On Array-RQMC for Markov Chains: Mapping Alternatives and Convergence Rates

Pierre L'Ecuyer, Christian Lécot, and Adam L'Archevêque-Gaudet

Abstract We study the convergence behavior of a randomized quasi-Monte Carlo (RQMC) method for the simulation of discrete-time Markov chains, known as array-RQMC. The goal is to estimate the expectation of a smooth function of the sample path of the chain. The method simulates n copies of the chain in parallel, using highly uniform point sets randomized independently at each step. The copies are sorted after each step, according to some multidimensional order, for the purpose of assigning the RQMC points to the chains. In this paper, we provide some insight on why the method works, explain what would need to be done to bound its convergence rate, discuss and compare different ways of realizing the sort and assignment, and report empirical experiments on the convergence rate of the variance and of the mean square discrepancy between the empirical and theoretical distribution of the states, as a function of n, for various types of discrepancies.

1 Introduction

Quasi-Monte Carlo (QMC) and randomized QMC (RQMC) methods can be quite effective to estimate an integral when the integrand is reasonably smooth and has low effective dimension [11, 16, 20]. But when we simulate a system (modeled as a Markov chain) that evolves over several time steps, and the integrand is a function

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

Christian Lécot
Laboratoire de Mathématiques, UMR 5127 CNRS, and Université de Savoie, 73376 Le Bourget-du-Lac Cedex, France,
Christian.Lecot@univ-savoie.fr

of the sample path, the dimension is typically very large, the effective dimension can also be large, and RQMC is often not very effective.

A different type of QMC and RQMC methodology, whose RQMC version is called array-RQMC, has been introduced and developed in [8, 9, 13, 14]. This array-RQMC algorithm simulates n copies of the chain in parallel. It advances all copies by one step at each iteration, using an RQMC point set of cardinality n to generate the transitions of these chains at the given step, and a clever matching of the RQMC points to the chains. This matching is done by sorting both the chains and the points according to their successive coordinates. The idea is (loosely speaking) to induce negative dependence between the n copies, so that the empirical distribution of the n states at any given step provides a much more accurate approximation of the true distribution than if the n copies were simulated independently [14]. Empirical experiments have shown that this can improve the simulation efficiency for Markov chains simulated over several hundred steps, sometimes by factors of over 1000. Potential applications include queueing systems, option pricing in finance, reliability and risk assessment models, image generation in computer graphics, and more [2, 12, 14, 21].

The aim of this paper is to provide further insight on why the method works, examine and compare alternative ways of matching the RQMC points to the chains at each step, and report empirical experiments on the convergence rate of the variance and of the mean square discrepancy between the empirical and theoretical distribution of the states, as a function of n, for various types of discrepancies.

The remainder is organized as follows. The Markov chain setting and the estimation problems are defined in Section 2. In Section 3, we explain the array-RQMC algorithm, provide (heuristic) arguments for why an how the variance of the resulting estimator could converge faster than the Monte Carlo rate of $O(1/n)$, and discuss what would be the required ingredients to bound this convergence rate. In Section 4, we examine how to map the chains to the RQMC points at each step. Empirical investigations of the convergence rate of the variance and the mean square discrepancy are reported in Section 5. A conclusion is given in Section 6.

2 A Markov chain setting

We consider a Markov chain model with state space $\mathscr{X} \subseteq \mathbb{R}^\ell$, whose state evolves according to the stochastic recursion

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

where x_0 is fixed, $\mathbf{U}_1, \mathbf{U}_2, \ldots$ are i.i.d. uniform random variables over the unit hypercube $(0,1)^d$, and $\varphi_j : \mathscr{X} \times (0,1)^d \to \mathscr{X}$ is a measurable mapping for each j. As usual, we assume that the uniform random variables never take the value 0 or 1, to avoid infinite realizations after they are transformed by inversion to normals, exponentials, etc. We want to estimate

$$\mu = \mathbb{E}[Y], \quad \text{where} \quad Y = \sum_{j=1}^{\tau} c_j(X_j)$$

for some measurable *cost functions* $c_j : \mathscr{X} \to \mathbb{R}$, and τ is a fixed positive integer. This can in fact be generalized to the case where τ is a random stopping time, for example the first (smallest) time when X_j hits a given subset of states. The array-RQMC also works in that case, but its performance (in terms of variance reduction) is usually not as good as when τ is fixed, according to our experiments (see also [11] for one example).

To estimate μ by ordinary Monte Carlo (MC), we proceed as follows. Given a large integer n, for each i, $i = 0, \ldots, n-1$, we generate a sample path of the chain via

$$X_{i,0} = x_0, \qquad X_{i,j} = \varphi_j(X_{i,j-1}, \mathbf{U}_{i,j}), \quad j = 1, \ldots, \tau, \tag{1}$$

where $\mathbf{U}_{i,1}, \ldots, \mathbf{U}_{i,\tau}$'s are i.i.d. uniform over $(0,1)^d$, and we compute $Y_i = \sum_{j=1}^{\tau} c_j(X_{i,j})$, the realization number i of Y. These sample paths are independent. The MC estimator of μ is then

$$\hat{\mu}_n = \frac{1}{n} \sum_{i=0}^{n-1} Y_i. \tag{2}$$

For the classical RQMC method, let $s = \tau d$ and put $\mathbf{V}_i = (\mathbf{U}_{i,1}, \mathbf{U}_{i,2}, \ldots, \mathbf{U}_{i,s/d})$. Let $P_n = \{\mathbf{V}_0, \ldots, \mathbf{V}_{n-1}\} \subset (0,1)^s$ be an s-dimensional *RQMC point set*, defined as a point set with the following properties [15, 17]: (a) each point $\mathbf{V}_i$ has the uniform distribution over $(0,1)^s$, and (b) P_n has low discrepancy in some sense (the precise meaning would depend on the definition of discrepancy that one would adopt, and this may depend on the problem context). The RQMC estimator of μ is defined as in (2):

$$\hat{\mu}_{\text{rqmc},n} = \frac{1}{n} \sum_{i=0}^{n-1} Y_i = \frac{1}{n} \sum_{i=0}^{n-1} \sum_{j=1}^{\tau} c_j(X_{i,j}), \tag{3}$$

where the $X_{i,j}$ are also defined as in the MC estimator. One difficulty here is that the dimension s can be very large when the chain has many steps.

3 The Array-RQMC Algorithm

With the array-RQMC method introduced in [13, 14], we simulate n chains in parallel, and use a d-dimensional RQMC point set P_n at each step to advance all the chains by one step, in a way that at each step j, the empirical distribution of the set of states $S_{n,j} = \{X_{0,j}, \ldots, X_{n-1,j}\}$ is a very accurate approximation of the theoretical distribution of X_j, hopefully more accurate than with standard Monte Carlo. We want the discrepancy between these two distributions to be as small as possible, for an appropriate measure of discrepancy whose choice may depend on the application.

To explain what this means and why we want to do that, let $\mu_j = \mathbb{E}[c_j(X_j)]$ be the expected cost at step j, and

$$\hat{\mu}_{\text{rqmc},j,n} = \frac{1}{n}\sum_{i=0}^{n-1} c_j(X_{i,j}), \tag{4}$$

the sample average cost over the n chains at step j. The methods considered in this paper estimate μ_j by $\hat{\mu}_{\text{rqmc},j,n}$ and are unbiased: $\mathbb{E}[\hat{\mu}_{\text{rqmc},j,n}] = \mu_j$ (for array-RQMC, see Proposition 1 below). Our goal is to reduce the variance $\text{Var}[\hat{\mu}_{\text{rqmc},j,n}]$, which in this case is the same as the mean square error $\mathbb{E}[(\hat{\mu}_{\text{rqmc},j,n} - \mu_j)^2]$. It would also be nice if we could show (under appropriate conditions) that this variance converges faster than $O(1/n)$, which is the ordinary MC rate. In the remainder of this section, we explain (with heuristic arguments) why the array-RQMC appears a sensible way to achieve that.

Let us assume for now that X_j has the uniform distribution over $\mathscr{X} = (0,1)^\ell$ for each j. This assumption is in force up to the statement of Proposition 1; after that we will relax it to cover the case of a more general distribution of X_j over $\mathbb{R}^\ell$. As is usually done to bound the mean square error for RQMC schemes [3, 4, 5, 11], we can select a reproducing kernel Hilbert space (RKHS) of functions $c_j : (0,1)^\ell \to \mathbb{R}$, from which we obtain a definition of function variation V and a corresponding definition of discrepancy D for randomized point sets in $(0,1)^\ell$, such that

$$\mathbb{E}[(\hat{\mu}_{\text{rqmc},j,n} - \mu_j)^2] \le \mathbb{E}[D^2(S_{n,j})]\, V^2(c_j), \tag{5}$$

which provides a variance bound whenever $V(c_j) < \infty$. The next step would be to make sure that $\mathbb{E}[D^2(S_{n,j})]$ is small for all j, and (ideally) that it converges faster than $O(1/n)$ for any fixed j.

In the RKHS case, $D(S_{n,j})$ is equal to the integration error of some representer function ξ_j (say) that depends on $S_{n,j}$, and such that $V(\xi_j) < \infty$. If (1) holds for all i, for some points $\mathbf{U}_{i,j} \in (0,1)^d$, then $D(S_{n,j})$ can also be written as the integration error of $\xi_j \circ \varphi_j$ by the (randomized) point set $Q_n = \{(X_{0,j-1}, \mathbf{U}_{0,j}), \ldots, (X_{n-1,j-1}, \mathbf{U}_{n-1,j})\}$. To bound this integration error, we may select another discrepancy $D_{(2)}$ defined over the $(\ell+d)$-dimensional unit hypercube, with corresponding variation $V_{(2)}$, such that for any function $g : (0,1)^{\ell+d} \to \mathbb{R}$ with $V_{(2)}(g) < \infty$, the mean square integration error of g by Q_n is bounded by $\mathbb{E}[D^2_{(2)}(Q_n)] \cdot V^2_{(2)}(g)$. This discrepancy measure $D_{(2)}$ can of course be different from D. Its role is to measure the departure of the empirical distribution of Q_n from the uniform distribution over $(0,1)^{\ell+d}$. If we can show that $V_{(2)}(\xi_j \circ \varphi_j) < \infty$ and that $\mathbb{E}[D^2_{(2)}(Q_n)] = O(n^{-\alpha+\varepsilon})$ for any $\varepsilon > 0$ for some constant $\alpha > 1$, then this would imply that $\text{Var}[\hat{\mu}_{\text{rqmc},j,n}]$ converges faster than $O(1/n)$, which is what we are trying to achieve. Of course, this may work only if $\xi_j \circ \varphi_j$ has sufficient "smoothness."

Note that in the points $(X_{i,j-1}, \mathbf{U}_{i,j})$ of Q_n, the last d coordinates (the $\mathbf{U}_{i,j}$) can be defined via some RQMC scheme, but the $X_{i,j-1}$ cannot be chosen; they are determined by the previous history of the chains. The aim is to select (or generate) the $\mathbf{U}_{i,j}$ in a way that $\mathbb{E}[D^2_{(2)}(Q_n)]$ is small.

In the array-RQMC algorithm defined below, we (try to) achieve this in the following way. We select an $(\ell+d)$-dimensional point set

$$\tilde{Q}_n^0 = \{(\mathbf{w}_0, \tilde{\mathbf{u}}_0), \ldots, (\mathbf{w}_{n-1}, \tilde{\mathbf{u}}_{n-1})\}$$

having low-discrepancy with respect to $D_{(2)}$, where $\mathbf{w}_i \in [0,1)^\ell$ and $\tilde{\mathbf{u}}_i \in [0,1)^d$ (these points are allowed to have zero coordinates). Then we define a randomization of $\tilde{P}_n^0 = \{\tilde{\mathbf{u}}_0, \ldots, \tilde{\mathbf{u}}_{n-1}\}$ with the property that if $P_n = (\mathbf{U}_0, \ldots, \mathbf{U}_{n-1})$ is (a realization of) the randomized version and if $\tilde{Q}_n$ is the version of $\tilde{Q}_n^0$ in which $\tilde{P}_n^0$ is replaced by its randomized version P_n, then: (a) each $\mathbf{U}_i$ is uniformly distributed over $(0,1)^d$ and (b) $\tilde{Q}_n$ has low discrepancy, in the sense that $\mathbb{E}[D_{(2)}^2(\tilde{Q}_n)]$ is small. Note that $\tilde{Q}_n^0$ does not have to be the same at all steps j, but taking the same point set (with independent randomizations at the different steps) is more convenient and works fine in practice.

Then we define a permutation π_j over $\{0, \ldots, n-1\}$, for which $X_{\pi_j(i),j-1}$ is close to $\mathbf{w}_i$ for each i, as much as possible, so that there is not much difference (loosely speaking) between the point sets $\tilde{Q}_n$ and

$$Q_{n,j}^\pi = \{(X_{\pi_j(0),j-1}, \mathbf{U}_{0,j}), \ldots, X_{\pi_j(n-1),j-1}, \mathbf{U}_{n-1,j}\}.$$

The motivation is that if these two point sets are close to each other, then $Q_{n,j}^\pi$ should also have low discrepancy. This RQMC point set $Q_{n,j}^\pi$ is the one that turns out to be used to approximate the integral of $\xi_j \circ \varphi_j$ at step j of the algorithm. The $\mathbf{w}_i$ are fixed once for all and are the same at all steps; their role is only to define the mapping between the chains and the points of $\tilde{Q}_n$. In the case of a one-dimensional state space ($\ell = 1$), we usually take $\mathbf{w}_i = (i + 1/2)/n$ and then the best permutation π_j is the one for which the states $X_{\pi_j(i),j-1}$ are sorted in increasing order, because the $\mathbf{w}_i$ are sorted in increasing order. The choice of permutations for higher-dimensional state spaces is less obvious. We discuss it in Section 4.

The array-RQMC algorithm simulates (in parallel) n copies of the chain; it can be summarized as follows.

Array-RQMC algorithm:
For $i = 0, \ldots, n-1$, let $X_{i,0} = x_0$;
For $j = 1, 2, \ldots, \tau$ {
  Randomize $\tilde{P}_n^0$ afresh (independently of the previous
    randomizations) into a new $P_n = P_{n,j} = \{\mathbf{U}_{0,j}, \ldots, \mathbf{U}_{n-1,j}\}$;
  For $i = 0, \ldots, n-1$, let $X_{i,j} = \varphi_j(X_{\pi_j(i),j-1}, \mathbf{U}_{i,j})$;
  Compute the permutation π_{j+1} for the next step;
  }
Estimate μ by the same average $\bar{Y}_n = \hat{\mu}_{\text{rqmc},n}$ as in (3).

This can be replicated m times to estimate the variance and compute a confidence interval on μ. The following is proved in [14]:

Proposition 1. *(a) $\bar{Y}_n$ is an unbiased estimator of μ and (b) the empirical variance of the m copies of $\bar{Y}_n$ is an unbiased estimator of* $\text{Var}[\bar{Y}_n]$.

So far we have assumed that X_j has the uniform distribution over $(0,1)^\ell$, which is of course unrealistic for practical applications. In the case where X_j has a more general distribution over $\mathbb{R}^\ell$, the array-RQMC algorithm operates in exactly the same

way. The only changes are in how to define the mappings π_j of chains to points and in the interpretation of the discrepancies.

It is standard in QMC studies to use discrepancies for the uniform distribution over the unit hypercube $(0,1)^\ell$, with the understanding that more general distributions over $\mathbb{R}^\ell$ can be transformed to the uniform distribution, usually via a change of variables. To follow this path, we assume that X_j has a continuous distribution and that for each j, there is a bijection $\psi_j : \mathscr{X} \to (0,1)^\ell$ such that $\psi_j(X_j)$ has the uniform distribution over $(0,1)^\ell$. We then define the discrepancy of the states at step j as

$$D_j = D_j(S_{n,j}) = D_j(X_{0,j},\ldots,X_{n-1,j}) \stackrel{\text{def}}{=} D(\psi_j(X_{0,j}),\ldots,\psi_j(X_{n-1,j})),$$

where D is the same as earlier. In (5), $V(c_j)$ also needs to be replaced by $V(c_j \circ \psi_j^{-1})$.

We emphasize that there is no need to know ψ_j to run the algorithm. For a one-dimensional state space, the most natural definition is obviously the standard *probability integral transformation*, $\psi_j(x) = F_j(x)$, where F_j is the cumulative distribution function (CDF) of X_j. With this definition, the permutation π_j will simply sort the states by increasing order, at each step. In more than one dimension, this can be generalized as follows [19]: Given $X_j = (X_j^{(1)},\ldots,X_j^{(\ell)})$, let $U_j^{(1)} = F_{j,1}(X_j^{(1)})$ where $F_{j,1}$ is the CDF of $X_j^{(1)}$, then let $U_j^{(2)} = F_{j,2}(X_j^{(2)} \mid X_j^{(1)})$ where $F_{j,2}(\cdot \mid X_j^{(1)})$ is the CDF of $X_j^{(2)}$ conditional on $X_j^{(1)}$, and so on. Then put $\psi_j(X_j) = U_j = (U_j^{(1)},\ldots,U_j^{(\ell)})$. When the distribution of X_j is not continuous, this does not define a bijection, but one could still define ψ_j by taking $U_j^{(1)}$ as some solution of $U_j^{(1)} = F_{j,1}(X_j^{(1)})$, and so on.

It would be nice if we could show, under appropriate smoothness assumptions on the φ_j and ψ_j, and with proper choices of discrepancies D and $D_{(2)}$, that

$$\mathbb{E}[D_j^2] \le \kappa_j n^{-\alpha+\varepsilon} \tag{6}$$

for any $\varepsilon > 0$, for some $\alpha > 1$, where κ_j does not depend on n and grows only very slowly (or not at all) with j. From this, assuming that the $V(c_j \circ \psi_j^{-1}) < \infty$, it would follow that $\mathrm{Var}[(Y_0 + \cdots + Y_{n-1})/n]$ converges as $O(n^{-\alpha+\varepsilon})$. A natural path to establish such a result would be to show that low mean-square discrepancy $\mathbb{E}[D_j^2]$ is preserved from one step j to the next.

At this time, we do not have a proof. We only have empirical evidence. In our numerical experiments reported in Section 5, we observed a convergence rate of $O(n^{-2})$ for the variance. On the other hand, the convergence rate of the mean square discrepancy $\mathbb{E}[D_j^2]$ (which we estimated only for one-dimensional examples) depends on the choice of discrepancy D. For example, if D is defined as the $\mathscr{L}_2$-star discrepancy, the rates observed empirically are (approximately) $O(n^{-3/2})$, whereas with D equal to the discrepancy defined in Eq. (15) of [5], we observe $O(n^{-2})$.

4 Mapping the chains to the points

We now discuss how to define and implement the one-to-one mapping of the n points to the n chains in the array-RQMC algorithm, so that each state is assigned to a representative point that is close to it. As in the previous section, we start with the simplified case where the chain's state X_j has the uniform distribution over $(0,1)^\ell$.

We consider the following way of mapping the chains to the points, called a *multivariate sort* [2, 7]. Select some positive integers $n_1, \ldots, n_\nu$ such that $\nu \geq \ell$ and $n_1 \cdots n_\nu = n$. Sort the states (i.e., the chains) by their first coordinate, in n_1 packets of size n/n_1. This means that any state in a given packet will have its first coordinate smaller or equal to the first coordinate of any other state in the next packet. Then sort each packet by the second coordinate, in n_2 packets of size n/n_1n_2, and so on. When we reach coordinate ℓ, we sort each packet in n_ℓ packets of size $n/n_1 \cdots n_\ell$ by the last coordinate. If $\nu > \ell$, then at the next step we sort each packet into $n_{\ell+1}$ packets according to the first coordinate, and so on. As a special case of this, one can take $n_j = 2$ for all j, with n equal to a power of 2. This corresponds to splitting each packet of states in two with respect to the next coordinate, and doing this for each coordinate in a round-robin fashion.

If ℓ is deemed too large, we can map the state space to a lower-dimensional space as follows. Define a *sorting function* $v : \mathscr{X} \to [0,1)^c$, for $c < \ell$, and apply the multivariate sort to the transformed points $v(X_{i,j})$, in c dimensions. The function v should be selected so that two states mapped to nearby values in $[0,1)^c$ should be approximately equivalent in terms of the probability distribution of future costs when we are in these two states. In [14], it was assumed that such a mapping was always used, with $c = 1$, so v uniquely determined the sort, whence the appellation "sorting function."

Figure 1 illustrates the mappings obtained for two choices of n_1, namely $n_1 = n$ and $n_1 = n^{1/2}$, for an example with $\ell = 2$ and $n = 16$.

In the more general (and realistic) case where the state space is not $[0,1)^\ell$ but $\mathbb{R}^\ell$ (or a subset) and the ψ_j cannot be computed explicitly, a reasonable heuristic is to simply sort the states in the real space in exactly the same way as in the unit hypercube. This is what we will do in our examples.

Further discussion and suggestions for the mapping between the points and states can be found in [21]. On page 675, the authors assume that the points lie in a predefined two-dimensional grid, exactly one point per square of the grid (in our understanding), and use what they call a Z-curve to order the points. This would work fine to sort the points of a digital net in base 2, for example. However, sorting the states (or chains) with this scheme seems problematic, because there is generally not exactly one state per square of the grid. It would also need to be adapted in some way for the (usual) case where the state space is unbounded and ψ_j is unknown (X_j as an unknown distribution).

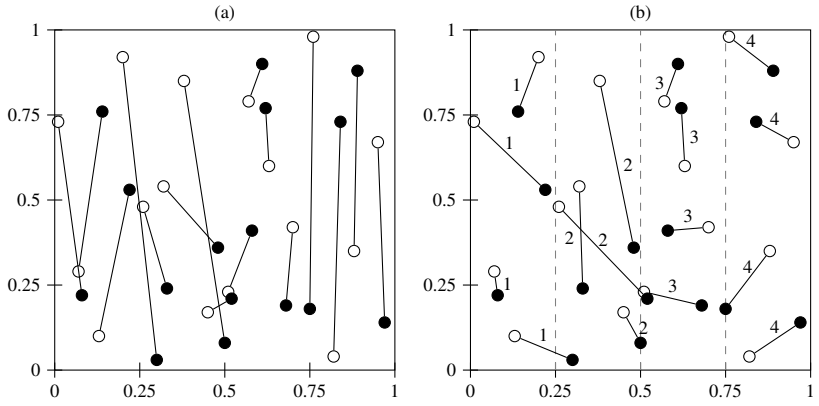

Fig. 1 Two mappings between points and states, in $\ell = 2$ dimensions, with $n = 16$. The black dots represent the states of the chains, and the white dots are the first 16 points of the two-dimensional Sobol' sequence, with a random digital shift. The lines indicate the mapping between the two sets of points. In the left picture, we have $n_1 = n$, so we sort according to the first coordinate only: the leftmost state is mapped to the leftmost point, the second leftmost state is mapped to the second leftmost point, and so on. The right picture is for $n_1 = n^{1/2} = 4$: we first sort both the points and the states in four packets according to the first (horizontal) coordinate. The numbers from 1 to 4 indicate the packet number in which each pair ended up in this first sort. Within each packet, the states are mapped to the points according to the second (vertical) coordinate. The dashed vertical lines at 1/4, 1/2, and 3/4 separate the Sobol' points in packets of four, but not the states. These dashed lines are only for visual intuition; they are not used by the sorting procedure.

5 Empirical investigations of the convergence rate

We now show how the variance and the mean square discrepancy $\mathbb{E}[D_j^2]$ (for different definitions of D) behave as functions of j and n, for small examples. All mean square discrepancies and variances were estimated from 100 independent replications of the array-RQMC estimator.

5.1 Example 1: An Autoregressive Process

Consider a Markov chain defined over the real line by

$$Y_0 = 0, \qquad Y_1 = Z_1, \qquad Y_j = \frac{\beta Y_{j-1} + Z_j}{\sqrt{\beta^2 + 1}} \text{ for } j \geq 2, \tag{7}$$

where $\beta \geq 0$ (a constant) and $Z_1, Z_2, \ldots$ are i.i.d. $N(0,1)$ (standard normal). This is a simple autoregressive process of order one. We have that $Y_j \sim N(0,1)$ and $X_j = \Phi(Y_j) \sim U(0,1)$, where Φ is the standard normal CDF. The transformed state X_j has the uniform distribution over $(0,1)$ at each step j, so here we are able to compute

explicitly the mean square discrepancy $\mathbb{E}[D_j^2]$ and to see how it evolves with j and n. This can be done for this small academic example, but cannot be done in general for more realistic examples. The Markov chain can also be defined directly in terms of a stochastic recurrence for X_j, namely $X_1 = U_1$ and

$$X_j = \varphi_j(X_{j-1}, U_j) = \Phi\left(\frac{\beta\,\Phi^{-1}(X_{j-1}) + \Phi^{-1}(U_j)}{\sqrt{\beta^2+1}}\right) \quad \text{for } j \geq 2,$$

where $U_1, U_2, \ldots$ are i.i.d. $U(0,1)$. Note that for $\beta = 0$ we have $Y_j = Z_j$ for all j, whereas for $\beta \to \infty$ (in the limit), we have $Y_j = Z_1$ for all j.

Our primary interest is in how the variance of $\hat{\mu}_{\text{rqmc},j,n}$ behaves as a function of j and n for various choices of the cost function c_j. In view of our discussion in Section 3, we are also interested in the behavior of $\mathbb{E}[D_j^2]$ as a function of j and of n, for various choices of the discrepancy D. These different discrepancies correspond to different assumptions on the smoothness of c_j and/or different choices of the RQMC point set $\tilde{Q}_n$.

To fix ideas, we consider two specific choices of D. The first one is the *$\mathscr{L}_2$-star discrepancy* [3]. In one dimension, its square value is the same as the Cramer-von Mises statistic:

$$D^2(u_0, \ldots, u_{n-1}) = D^2_{2,*}(u_0, \ldots, u_{n-1}) = \frac{1}{12n^2} + \frac{1}{n}\sum_{i=0}^{n-1}(w_i - u_i)^2,$$

where $w_i = (i+1/2)/n$ and $0 \leq u_0 \leq u_1 \leq \cdots \leq u_{n-1} \leq 1$.

We name our second example of D the *shift-baker2* discrepancy. In one dimension, its square is given by

$$D^2_{\text{shb}}(u_0, \ldots, u_{n-1}) = \frac{1}{n^2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\left[\frac{16}{45}\left[B_6(\{u_i - u_j - 1/2\}) - B_6(\{u_i - u_j\})\right] + \frac{1}{9}\left[10B_4(\{u_i - u_j - 1/2\}) - 19B_4(\{u_i - u_j\})\right]\right],$$

where the bold braces mean "mod 1", and B_4 and B_6 are the Bernoulli polynomials

$$B_4(u) = u^4 - 2u^3 + u^2 - \frac{1}{30} \quad \text{and} \quad B_6(u) = u^6 - 3u^5 + \frac{5}{2}u^4 - \frac{1}{2}u^2 + \frac{1}{42}.$$

This is the discrepancy given in Eq. (15) of [5], without the weights and with a correction on the coefficient of $B_4(\{u_i - u_j - 1/2\})$. This discrepancy represents the worst-case mean square error for a class of functions with square integrable second derivative, with the point set $\{u_0, \ldots, u_{n-1}\}$ randomized by a random shift modulo 1 followed by a baker's transformation [5]. Strictly speaking, this discrepancy would be appropriate only if we would apply the baker's transformation to the states X_j before computing the average cost $\hat{\mu}_{\text{rqmc},j,n}$ at each step, and we do not do that.

We nevertheless examine D_{shb} as an example to illustrate how the convergence rate might depend on the choice of discrepancy.

We also consider two choices for the two-dimensional RQMC point set used at each stage. In the first choice, we take the first n points of the two-dimensional Sobol' sequence, where the second coordinate of the points is randomized by a (random) left matrix scramble followed by a random digital shift [18]. For our second choice, we take a Korobov lattice rule with a random shift modulo 1 followed by a baker's transformation [5]. For the Korobov rule, for each n, we took the parameter a (in the usual notation) that gave the smallest shift-baker2 discrepancy in a random search over 1000 different values. The simulations were done using SSJ [10].

Our first results are for the $\mathscr{L}_2$-star $D_{2,*}$ and shift-baker2 D_{shb} discrepancies, for the Sobol' point sets. Figure 2 shows our estimate of $\mathbb{E}[D_j^2]$ (the sample average of D_j^2 over 100 independent replicates of the algorithms, as said earlier) as a function of j, with $n = 4096$ points, for $\beta = 0.1$, $\beta = 1$, and $\beta = 10$. The mean square discrepancy turns out to be quite stable even when we simulate this chain over a large number of steps. We observed the same behavior as a function of j for other discrepancies, other RQMC point sets, and also for the variance. This is very encouraging. For the shift-baker2 discrepancy and $\beta = 10$ (or any large β), $\mathbb{E}[D_j^2]$ increases in a visible way toward an horizontal asymptote as a function of j. Due to the nature of the recurrence (7), $\mathbb{E}[D_j^2]$ turns out to be an exponential smoothing of the discrepancies of previous steps, plus an additional term.

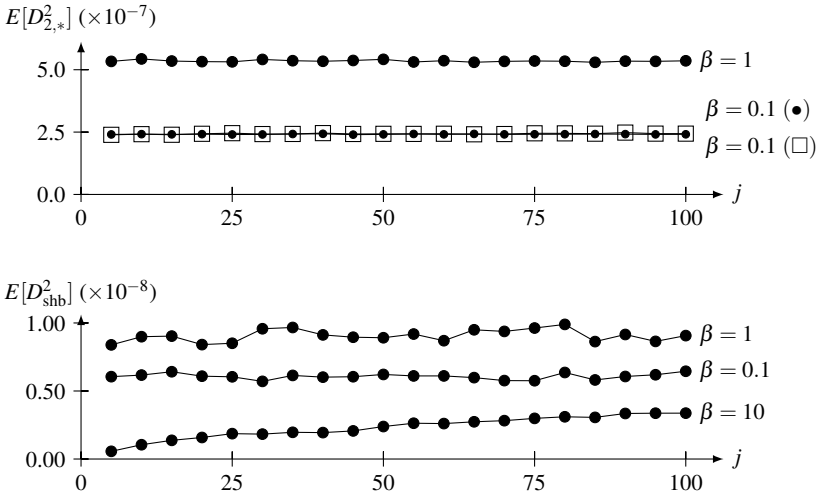

Fig. 2 Estimate of $\mathbb{E}[D_j^2]$ as a function of j for Example 5.1 with $n = 4096$, for $D = D_{2,*}$ (above) and $D = D_{\text{shb}}$ (below).

For $D_{2,*}$, the behavior for $\beta = 10$ is essentially the same as for $\beta = 0.1$. This also occurs more generally for any pair $(\beta, 1/\beta)$ where $\beta > 1$, and could be explained by the fact that it gives a linear combination of two independent standard normals

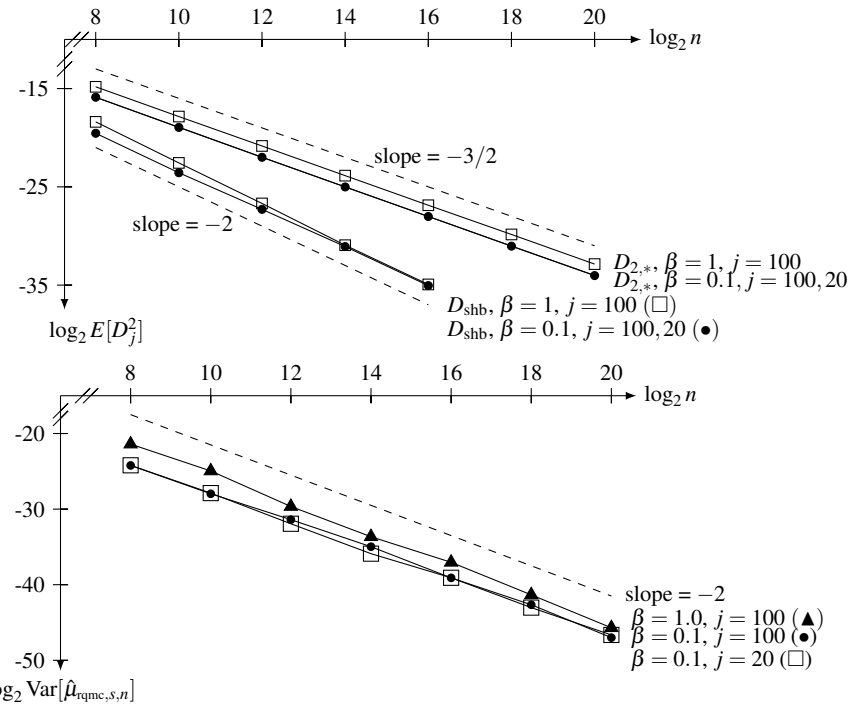

Fig. 3 Above: Estimate of $\log_2 \mathbb{E}[D_j^2]$ as a function of $\log_2 n$ for Example 5.1, for $D = D_{2,*}$ and $D = D_{\text{shb}}$. Below: Estimate of $\log_2 \text{Var}[\hat{\mu}_{\text{rqmc},j,n}]$ as a function of $\log_2 n$.

with the coefficients swapped in (7). For D_{shb}, however, the discrepancy is smaller for $\beta = 10$ than for $\beta = 0.1$. In what follows we will only report results for $\beta = 0.1$ and $\beta = 1$.

Figure 3 shows our estimate of $\log_2 \mathbb{E}[D_j^2]$ as a function of $\log_2 n$, for the same values of β, for $j = 20$ and $j = 100$, again for the Sobol' points, for $D_{2,*}$ and D_{shb}. The expectation was still estimated by the average over 100 independent replications. For $D_{2,*}$, $\mathbb{E}[D_j^2]$ seems to converge approximately as $O(n^{-3/2})$ as a function of n, and appears independent of j, as in the previous figure. With a Korobov lattice, the results are almost the same. They are also almost the same for other similar types of discrepancies such as unanchored $\mathscr{L}_2$-discrepancies defined in [4], for example. For the shift-baker2 discrepancy D_{shb}, the convergence rate differs and is (empirically) quite close to $O(n^{-2})$. The bottom part of Figure 3 shows our estimate of $\log_2 \text{Var}[\hat{\mu}_{\text{rqmc},j,n}]$ as a function of $\log_2 n$, for cost function $c_j(x) = x$, for $j = 20$ and $j = 100$. The slope indicates a convergence rate of approximately $O(n^{-2})$. This corresponds to the convergence rate of the mean square shift-baker2 discrepancy. We also tried other smooth cost functions such as $c_j(x) = x^2$, $\sqrt{x}$ and $\ln(x)$ and the observed rate was the same. The variance reduction factor compared to MC was also roughly the same for x, x^2, and $\sqrt{x}$, but it was approximately ten times smaller

for $\ln(x)$. Here, the n chains were simulated for j steps, the cost was then averaged over the n chains (at step j) to get one realization of the estimator $\hat{\mu}_{\text{rqmc},j,n}$. This was repeated $m = 100$ times, and the variance shown is the empirical variance of those m observations. The variance reduction factor, defined as the Monte Carlo variance divided by the array-RQMC variance when the two estimators are based on an average for n chains, is very roughly $600n$ when $\beta = 0.1$ and $j = 100$ (although there is significant fluctuation around this value when we change n and especially the RQMC point set that is used). The variance is also practically independent of j.

We also tried $c_j(x) = \mathbb{I}(x > 0.5)$, where $\mathbb{I}$ is the indicator function. The (empirical) convergence rate then dropped to approximately $O(n^{-3/2})$, and the variance reduction factor with respect to MC was about a hundred times smaller than for x for $n = 2^{10}$, and a thousand times smaller for $n = 2^{20}$ (that is, about half a million instead of 500 million). It is encouraging to see that even with such a discontinuous indicator function, the variance is much smaller than for MC and its convergence rate is faster (empirically).

5.2 Example 2: An Asian Option

In this example, let $0 < t_1 < t_2 < \cdots < t_s = T$ be fixed numbers (observation times), r and σ be positive constants, $S_0 = s_0$ (a constant), and

$$S_j = S_{j-1} \exp[(r - \sigma^2/2)(t_j - t_{j-1}) + \sigma(t_j - t_{j-1})^{1/2} \Phi^{-1}(U_j)] \tag{8}$$

where $U_j \sim U(0,1)$, for $j = 1, \ldots, s$. Define

$$\bar{S}_j = \frac{1}{j} \sum_{i=1}^{j} S_i.$$

We want to estimate

$$\mu = \mathbb{E}\left[\max\left(0, \bar{S}_s - K\right)\right].$$

This estimation problem occurs in pricing an Asian call option for a single asset whose price evolves as a geometric Brownian motion [6]. Note that μ is then multiplied by a constant discount factor, which we ignore here.

To put this model in our framework, we define a Markov chain with state $X_j = (S_j, \bar{S}_j)$ at step j, and whose transitions obey $(S_j, \bar{S}_j) = \varphi_j(S_{j-1}, \bar{S}_{j-1}, U_j)$ where φ_j is defined via (8) and $\bar{S}_j = [(j-1)\bar{S}_{j-1} + S_j]/j$. The function c_j is zero for $j < s$ and we have $c_s(S_s, \bar{S}_s) = \max\left(0, \bar{S}_s - K\right)$. The estimator is defined by (3) as usual. Here, $\tau = s$, we have a two-dimensional state space ($\ell = 2$), and we use a two-dimensional sort at each step: we first sort the states in n_1 packets of size n/n_1 based on $S(t_j)$, then we sort the packets based on $\bar{S}_j$.

In contrast with the previous example, we have no explicit mapping ψ_j available to transform the state into a uniform point over the unit square, so we cannot compute the discrepancy D_j explicitly. However, we can estimate the variance and

examine its convergence speed as a function of n. Our RQMC point set at each step is the first n points of a Sobol' sequence, this time in three dimensions, with coordinates 2 and 3 randomized by a left matrix scramble followed by a random digital shift.

For a numerical example, we take $S(0) = 100$, $K = 90$, $T = 240/365$, $t_1 = T - (s-1)/365$, $t_j - t_{j-1} = 1/365$, $r = \ln 1.09$, $\sigma = 0.2$, and $s = 10$ and 60.

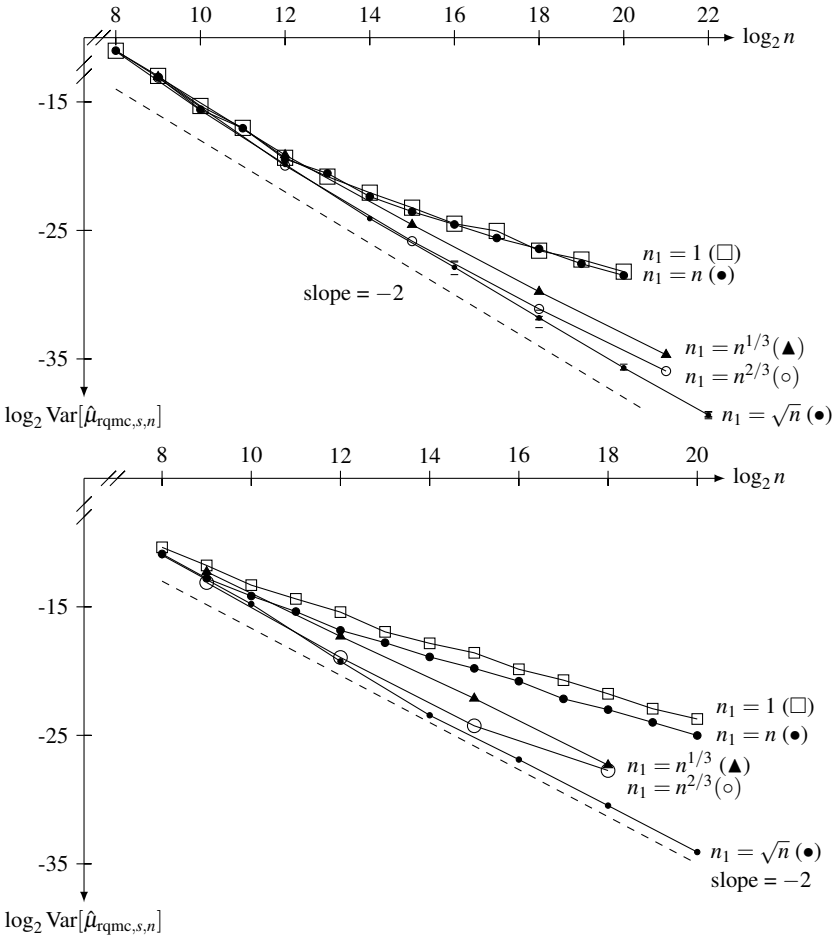

Fig. 4 Estimate of $\log_2 \mathrm{Var}[\hat{\mu}_{\mathrm{rqmc},n}]$ as a function of $\log_2 n$, for Example 5.2 with $K = 90$, for $s = 10$ (above) and $s = 60$ (below). For $s = 10$, $n \geq 2^{16}$, and $n_1 = \sqrt{n}$, we made four independent replicates of the experiment and their results are indicated by small horizontal bars on the graphs. The line goes through the log of the average variance over these four replicates.

Figures 4 show the variance as a function of n, again in a log-log scale, for different choices of n_1, for $s = 10$ and $s = 60$, respectively. The best results are with $n_1 \approx n^{1/2}$, for which the variance seems to converge approximately as $O(n^{-2})$.

For $n_1 \approx n^{1/3}$ and $n_1 \approx n^{2/3}$, the variance is larger (by a factor of about 10 for $s = 60$, $n \approx 2^{18}$, and $n_1 \approx n^{1/3}$, for example). The results are even worse if we take $n_1 = 1$ or $n_1 = n$, which corresponds to sorting the states by one of their two coordinates (this is the strategy that was used for this example in [14]). For $s = 60$ and $n \approx 2^{18}$, the variance with the best two-dimensional sort adopted here is about 400 times smaller than with a sort based on the second coordinate only. We emphasize that not only the convergence rate of the variance is better than for MC, but the variance is also much smaller for the range of values of n shown in the figure. For example, with $s = 10$, $K = 90$, and the best sorting strategy ($n_1 = \sqrt{n}$), the variance reduction factor is approximately $5n$. Thus, for $n = 2^{20}$, the variance with array-RQMC is about five million times smaller than with MC.

Of course, the variance behavior depends on the option and model parameters. For example, the probability p of a nonzero final payoff becomes very small when K is large, and the relative error (the standard deviation divided by the mean) increases without bound. This is a case of *rare event simulation*, for which RQMC is not the right tool. In that situation, we should first apply an appropriate technique such as *importance sampling* [1] to smooth out the estimator. Then we can apply RQMC for further improvement. On the other hand, the convergence *rate* of the variance for either MC or array-RQMC does not depend on K or p. With $K = 90$ and $s = 10$ as in Figure 4, we have $p \approx 0.87$. If we change to $K = 111$, for example, we get $p \approx 0.23$ and the variance reduction factor of array-RQMC over MC turns out to be about four times smaller than with $K = 90$, but we still have (approximately) an $O(n^{-2})$ convergence rate for the variance.

We also experimented with the discontinuous payoff function $c_s(S_s, \bar{S}_s) = \max(0, \bar{S}_s - K)$ for $\bar{S}_s \geq S_s$, and 0 otherwise. In this case, the convergence rate drops (empirically) to $O(n^{-1.3})$ with $s = 10$ and $K = 90$, and the variance reduction factor becomes much more modest (5 for $n = 2^{10}$ and 50 for $n = 2^{20}$). Nevertheless, these factors are non-negligible and this is encouraging.

6 Future Work and Conclusion

The array-RQMC algorithm is a promising methodology for reducing the variance in the simulation of Markov chains. We believe that plenty of interesting results on its convergence are waiting to be established, in particular for multidimensional state spaces, under various sets of assumptions on the transition and cost functions. Further empirical experimentation is also needed, with large examples, alternative sorting strategies, and various classes of applications.

Acknowledgements This research has been supported by NSERC-Canada grant No. ODGP0110050 and a Canada Research Chair to the first author, and an NSERC scholarship to the third author. We thank the two anonymous reviewers and the Editor Art B. Owen for their helpful suggestions.

References

1. Asmussen, S., Glynn, P.W.: Stochastic Simulation. Springer-Verlag, New York (2007)
2. El Haddad, R., Lécot, C., L'Ecuyer, P.: Quasi-Monte Carlo simulation of discrete-time Markov chains on multidimensional state spaces. In: A. Keller, S. Heinrich, H. Niederreiter (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2006, pp. 413–429. Springer-Verlag, Berlin (2008)
3. Hickernell, F.J.: A generalized discrepancy and quadrature error bound. Mathematics of Computation **67**, 299–322 (1998)
4. Hickernell, F.J.: What affects the accuracy of quasi-Monte Carlo quadrature? In: H. Niederreiter, J. Spanier (eds.) Monte Carlo and Quasi-Monte Carlo Methods 1998, pp. 16–55. Springer-Verlag, Berlin (2000)
5. Hickernell, F.J.: Obtaining $O(N^{-2+\varepsilon})$ convergence for lattice quadrature rules. In: K.T. Fang, F.J. Hickernell, H. Niederreiter (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2000, pp. 274–289. Springer-Verlag, Berlin (2002)
6. Hull, J.C.: Options, Futures, and Other Derivatives, sixth edn. Prentice-Hall, Upper Saddle River, N.J. (2006)
7. Lécot, C., Coulibaly, I.: A quasi-Monte Carlo scheme using nets for a linear Boltzmann equation. SIAM Journal on Numerical Analysis **35**(1), 51–70 (1998)
8. Lécot, C., Ogawa, S.: Quasirandom walk methods. In: K.T. Fang, F.J. Hickernell, H. Niederreiter (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2000, pp. 63–85. Springer-Verlag, Berlin (2002)
9. Lécot, C., Tuffin, B.: Quasi-Monte Carlo methods for estimating transient measures of discrete time Markov chains. In: H. Niederreiter (ed.) Monte Carlo and Quasi-Monte Carlo Methods 2002, pp. 329–343. Springer-Verlag, Berlin (2004)
10. L'Ecuyer, P.: SSJ: A Java Library for Stochastic Simulation (2008). Software user's guide, available at `http://www.iro.umontreal.ca/~lecuyer`
11. L'Ecuyer, P.: Quasi-Monte Carlo methods with applications in finance. Finance and Stochastics (2009). To appear
12. L'Ecuyer, P., Demers, V., Tuffin, B.: Rare-events, splitting, and quasi-Monte Carlo. ACM Transactions on Modeling and Computer Simulation **17**(2), Article 9 (2007)
13. L'Ecuyer, P., Lécot, C., Tuffin, B.: Randomized quasi-Monte Carlo simulation of Markov chains with an ordered state space. In: H. Niederreiter, D. Talay (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2004, pp. 331–342. Springer-Verlag, Berlin (2006)
14. L'Ecuyer, P., Lécot, C., Tuffin, B.: A randomized quasi-Monte Carlo simulation method for Markov chains. Operations Research **56**(4), 958–975 (2008)
15. L'Ecuyer, P., Lemieux, C.: Variance reduction via lattice rules. Management Science **46**(9), 1214–1235 (2000)
16. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods, *SIAM CBMS-NSF Regional Conference Series in Applied Mathematics*, vol. 63. SIAM, Philadelphia, PA (1992)
17. Owen, A.B.: Latin supercube sampling for very high-dimensional simulations. ACM Transactions on Modeling and Computer Simulation **8**(1), 71–102 (1998)
18. Owen, A.B.: Variance with alternative scramblings of digital nets. ACM Transactions on Modeling and Computer Simulation **13**(4), 363–378 (2003)
19. Rosenblatt, M.: Remarks on a multivariate transformation. The Annals of Mathematical Statistics **23**(3), 470–472 (1952)

20. Sloan, I.H., Joe, S.: Lattice Methods for Multiple Integration. Clarendon Press, Oxford (1994)
21. Wächter, C., Keller, A.: Efficient simultaneous simulation of Markov chains. In: A. Keller, S. Heinrich, H. Niederreiter (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2006, pp. 669–684. Springer-Verlag, Berlin (2008)