Array-RQMC for option pricing under stochastic volatility models

Amal BEN ABDELLAH

Join work with : **Pierre L'Ecuyer and Florian Puchhammer**
Département d'informatique et de recherche opérationnelle,
Université de Montréal

Optimization Days
May 2019

Introduction

Quasi-Monte Carlo (QMC) and randomized QMC (RQMC) methods have been studied extensively for *estimating an integral*, say $\mathbb{E}[Y]$, in a moderate number of dimensions.

Array-RQMC has been proposed as a way to effectively apply RQMC when simulating a Markov chain over a large number of steps to estimate an expected cost or reward.

This method simulates n copies of the chain in parallel using a set of RQMC point independently at each step, and sorts the chains in a specific sorting function after each step.

Introduction

Array-RQMC has already been applied for pricing Asian options when the underlying process evolves as a geometric Brownian motion (GBM) with fixed volatility. In that case, the state is two-dimensional and a single random number is needed at each step, so the required RQMC points are three-dimensional.

In this talk, we show how to apply this method in case the underlying process has stochastic volatility. We show that Array-RQMC can also work very well for these models, even if it requires RQMC points in larger dimension.

We examine in particular the variance-gamma, Heston, and Ornstein-Uhlenbeck stochastic volatility model and we provide numerical results.

Array-RQMC for Markov Chain Setting

1. Setting: A Markov Chain with state space $\chi \subseteq \mathbb{R}^l$, evolves as

$$X_0 = x_0, \qquad X_j = \varphi_j(X_{j-1}, U_j), \quad j = 1, ..., \tau.$$

where the U_j are i.i.d uniform random variate's over $(0,1)^d$, the functions $\varphi_j : \mathcal{X} \times (0,1)^d \to \mathcal{X}$ are measurable and τ is fixed time horizon .
We want to estimate

$$\mu = \mathbb{E}[Y] \quad \text{where} \quad Y = g(X_\tau),$$

and $g : \mathcal{X} \to \mathbb{R}$ is a *cost* (or reward) function. Here we have a cost only at the last step τ.

Array-RQMC for Markov Chain Setting

1. Monte Carlo: For $i = 0, .., n-1$, generate $X_{i,j} = \varphi_j(X_{i,j-1}, U_{i,j})$,

$j = 1, .., \tau$, where the $U_{i,j}$'s are i.i.d. $U(0,1)^d$, Estimate μ by

$$\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^{n} g(X_{i,\tau}) = \frac{1}{n}\sum_{i=1}^{n} Y_i.$$

The simulation of each realization of Y requires a vector $\mathbf{V} = (\mathbf{U}_1, \ldots, \mathbf{U}_\tau)$ of $d\tau$ independent uniform random variables over $(0,1)$.

$\mathbb{E}[\hat{\mu}_n] = \mu$ and $\mathrm{Var}[\hat{\mu}_n] = \frac{1}{n}\mathrm{Var}[Y_i] = \mathcal{O}(n^{-1})$.

Array-RQMC for Markov Chain Setting

1. RQMC : One RQMC point set for each sample path.

Put $\mathbf{V}_i = (\mathbf{U}_{i,1}, ..., \mathbf{U}_{i,\tau}) \in (0,1)^s = (0,1)^{d\tau}$. Estimate μ par

$$\hat{\mu}_{rqmc,n} = \frac{1}{n} \sum_{i=1}^{n} g(X_{i,\tau})$$

Where $P_n = \{\mathbf{V}_0, ..., \mathbf{V}_{n-1}\} \subset (0,1)^s$ satisfies:

- each point $\mathbf{V}_i$ has the uniform distribution over $(0,1)^s$;
- P_n covers $(0,1)^s$ very evenly (i.e., has low discrepancy)

This dimension s is often very large! and RQMC generally becomes ineffective, because $\mathbb{E}[Y]$ is a large integral.

Array-RQMC

Simulate an "array" of n chains in "parallel".
At each step, use an RQMC point set P_n to advance all the chains by one step, with global negative dependence across the chains.

Goal: Want small discrepancy (or "distance") between empirical distribution of $S_{n,j} = \{X_{0,j}, ..., X_{n-1,j}\}$ and theoretical distribution of X_j. If we succeed, these unbiased estimators will have small variance :

$$\mu_j = \mathbb{E}[g_j(X_j)] \approx \hat{\mu}_{rqmc,j,n} = \frac{1}{n}\sum_{i=1}^{n} g_j(X_{i,j})$$

$$\mathrm{Var}[\hat{\mu}_{rqmc,j,n}] = \frac{\mathrm{Var}[g_j(X_{i,j})}{n} + \frac{2}{n^2}\sum_{i=0}^{n-1}\sum_{k=i+1}^{n-1}\mathrm{Cov}[g_j(X_{i,j}), g_j(X_{k,j})] \,.$$

RQMC insight

Suppose that $X_j \sim U(0,1)^l$. This can be achieved by a change of variable. We estimate

$$\mu_j = \mathbb{E}[g(X_j)] = \mathbb{E}[g(\varphi_j(X_{j-1}, U))] = \int_{[0,1)^{l+d}} g(\varphi_j(x,u))dxdu$$

$$\hat{\mu}_{rqmc,j,n} = \frac{1}{n}\sum_{i=0}^{n-1} g(X_{i,j}) = \frac{1}{n}\sum_{i=0}^{n-1} g(\varphi_j(X_{j-1}, U_{i,j})).$$

This is RQMC with the point set $Q_n = \{(X_{i,j-1}, U_{i,j}), 0 \le i < n\}$.
We want Q_n to have low discrepancy (LD) over $[0,1)^{l+d}$.

$X_{i,j-1}$'s isn't chosen from Q_n : they come from the simulation .
To construct the randomized $U_{i,j}$, select a LD point set

$$\tilde{Q}_n = \{(w_0, U_{0,j}), ..., (w_{n-1}, U_{n-1,j})\},$$

where the $w_i \in [0,1)^l$ are fixed and each $U_{i,j} \sim U(0,1)^d$.

RQMC insight

We suppose that there is a *sorting function* $h : \mathcal{X} \to \mathbb{R}$ that assigns to each state a *value* which summarizes in a single real number the most important information that we should retain from that state.

At each step j, the n chains are sorted by increasing order of their values of $h(X_{i,j-1})$.

Compute an appropriate permutation π_j of the n states, based on the $h(X_{i,j-1})$, to match them with the RQMC points and we permute the states $X_{i,j-1}$ so that $X_{\pi_j(i),j-1}$ is "close" to w_i for each i(LD between the two sets), and compute $X_{i,j} = \varphi_j(X_{\pi_j(i),j-1}, U_{i,j})$ for each i.

Array-RQMC algorithm

Algorithm 1 Array-RQMC Algorithm

$X_{i,0} \leftarrow x_0$ for $i = 0, \ldots, n-1$;
for $j = 1, 2, \ldots, \tau$ **do**
  Compute an appropriate permutation π_j of the n states, based on the $h(X_{i,j-1})$, to match them with the RQMC points;
  Randomized afresh $\{\mathbf{U}_{0,j}, \ldots, \mathbf{U}_{n-1,j}\}$ in $\tilde{Q}_n$;
  for $i = 0, 2, \ldots, n-1$ **do**
    $X_{i,j} = \varphi_j(\tilde{X}_{\pi j(i),j-1}, \mathbf{U}_{i,j})$;
  end for
end for
return the average $\hat{\mu}_{\text{arqmc},n} = \bar{Y}_n = (1/n)\sum_{i=0}^{n-1} g(X_{i,\tau})$ as an estimate of μ.

The average $\hat{\mu}_{\text{arqmc},n} = \bar{Y}_n$ is an *unbiased* estimator of μ.

The *empirical variance* of m independent realizations of $\hat{\mu}_{\text{arqmc},n}$ gives An unbiased estimator of $Var[\bar{Y}_n]$.

Mapping chains to points

If $l = 1$, can take $w_i = (i + 0.5)/n$ and just sort the states according to their first coordinate .

For $l > 1$, there are various ways to define the matching (multivariate sort):

1. Multivariate batch sort:
   We select positive integers $n_l, n_2, ..., n_l$ such that $n = n_l n_2 ... n_l$.
   Sort the states (chains) by first coordinate, in n_1 packets of size n/n_1.
   Sort each packet by second coordinate, in n_2 packets of size $n/n_1 n_2$.
   ...
   At the last level, sort each packet of size n_l by the last coordinate.

2. Multivariate split sort:
   $n_1 = n_2 = ... = 2$.
   Sort by first coordinate in 2 packets.
   Sort each packet by second coordinate in 2 packets.
   etc.

In these two sorts, the state space does not have to be $[0, 1)^l$.

Sorting by a Hilbert curve

Suppose that the state space is : $\mathcal{X} = [0,1)^l$.
Partition this cube into 2^{ml} subcubes of equal size.
While any subcube contains more than one point, partition it in 2^l.

The Hilbert curve defines a way to enumerate the subcubes so that successive subcubes are always adjacent. This gives a way to sort the points. Colliding points are ordered arbitrarily. We precompute and store the map from point coordinates (first m bits) to its position in the list.

Map the states to points as if the state has one dimension.
Use RQMC points in $1 + d$ dimensions, ordered by first coordinate.

What if state space is not $[0,1)^l$?

Define a transformation $\psi : \mathcal{X} \to [0,1)^l$ so that the transformed state is approximately uniformly distributed over $[0,1)^l$.

Gerber and Chopin [2015] propose to use the hilbert curve sort after mapping the state to the $[0,1)^l$ via a logistic transformation defined as follows : $\psi(x) = (\psi_1(x_1), ..., \psi_\ell(x_\ell)) \in [0,1]^\ell$ for all $x = (x_1, \ldots, x_\ell) \in \mathcal{X}$, where

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

with constants $\bar{x}_j = \mu_j + 2\sigma_j$ and $\underline{x}_j = \mu_j - 2\sigma_j$ in which μ_j and σ_j are estimates of the mean and the variance of the distribution of the jth coordinate of the state.

Experimental setting

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

European option payoff :

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

Asian option payoff :

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

where $\bar{S} = (1/\tau) \sum_{j=1}^{\tau} S(t_j)$.

Experimental setting

In our examples, we consider the following RQMC points sets :

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

We define the *variance reduction factor* (VRF20) observed for $n = 2^{20}$ for a given method compared with MC by $\sigma_y^2/(nVar[\bar{Y}_n])$. In each case, we fitted a linear regression model for the variance per run as a function of n, in log-log scale. We denote by $\hat{\beta}$ the regression slope estimated by this linear model.

Example 1: Asian Option Under Variance Gamma Process

We consider the pricing of an Asian option on a single asset price that evolves according to a variance-gamma (VG) process defined at time t_j as follows :

$$S(t_j) = S(0)\exp[(w+r)t_j + Y(t_j)],$$

where $Y(t_j) = X(G(t_j))$, X is a BM with drift and variance parameters θ and σ, G is a gamma process with mean and variance parameters 1 and ν .

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

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

State: $X_j = (S(t_j), \bar{S}_j)$

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

Numerical Experiments

We ran the simulation with the following parameters $\theta = 0.1436$, $\sigma = 0.12136$, $\nu = 0.3$, $r = 0.1$, $T = 240/365$, $\tau = 10$, $K = 100$, and $S(0) = 100$.
We tried a simple linear mapping $h_j : \mathbb{R}^2 \to \mathbb{R}$ defined by $h_j(S(t_j), \bar{S}_j) = b_j \bar{S}_j + (1 - b_j) S(t_j)$ where $b_j = (j-1)/(\tau - 1)$.

Sort	Point sets	$\hat{\beta}$	VRF20
Split sort	MC	-1	1
	Stratif	-1.17	42
	Sobol'+LMS	-1.77	91550
	Sobol'+NUS	-1.80	106965
	Lattice+baker	-1.83	32812
Batch sort ($n_1 = n_2$)	MC	-1	1
	Stratif	-1.17	42
	Sobol'+LMS	-1.71	100104
	Sobol'+NUS	-1.54	90168
	Lattice+baker	-1.95	58737
Hilbert sort (with logistic map)	MC	-1	1
	Stratif	-1.43	204
	Sobol'+LMS	-1.59	68297
	Sobol'+NUS	-1.67	79869
	Lattice+baker	-1.55	45854
Linear map sort	MC	-1	1
	Stratif	-1.35	192
	Sobol'+LMS	-1.64	115216
	Sobol'+NUS	-1.75	166541
	Lattice+baker	-1.72	68739

Example 2: Heston Volatility Model

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

$$dS(t) = rS(t)dt + V(t)^{1/2}S(t)dB_1(t),$$
$$dV(t) = \lambda(\sigma^2 - V(t))dt + \xi V(t)^{1/2}dB_2(t),$$

for $t \geq 0$, S(t) and V (t) are, respectively, the value and the instantaneous variance of an asset price, and (B_1, B_2) is a pair of standard Brownian motions with correlation ρ between them. to reduce the bias due to the discretization, we make the change of variable $W(t) = e^{\lambda t}(V(t) - \sigma^2)$, with $\mathrm{d}W(t) = e^{\lambda t}\xi V(t)^{1/2}\mathrm{d}B_2(t)$, and apply the Euler method to (S, W) instead of (S, V). The Euler approximation scheme with step size $\delta = T/\tau$ applied to W gives

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

Example 2: Heston Volatility Model

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

$$dS(t) = rS(t)dt + V(t)^{1/2}S(t)dB_1(t),$$
$$dV(t) = \lambda(\sigma^2 - V(t))dt + \xi V(t)^{1/2}dB_2(t),$$

for $t \geq 0$, S(t) and V (t) are, respectively, the value and the instantaneous variance of an asset price, and (B_1, B_2) is a pair of standard Brownian motions with correlation ρ between them. to reduce the bias due to the discretization, we make the change of variable $W(t) = e^{\lambda t}(V(t) - \sigma^2)$, with $\mathrm{d}W(t) = e^{\lambda t}\xi V(t)^{1/2}\mathrm{d}B_2(t)$, and apply the Euler method to (S, W) instead of (S, V). The Euler approximation scheme with step size $\delta = T/\tau$ applied to W gives

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

We obtain the following discrete-time stochastic recurrence, which we will simulate by Array-RQMC:

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

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

Example 2: Heston Volatility Model

The running average $\bar{S}_j$ at step j must be the average of the $S(t_k)$ at the observation times $t_k \le w_j = j\delta$. If we denote $N_j = \sum_{k=1}^{\tau} \mathbb{I}[t_k \le j\delta]$, we have $\bar{S}_j = (1/N_j)\sum_{k=1}^{N_j} S(t_k)$, which we approximate by $\overline{\widetilde{S}}_j = (1/N_j)\sum_{k=1}^{N_j} \widetilde{S}(t_k)$.

1. Asian Option

   State: $X_j = (\widetilde{S}(j\delta), \widetilde{V}(j\delta), \overline{\widetilde{S}}_j)$

   Transition:

$$X_j = (\widetilde{S}(j\delta), \widetilde{V}(j\delta), \overline{\widetilde{S}}_j) = \varphi_j(\widetilde{S}((j-1)\delta), \widetilde{V}((j-1)\delta), \overline{\widetilde{S}}_{(j-1)}, U_{j,1}, U_{j,2}).$$

2. European Option

   State: $X_j = (\widetilde{S}(j\delta), \widetilde{V}(j\delta))$

   Transition:

$$X_j = (\widetilde{S}(j\delta), \widetilde{V}(j\delta)) = \varphi_j(\widetilde{S}((j-1)\delta), \widetilde{V}((j-1)\delta), U_{j,1}, U_{j,2}).$$

Numerical Experiments

We ran experiments with $T = 1$ (one year), $K = 100$, $S(0) = 100$, $V(0) = 0.04$, $r = 0.05$, $\sigma = 0.2$, $\lambda = 5$, $\xi = 0.25$, $\rho = -0.5$, and $\tau = 256$ $(\delta = 1/256)$.

		European		Asian	
Sort	Point sets	$\hat{\beta}$	VRF20	$\hat{\beta}$	VRF20
Split sort	MC	-1	1	-1	1
	Stratif	-1.26	91	-1.36	48
	Sobol'+LMS	-1.61	60034	-1.72	5034
	Sobol'+NUS	-1.69	64908	-1.70	5755
	Lattice+baker	-1.70	36477	-1.73	3782
Batch sort	MC	-1	1	-1	1
	Stratif	-1.34	93	-1.31	39
	Sobol'+LMS	-1.74	34916	-1.23	472
	Sobol'+NUS	-1.82	50101	-1.36	633
	Lattice+baker	-1.78	14626	-1.23	550
Hilbert sort (with logistic map)	MC	-1	1	-1	1
	Stratif	-1.04	34	-1.13	40
	Sobol'+LMS	-1.20	339	-1.03	105
	Sobol'+NUS	-1.09	241	-1.08	102
	Lattice+baker	-1.01	229	-1.09	113

Example 3: Ornstein-Uhlenbeck Volatility Model

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

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

(B_1, B_2) is a pair of standard Brownian motions with correlation ρ between them, r is the risk-free rate, b is the long-term average volatility, α is the rate of return to the average volatility, and σ is a variance parameter for the volatility process. The discrete-time approximation of the stochastic recurrence is

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

where $(Z_{j,1}, Z_{j,2})$ is a pair of standard normals with correlation ρ.

Example 3: Ornstein-Uhlenbeck Volatility Model

The running average $\bar{S}_j$ at step j must be the average of the $S(t_k)$ at the observation times $t_k \leq w_j = j\delta$. If we denote $N_j = \sum_{k=1}^{\tau} \mathbb{I}[t_k \leq j\delta]$, we have $\bar{S}_j = (1/N_j) \sum_{k=1}^{N_j} S(t_k)$, which we approximate by $\overline{\widetilde{S}}_j = (1/N_j) \sum_{k=1}^{N_j} \widetilde{S}(t_k)$.

1. Asian Option

   State: $X_j = (\widetilde{S}(j\delta), \widetilde{V}(j\delta), \overline{\widetilde{S}}_j)$

   Transition:

$$X_j = (\widetilde{S}(j\delta), \widetilde{V}(j\delta), \overline{\widetilde{S}}_j) = \varphi_j(\widetilde{S}((j-1)\delta), \widetilde{V}((j-1)\delta), \overline{\widetilde{S}}_{(j-1)}, U_{j,1}, U_{j,2}).$$

2. European Option

   State: $X_j = (\widetilde{S}(j\delta), \widetilde{V}(j\delta))$

   Transition:

$$X_j = (\widetilde{S}(j\delta), \widetilde{V}(j\delta)) = \varphi_j(\widetilde{S}((j-1)\delta), \widetilde{V}((j-1)\delta), U_{j,1}, U_{j,2}).$$

Numerical Experiments

We ran a numerical experiment with $T = 1$, $K = 100$, $S(0) = 100$, $V(0) = 0.04$, $r = 0.05$, $b = 0.4$, $\alpha = 5$, $\sigma = 0.2$, $\rho = -0.5$, and $\tau = 256$ (so $\delta = 1/256$).

		European		Asian	
Sort	Point sets	$\hat{\beta}$	VRF20	$\hat{\beta}$	VRF20
Split sort	MC	-1	1	-1	1
	Stratif	-1.33	102	-1.23	46
	Sobol'+LMS	-1.39	60155	-1.50	65173
	Sobol'+NUS	-1.35	66507	-1.43	58063
	Lattice+baker	-1.07	47494	-1.42	42024
Batch sort	MC	-1	1	-1	1
	Stratif	-1.32	102	-1.23	46
	Sobol'+LMS	-1.28	49370	-1.20	7144
	Sobol'+NUS	-1.33	66155	-1.30	28665
	Lattice+baker	-1.32	51356	-1.21	6813
Hilbert sort (with logistic map)	MC	-1	1	-1	1
	Stratif	-1.31	404	-1.37	429
	Sobol'+LMS	-1.67	196131	-1.16	23896
	Sobol'+NUS	-1.69	259918	-1.30	28665
	Lattice+baker	-1.70	223170	-1.27	34416

Conclusion

- We have shown how Array-RQMC can be applied for pricing options under stochastic volatility models.
- The method Array-RQMC requires higher-dimensional RQMC points than with the simpler Geometric Brownian Motion model, and when time has to be discretized to apply Euler's method, the number of steps of the Markov chain is much larger.
- The empirical results shows that it's brings very significant variance reductions compared with crude Monte Carlo.

References

- M. Gerber and N. Chopin. Sequential quasi-Monte Carlo.
Journal of the Royal Statistical Society, Series B, 77(Part 3):509–579, 2015.
- P. L'Ecuyer, V. Demers, and B. Tuffin. Rare-events, splitting, and quasi-Monte Carlo.
ACM Transactions on Modeling and Computer Simulation, 17(2):Article 9, 2007.
- P. L'Ecuyer, C. Lécot, and A. L'Archevêque-Gaudet. On array-RQMC for Markov chains: Mapping alternatives and convergence rates.
Monte Carlo and Quasi-Monte Carlo Methods 2008, pages 485–500, Berlin, 2009. Springer-Verlag.
- P. L'Ecuyer, C. Lécot, and B. Tuffin. A randomized quasi-Monte Carlo simulation method for Markov chains.
Operations Research, 56(4):958–975, 2008.
- P. L'Ecuyer, D. Munger, C. Lécot, and B. Tuffin. Sorting methods and convergence rates for array-rqmc: Some empirical comparisons.
Mathematics and Computers in Simulation, 2017.
http://dx.doi.org/10.1016/j.matcom.2016.07.010.
- P. L'Ecuyer and C. Sanvido. Coupling from the past with randomized quasi-Monte Carlo.
Mathematics and Computers in Simulation, 81(3):476–489, 2010.
- C. Wächter and A. Keller. Efficient simultaneous simulation of Markov chains.
Monte Carlo and Quasi-Monte Carlo Methods 2006, pages 669–684, Berlin, 2008. Springer-Verlag.