker	3540	918	152	150
	Array-Sobol	13830	8067	5566	667
	Array-Sobol-NoGray	3507	6206	5205	702
0.8	Classical-Korobov-Baker	21	22	20	28
	Classical-Sobol	10	12	14	10
	Array-Korobov-Baker	10080	2296	1074	597
	Array-Sobol	57290	33360	22550	2515
	Array-Sobol-NoGray	13210	23850	15570	2117

possibilities for Y: (i) the total waiting time of all customers in a regenerative cycle and (ii) the number of customers in a cycle whose waiting time exceeds c, for some constant $c > 0$. Note that changing the uniforms slightly may split or merge regenerative cycles, making Y highly discontinuous in both cases. Moreover, in the second case, Y is integer-valued, so it is not as smooth as in the first case. For our numerical illustration of case (ii), we take $c = 1$. The exact values of μ for case (i) are 0.0625, 1, and 16 for $\rho = 0.2$, 0.5 and 0.8 (computed via standard queueing formulas). For case (ii), they are approximately 0.00458, 0.368, and 3.115 for $\rho = 0.2$, 0.5, and 0.8 (estimated by simulation).

Table 3: Estimated VRFs for the regenerative $M/M/1$ queue with utilization factor ρ, case (i).

ρ		$k=10$	$k=12$	$k=14$	$k=16$	$k=18$	$k=20$
0.2	Classical-Korobov-Baker	3	5	6	5	14	24
	Array-Korobov-Baker	13	28	49	116	289	1093
	Array-Sobol	7	21	46	99	239	756
0.5	Classical-Korobov-Baker	2	3	3	1	6	5
	Array-Korobov-Baker	11	16	37	79	159	438
	Array-Sobol	6	11	24	72	228	469
0.8	Classical-Korobov-Baker	1	1	2	1	2	2
	Array-Korobov-Baker	6	12	22	36	151	237
	Array-Sobol	3	5	19	32	92	225

Table 4: Estimated VRFs for the regenerative $M/M/1$ queue with utilization factor ρ, case (ii).

ρ		$k = 10$	$k = 12$	$k = 14$	$k = 16$	$k = 18$	$k = 20$
0.2	Classical-Korobov-Baker	1	1	2	2	3	2
	Array-Korobov-Baker	3	5	15	22	72	113
	Array-Sobol	2	5	9	23	46	108
0.5	Classical-Korobov-Baker	3	3	4	2	7	6
	Array-Korobov-Baker	22	35	146	253	540	1655
	Array-Sobol	13	33	85	245	645	1847
0.8	Classical-Korobov-Baker	2	1	3	2	3	3
	Array-Korobov-Baker	16	40	100	76	442	997
	Array-Sobol	10	27	81	198	629	1844

Tables 3 and 4 give the estimated VRFs of classical RQMC and array-RQMC compared with standard MC, again with $m = 100$. The improvement factors are not as large as in the two previous tables, but they are still significant, increase with n, and are much larger for the array versions than for the classical ones. The smaller improvements observed here could be due in part to the randomness of τ, as explained earlier: some chains reach τ much later than others and the advantage of a good uniformity of the point set $P'_{n,j}$ decreases with the number of chains left, since we use a smaller and smaller subset of the points. The gain decreases with ρ in case (i) and increases with ρ in case (ii). Note that in case (ii), "$Y > 0$" is a rare event when ρ is very small so in that case something else (such as importance sampling) would have to be done to reduce the variance. The fact that the gain decreases with ρ in case (i) here while the opposite was true in Table 1, for the same performance measure, might be an indication of a loss of efficiency due to a larger variance of τ when ρ increases. For classical RQMC, we need infinite-dimensional RQMC point sets, because the number of steps of the chain is unbounded; so we cannot use Sobol' nets.

4.4 Markovian queues in series

For an example with a higher-dimensional state space, we consider a system of ℓ Markovian queues in series. Customers arrive to the first queue according to a Poisson process with rate λ, go through queue 1, then queue 2, etc., in FIFO order. Service rate at queue q is $\mu_q > \lambda$, for $q = 1, \ldots, \ell$. We uniformize this continuous-time Markov chain (CTMC) model so that the global transition rate is always $\nu = \lambda + \mu_1 + \cdots + \mu_\ell$. At each step, the next transition is an arrival with probability λ/ν and a service completion at queue q with probability μ_q/ν. A service completion at an empty queue is just a dummy event that does not change the state of the system. The embedded discrete-time

Markov chain is $\{X_j = (N_{j,1}, \ldots, N_{j,\ell}),\ j \geq 0\}$ where $N_{j,q}$ is the number of customers in queue q just after the jth transition. So if we are in state X_{j-1} at step j, with probability λ/ν the next state is $X_j = (N_{j,1} + 1, \ldots, N_{j,\ell})$, and with probability μ_q/ν the next state is

$$X_j = \begin{cases} (N_{j,1}, \ldots, N_{j,\ell}) \text{ (unchanged)} & \text{if} \quad N_{j,q} = 0, \\ (N_{j,1}, \ldots, N_{j,q} - 1, N_{j,q+1} + 1, \ldots, N_{j,\ell}) & \text{if} \quad q < \ell \text{ and } N_{j,q} > 0, \\ (N_{j,1}, \ldots, N_{j,\ell} - 1) & \text{if} \quad q = \ell \text{ and } N_{j,\ell} > 0. \end{cases}$$

We assume that the system starts empty and evolves for t transitions, so $X_0 = (0, \ldots, 0)$ and $\tau = t$ in (2). The performance measure μ that we want to estimate is the expected average number of customers in the system over the first t transition. The corresponding cost function at each step is $c_j(X_j) = N_{j,1} + \cdots + N_{j,\ell}$. Each transition is simulated from a single uniform $U_j \in [0, 1)$ (so $d = 1$) as follows: if $U_j > 1 - \lambda/\nu$ we have an arrival, otherwise we have a service completion at queue $q = \min\{i : U_j \leq (\mu_1 + \cdots + \mu_i)/\nu\}$.

The sorting function h is defined by

$$h(N_{j,1}, \ldots, N_{j,\ell}) = N_{j,\ell}(1 + \epsilon) + N_{j,\ell-1}(1 + \epsilon^2) + \cdots + N_{j,1}(1 + \epsilon^\ell)$$

where ϵ is a very small constant (e.g., 10^{-6}). This has the effect of sorting the states primarily according to the total number of customers in the system, then (in case of equalities) according to the number of customers in the last queue, then the number in the next-to-last queue, and so on. This choice is only a heuristic to regroup "similar" states.

We tried two numerical examples. The first one has $\ell = 2$ (a tandem queue), $\lambda = 1$, $\mu_1 = 1.75$, $\mu_2 = 1.25$, and $t = 200$. The second one has $\ell = 3$, $\lambda = 6$, $\mu_1 = 10$, $\mu_2 = 9$, $\mu_3 = 7$, and $t = 200$. Table 5 reports the estimated VRFs of RQMC compared with standard MC. Array-RQMC clearly outperforms classical RQMC for both examples and Array-Sobol does better than Array-Korobov-Baker. We also observe a smaller variance reduction for the three-dimensional example than for the two-dimensional one.

We then repeated the same experiment, but with μ defined as the fraction of the time where the number of customers at the last queue exceeds some constant K; that is, with $c_j(X_j) = I[N_{j,\ell} > K]$. We took $K = 4$ for the tandem queue example and $K = 6$ for the three-queue example. The results are in Table 6. They are quite similar to those of Table 5.

Table 5: Estimated VRFs for the average number of customers in a network of ℓ queues in series, with $n = 2^k$ points.

ℓ		$k = 10$	$k = 12$	$k = 14$	$k = 16$	$k = 18$	$k = 20$
2	Classical-Korobov-Baker	6	1	16	11	16	12
	Classical-Sobol	3	3	5	8	11	9
	Array-Korobov-Baker	34	82	316	476	1733	3233
	Array-Sobol	40	143	480	1104	3785	14340
3	Classical-Korobov-Baker	8	2	15	13	20	15
	Classical-Sobol	4	5	8	7	12	14
	Array-Korobov-Baker	13	78	196	190	884	783
	Array-Sobol	21	55	169	479	1884	4663

Table 6: Estimated VRFs for the fraction of the time where the number of customers at the last queue exceeds K, for a network of ℓ queues in series, with $n = 2^k$ points.

ℓ		$k = 10$	$k = 12$	$k = 14$	$k = 16$	$k = 18$	$k = 20$
2	Classical-Korobov-Baker	4	4	9	10	11	7
	Classical-Sobol	4	6	4	9	9	6
	Array-Korobov-Baker	39	161	595	929	1477	6408
	Array-Sobol	55	226	582	2438	8659	38030
3	Classical-Korobov-Baker	4	2	4	5	5	5
	Classical-Sobol	4	3	3	4	7	5
	Array-Korobov-Baker	11	46	134	358	610	607
	Array-Sobol	16	50	132	450	2091	6082

4.5 Pricing an Asian option

We consider the pricing of an Asian option on a single asset whose value $S(\tau)$ at time τ obeys a geometric Brownian motion: $dS(\tau) = rS(\tau)d\tau + \sigma S(\tau)dB(\tau)$, where r is the risk-free *interest rate*, σ is the (risk-neutral) *volatility* parameter, and $B(\cdot)$ is a standard Brownian motion. The option's value can be written as

$$\mu = E[e^{-rT}C_{\mathrm{a}}(T) \mid S(0)]$$

where

$$C_{\mathrm{a}}(T) = \max\left[0, \left(\frac{1}{s}\sum_{j=1}^{s} S(\tau_j)\right) - K\right]$$

and $0 < \tau_1 < \cdots < \tau_s = T$ are the discrete observation times. See, e.g., Hull (2000) and Glasserman (2004) for further details. We have

$$S(\tau_j) = S(\tau_{j-1})\exp[(r - \sigma^2/2)(\tau_j - \tau_{j-1}) + \sigma(\tau_j - \tau_{j-1})^{1/2}\Phi^{-1}(U_j)] \tag{28}$$

for $j = 1, \ldots, s$, where the U_j's are independent U$(0,1)$ random variables and Φ is the standard normal distribution function. To get an unbiased estimator of μ it suffices to generate $S(\tau_1), \ldots, S(\tau_s)$ via (28), with s i.i.d. U$(0,1)$ random variates, and compute the estimator $X = e^{-rT} C_{\mathrm{a}}(T)$.

To apply the *array-RQMC* method, we define the *state* of the chain at step j as the two-dimensional vector $X_j = (S(\tau_j), \bar{S}_j)$, where $\bar{S}_j = (S(\tau_1) + \cdots + S(\tau_j))/j$. We order these states simply by increasing order of their value of $S(\tau_j)$, i.e., we define the sorting function by $h(x_1, x_2) = x_1$ (there are other possibilities, probably better ones).

Table 7: Estimated VRFs for an Asian option with s observation times, with $n = 2^k$ points.

s		$k = 10$	$k = 12$	$k = 14$	$k = 16$	$k = 18$	$k = 20$
10	Classical-Korobov	188	594	2601	5505	18050	11040
	Classical-Korobov-Baker	2629	10600	5104	83450	27560	93620
	Classical-Sobol	4844	11460	28740	46020	142900	222800
	Array-Korobov-Baker	4783	13280	23960	45990	36670	39950
	Array-Sobol	5080	13030	37460	38320	36360	32430
60	Classical-Korobov	81	17	352	406	552	497
	Classical-Korobov-Baker	481	567	919	610	1362	1745
	Classical-Sobol	282	488	907	787	1654	2413
	Array-Korobov-Baker	1187	1742	1218	2231	1680	1998
	Array-Sobol	1234	1742	2050	2203	2189	1866
120	Classical-Korobov	73	27	152	209	252	276
	Classical-Korobov-Baker	244	380	452	407	581	498
	Classical-Sobol	68	92	234	253	531	410
	Array-Korobov-Baker	816	1263	1355	1736	1456	1635
	Array-Sobol	1423	1485	1260	1390	1333	1477
240	Classical-Korobov	30	9	93	116	95	148
	Classical-Korobov-Baker	76	167	233	303	375	319
	Classical-Sobol	29	32	54	69	151	217
	Array-Korobov-Baker	445	703	375	758	773	601
	Array-Sobol	744	769	670	725	702	667

For a numerical illustration, let $S(0) = 100$, $K = 90$, $T = 240/365$ (in years), $\tau_j - \tau_{j-1} = 1/365$ for all j, $\tau_1 = T - (s-1)/365$, $r = \log 1.09$, $\sigma = 0.2$, and $s = 10$, 60, 120, and 240. Table 7 gives the estimated VRFs of RQMC compared with standard MC. The *classical* RQMC methods uses the straightforward simulation approach described above. Efficiency can be further improved by combining RQMC with *bridge sampling* and other variance-reduction techniques such as control variates and importance sampling (Caflisch and Moskowitz, 1995; Glasserman, 2004; L'Ecuyer and Lemieux, 2000; L'Ecuyer, 2004a) but we do not go in that direction.

Here, classical RQMC is already very effective when s is small, and array-RQMC is not really

better. For larger s, however, array-RQMC eventually provides larger VRFs, especially when k is small. For $s = 240$ and $k = 10$, for example, the variance is approximately seven times smaller for Array-Sobol than for the best classical RQMC method. On the other hand, it is disappointing to see that the VRFs eventually stabilize when we increase n. This suggests a $O(1/n)$ asymptotic rate of convergence of the variance for this particular implementation and choice of sorting function h. Another important point to notice is that once again, the baker's transformation applied on top of the Korobov rules really helps. In previous experiments with this Asian option example, the lattice rules were used only *without* the baker's transformation (L'Ecuyer and Lemieux, 2000).

4.6 Estimating a small ruin probability with importance sampling and array-RQMC

A (simplified) insurance company receives *premiums* at constant rate $c > 0$ and *claims* according to a Poisson process $\{N(t), t \geq 0\}$ with rate $\lambda > 0$. The claim sizes C_j, $j \geq 1$, are i.i.d. random variables with density h. The *reserve* (amount of money in hand) at time t is

$$R(t) = R(0) + ct - \sum_{j=1}^{N(t)} C_j,$$

where $R(0)$ is the initial reserve. We want to estimate the *ruin* probability, i.e., the probability μ that $R(t)$ eventually becomes negative.

Note that ruin can occur only at the time of a claim. The reserve just after claim j is

$$X_j = X_{j-1} + A_j c - C_j, \qquad j \geq 1,$$

where $X_0 = R(0)$ and A_j is the time between claims $j-1$ and j. The process $\{X_j, j \geq 0\}$ is a random walk. This process cannot be simulated directly to estimate the ruin probability because: (1) we cannot be 100% sure that ruin does not occur if we simulate only for a finite time and (2) in practice ruin occurs very rarely. We can get around these difficulties by using *importance sampling* (IS) with *exponential twisting* as follows (Asmussen, 1985). Assuming that h has a finite moment generating function M_h around 0, we replace the density $h(x)$ by

$$h_\theta(x) = h(x)e^{\theta x}/M_h(\theta)$$

and increase the rate λ of the Poisson process to $\lambda_\theta = \lambda + \theta c$, where θ is the largest solution to the Lundberg equation $M_h(\theta) = (\lambda + \theta c)/\lambda$. Under this IS scheme, ruin occurs with probability 1 and

the (unbiased) estimator of μ is

$$e^{\theta(X_\tau - X_0)} \tag{29}$$

where $\tau = \inf\{j : X_j < 0\}$, a random stopping time. Here, there is no need to store the intermediate values of the likelihood ratio during the simulation, because its final value depends only on X_τ. Thus, the state space is one-dimensional.

We are interested in seeing if a combination of IS with array-RQMC is more efficient than IS alone. A priori, since the function $f(\mathbf{u}) = e^{\theta(X_\tau - X_0)}$ is *sawtooth-like* (not smooth at all) with respect to each coordinate of $\mathbf{u}$, we do not expect RQMC to perform well.

For a numerical experiment, we take $\lambda = 1$, exponential claim sizes with mean $1/\beta = 2$, and $R(0) = 200$. We use $d = 1$, i.e., one step of the chain each time a uniform random number is generated. The number of steps before ruin occurs is random.

Table 8 gives the estimated VRFs compared with MC (with IS) for $c = 3$, 5, and 10. For classical RQMC we need an infinite-dimensional point set; this rules out Classical-Sobol. The exact ruin probability μ is approximately $\mu = 2.2 \times 10^{-15}$ for $c = 3$, $\mu = 3.5 \times 10^{-27}$ for $c = 5$, and $\mu = 3.6 \times 10^{-36}$ for $c = 10$. The gains are not as spectacular as for the previous examples, but they are nevertheless significant for large n.

Table 8: Estimated VRFs for the ruin probability example, with inflow rate c and $n = 2^k$ points.

c		$k = 10$	$k = 12$	$k = 14$	$k = 16$	$k = 18$	$k = 20$
10	Classical-Korobov-Baker	1	1	1	1	1	1
	Array-Korobov-Baker	3	3	7	3	15	27
	Array-Sobol	2	2	6	10	19	45
5	Classical-Korobov-Baker	1	1	1	1	2	1
	Array-Korobov-Baker	3	4	10	5	21	37
	Array-Sobol	2	4	8	13	33	73
3	Classical-Korobov-Baker	1	1	1	1	1	1
	Array-Korobov-Baker	2	4	8	7	24	38
	Array-Sobol	2	5	7	17	30	49

5 Conclusion

We have proposed a new RQMC method for Markov chains, proved results on its convergence for special cases, and tested it numerically on examples. The new method provides large efficiency

gains compared with standard MC in our examples. It performs better than classical RQMC in the examples where the Markov chain has a one-dimensional state space and evolves over several steps. Generally speaking, the performance of the array-RQMC method tends to degrade when the integrand has higher variability, or when the dimension of the state space becomes larger than 1 and there is no natural (or obvious) sorting function for the states. But even in these cases, there can be significant variance reductions compared with standard MC, and sometimes compared with classical RQMC as well. Our paper also provides novel empirical evidence of the effectiveness of applying the baker's transformation over a randomly shifted lattice rule, an idea that was studied theoretically by Hickernell (2002).

Obtaining better convergence bounds for the variance is a goal that would certainly deserve further work. From the practical viewpoint, an interesting challenge would be to find good ways of defining the sorting function h for specific classes of problems where the Markov chain has a multidimensional state space. Our on-going and future work also includes studying the application of array-RQMC to other settings that fit a general Markov chain framework, such as Markov Chain Monte Carlo methods and stochastic approximation algorithms, for example.

6 Acknowledgments

The work of the first author has been supported by NSERC-Canada grant No. ODGP0110050, NATEQ-Québec grant No. 02ER3218, and a Canada Research Chair. The work of the third author has been supported by EuroNGI Network of Excellence and the "SurePath ACI sécurité" project. A (short) preliminary version of this paper appeared in the Proceedings of the MCQMC'2004 Conference (L'Ecuyer et al., 2006). Richard Simard helped preparing the software for the numerical experiments and Charles Sanvido implemented the stratification. We thank the Associate Editor and two referees for their helpful comments.

References

Antonov, I. A. and Saleev, V. M. (1979). An economic method of computing LP_τ-sequences. *Zh. Vychisl. Mat. i. Mat. Fiz.*, 19:243–245. In Russian.

Asmussen, S. (1985). Conjugate processes and the simulation of ruin problems. *Stochastic Processes and their Applications*, 20:213–229.

Ben-Ameur, H., L'Ecuyer, P., and Lemieux, C. (2004). Combination of general antithetic transformations and control variables. *Mathematics of Operations Research*, 29(4):946–960.

Bratley, P. and Fox, B. L. (1988). Algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software*, 14(1):88–100.

Caflisch, R. E. and Moskowitz, B. (1995). Modified Monte Carlo methods using quasi-random sequences. In Niederreiter, H. and Shiue, P. J.-S., editors, *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, number 106 in Lecture Notes in Statistics, pages 1–16, New York. Springer-Verlag.

Garvels, M. J. J., Kroese, D. P., and Van Ommeren, J.-K. C. W. (2002). On the importance function in splitting simulation. *European Transactions on Telecommunications*, 13(4):363–371.

Glasserman, P. (2004). *Monte Carlo Methods in Financial Engineering*. Springer-Verlag, New York.

Glasserman, P., Heidelberger, P., Shahabuddin, P., and Zajic, T. (1999). Multilevel splitting for estimating rare event probabilities. *Operations Research*, 47(4):585–600.

Hickernell, F. J. (1998). A generalized discrepancy and quadrature error bound. *Mathematics of Computation*, 67:299–322.

Hickernell, F. J. (2002). Obtaining $o(n^{-2+\epsilon})$ convergence for lattice quadrature rules. In Fang, K.-T., Hickernell, F. J., and Niederreiter, H., editors, *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 274–289, Berlin. Springer-Verlag.

Hlawka, E. (1971). Discrepancy and Riemann integration. In Mirsky, L., editor, *Studies in Pure Mathematics: papers in combinatorial theory, analysis geometry, algebra and the theory of numbers*, pages 121–129. Academic Press.

Hoschek, W. (2004). *The Colt Distribution: Open Source Libraries for High Performance Scientific and Technical Computing in Java*. CERN, Geneva. Available at `http://dsd.lbl.gov/~hoschek/colt/`.

Hull, J. (2000). *Options, Futures, and Other Derivative Securities*. Prentice-Hall, Englewood-Cliff, N.J., fourth edition.

Kleinrock, L. (1975). *Queueing Systems, Vol. 1*. Wiley, New York.

Kollig, T. and Keller, A. (2002). Efficient bidirectional path-tracing by randomized quasi-Monte Carlo integration. In Fang, K.-T., Hickernell, F. J., and Niederreiter, H., editors, *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 290–305, Berlin. Springer-Verlag.

Lécot, C. (1996). Error bound for quasi-Monte Carlo integration with nets. *Mathematics of Computation*, 65(213):179–187.

Lécot, C. and Ogawa, S. (2002). Quasirandom walk methods. In Fang, K.-T., Hickernell, F. J., and Niederreiter, H., editors, *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 63–85, Berlin. Springer-Verlag.

Lécot, C. and Tuffin, B. (2004). Quasi-Monte Carlo methods for estimating transient measures of discrete time Markov chains. In Niederreiter, H., editor, *Monte Carlo and Quasi-Monte Carlo Methods 2002*, pages 329–343, Berlin. Springer-Verlag.

L'Ecuyer, P. (2004a). Quasi-Monte Carlo methods in finance. In Ingalls, R. G., Rossetti, M. D., Smith, J. S., and Peters, B. A., editors, *Proceedings of the 2004 Winter Simulation Conference*, Piscataway, New Jersey. IEEE Press.

L'Ecuyer, P. (2004b). *SSJ: A Java Library for Stochastic Simulation*. Software user's guide, Available at `http://www.iro.umontreal.ca/~lecuyer`.

L'Ecuyer, P., Lécot, C., and Tuffin, B. (2006). Randomized quasi-Monte Carlo simulation of Markov chains with an ordered state space. In Niederreiter, H. and Talay, D., editors, *Monte Carlo and Quasi-Monte Carlo Methods 2004*, pages 331–342.

L'Ecuyer, P. and Lemieux, C. (2000). Variance reduction via lattice rules. *Management Science*, 46(9):1214–1235.

L'Ecuyer, P. and Lemieux, C. (2002). Recent advances in randomized quasi-Monte Carlo methods. In Dror, M., L'Ecuyer, P., and Szidarovszky, F., editors, *Modeling Uncertainty: An Examination of Stochastic Theory, Methods, and Applications*, pages 419–474. Kluwer Academic, Boston.

Matoušek, J. (1999). *Geometric Discrepancy: An Illustrated Guide*. Springer-Verlag, Berlin.

Niederreiter, H. (1992). *Random Number Generation and Quasi-Monte Carlo Methods*, volume 63 of *SIAM CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia.

Owen, A. B. (1998). Latin supercube sampling for very high-dimensional simulations. *ACM Transactions on Modeling and Computer Simulation*, 8(1):71–102.

Owen, A. B. (2003a). Quasi-Monte Carlo sampling. In Jensen, H. W., editor, *Monte Carlo Ray Tracing: Siggraph 2003, Course 44*. Available at `http://www-stat.stanford.edu/~owen/reports/siggraph03.pdf`.

Owen, A. B. (2003b). Variance with alternative scramblings of digital nets. *ACM Transactions on Modeling and Computer Simulation*, 13(4):363–378.

Owen, A. B. (2005). Multidimensional variation for quasi-Monte Carlo. In Fan, J. and Li, G., editors, *International Conference on Statistics in honour of Professor Kai-Tai Fang's 65th birthday*. Available at `http://www-stat.stanford.edu/~owen/reports/`.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes in C: The Art of Scientific Computing*. Combridge University Press, New York, second edition.

Sloan, I. H. and Joe, S. (1994). *Lattice Methods for Multiple Integration*. Clarendon Press, Oxford.

Tezuka, S. (2002). Quasi-Monte Carlo—discrepancy between theory and practice. In Fang, K.-T., Hickernell, F. J., and Niederreiter, H., editors, *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 124–140, Berlin. Springer-Verlag.

Tuffin, B. (1996). On the use of low discrepancy sequences in Monte Carlo methods. *Monte Carlo Methods and Applications*, 2(4):295–320.

Wilson, J. R. (1983). Antithetic sampling with multivariate inputs. *American Journal of Mathematical and Management Sciences*, 3:121–144.