

A review of Array-RQMC

Sorting methods and convergence rates

Pierre L'Ecuyer

Christian Lécot, David Munger, Bruno Tuffin

DIRO, Université de Montréal, Canada

LAMA, Université de Savoie, France

Inria–Rennes, France

Monte Carlo for Markov Chains

Setting: A Markov chain with state space $\mathcal{X} \subseteq \mathbb{R}^\ell$, evolves as

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

where the \mathbf{U}_j are i.i.d. uniform r.v.'s over $(0, 1)^d$. Want to estimate

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

for some fixed time horizon τ .

Monte Carlo for Markov Chains

Setting: A Markov chain with state space $\mathcal{X} \subseteq \mathbb{R}^\ell$, evolves as

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

where the \mathbf{U}_j are i.i.d. uniform r.v.'s over $(0, 1)^d$. Want to estimate

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

for some fixed time horizon τ .

Ordinary MC: For $i = 0, \dots, n-1$, generate $X_{i,j} = \varphi_j(X_{i,j-1}, \mathbf{U}_{i,j})$, $j = 1, \dots, \tau$, where the $\mathbf{U}_{i,j}$'s are i.i.d. $U(0, 1)^d$. Estimate μ by

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

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

Example 1 (very simple, one-dimensional state)

Let $Y = \theta U + (1 - \theta)V$, where U, V indep. $U(0, 1)$ and $\theta \in [0, 1)$.
This Y has cdf G_θ .

Markov chain is defined by

$$\begin{aligned} X_0 &= U_0 \sim U(0, 1); \\ X_j &= \varphi_j(X_{j-1}, U_j) = G_\theta(\theta X_{j-1} + (1 - \theta)U_j), \quad j \geq 1 \end{aligned}$$

where $U_j \sim U(0, 1)$. Then, $X_j \sim U(0, 1)$.

We consider various functions g_j : $g_j(x) = x - 1/2$, $g_j(x) = x^2 - 1/3$,
 $g_j(x) = \sin(2\pi x)$, $g_j(x) = e^x - e + 1$,
 $g_j(x) = (x - 1/2)^+ - 1/8$, $g_j(x) = \mathbb{I}[x \leq 1/3] - 1/3$.
They all have $\mathbb{E}[g_j(X_j)] = 0$.

Also discrepancies of states $X_{0,j}, \dots, X_{n-1,j}$.

Example 2: Asian Call Option (two-dim state)

Given observation times t_1, t_2, \dots, t_τ suppose

$$S(t_j) = S(t_{j-1}) \exp[(r - \sigma^2/2)(t_j - t_{j-1}) + \sigma(t_j - t_{j-1})^{1/2} \Phi^{-1}(U_j)],$$

where $U_j \sim U[0, 1)$ and $S(t_0) = s_0$ is fixed.

Running average: $\bar{S}_j = \frac{1}{j} \sum_{i=1}^j S(t_i)$.

Payoff at step $j = \tau$ is $Y = g_\tau(X_\tau) = \max[0, \bar{S}_\tau - K]$.

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

Transition:

$$X_j = (S(t_j), \bar{S}_j) = \varphi_j(S(t_{j-1}), \bar{S}_{j-1}, U_j) = \left(S(t_j), \frac{(j-1)\bar{S}_{j-1} + S(t_j)}{j} \right).$$

Plenty of potential applications:

Finance

Queueing systems

Inventory, distribution, logistic systems

Reliability models

MCMC in Bayesian statistics

Etc.

Classical Randomized Quasi-Monte Carlo (RQMC) for Markov Chains

One RQMC point for each sample path.

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

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

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

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

The dimension s is often very large!

Array-RQMC for Markov Chains

L., Lécot, Tuffin, et al. [2004, 2006, 2008, etc.]

Earlier deterministic versions: Lécot et al.

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. Seek global **negative dependence** across the chains.

Goal: Want **small discrepancy** (or “distance”) between empirical distribution of $S_{n,j} = \{X_{0,j}, \dots, X_{n-1,j}\}$ and theoretical distribution of X_j .

If we succeed, these (unbiased) estimators will have small variance:

$$\mu_j = \mathbb{E}[g_j(X_j)] \approx \hat{\mu}_{\text{arqmc},j,n} = \frac{1}{n} \sum_{i=0}^{n-1} g_j(X_{i,j})$$

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

Some RQMC insight: To simplify the discussion, suppose $X_j \sim U(0, 1)^\ell$. This can be achieved (in principle) by a change of variable. We estimate

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

(we take a single j here) by

$$\hat{\mu}_{\text{arqmc},j,n} = \frac{1}{n} \sum_{i=0}^{n-1} g_j(X_{i,j}) = \frac{1}{n} \sum_{i=0}^{n-1} g_j(\varphi_j(X_{i,j-1}, \mathbf{U}_{i,j})).$$

This is (roughly) RQMC with the point set $Q_n = \{(X_{i,j-1}, \mathbf{U}_{i,j}), 0 \leq i < n\}$. We want Q_n to have low discrepancy (LD) (be highly uniform) over $[0, 1]^{\ell+d}$.

Some RQMC insight: To simplify the discussion, suppose $X_j \sim U(0, 1)^\ell$. This can be achieved (in principle) by a change of variable. We estimate

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

(we take a single j here) by

$$\hat{\mu}_{\text{arqmc},j,n} = \frac{1}{n} \sum_{i=0}^{n-1} g_j(X_{i,j}) = \frac{1}{n} \sum_{i=0}^{n-1} g_j(\varphi_j(X_{i,j-1}, \mathbf{U}_{i,j})).$$

This is (roughly) RQMC with the point set $Q_n = \{(X_{i,j-1}, \mathbf{U}_{i,j}), 0 \leq i < n\}$. We want Q_n to have low discrepancy (LD) (be highly uniform) over $[0, 1)^{\ell+d}$.

We do not choose the $X_{i,j-1}$'s in Q_n : they come from the simulation. To construct the (randomized) $\mathbf{U}_{i,j}$, select a LD point set

$$\tilde{Q}_n = \{(\mathbf{w}_0, \mathbf{U}_{0,j}), \dots, (\mathbf{w}_{n-1}, \mathbf{U}_{n-1,j})\},$$

where the $\mathbf{w}_i \in [0, 1)^\ell$ are fixed and each $\mathbf{U}_{i,j} \sim U(0, 1)^d$.

Permute the states $X_{i,j-1}$ so that $X_{\pi_j(i),j-1}$ is "close" to \mathbf{w}_i for each i (LD between the two sets), and compute $X_{i,j} = \varphi_j(X_{\pi_j(i),j-1}, \mathbf{U}_{i,j})$ for each i .

Example: If $\ell = 1$, can take $\mathbf{w}_i = (i + 0.5)/n$ and just sort the states. For $\ell > 1$, there are various ways to define the matching (multivariate sort).

Array-RQMC algorithm

$X_{i,0} \leftarrow x_0$ (or $X_{i,0} \leftarrow x_{i,0}$) for $i = 0, \dots, n-1$;

for $j = 1, 2, \dots, \tau$ **do**

 Compute the permutation π_j of the states (for matching);

 Randomize afresh $\{\mathbf{U}_{0,j}, \dots, \mathbf{U}_{n-1,j}\}$ in \tilde{Q}_n ;

$X_{i,j} = \varphi_j(X_{\pi_j(i),j-1}, \mathbf{U}_{i,j})$, for $i = 0, \dots, n-1$;

$\hat{\mu}_{\text{arqmc},j,n} = \bar{Y}_{n,j} = \frac{1}{n} \sum_{i=0}^{n-1} g(X_{i,j})$;

end for

Estimate μ by the average $\bar{Y}_n = \hat{\mu}_{\text{arqmc},n} = \sum_{j=1}^{\tau} \hat{\mu}_{\text{arqmc},j,n}$.

Array-RQMC algorithm

$X_{i,0} \leftarrow x_0$ (or $X_{i,0} \leftarrow x_{i,0}$) for $i = 0, \dots, n-1$;

for $j = 1, 2, \dots, \tau$ **do**

 Compute the permutation π_j of the states (for matching);

 Randomize afresh $\{\mathbf{U}_{0,j}, \dots, \mathbf{U}_{n-1,j}\}$ in \tilde{Q}_n ;

$X_{i,j} = \varphi_j(X_{\pi_j(i),j-1}, \mathbf{U}_{i,j})$, for $i = 0, \dots, n-1$;

$\hat{\mu}_{\text{arqmc},j,n} = \bar{Y}_{n,j} = \frac{1}{n} \sum_{i=0}^{n-1} g(X_{i,j})$;

end for

Estimate μ by the average $\bar{Y}_n = \hat{\mu}_{\text{arqmc},n} = \sum_{j=1}^{\tau} \hat{\mu}_{\text{arqmc},j,n}$.

- Proposition:** (i) The average \bar{Y}_n is an unbiased estimator of μ .
 (ii) The empirical variance of m independent realizations gives an unbiased estimator of $\text{Var}[\bar{Y}_n]$.

Key issues:

1. How can we preserve LD of $S_{n,j}$ as j increases?
2. Can we prove that $\text{Var}[\hat{\mu}_{\text{arqmc},j,n}] = \mathcal{O}(n^{-\alpha})$ for some $\alpha > 1$?
How? What α ?

Intuition: Write discrepancy measure of $S_{n,j}$ as the mean square integration error (or variance) when integrating some function $\psi : [0, 1)^{\ell+d} \rightarrow \mathbb{R}$ using Q_n .

Use RQMC theory to show it is small if Q_n has LD. Then use induction.

Some generalizations

L., Lécot, and Tuffin [2008]: τ can be a random stopping time w.r.t. the filtration $\mathcal{F}\{(j, X_j), j \geq 0\}$.

L., Demers, and Tuffin [2006, 2007]: Combination with splitting techniques (multilevel and without levels), combination with importance sampling and weight windows. Covers particle filters.

L. and Sanvido [2010]: Combination with coupling from the past for exact sampling.

Dion and L. [2010]: Combination with approximate dynamic programming and for optimal stopping problems.

Gerber and Chopin [2015]: Sequential QMC.

Convergence results and applications

L., Lécot, and Tuffin [2006, 2008]: Special cases: convergence at MC rate, one-dimensional, stratification, etc. Var in $\mathcal{O}(n^{-3/2})$.

Lécot and Tuffin [2004]: Deterministic, one-dimension, discrete state.

El Haddad, Lécot, L. [2008, 2010]: Deterministic, multidimensional.

Fakherredine, El Haddad, Lécot [2012, 2013, 2014]: LHS, stratification, Sudoku sampling, ...

Wächter and Keller [2008]: Applications in computer graphics.

Gerber and Chopin [2015]: Sequential QMC (particle filters), Owen nested scrambling an dHilbert sort.

Variance in $o(n^{-1})$.

Mapping chains to points when $\ell > 2$

1. Multivariate batch sort:

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 $n/n_1 n_2$.

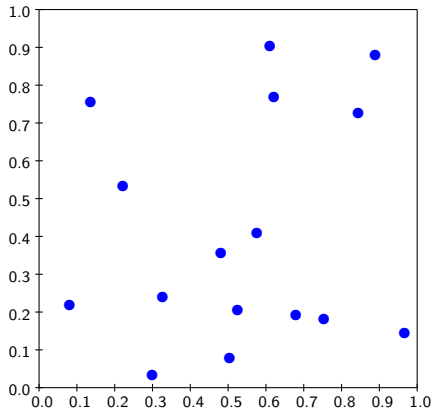
...

At the last level, sort each packet of size n_ℓ by the last coordinate.

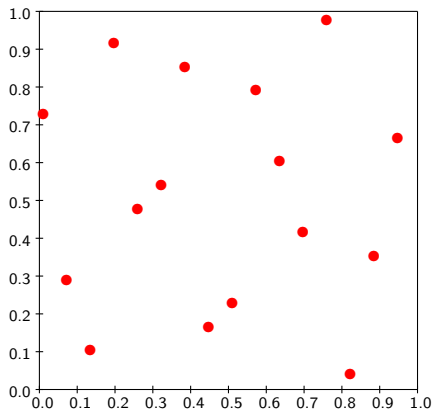
Choice of n_1, n_2, \dots, n_ℓ ?

A (4,4) mapping

States of the chains

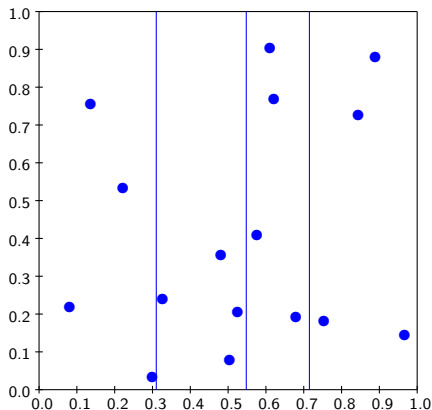


Sobol' net in 2 dimensions after random digital shift

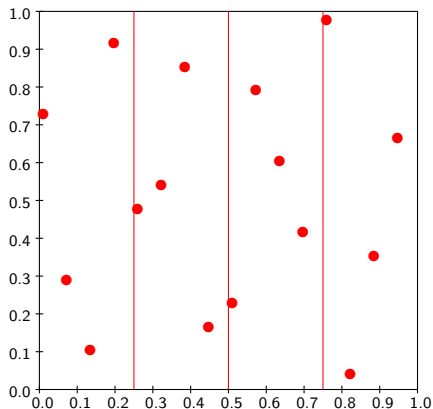


A (4,4) mapping

States of the chains

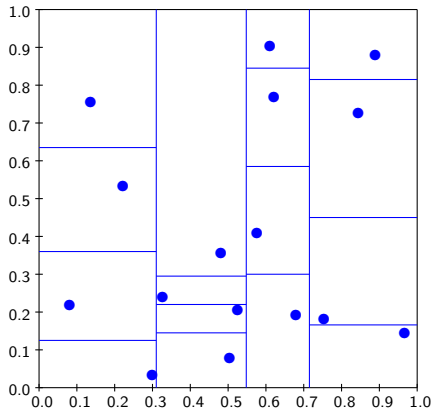


Sobol' net in 2 dimensions after random digital shift

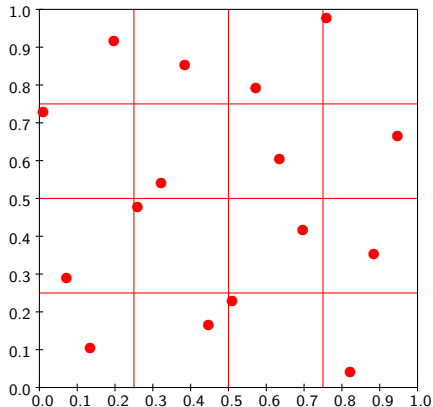


A (4,4) mapping

States of the chains

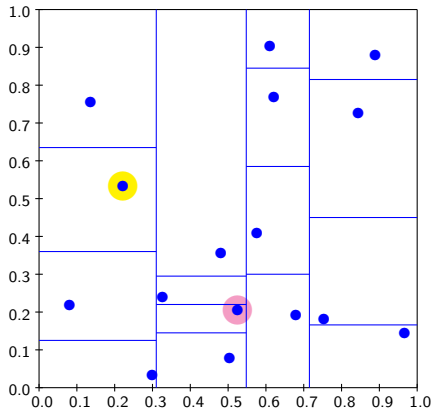


Sobol' net in 2 dimensions after random digital shift

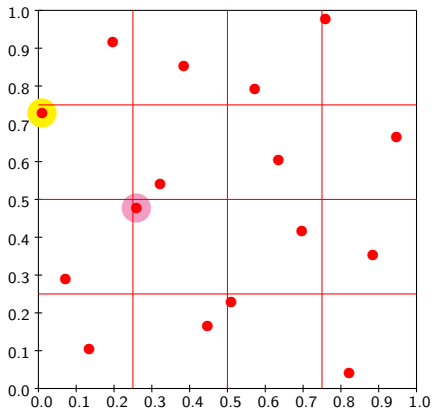


A (4,4) mapping

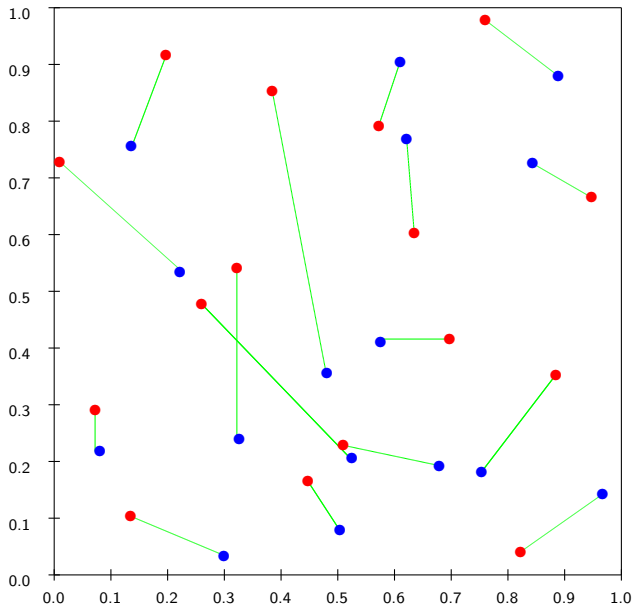
States of the chains



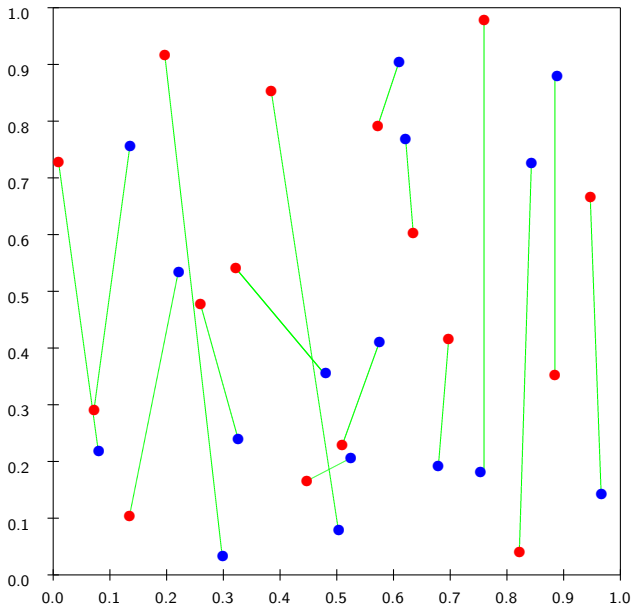
Sobol' net in 2 dimensions after random digital shift



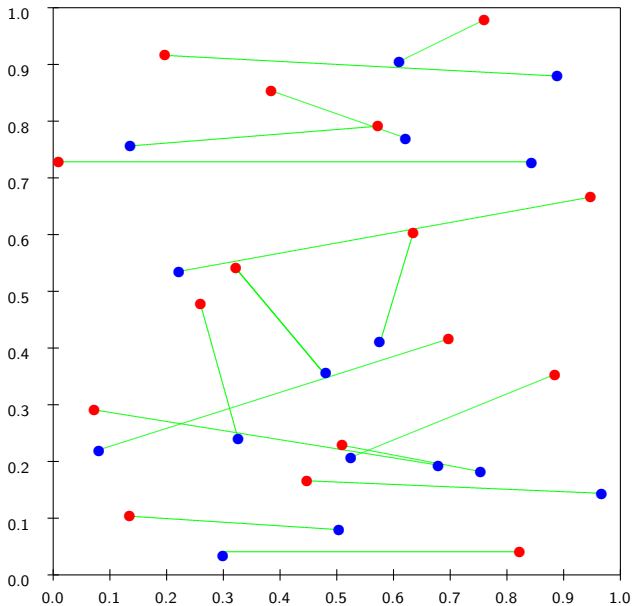
A (4,4) mapping



A (16,1) mapping, sorting along first coordinate



A (1,16) mapping, sorting along second coordinate



Mapping chains to points when $\ell > 2$

2. Multivariate split sort:

$$n_1 = n_2 = \dots = 2.$$

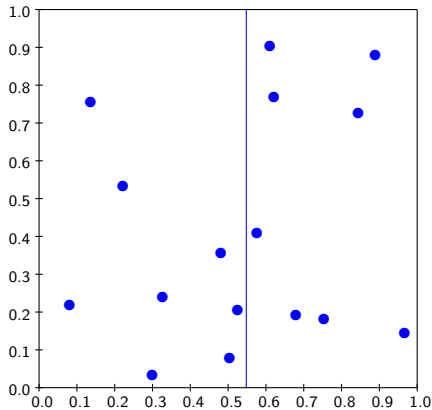
Sort by first coordinate in 2 packets.

Sort each packet by second coordinate in 2 packets.

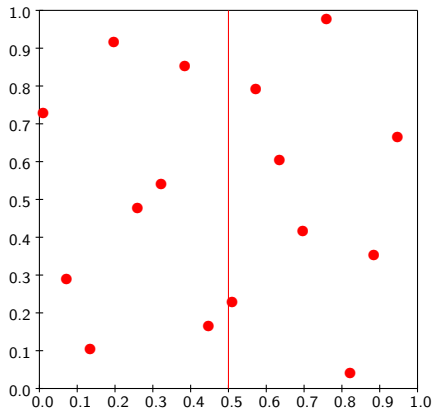
etc.

Mapping by split sort

States of the chains

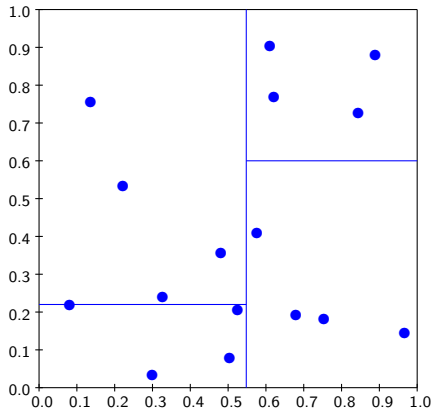


Sobol' net in 2 dimensions after random digital shift

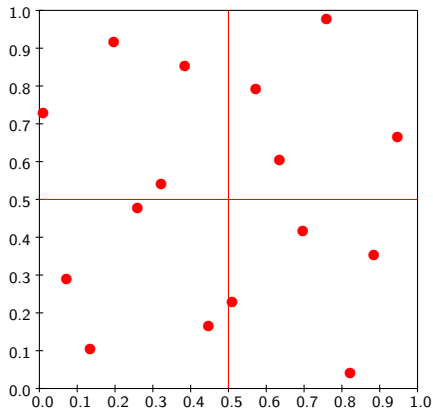


Mapping by split sort

States of the chains

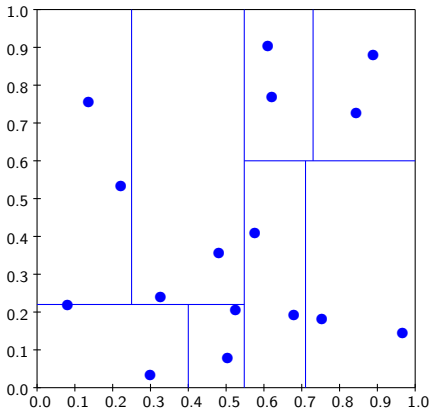


Sobol' net in 2 dimensions after random digital shift

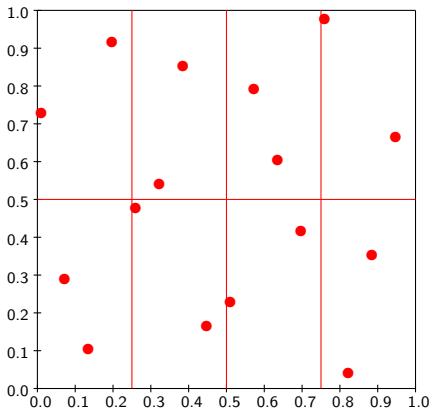


Mapping by split sort

States of the chains

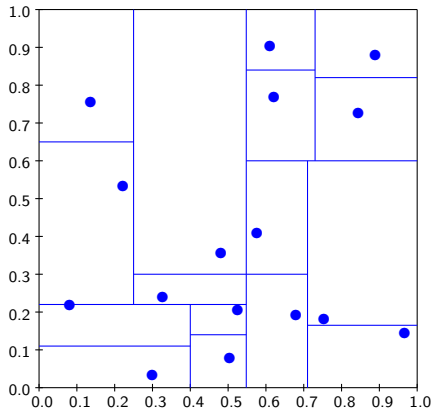


Sobol' net in 2 dimensions after random digital shift

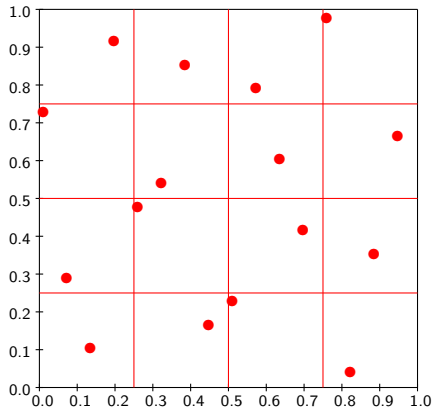


Mapping by split sort

States of the chains

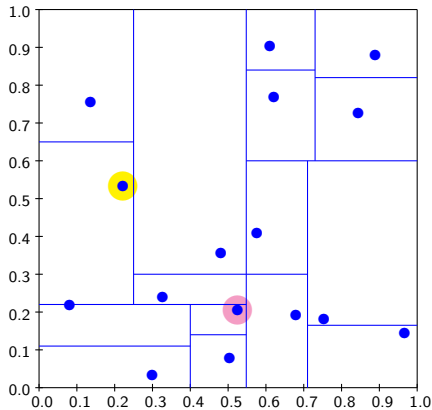


Sobol' net in 2 dimensions after random digital shift

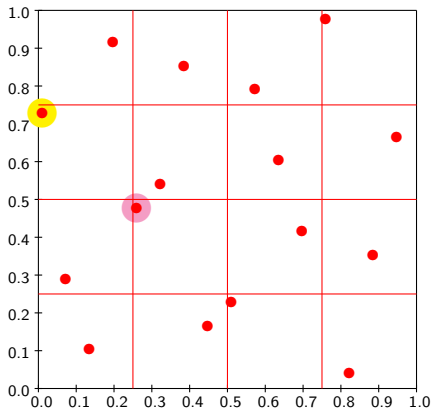


Mapping by split sort

States of the chains



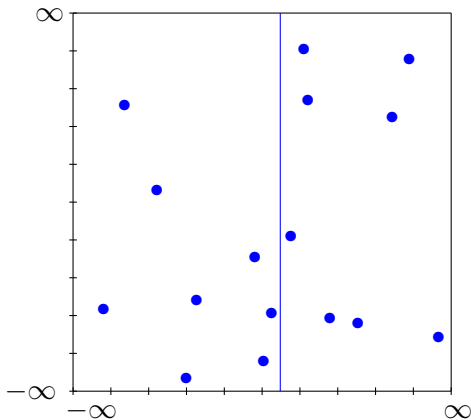
Sobol' net in 2 dimensions after random digital shift



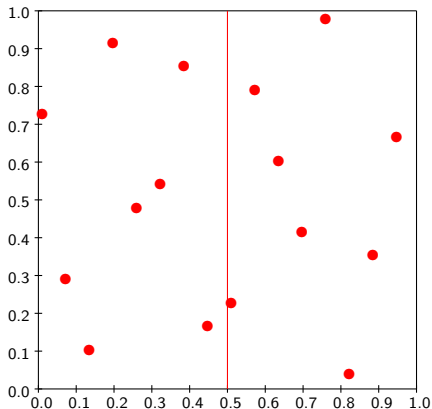
Mapping by batch sort and split sort

One advantage: The state space does not have to be $[0, 1)^d$:

States of the chains



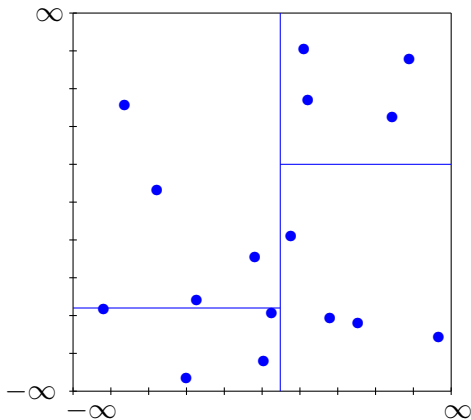
Sobol' net + digital shift



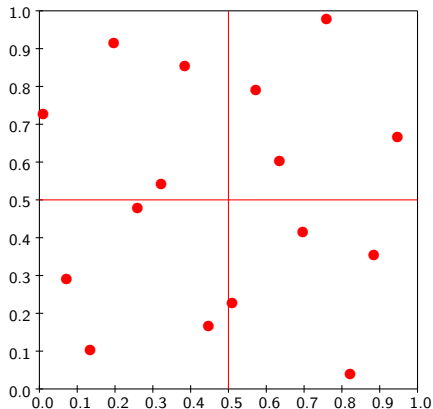
Mapping by batch sort and split sort

One advantage: The state space does not have to be $[0, 1)^d$:

States of the chains



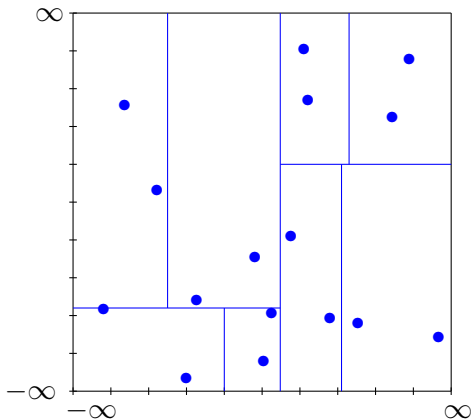
Sobol' net + digital shift



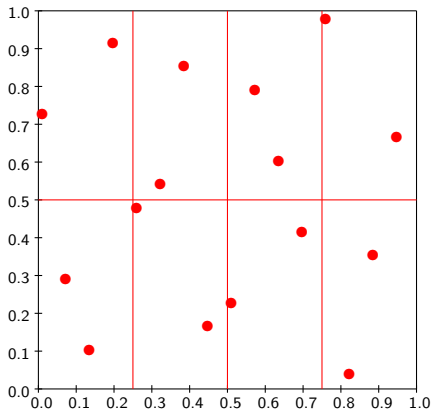
Mapping by batch sort and split sort

One advantage: The state space does not have to be $[0, 1)^d$:

States of the chains



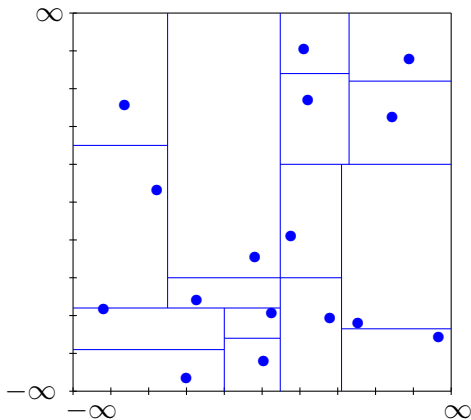
Sobol' net + digital shift



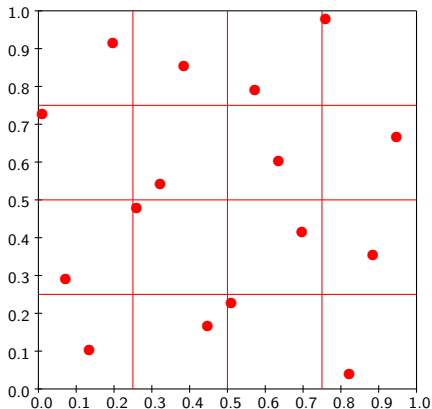
Mapping by batch sort and split sort

One advantage: The state space does not have to be $[0, 1)^d$:

States of the chains



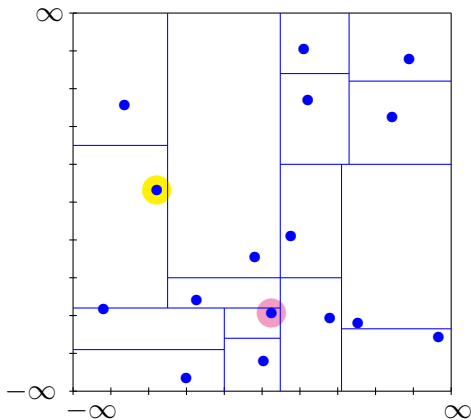
Sobol' net + digital shift



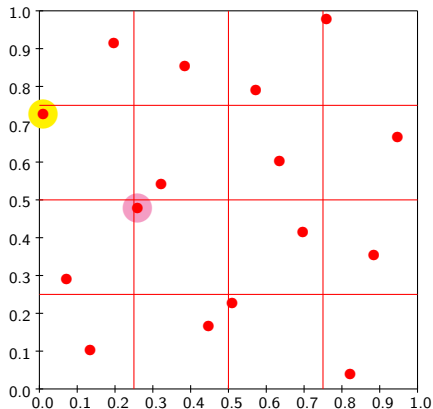
Mapping by batch sort and split sort

One advantage: The state space does not have to be $[0, 1)^d$:

States of the chains



Sobol' net + digital shift



Lowering the state dimension

For large ℓ : Define a transformation $h : \mathcal{X} \rightarrow [0, 1)^c$ for $c < \ell$.

Sort the transformed points $h(X_{i,j})$ in c dimensions.

Now we only need $c + d$ dimensions for the RQMC point sets;
 c for the mapping and d to advance the chain.

Choice of h : states mapped to nearby values should be nearly equivalent.

For $c = 1$, \mathcal{X} is mapped to $[0, 1)$, which leads to a one-dim sort.

The mapping h with $c = 1$ can be based on a space-filling curve:

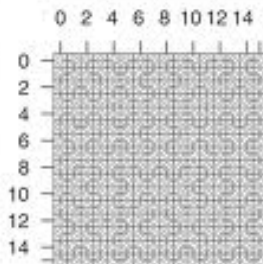
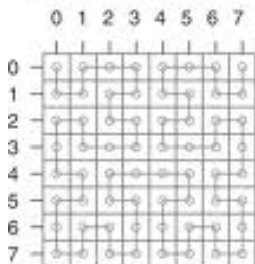
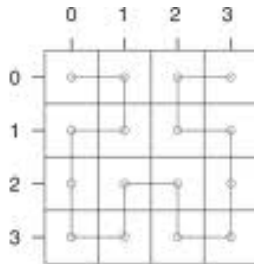
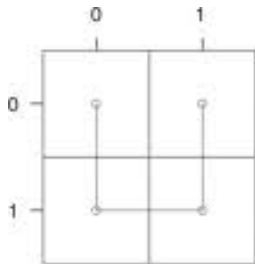
Wächter and Keller [2008] use a Lebesgue **Z-curve** and mention others;

Gerber and Chopin [2015] use a **Hilbert curve** and prove $o(n^{-1})$
convergence for the variance when used with digital nets and Owen nested
scrambling. A Peano curve would also work in base 3.

Reality check: **We only need a good pairing between states and RQMC points.** Any good way of doing this is welcome!

Hilbert curve

In ℓ dimensions, m levels: $2^{m\ell}$ subcubes and curve has length $2^{m(\ell-1)}$.



Sorting by a Hilbert curve

Suppose the state space is $\mathcal{X} = [0, 1)^\ell$.

Partition this cube into $2^{m\ell}$ subcubes of equal size.

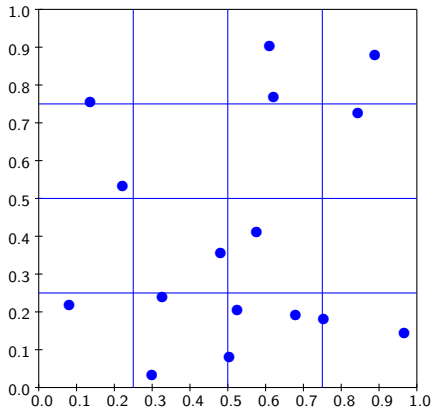
When a subcube contains more than one point (a collision), we could split it again in 2^ℓ . But in practice, we rather fix m and neglect collisions.

The Hilbert curve defines a way to enumerate (order) 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.

Then we can map states to points as if the state had one dimension. We use RQMC points in $1 + d$ dimensions, ordered by first coordinate, which is used to match the states, and d (randomized) coordinates are used to advance the chains.

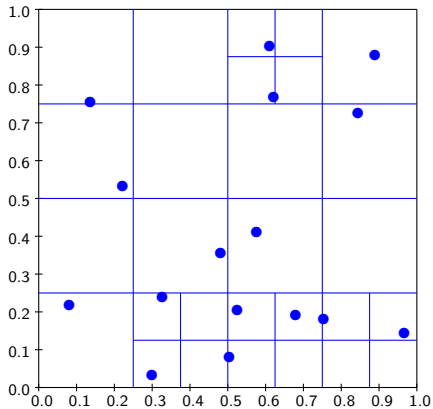
Hilbert curve sort

States of the chains



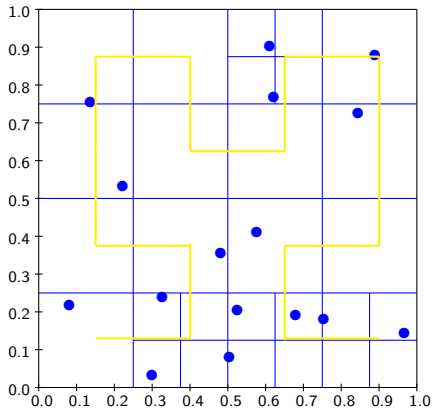
Hilbert curve sort

States of the chains



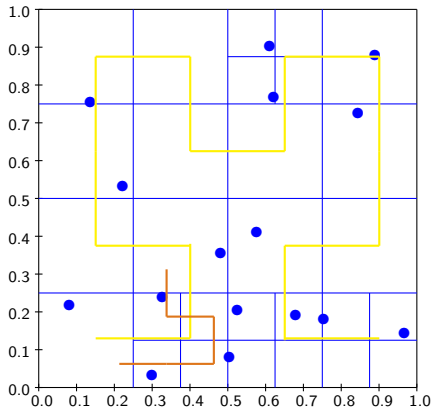
Hilbert curve sort

States of the chains



Hilbert curve sort

States of the chains



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

Ex.: For the Asian option, $\mathcal{X} = [0, \infty)^2$.

Then one must define a **transformation** $\psi : \mathcal{X} \rightarrow [0, 1)^\ell$ so that the transformed state is approximately uniformly distributed over $[0, 1)^\ell$.

Not easy to find a good ψ in general!

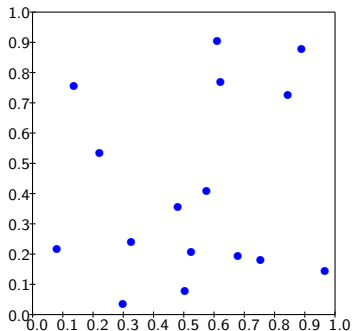
Gerber and Chopin [2015] propose using a logistic transformation for each coordinate, combined with trial and error.

A lousy choice could possibly damage efficiency.

Intuition for multivariate sort

For a path that connects the points in a given order, the variation along the path may have a bound that is proportional to its length.

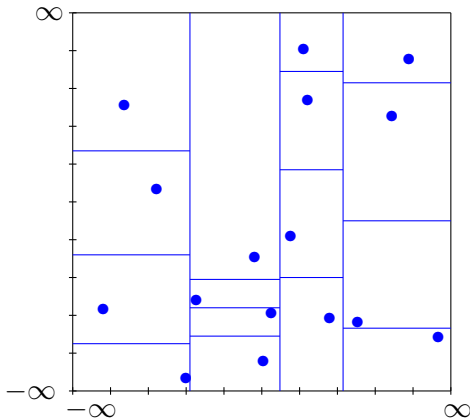
Shortest path that connect all the points? Traveling salesman problem!
Quickest heuristic for a good solution when n is very large: Hilbert or Peano curve sorts! Length of shortest path is $\mathcal{O}(\sqrt{n})$ on average. and heuristic gives $\mathcal{O}(\log n \sqrt{n})$.



Hilbert curve batch sort

Perform a multivariate batch sort, or a split sort, and then enumerate the boxes as in the Hilbert curve sort.

Advantage: the state space can be \mathbb{R}^{ℓ} .



Proved convergence results

L., Lécot, Tuffin [2008] + some extensions.

Simple case: suppose $\ell = d = 1$, $\mathcal{X} = [0, 1]$, and $X_j \sim U(0, 1)$. Define

$$\Delta_j = \sup_{x \in \mathcal{X}} |\hat{F}_j(x) - F_j(x)| \quad (\text{star discrepancy of states})$$

$$V_\infty(g_j) = \int_0^1 \left| \frac{dg_j(x)}{dx} \right| dx \quad (\text{corresponding variation of } g_j)$$

$$D_j^2 = \int_0^1 (\hat{F}_j(x) - F_j(x))^2 dx = \frac{1}{12n^2} + \frac{1}{n} \sum_{i=0}^{n-1} ((i + 0.5/n) - F_j(X_{(i),j}))^2$$

$$V_2^2(g_j) = \int_0^1 \left| \frac{dg_j(x)}{dx} \right|^2 dx \quad (\text{corresp. square variation of } g_j).$$

We have

$$\begin{aligned} |\bar{Y}_{n,j} - \mathbb{E}[g_j(X_j)]| &\leq \Delta_j V_\infty(g_j), \\ \text{Var}[\bar{Y}_{n,j}] = \mathbb{E}[(\bar{Y}_{n,j} - \mathbb{E}[g_j(X_j)])^2] &\leq \mathbb{E}[D_j^2] V_2^2(g_j). \end{aligned}$$

Convergence results and proofs, $\ell = 1$

Assumption 1. $\varphi_j(x, u)$ non-decreasing in u . Also $n = k^2$ for some integer k and that each square of the $k \times k$ grid contains exactly one RQMC point.

Let $\Lambda_j = \sup_{0 \leq z \leq 1} V(F_j(z | \cdot))$.

Proposition. (Worst-case error.) Under Assumption 1,

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

Corollary. If $\Lambda_j \leq \rho < 1$ for all j , then

$$\Delta_j \leq \frac{1 + \rho}{1 - \rho} n^{-1/2}.$$

Convergence results and proofs, $\ell = 1$

Assumption 2. (Stratification) Assumption 1 holds, φ_j also non-decreasing in x , and randomized parts of the points are uniformly distributed in the cubes and pairwise independent (or negatively dependent) conditional on the cubes in which they lie.

Proposition. (Variance bound.) Under Assumption 2,

$$\mathbb{E}[D_j^2] \leq \left(\frac{1}{4} \sum_{\ell=1}^j (\Lambda_\ell + 1) \prod_{i=\ell+1}^j \Lambda_i^2 \right) n^{-3/2}$$

Corollary. If $\Lambda_j \leq \rho < 1$ for all j , then

$$\begin{aligned} \mathbb{E}[D_j^2] &\leq \frac{1 + \rho}{4(1 - \rho^2)} n^{-3/2} = \frac{1}{4(1 - \rho)} n^{-3/2}, \\ \text{Var}[\bar{Y}_{n,j}] &\leq \frac{1}{4(1 - \rho)} V_2^2(g_j) n^{-3/2}. \end{aligned}$$

These bounds are uniform in j .

Convergence results and proofs, $\ell > 1$

Worst-case error of $\mathcal{O}(n^{-1/(\ell+1)})$ has been proved in a deterministic setting for a discrete state space in $\mathcal{X} \subseteq \mathbb{Z}^\ell$, and for a continuous state space $\mathcal{X} \subseteq \mathbb{R}^\ell$ under strong conditions on φ_j , using a batch sort (El Haddad, Lécot, L'Ecuyer 2008, 2010).

Gerber and Chopin (2015) proved $o(n^{-1})$ for the variance, for Hilbert sort and digital net with nested scrambling.

The one-dimensional example

$$X_0 = U_0; \quad X_j = \varphi_j(X_{j-1}, U_j) = G_\theta(\theta X_{j-1} + (1 - \theta)U_j), \quad j \geq 1$$

For array-RQMC, we take $X_{i,0} = w_i = (i - 1/2)/n$.

We have

$$\mathbb{E}[D_j^2] \leq \frac{n^{-3/2}}{4(1 - \rho)} = \frac{1 - \theta}{4(1 - 2\theta)} n^{-3/2}.$$

We tried different RQMC methods, for $n = 2^9$ to $n = 2^{21}$.

We did $m = 200$ independent replications for each n .

We fitted a **linear regression** of $\log_2 \text{Var}[\bar{Y}_{n,j}]$ vs $\log_2 n$, for various g_j

We also looked at $\mathbb{E}[D_j^2]$ and $\mathbb{E}[P_\alpha]$ for $\alpha = 2, 4, 6$.

Some MC and RQMC point sets:

MC:	Crude Monte Carlo
LHS:	Latin hypercube sampling
SS:	Stratified sampling
SSA:	Stratified sampling with antithetic variates in each stratum
Sobol:	Sobol' points, left matrix scrambling + digital random shift
Sobol+baker:	Add baker transformation
Sobol+NUS:	Sobol' points with Owen's nested uniform scrambling
Korobov:	Korobov lattice in 2 dim. with a random shift modulo 1
Korobov+baker:	Add a baker transformation

slope vs $\log_2 n$	$\log_2 \mathbb{E}[D_j^2]$	$\log_2 \text{Var}[\bar{Y}_{n,j}]$			
		$X_j - \frac{1}{2}$	$X_j^2 - \frac{1}{3}$	$(X_j - \frac{1}{2})^+ - \frac{1}{8}$	$\mathbb{I}[X_j \leq \frac{1}{3}] - \frac{1}{3}$
MC	-1.01	-1.02	-1.01	-1.00	-1.02
LHS	-1.02	-0.99	-1.00	-1.00	-1.00
SS	-1.50	-1.98	-2.00	-2.00	-1.49
SSA	-1.50	-2.65	-2.56	-2.50	-1.50
Sobol	-1.51	-3.22	-3.14	-2.52	-1.49
Sobol+baker	-1.50	-3.41	-3.36	-2.54	-1.50
Sobol+NUS	-1.50	-2.95	-2.95	-2.54	-1.52
Korobov	-1.87	-2.00	-1.98	-1.98	-1.85
Korobov+baker	-1.92	-2.01	-2.02	-2.01	-1.90

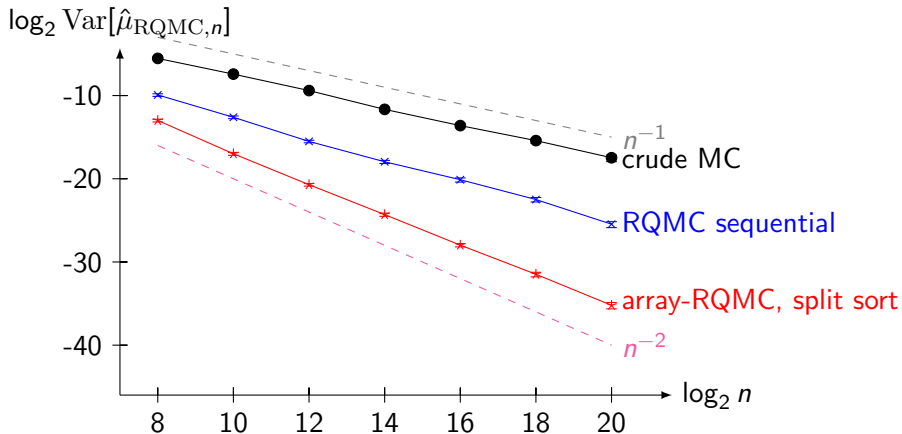
	$-\log_{10} \text{Var}[\bar{Y}_{n,j}]$ for $n = 2^{21}$			CPU time (sec)
	$X_j^2 - \frac{1}{3}$	$(X_j - \frac{1}{2})^+ - \frac{1}{8}$	$\mathbb{I}[X_j \leq \frac{1}{3}] - \frac{1}{3}$	
MC	7.35	7.86	6.98	270
LHS	8.82	8.93	7.61	992
SS	13.73	14.10	10.20	2334
SSA	18.12	17.41	10.38	1576
Sobol	19.86	17.51	10.36	443
Korobov	13.55	14.03	11.98	359

Example: Asian Call Option

$S(0) = 100$, $K = 100$, $r = 0.05$, $\sigma = 0.15$, $t_j = j/52$, $j = 0, \dots, \tau = 13$.

RQMC: Sobol' points with linear scrambling + random digital shift.

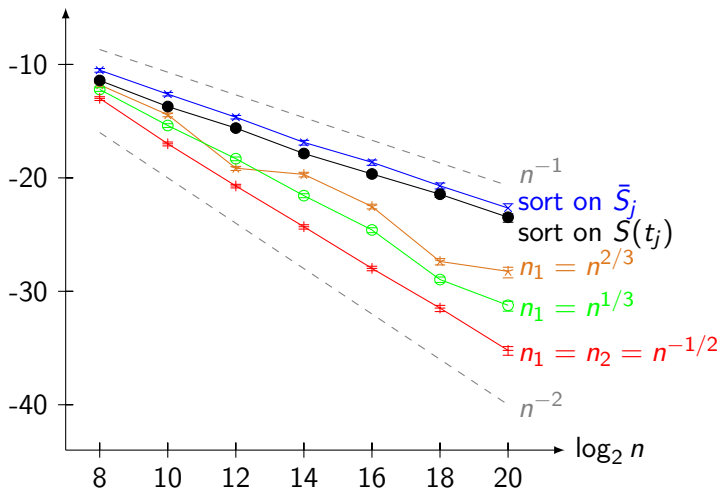
Similar results for randomly-shifted lattice + baker's transform.



Array-RQMC for Asian option, 2-dim. batch sort

Sort in n_1 packets based on $S(t_j)$, then sort the packets based on \bar{S}_j .

$$\log_2 \text{Var}[\hat{\mu}_{\text{arqmc},n}]$$



Example: Asian Call Option

$$S(0) = 100, K = 100, r = \ln(1.09), \sigma = 0.2, \\ t_j = (230 + j)/365, \text{ for } j = 1, \dots, \tau = 10.$$

Example: Asian Call Option

Sort	RQMC points	$\frac{\log_2 \text{Var}[\bar{Y}_{n,j}]}{\log_2 n}$	VRF	CPU (sec)
Split sort	SS	-1.38	2.0×10^2	3093
	Sobol	-2.04	4.0×10^6	1116
	Sobol+NUS	-2.03	2.6×10^6	1402
	Korobov+baker	-2.00	2.2×10^6	903
Batch sort ($n_1 = n_2$)	SS	-1.38	2.0×10^2	744
	Sobol	-2.03	4.2×10^6	532
	Sobol+NUS	-2.03	2.8×10^6	1035
	Korobov+baker	-2.04	4.4×10^6	482
Hilbert batch sort	SS	-1.54	2.3×10^3	835
	Sobol	-1.79	1.4×10^5	555
	Sobol+NUS	-1.80	1.2×10^5	711
	Korobov+baker	-1.92	3.4×10^6	528
Hilbert sort (logistic map)	SS	-1.55	2.4×10^3	840
	Sobol	-2.03	2.6×10^6	534
	Sobol+NUS	-2.02	2.8×10^6	724
	Korobov+baker	-2.01	3.3×10^6	567

VRF for $n = 2^{20}$. CPU time for $m = 100$ replications.

Conclusion

We have **convergence proofs** for special cases, but not yet for the rates we observe in examples.

Many other sorting strategies remain to be explored.

Other examples and applications. Higher dimension.

Array-RQMC is good not only to estimate the mean more accurately, but also to estimate the entire distribution of the state.