Lattice Rules for Quasi-Monte Carlo

Pierre L'Ecuyer

Université de Montréal, DIRO, GERAD Group for Research in Decision Analysis, CIRRELT, Inria

SAMSI Opening workshop on QMC, Duke University, August 2017

Monte Carlo integration

Want to estimate

$$\mu = \mu(f) = \int_{[0,1)^s} f(\mathbf{u})\, d\mathbf{u} = \mathbb{E}[f(\mathbf{U})]$$

where $f : [0,1)^s \to \mathbb{R}$ and $\mathbf{U}$ is a uniform r.v. over $[0,1)^s$.

Standard (or crude) Monte Carlo:

- Generate n independent copies of $\mathbf{U}$, say $\mathbf{U}_1, \ldots, \mathbf{U}_n$;
- estimate μ by $\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{U}_i)$.

Monte Carlo integration

Want to estimate

$$\mu = \mu(f) = \int_{[0,1)^s} f(\mathbf{u})\, d\mathbf{u} = \mathbb{E}[f(\mathbf{U})]$$

where $f : [0,1)^s \to \mathbb{R}$ and $\mathbf{U}$ is a uniform r.v. over $[0,1)^s$.

Standard (or crude) Monte Carlo:

- Generate n independent copies of $\mathbf{U}$, say $\mathbf{U}_1, \ldots, \mathbf{U}_n$;
- estimate μ by $\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{U}_i)$.

Almost sure convergence as $n \to \infty$ (strong law of large numbers).
For confidence interval of level $1-\alpha$, can use central limit theorem:

$$\mathbb{P}\left[\mu \in \left(\hat{\mu}_n - \frac{c_\alpha S_n}{\sqrt{n}},\ \ \hat{\mu}_n + \frac{c_\alpha S_n}{\sqrt{n}}\right)\right] \approx 1-\alpha,$$

where S_n^2 is any consistent estimator of $\sigma^2 = \mathrm{Var}[f(\mathbf{U})]$.

Quasi-Monte Carlo (QMC)

Replace the independent random points $\mathbf{U}_i$ by a set of deterministic points $P_n = \{\mathbf{u}_0, \ldots, \mathbf{u}_{n-1}\}$ that cover $[0,1)^s$ more evenly.

Estimate

$$\mu = \int_{[0,1)^s} f(\mathbf{u}) \mathrm{d}\mathbf{u} \quad \text{by} \quad Q_n = \frac{1}{n} \sum_{i=0}^{n-1} f(\mathbf{u}_i).$$

Integration error $E_n = Q_n - \mu$.

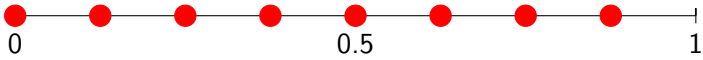
P_n is called a highly-uniform point set or low-discrepancy point set if some measure of discrepancy between the empirical distribution of P_n and the uniform distribution converges to 0 faster than $O(n^{-1/2})$ (the typical rate for independent random points).

Main construction methods: lattice rules and digital nets.

Simple case: one dimension ($s = 1$)

Obvious solutions:

$P_n = \mathbb{Z}_n/n = \{0, 1/n, \ldots, (n-1)/n\}$ (left Riemann sum):

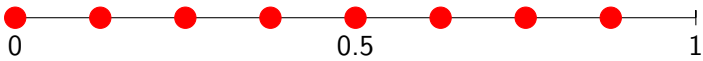


which gives $\quad Q_n = \dfrac{1}{n}\sum_{i=0}^{n-1} f(i/n)$, and $E_n = \mathcal{O}(n^{-1})$ if f' is bounded,

Simple case: one dimension ($s = 1$)

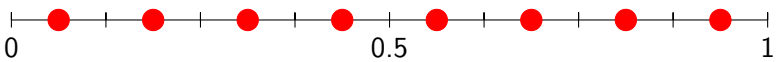
Obvious solutions:

$P_n = \mathbb{Z}_n/n = \{0, 1/n, \ldots, (n-1)/n\}$ (left Riemann sum):



which gives $\quad Q_n = \dfrac{1}{n}\sum_{i=0}^{n-1} f(i/n)$, and $E_n = \mathcal{O}(n^{-1})$ if f' is bounded,

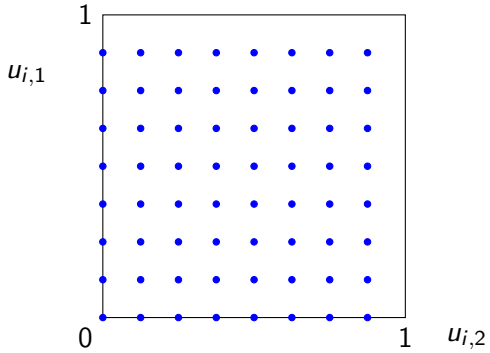
or $P'_n = \{1/(2n), 3/(2n), \ldots, (2n-1)/(2n)\}$ (midpoint rule):



for which $E_n = \mathcal{O}(n^{-2})$ if f'' is bounded.

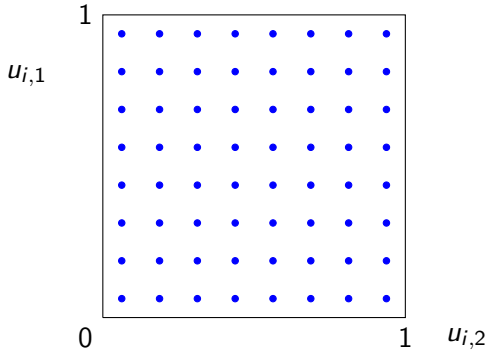
Simplistic solution for $s > 1$: rectangular grid

$P_n = \{(i_1/d, \ldots, i_s/d) \text{ such that } 0 \leq i_j < d \quad \forall j\}$ where $n = d^s$.



Simplistic solution for $s > 1$: rectangular grid

$P_n = \{(i_1/d, \ldots, i_s/d) \text{ such that } 0 \leq i_j < d \ \ \forall j\}$ where $n = d^s$.



Midpoint rule in s dimensions. Quickly becomes impractical when s increases.
Moreover, each one-dimensional projection has only d distinct points,
each two-dimensional projections has only d^2 distinct points, etc.

Lattice rules

[Korobov 1959; Sloan and Joe 1994; etc.] Integration lattice:

$$L_s = \left\{ \mathbf{v} = \sum_{j=1}^{s} z_j \mathbf{v}_j \text{ such that each } z_j \in \mathbb{Z} \right\},$$

where $\mathbf{v}_1, \ldots, \mathbf{v}_s \in \mathbb{R}^s$ are linearly independent over $\mathbb{R}$ and where L_s contains $\mathbb{Z}^s$.
Lattice rule: Take $P_n = \{\mathbf{u}_0, \ldots, \mathbf{u}_{n-1}\} = L_s \cap [0,1)^s$.

Each coordinate of each point must be a multiple of $1/n$.

Lattice rules

[Korobov 1959; Sloan and Joe 1994; etc.] Integration lattice:

$$L_s = \left\{ \mathbf{v} = \sum_{j=1}^{s} z_j \mathbf{v}_j \text{ such that each } z_j \in \mathbb{Z} \right\},$$

where $\mathbf{v}_1, \ldots, \mathbf{v}_s \in \mathbb{R}^s$ are linearly independent over $\mathbb{R}$ and where L_s contains $\mathbb{Z}^s$.
Lattice rule: Take $P_n = \{\mathbf{u}_0, \ldots, \mathbf{u}_{n-1}\} = L_s \cap [0,1)^s$.

Each coordinate of each point must be a multiple of $1/n$.

Lattice rule of rank 1: $\mathbf{u}_i = i\mathbf{v}_1 \bmod 1$ for $i = 0, \ldots, n-1$,
where $n\mathbf{v}_1 = \mathbf{a} = (a_1, \ldots, a_s) \in \{0, 1, \ldots, n-1\}^s$.

Korobov rule: $\mathbf{a} = (1, a, a^2 \bmod n, \ldots)$.

Lattice rules

[Korobov 1959; Sloan and Joe 1994; etc.] Integration lattice:

$$L_s = \left\{ \mathbf{v} = \sum_{j=1}^{s} z_j \mathbf{v}_j \text{ such that each } z_j \in \mathbb{Z} \right\},$$

where $\mathbf{v}_1, \ldots, \mathbf{v}_s \in \mathbb{R}^s$ are linearly independent over $\mathbb{R}$ and where L_s contains $\mathbb{Z}^s$.
Lattice rule: Take $P_n = \{\mathbf{u}_0, \ldots, \mathbf{u}_{n-1}\} = L_s \cap [0,1)^s$.

Each coordinate of each point must be a multiple of $1/n$.

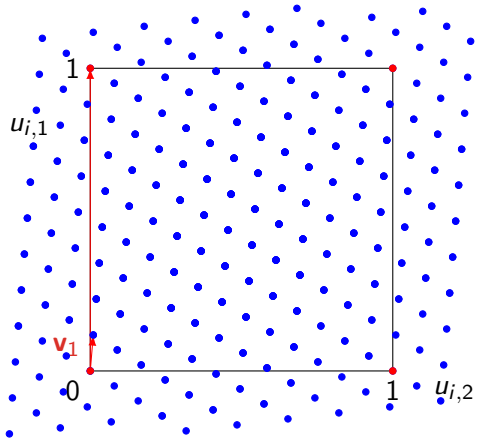
Lattice rule of rank 1: $\mathbf{u}_i = i\mathbf{v}_1 \bmod 1$ for $i = 0, \ldots, n-1$,
where $n\mathbf{v}_1 = \mathbf{a} = (a_1, \ldots, a_s) \in \{0, 1, \ldots, n-1\}^s$.

Korobov rule: $\mathbf{a} = (1, a, a^2 \bmod n, \ldots)$.

For any $\mathfrak{u} \subset \{1, \ldots, s\}$, the projection $L_s(\mathfrak{u})$ of L_s is also a lattice.

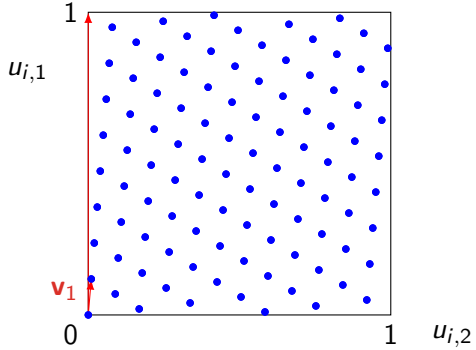
Fully projection-regular: Each $P_n(\mathfrak{u}) = L_s(\mathfrak{u}) \cap [0,1)^{|\mathfrak{u}|}$ contains n distinct points.
For a rule of rank 1: true iff $\gcd(n, a_j) = 1$ for all j.

Example: lattice with $s = 2$, $n = 101$, $\mathbf{v}_1 = (1, 12)/n$



Here, each one-dimensional projection is $\{0, 1/n, \ldots, (n-1)/n\}$.

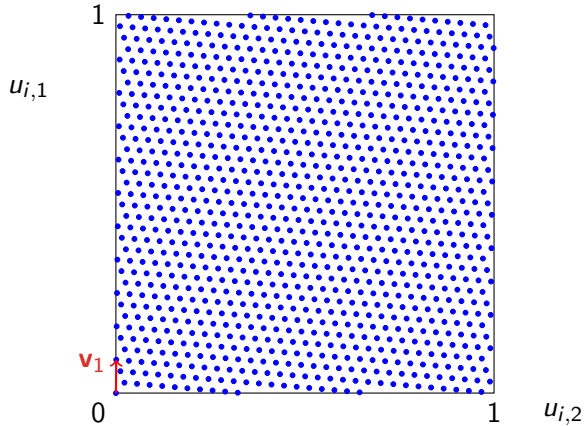
Example: lattice with $s = 2$, $n = 101$, $\mathbf{v}_1 = (1, 12)/n$



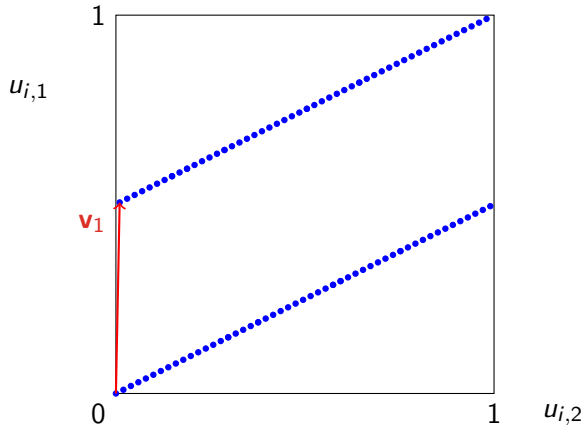
Here, each one-dimensional projection is $\{0, 1/n, \ldots, (n-1)/n\}$.

Another example: $s = 2$, $n = 1021$, $\mathbf{v}_1 = (1, 90)/n$

$$
\begin{aligned}
P_n &= \{\mathbf{u}_i = i\mathbf{v}_1 \bmod 1 : i = 0, \ldots, n-1\} \\
&= \{(i/1021, (90i/1021) \bmod 1) : i = 0, \ldots, 1020\}.
\end{aligned}
$$



A bad lattice: $s = 2$, $n = 101$, $\mathbf{v}_1 = (1, 51)/n$



Good uniformity for one-dimensional projections, but not in two dim.

Sequence of imbedded lattices

Sequence of lattices $L_s^1 \subset L_s^2 \subset L_s^3 \subset \ldots$.

$L_s^\xi \cap [0,1)^s$ contains n_ξ points: $n_{\xi-1}$ divides n_ξ for all ξ.

Sequence of imbedded lattices

Sequence of lattices $L_s^1 \subset L_s^2 \subset L_s^3 \subset \ldots$.

$L_s^\xi \cap [0,1)^s$ contains n_ξ points: $n_{\xi-1}$ divides n_ξ for all ξ.

Simple case: lattices of rank 1, $n_\xi = 2^\xi$, $\mathbf{a}_\xi$ mod $n_{\xi-1} = \mathbf{a}_{\xi-1}$.

Then, $a_{\xi,j} = a_{\xi-1,j}$ or $a_{\xi,j} = a_{\xi-1,j} + n_{\xi-1}$.

[Cranley and Patterson 1976, Joe 1990, Hickernell et al. 2001, Kuo et al. 2006]

Error in terms of Fourier expansion of f

$$f(\mathbf{u}) = \sum_{\mathbf{h} \in \mathbb{Z}^s} \hat{f}(\mathbf{h}) \exp(2\pi \mathrm{i}\, \mathbf{h} \cdot \mathbf{u}),$$

with Fourier coefficients

$$\hat{f}(\mathbf{h}) = \int_{[0,1)^s} f(\mathbf{u}) \exp(-2\pi \mathrm{i}\, \mathbf{h} \cdot \mathbf{u}) d\mathbf{u}.$$

If this series converges absolutely, then

$$E_n = Q_n - \mu = \sum_{\mathbf{0} \neq \mathbf{h} \in L_s^*} \hat{f}(\mathbf{h})$$

where $L_s^* = \{\mathbf{h} \in \mathbb{R}^s : \mathbf{h}^{\mathrm{t}}\mathbf{v} \in \mathbb{Z} \text{ for all } \mathbf{v} \in L_s\} \subseteq \mathbb{Z}^s$ is the dual lattice.

Let $\alpha > 0$ be an even integer. If f has square-integrable mixed partial derivatives up to order $\alpha/2 > 0$, and the periodic continuation of its derivatives up to order $\alpha/2 - 1$ is continuous across the unit cube boundaries, then

$$|\hat{f}(\mathbf{h})| = \mathcal{O}((\max(1, h_1) \cdots \max(1, h_s))^{-\alpha/2}).$$

Moreover, there are rank-1 integration lattices for which

$$\mathcal{P}_{\alpha/2} := \sum_{\mathbf{0} \neq \mathbf{h} \in L_s^*} (\max(1, h_1) \cdots \max(1, h_s))^{-\alpha/2} = \mathcal{O}(n^{-\alpha/2+\epsilon}),$$

and they are easy to find via CBC construction. This criterion was proposed long ago as a figure of merit, usually with $\alpha = 2$. This is the error for a worst-case f for which $|\hat{f}(\mathbf{h})| = (\max(1, |h_1|) \cdots \max(1, |h_s|))^{-\alpha/2}$.

Let $\alpha > 0$ be an even integer. If f has square-integrable mixed partial derivatives up to order $\alpha/2 > 0$, and the periodic continuation of its derivatives up to order $\alpha/2 - 1$ is continuous across the unit cube boundaries, then

$$|\hat{f}(\mathbf{h})| = \mathcal{O}((\max(1, h_1) \cdots \max(1, h_s))^{-\alpha/2}).$$

Moreover, there are rank-1 integration lattices for which

$$\mathcal{P}_{\alpha/2} := \sum_{\mathbf{0} \neq \mathbf{h} \in L_s^*} (\max(1, h_1) \cdots \max(1, h_s))^{-\alpha/2} = \mathcal{O}(n^{-\alpha/2+\epsilon}),$$

and they are easy to find via CBC construction. This criterion was proposed long ago as a figure of merit, usually with $\alpha = 2$. This is the error for a worst-case f for which $|\hat{f}(\mathbf{h})| = (\max(1, |h_1|) \cdots \max(1, |h_s|))^{-\alpha/2}$.

Unfortunately, this bound becomes rapidly useless (in many ways) when s increases. But it can be generalized in various directions: put different weights $w(\mathbf{h})$ on vectors $\mathbf{h}$, or on subsets of coordinates; truncate the series if α is not even; etc.
Notions of tractability... Also hard to estimate the error.

Randomized quasi-Monte Carlo (RQMC)

$$\hat{\mu}_{n,\mathrm{rqmc}} = \frac{1}{n}\sum_{i=0}^{n-1} f(\mathbf{U}_i),$$

with $P_n = \{\mathbf{U}_0, \ldots, \mathbf{U}_{n-1}\} \subset (0,1)^s$ an RQMC point set:

(i) each point $\mathbf{U}_i$ has the uniform distribution over $(0,1)^s$;

(ii) P_n as a whole is a low-discrepancy point set.

$$\mathbb{E}[\hat{\mu}_{n,\mathrm{rqmc}}] = \mu \qquad \text{(unbiased)}.$$

$$\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}] \;=\; \frac{\mathrm{Var}[f(\mathbf{U}_i)]}{n} + \frac{2}{n^2}\sum_{i<j}\mathrm{Cov}[f(\mathbf{U}_i), f(\mathbf{U}_j)].$$

We want to make the last sum as negative as possible.

Weaker attempts to do the same: antithetic variates ($n = 2$), Latin hypercube sampling (LHS), stratification, ...

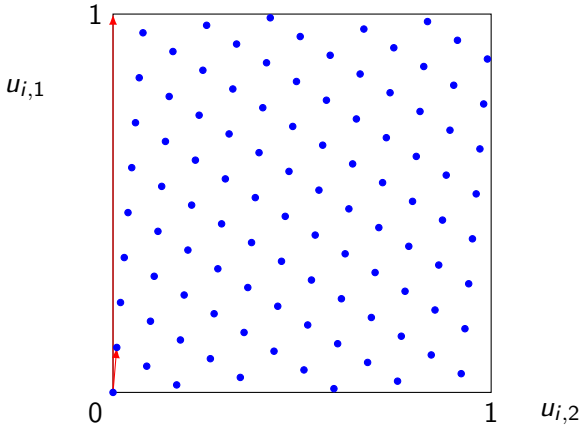
Variance estimation:

Can compute m independent realizations $X_1, \ldots, X_m$ of $\hat{\mu}_{n,\mathrm{rqmc}}$, then estimate μ and $\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}]$ by their sample mean $\bar{X}_m$ and sample variance S_m^2. Could be used to compute a confidence interval.

Temptation: assume that $\bar{X}_m$ has the normal distribution.
Beware: often wrong unless $m \to \infty$.
CLT for fixed m and $n \to \infty$ holds for nets with Owen nested scrambling, but not for randomly-shifted lattice rules.

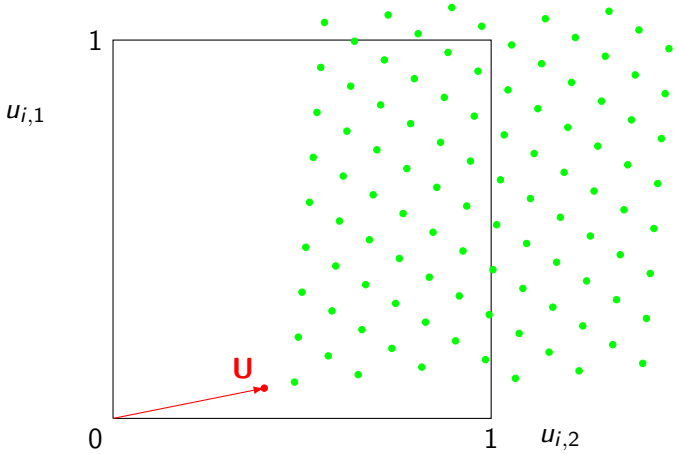
Randomly-Shifted Lattice

Example: lattice with $s = 2$, $n = 101$, $\mathbf{v}_1 = (1, 12)/101$



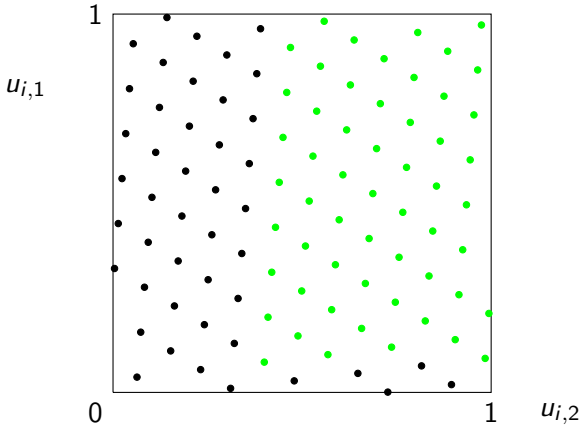
Randomly-Shifted Lattice

Example: lattice with $s = 2$, $n = 101$, $\mathbf{v}_1 = (1, 12)/101$



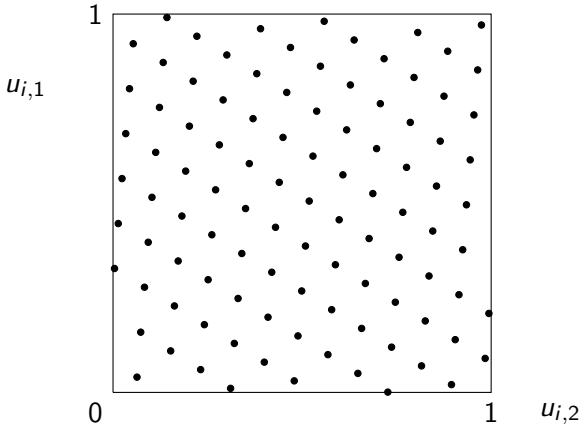
Randomly-Shifted Lattice

Example: lattice with $s = 2$, $n = 101$, $\mathbf{v}_1 = (1, 12)/101$

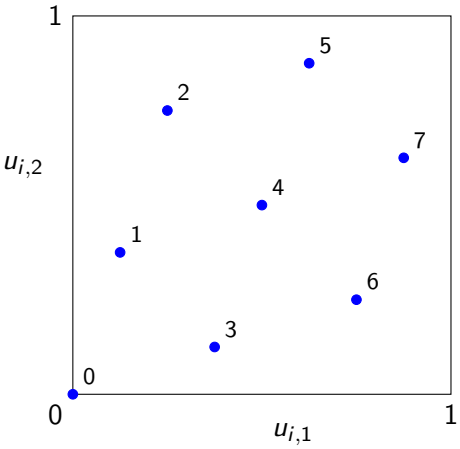


Randomly-Shifted Lattice

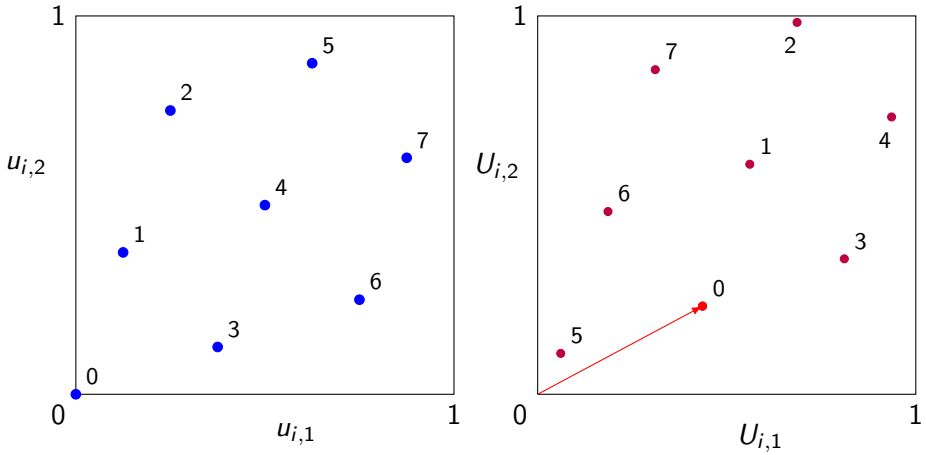
Example: lattice with $s = 2$, $n = 101$, $\mathbf{v}_1 = (1, 12)/101$



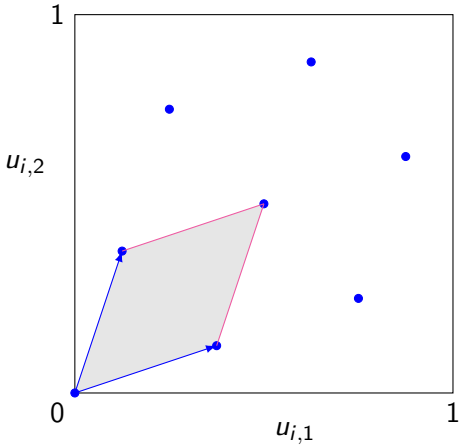
A small lattice shifted by the red vector, modulo 1.



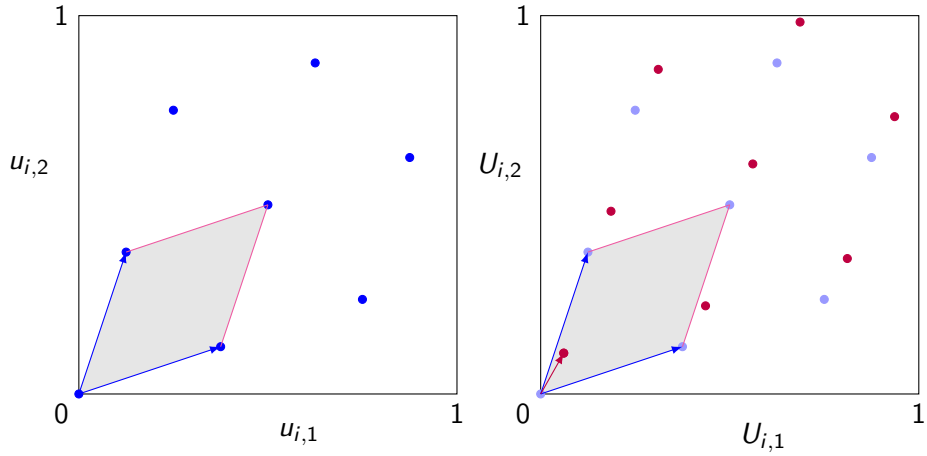
A small lattice shifted by the red vector, modulo 1.



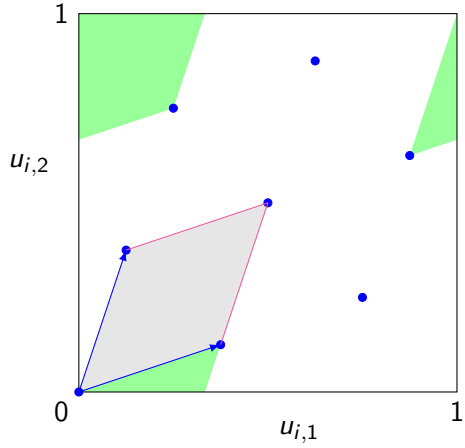
Can generate the shift uniformly in the parallelotope determined by basis vectors,

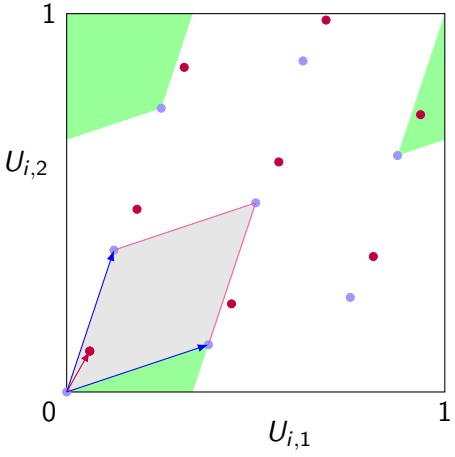


Can generate the shift uniformly in the parallelotope determined by basis vectors,

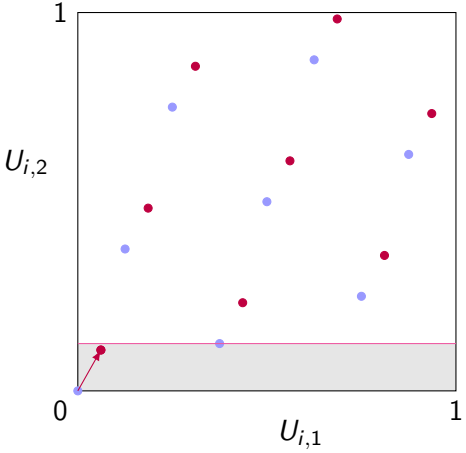


Can generate the shift uniformly in the parallelotope determined by basis vectors, or in any shifted copy of it.

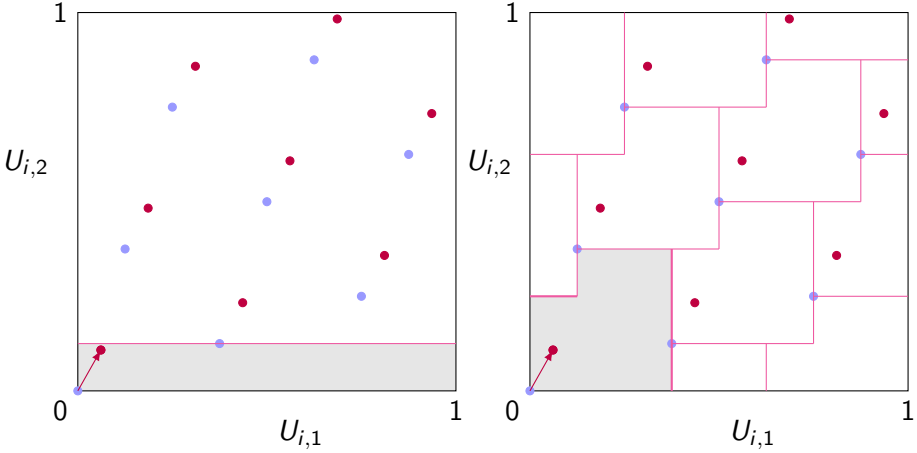




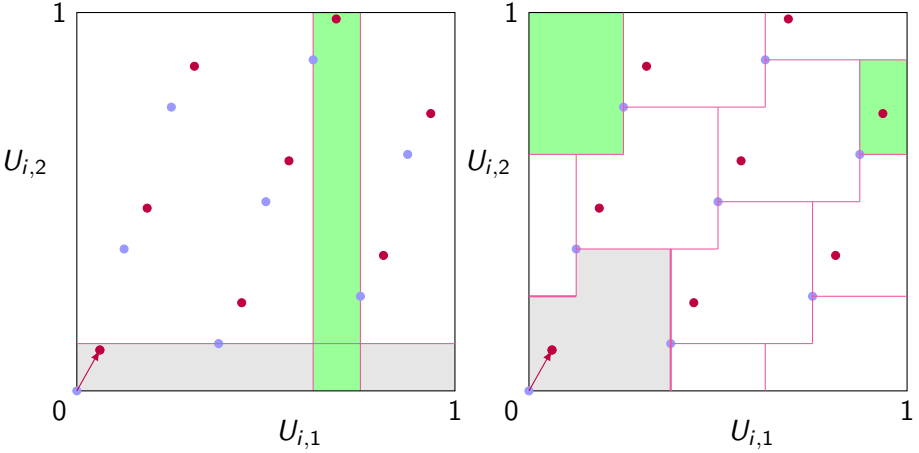
Perhaps less obvious: Can generate it in any of the colored shapes below.



Perhaps less obvious: Can generate it in any of the colored shapes below.



Perhaps less obvious: Can generate it in any of the colored shapes below.



Generating the shift uniformly in one tile

Proposition. Let $R \subset [0,1)^s$ such that

$$\{R_i = (R + \mathbf{u}_i) \bmod 1,\ i = 0, \ldots, n-1\}$$

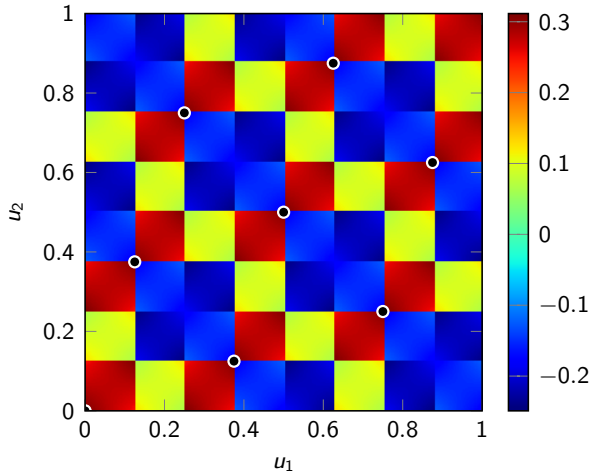
is a partition of $[0,1)^s$ in n regions of volume $1/n$.
Then, sampling the random shift $\mathbf{U}$ uniformly in any given R_i is equivalent to sampling it uniformly in $[0,1)^s$.

The error function

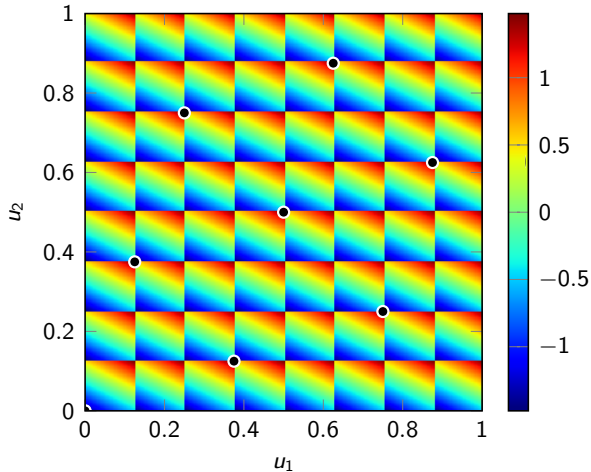
$$g_n(\mathbf{U}) = \hat{\mu}_{n,\mathrm{rqmc}} - \mu$$

over any R_i is the same as over R.

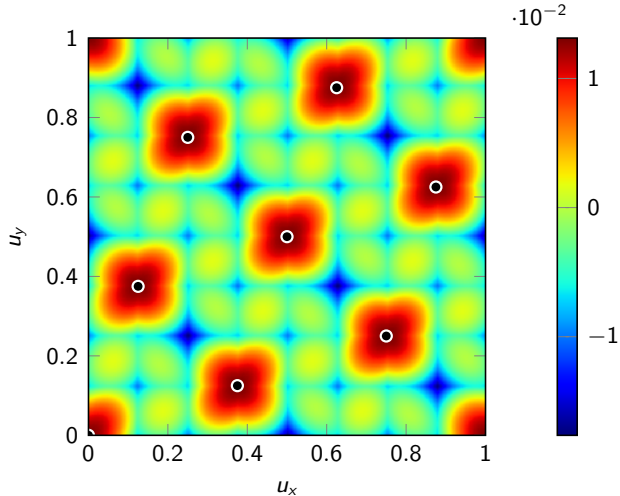
Error function $g_n(\mathbf{u})$ for $f(u_1, u_2) = (u_1 - 1/2)(u_2 - 1/2)$.



Error function $g_n(\mathbf{u})$ for $f(u_1, u_2) = (u_1 - 1/2) + (u_2 - 1/2)$.



Error function $g_n(\mathbf{u})$ for $f(u_1, u_2) = u_1 u_2 (u_1 - 1/2)(u_2 - 1/2)$.



Variance for randomly-shifted lattice

Suppose f has Fourier expansion

$$f(\mathbf{u}) = \sum_{\mathbf{h} \in \mathbb{Z}^s} \hat{f}(\mathbf{h}) e^{2\pi i \mathbf{h}^t \mathbf{u}}.$$

For a randomly shifted lattice, the exact variance is (always)

$$\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}] = \sum_{\mathbf{0} \neq \mathbf{h} \in L_s^*} |\hat{f}(\mathbf{h})|^2,$$

where L_s^* is the dual lattice.

From the viewpoint of variance reduction, an optimal lattice for given f minimizes the square "discrepancy" $D^2(P_n) = \mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}]$.

$$\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}] = \sum_{\mathbf{0}\neq\mathbf{h}\in L_s^*} |\hat{f}(\mathbf{h})|^2.$$

Let $\alpha > 0$ be an even integer. If f has square-integrable mixed partial derivatives up to order $\alpha/2 > 0$, and the periodic continuation of its derivatives up to order $\alpha/2 - 1$ is continuous across the unit cube boundaries, then

$$|\hat{f}(\mathbf{h})|^2 = \mathcal{O}((\max(1, h_1)\cdots\max(1, h_s))^{-\alpha}).$$

Moreover, there is a vector $\mathbf{v}_1 = \mathbf{v}_1(n)$ such that

$$\mathcal{P}_\alpha := \sum_{\mathbf{0}\neq\mathbf{h}\in L_s^*} (\max(1, h_1)\cdots\max(1, h_s))^{-\alpha} = \mathcal{O}(n^{-\alpha+\epsilon}).$$

This $\mathcal{P}_\alpha$ is the variance for a worst-case f having

$$|\hat{f}(\mathbf{h})|^2 = (\max(1, |h_1|)\cdots\max(1, |h_s|))^{-\alpha}.$$

A larger α means a smoother f and a faster convergence rate.

If α is an even integer, this worst-case f is

$$f^*(\mathbf{u}) = \sum_{\mathfrak{u} \subseteq \{1,\ldots,s\}} \prod_{j \in \mathfrak{u}} \frac{(2\pi)^{\alpha/2}}{(\alpha/2)!} B_{\alpha/2}(u_j).$$

where $B_{\alpha/2}$ is the Bernoulli polynomial of degree $\alpha/2$.
In particular, $B_1(u) = u - 1/2$ and $B_2(u) = u^2 - u + 1/6$.
Easy to compute P_α and to search for good lattices in this case!

However: This worst-case function is not necessarily representative of what happens in applications. Also, the hidden factor in $\mathcal{O}$ increases quickly with s, so this result is not very useful for large s.

To get a bound that is uniform in s, the Fourier coefficients must decrease rapidly with the dimension and "size" of vectors $\mathbf{h}$; that is, f must be "smoother" in high-dimensional projections. This is typically what happens in applications for which RQMC is effective!

A very general weighted $\mathcal{P}_\alpha$

$\mathcal{P}_\alpha$ can be generalized by giving different weights $w(\mathbf{h})$ to the vectors $\mathbf{h}$:

$$\tilde{\mathcal{P}}_\alpha := \sum_{\mathbf{0}\neq\mathbf{h}\in L_s^*} w(\mathbf{h})(\max(1,|h_1|)\cdots\max(1,|h_s|))^{-\alpha}.$$

But how do we choose these weights? There are too many!

The optimal weights to minimize the variance are:

$$w(\mathbf{h}) = (\max(1,|h_1|)\cdots\max(1,|h_s|))^{\alpha}|\hat{f}(\mathbf{h})|^2.$$

ANOVA decomposition

The Fourier expansion has too many terms to handle.

As a cruder expansion, we can write $f(\mathbf{u}) = f(u_1, \ldots, u_s)$ as:

$$f(\mathbf{u}) = \sum_{\mathfrak{u} \subseteq \{1,\ldots,s\}} f_{\mathfrak{u}}(\mathbf{u}) = \mu + \sum_{i=1}^{s} f_{\{i\}}(u_i) + \sum_{i,j=1}^{s} f_{\{i,j\}}(u_i, u_j) + \cdots$$

where

$$f_{\mathfrak{u}}(\mathbf{u}) = \int_{[0,1)^{|\bar{\mathfrak{u}}|}} f(\mathbf{u}) \, \mathrm{d}\mathbf{u}_{\bar{\mathfrak{u}}} - \sum_{\mathfrak{v} \subset \mathfrak{u}} f_{\mathfrak{v}}(\mathbf{u}_{\mathfrak{v}}),$$

and the Monte Carlo variance decomposes as

$$\sigma^2 = \sum_{\mathfrak{u} \subseteq \{1,\ldots,s\}} \sigma^2_{\mathfrak{u}}, \qquad \text{where } \sigma^2_{\mathfrak{u}} = \mathrm{Var}[f_{\mathfrak{u}}(\mathbf{U})].$$

The $\sigma^2_{\mathfrak{u}}$'s can be estimated (perhaps very roughly) by MC or RQMC.

Intuition: Make sure the projections $P_n(\mathfrak{u})$ are very uniform for subsets $\mathfrak{u}$ with large $\sigma^2_{\mathfrak{u}}$.

Weighted $\mathcal{P}_{\gamma,\alpha}$ with projection-dependent weights $\gamma_{\mathfrak{u}}$

Denote $\mathfrak{u}(\mathbf{h}) = \mathfrak{u}(h_1, \ldots, h_s)$ the set of indices j for which $h_j \neq 0$.

$$\mathcal{P}_{\gamma,\alpha} \quad = \sum_{\mathbf{0} \neq \mathbf{h} \in L_s^*} \gamma_{\mathfrak{u}(\mathbf{h})} (\max(1, |h_1|) \cdots \max(1, |h_s|))^{-\alpha}.$$

For $\alpha/2$ integer > 0, with $\mathbf{u}_i = (u_{i,1}, \ldots, u_{i,s}) = i\mathbf{v}_1 \bmod 1$,

$$\mathcal{P}_{\gamma,\alpha} = \sum_{\emptyset \neq \mathfrak{u} \subseteq \{1,\ldots,s\}} \frac{1}{n} \sum_{i=0}^{n-1} \gamma_{\mathfrak{u}} \left[\frac{-(-4\pi^2)^{\alpha/2}}{(\alpha)!} \right]^{|\mathfrak{u}|} \prod_{j \in \mathfrak{u}} B_\alpha(u_{i,j}),$$

and the corresponding variation is

$$V_\gamma^2(f) = \sum_{\emptyset \neq \mathfrak{u} \subseteq \{1,\ldots,s\}} \frac{1}{\gamma_{\mathfrak{u}} (4\pi^2)^{\alpha|\mathfrak{u}|/2}} \int_{[0,1]^{|\mathfrak{u}|}} \left| \frac{\partial^{\alpha|\mathfrak{u}|/2}}{\partial \mathbf{u}^{\alpha/2}} f_{\mathfrak{u}}(\mathbf{u}) \right|^2 d\mathbf{u},$$

for $f : [0,1)^s \to \mathbb{R}$ smooth enough. Then,

$$\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}] = \sum_{\mathfrak{u} \subseteq \{1,\ldots,s\}} \mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}(f_{\mathfrak{u}})] \leq V_\gamma^2(f) \mathcal{P}_{\gamma,\alpha}\,.$$

Weighted $\mathcal{P}_{\gamma,\alpha}$:

$$\mathcal{P}_{\gamma,\alpha} = \sum_{\mathbf{0}\neq\mathbf{h}\in L_s^*} \gamma_{\mathrm{u}(\mathbf{h})}(\max(1,h_1),\ldots,\max(1,h_s))^{-\alpha}$$

Variance for a worst-case function whose square Fourier coefficients are

$$|\hat{f}(\mathbf{h})|^2 = \gamma_{\mathrm{u}(\mathbf{h})}(\max(1,h_1),\ldots,\max(1,h_s))^{-\alpha}.$$

This is the RQMC variance for the function

$$f^*(\mathbf{u}) = \sum_{\mathrm{u}\subseteq\{1,\ldots,s\}} \sqrt{\gamma_{\mathrm{u}}} \prod_{j\in\mathrm{u}} \frac{(2\pi)^{\alpha/2}}{(\alpha/2)!} B_{\alpha/2}(u_j).$$

We also have for this f:

$$\sigma_{\mathrm{u}}^2 = \gamma_{\mathrm{u}} \left[\mathrm{Var}[B_{\alpha/2}(U)]\frac{(4\pi^2)^{\alpha/2}}{((\alpha/2)!)^2}\right]^{|\mathrm{u}|} = \gamma_{\mathrm{u}} \left[|B_\alpha(0)|\frac{(4\pi^2)^{\alpha/2}}{(\alpha)!}\right]^{|\mathrm{u}|}.$$

Weighted $\mathcal{P}_{\gamma,\alpha}$:

$$\mathcal{P}_{\gamma,\alpha} = \sum_{\mathbf{0}\neq\mathbf{h}\in L_s^*} \gamma_{\mathrm{u}(\mathbf{h})}(\max(1,h_1),\ldots,\max(1,h_s))^{-\alpha}$$

Variance for a worst-case function whose square Fourier coefficients are

$$|\hat{f}(\mathbf{h})|^2 = \gamma_{\mathrm{u}(\mathbf{h})}(\max(1,h_1),\ldots,\max(1,h_s))^{-\alpha}.$$

This is the RQMC variance for the function

$$f^*(\mathbf{u}) = \sum_{\mathrm{u}\subseteq\{1,\ldots,s\}} \sqrt{\gamma_{\mathrm{u}}} \prod_{j\in\mathrm{u}} \frac{(2\pi)^{\alpha/2}}{(\alpha/2)!} B_{\alpha/2}(u_j).$$

We also have for this f:

$$\sigma_{\mathrm{u}}^2 = \gamma_{\mathrm{u}} \left[\mathrm{Var}[B_{\alpha/2}(U)]\frac{(4\pi^2)^{\alpha/2}}{((\alpha/2)!)^2}\right]^{|\mathrm{u}|} = \gamma_{\mathrm{u}} \left[|B_\alpha(0)|\frac{(4\pi^2)^{\alpha/2}}{(\alpha)!}\right]^{|\mathrm{u}|}.$$

For $\alpha = 2$, we should take $\gamma_{\mathrm{u}} = (3/\pi^2)^{|\mathrm{u}|}\sigma_{\mathrm{u}}^2 \approx (0.30396)^{|\mathrm{u}|}\sigma_{\mathrm{u}}^2$.
For $\alpha = 4$, we should take $\gamma_{\mathrm{u}} = [45/\pi^4]^{|\mathrm{u}|}\sigma_{\mathrm{u}}^2 \approx (0.46197)^{|\mathrm{u}|}\sigma_{\mathrm{u}}^2$.
For $\alpha \to \infty$, we have $\gamma_{\mathrm{u}} \to (0.5)^{|\mathrm{u}|}\sigma_{\mathrm{u}}^2$.

The ratios weight / variance should decrease exponentially with $|\mathrm{u}|$.

Heuristics for choosing the weights

For f^*, we should take $\gamma_{\mathfrak{u}} = \rho^{|\mathfrak{u}|}\sigma_{\mathfrak{u}}^2$ for some constant ρ.

But there are still $2^s - 1$ subsets $\mathfrak{u}$ to consider!

One could define a simple parametric model for the square variations and then estimate the parameters by matching the ANOVA variances $\sigma_{\mathfrak{u}}^2$
[Wang and Sloan 2006, L. and Munger 2012].

For example, product weights: $\gamma_{\mathfrak{u}} = \prod_{j\in\mathfrak{u}} \gamma_j$ for some constants $\gamma_j \geq 0$.

Order-dependent weights: $\gamma_{\mathfrak{u}}$ depends only on $|\mathfrak{u}|$.
Example: $\gamma_{\mathfrak{u}} = 1$ for $|\mathfrak{u}| \leq d$ and $\gamma_{\mathfrak{u}} = 0$ otherwise. Wang (2007) suggests this with $d = 2$.

Mixture: POD weights (Kuo et al. 2011).

Note that all one-dimensional projections (before random shift) are the same.
So the weights $\gamma_{\mathfrak{u}}$ for $|\mathfrak{u}| = 1$ are irrelevant.

Weighted $\mathcal{R}_{\gamma,\alpha}$

When α is not even, one can take

$$\mathcal{R}_{\gamma,\alpha}(P_n) = \sum_{\emptyset \neq \mathfrak{u} \subseteq \{1,\ldots,s\}} \gamma_{\mathfrak{u}} \frac{1}{n} \sum_{i=0}^{n-1} \prod_{j \in \mathfrak{u}} \left(\sum_{h=-\lfloor (n-1)/2 \rfloor}^{\lfloor n/2 \rfloor} \max(1,|h|)^{-\alpha} e^{2\pi \mathrm{i} h u_{i,j}} - 1 \right).$$

Upper bounds on $\mathcal{P}_{\gamma,\alpha}$ can be computed in terms of $\mathcal{R}_{\gamma,\alpha}$.

Can be computed for any $\alpha > 0$ (finite sum). For example, can take $\alpha = 1$.

We can compute it using FFT.

Figure of merit based on the spectral test

Compute the shortest vector $\ell_{\mathfrak{u}}(P_n)$ in dual lattice for each projection $\mathfrak{u}$ and normalize by an upper bound $\ell^*_{|\mathfrak{u}|}(n)$ (with Euclidean length):

$$\mathcal{D}_{\mathfrak{u}}(P_n) = \frac{\ell^*_{|\mathfrak{u}|}(n)}{\ell_{\mathfrak{u}}(P_n)} \geq 1.$$

Figure of merit based on the spectral test

Compute the shortest vector $\ell_{\mathrm{u}}(P_n)$ in dual lattice for each projection u and normalize by an upper bound $\ell^*_{|\mathrm{u}|}(n)$ (with Euclidean length):

$$\mathcal{D}_{\mathrm{u}}(P_n) = \frac{\ell^*_{|\mathrm{u}|}(n)}{\ell_{\mathrm{u}}(P_n)} \geq 1.$$

L. and Lemieux (2000), etc., maximize

$$M_{t_1,\ldots,t_d} = \min\left[\min_{2\leq r\leq t_1} \frac{\ell_{\{1,\ldots,r\}}(P_n)}{\ell^*_r(n)}, \min_{2\leq r\leq d} \min_{\substack{\mathrm{u}=\{j_1,\ldots,j_r\}\subset\{1,\ldots,s\} \\ 1=j_1<\cdots<j_r\leq t_r}} \frac{\ell_{\mathrm{u}}(P_n)}{\ell^*_r(n)}\right].$$

Computing time of $\ell_{\mathrm{u}}(P_n)$ is almost independent of n, but exponential in $|\mathrm{u}|$. Poor lattices can be eliminated quickly.

Can use a different norm, compute shortest vector in primal lattice, etc.

Search methods

Korobov lattices. Search over all admissible a, for $\mathbf{a} = (1, a, a^2, \ldots, \ldots)$.

Random Korobov. Try r random values of a.

Search methods

Korobov lattices. Search over all admissible a, for $\mathbf{a} = (1, a, a^2, \ldots, \ldots)$.

Random Korobov. Try r random values of a.

Rank 1, exhaustive search.

Pure random search. Try admissible vectors $\mathbf{a}$ at random.

Search methods

Korobov lattices. Search over all admissible a, for $\mathbf{a} = (1, a, a^2, \ldots, \ldots)$.

Random Korobov. Try r random values of a.

Rank 1, exhaustive search.

Pure random search. Try admissible vectors $\mathbf{a}$ at random.

Component by component (CBC) construction. (Sloan, Kuo, etc.).

Let $a_1 = 1$;

For $j = 2, 3, \ldots, s$, find $z \in \{1, \ldots, n-1\}$, $\gcd(z, n) = 1$, such that $(a_1, a_2, \ldots, a_j = z)$ minimizes $\mathcal{D}(P_n(\{1, \ldots, j\}))$.

Fast CBC construction for $\mathcal{P}_{\gamma,\alpha}$: use FFT. (Nuyens, Cools).

Search methods

Korobov lattices. Search over all admissible a, for $\mathbf{a} = (1, a, a^2, \ldots, \ldots)$.

Random Korobov. Try r random values of a.

Rank 1, exhaustive search.

Pure random search. Try admissible vectors $\mathbf{a}$ at random.

Component by component (CBC) construction. (Sloan, Kuo, etc.).

Let $a_1 = 1$;

For $j = 2, 3, \ldots, s$, find $z \in \{1, \ldots, n-1\}$, $\gcd(z, n) = 1$, such that $(a_1, a_2, \ldots, a_j = z)$ minimizes $\mathcal{D}(P_n(\{1, \ldots, j\}))$.

Fast CBC construction for $\mathcal{P}_{\gamma,\alpha}$: use FFT. (Nuyens, Cools).

Randomized CBC construction.

Let $a_1 = 1$;

For $j = 2, \ldots, s$, try r random $z \in \{1, \ldots, n-1\}$, $\gcd(z, n) = 1$, and retain $(a_1, a_2, \ldots, a_j = z)$ that minimizes $\mathcal{D}(P_n(\{1, \ldots, j\}))$.

Can add filters to eliminate poor lattices more quickly.

Embedded lattices

$P_{n_1} \subset P_{n_2} \subset \cdots P_{n_m}$ with $n_1 < n_2 < \cdots < n_m$, for some $m > 0$.
Usually: $n_k = b^{c+k}$ for integers $c \geq 0$ and $b \geq 2$, typically with $b = 2$, $\mathbf{a}_k = \mathbf{a}_{k+1} \bmod n_k$ for all $k < m$, and the same random shift.

We need a measure that accounts for the quality of all m lattices.
We standardize the merit at all levels k so they have a comparable scale:

$$\mathcal{E}_q(P_n) = \mathcal{D}_q(P_n)/D_q^*(n),$$

where $D_q^*(n)$ is a normalization factor, e.g., a bound on $\mathcal{D}_q(P_n)$ or a bound on its average over all $(a_1, \ldots, a_s)$ under consideration.
For $\mathcal{P}_{\gamma,\alpha}$, bounds by Sinescu and L. (2012) and Dick et al. (2008).
For CBC, we do this for each coordinate $j = 1, \ldots, s$ (replace s by j).

Then we can take as a global measure (with sum or max):

$$\left[\bar{\mathcal{E}}_{q,m}(P_{n_1}, \ldots, P_{n_m})\right]^q = \sum_{k=1}^{m} w_k \left[\mathcal{E}_q(P_{n_k})\right]^q.$$

Available software tools

Construction: Nuyens (2012) provides Matlab code for fast-CBC construction of lattice rules based on $\mathcal{P}_{\gamma,\alpha}$, with product and order-dependent weights.

Precomputed tables for fixed criteria: Maisonneuve (1972), Sloan and Joe (1994), L. and Lemieux (2000), Kuo (2012), etc.

Software for **using** (randomized) lattice rules in simulations is also available in many places (e.g., in SSJ).

Lattice Builder

Implemented as C++ library, modular, object-oriented, accessible from a program via API.

Various choices of figures of merit, arbitrary weights, construction methods, etc. Easily extensible.

For better run-time efficiency, uses static polymorphism, via templates, rather than dynamic polymorphism. Several other techniques to reduce computations and improve speed.

Offers a pre-compiled program with Unix-like command line interface. Also graphical interface.

Available for download on GitHub, with source code, documentation, and precompiled executable codes for Linux or Windows, in 32-bit and 64-bit versions.

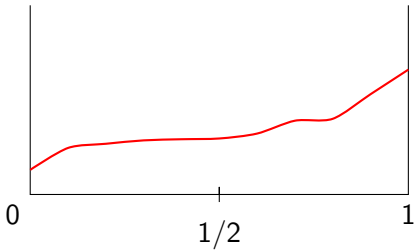
Coming very soon: Construction of polynomial lattice rules as well.

Show graphical interface

Baker's (or tent) transformation

To make the periodic continuation of f continuous.

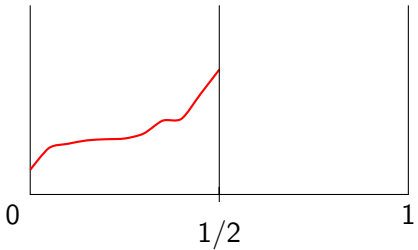
If $f(0) \neq f(1)$, define $\tilde{f}$ by $\tilde{f}(1-u) = \tilde{f}(u) = f(2u)$ for $0 \leq u \leq 1/2$.
This $\tilde{f}$ has the same integral as f and $\tilde{f}(0) = \tilde{f}(1)$.



Baker's (or tent) transformation

To make the periodic continuation of f continuous.

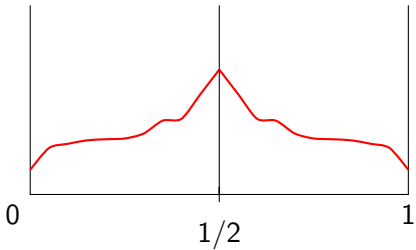
If $f(0) \neq f(1)$, define $\tilde{f}$ by $\tilde{f}(1-u) = \tilde{f}(u) = f(2u)$ for $0 \leq u \leq 1/2$.
This $\tilde{f}$ has the same integral as f and $\tilde{f}(0) = \tilde{f}(1)$.



Baker's (or tent) transformation

To make the periodic continuation of f continuous.

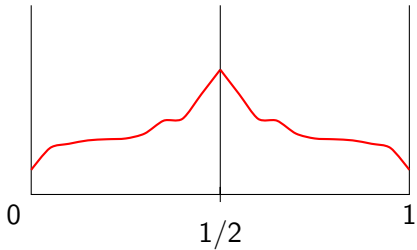
If $f(0) \neq f(1)$, define $\tilde{f}$ by $\tilde{f}(1-u) = \tilde{f}(u) = f(2u)$ for $0 \leq u \leq 1/2$.
This $\tilde{f}$ has the same integral as f and $\tilde{f}(0) = \tilde{f}(1)$.



Baker's (or tent) transformation

To make the periodic continuation of f continuous.

If $f(0) \neq f(1)$, define $\tilde{f}$ by $\tilde{f}(1-u) = \tilde{f}(u) = f(2u)$ for $0 \leq u \leq 1/2$.
This $\tilde{f}$ has the same integral as f and $\tilde{f}(0) = \tilde{f}(1)$.

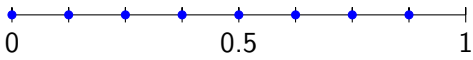


For smooth f, can reduce the variance to $O(n^{-4+\epsilon})$ (Hickernell 2002).
The resulting $\tilde{f}$ is symmetric with respect to $u = 1/2$.

In practice, we transform the points $\mathbf{U}_i$ instead of f.

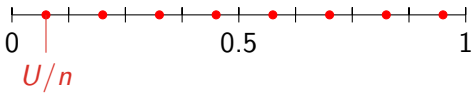
One-dimensional case

Random shift followed by baker's transformation.
Along each coordinate, stretch everything by a factor of 2 and fold.
Same as replacing U_j by $\min[2U_j, 2(1-U_j)]$.



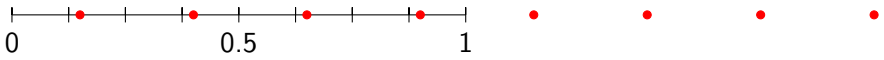
One-dimensional case

Random shift followed by baker's transformation.
Along each coordinate, stretch everything by a factor of 2 and fold.
Same as replacing U_j by $\min[2U_j, 2(1-U_j)]$.



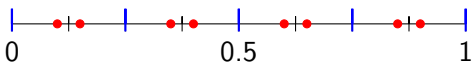
One-dimensional case

Random shift followed by baker's transformation.
Along each coordinate, stretch everything by a factor of 2 and fold.
Same as replacing U_j by $\min[2U_j, 2(1-U_j)]$.



One-dimensional case

Random shift followed by baker's transformation.
Along each coordinate, stretch everything by a factor of 2 and fold.
Same as replacing U_j by $\min[2U_j, 2(1-U_j)]$.



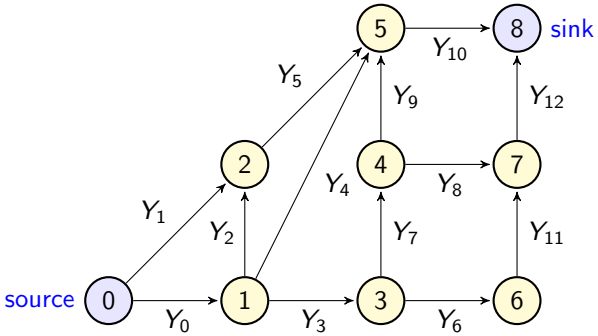
Gives locally antithetic points in intervals of size $2/n$.
This implies that linear pieces over these intervals are integrated exactly.
Intuition: when f is smooth, it is well-approximated by a piecewise linear function, which is integrated exactly, so the error is small.

Example: A stochastic activity network

Gives precedence relations between activities. Activity k has random duration Y_k (also length of arc k) with known cumulative distribution function (cdf) $F_k(y) := \mathbb{P}[Y_k \leq y]$.

Project duration $T =$ (random) length of longest path from source to sink.

May want to estimate $\mathbb{E}[T]$, $\mathbb{P}[T > x]$, a quantile, density of T, etc.



Simulation

Algorithm: to generate T:

for $k = 0, \ldots, 12$ **do**
  Generate $U_k \sim U(0,1)$ and let $Y_k = F_k^{-1}(U_k)$
Compute $X = T = h(Y_0, \ldots, Y_{12}) = f(U_0, \ldots, U_{12})$

Monte Carlo: Repeat n times independently to obtain n realizations $X_1, \ldots, X_n$ of T.
Estimate $\mathbb{E}[T] = \int_{(0,1)^s} f(\mathbf{u}) \mathrm{d}\mathbf{u}$ by $\bar{X}_n = \frac{1}{n} \sum_{i=0}^{n-1} X_i$.

To estimate $\mathbb{P}(T > x)$, take $X = \mathbb{I}[T > x]$ instead.

RQMC: Replace the n independent points by an RQMC point set of size n.

Simulation

Algorithm: to generate T:

for $k = 0, \ldots, 12$ **do**
  Generate $U_k \sim U(0,1)$ and let $Y_k = F_k^{-1}(U_k)$
Compute $X = T = h(Y_0, \ldots, Y_{12}) = f(U_0, \ldots, U_{12})$

Monte Carlo: Repeat n times independently to obtain n realizations $X_1, \ldots, X_n$ of T.
Estimate $\mathbb{E}[T] = \int_{(0,1)^s} f(\mathbf{u})\mathrm{d}\mathbf{u}$ by $\bar{X}_n = \frac{1}{n}\sum_{i=0}^{n-1} X_i$.

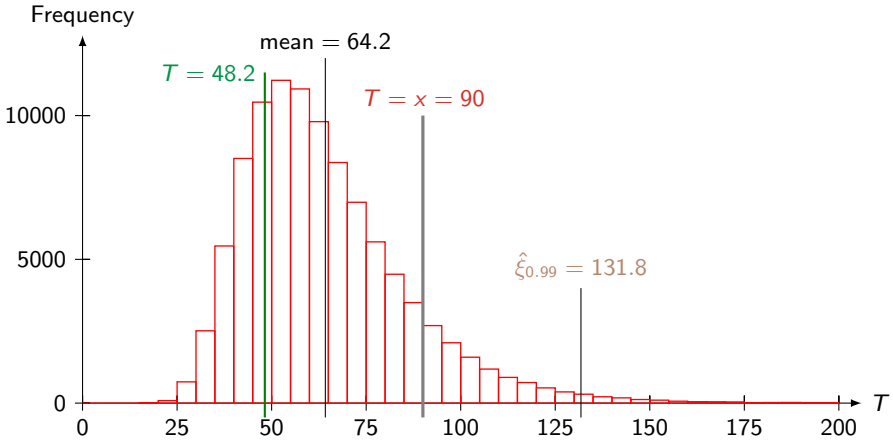
To estimate $\mathbb{P}(T > x)$, take $X = \mathbb{I}[T > x]$ instead.

RQMC: Replace the n independent points by an RQMC point set of size n.

Numerical illustration from Elmaghraby (1977):
$Y_k \sim N(\mu_k, \sigma_k^2)$ for $k = 0, 1, 3, 10, 11$, and $V_k \sim \text{Expon}(1/\mu_k)$ otherwise.
$\mu_0, \ldots, \mu_{12}$: 13.0, 5.5, 7.0, 5.2, 16.5, 14.7, 10.3, 6.0, 4.0, 20.0, 3.2, 3.2, 16.5.

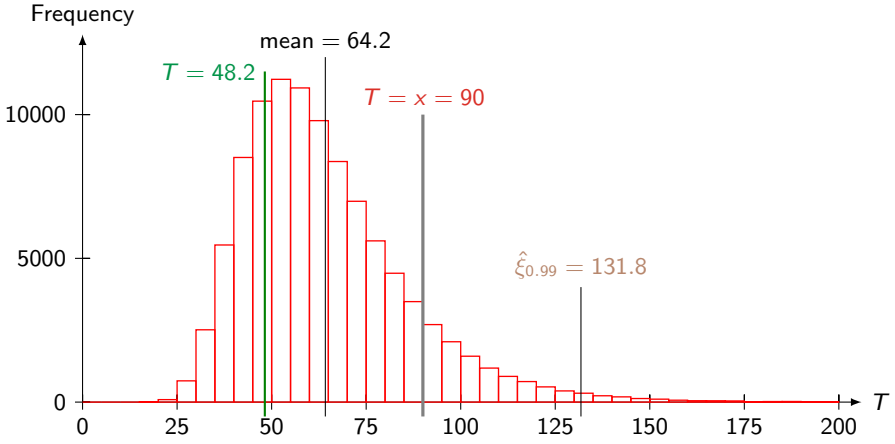
Naive idea: replace each Y_k by its expectation. Gives $T = 48.2$.

Results of an experiment with $n = 100\,000$.
Histogram of values of T is a density estimator that gives more information than a confidence interval on $\mathbb{E}[T]$ or $\mathbb{P}[T > x]$. Values range from 14.4 to 268.6; 11.57% exceed $x = 90$.



Naive idea: replace each Y_k by its expectation. Gives $T = 48.2$.

Results of an experiment with $n = 100\,000$.
Histogram of values of T is a density estimator that gives more information than a confidence interval on $\mathbb{E}[T]$ or $\mathbb{P}[T > x]$. Values range from 14.4 to 268.6; 11.57% exceed $x = 90$.
RQMC can also reduce the error (e.g., the MISE) of a density estimator!

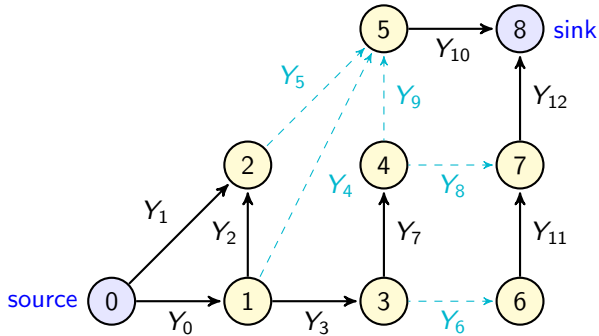


Alternative estimator of $\mathbb{P}[T > x] = \mathbb{E}[\mathbb{I}(T > x)]$ for SAN.

Naive estimator: Generate T and compute $X = \mathbb{I}[T > x]$.
Repeat n times and average.

Alternative estimator of $\mathbb{P}[T > x] = \mathbb{E}[\mathbb{I}(T > x)]$ for SAN.

Naive estimator: Generate T and compute $X = \mathbb{I}[T > x]$.
Repeat n times and average.



Conditional Monte Carlo estimator of $\mathbb{P}[T > x]$. Generate the Y_j's only for the 8 arcs that do not belong to the cut $\mathcal{L} = \{4, 5, 6, 8, 9\}$, and replace $\mathbb{I}[T > x]$ by its conditional expectation given those Y_j's,

$$X_e = \mathbb{P}[T > x \mid \{Y_j, j \notin \mathcal{L}\}].$$

This makes the integrand continuous in the U_j's.

Conditional Monte Carlo estimator of $\mathbb{P}[T > x]$. Generate the Y_j's only for the 8 arcs that do not belong to the cut $\mathcal{L} = \{4, 5, 6, 8, 9\}$, and replace $\mathbb{I}[T > x]$ by its conditional expectation given those Y_j's,

$$X_e = \mathbb{P}[T > x \mid \{Y_j, j \notin \mathcal{L}\}].$$

This makes the integrand continuous in the U_j's.

To compute X_e: for each $l \in \mathcal{L}$, say from a_l to b_l, compute the length α_l of the longest path from 1 to a_l, and the length β_l of the longest path from b_l to the destination.

The longest path that passes through link l does not exceed x iff $\alpha_l + Y_l + \beta_l \leq x$, which occurs with probability $\mathbb{P}[Y_l \leq x - \alpha_l - \beta_l] = F_l[x - \alpha_l - \beta_l]$.

Conditional Monte Carlo estimator of $\mathbb{P}[T > x]$. Generate the Y_j's only for the 8 arcs that do not belong to the cut $\mathcal{L} = \{4, 5, 6, 8, 9\}$, and replace $\mathbb{I}[T > x]$ by its conditional expectation given those Y_j's,

$$X_e = \mathbb{P}[T > x \mid \{Y_j, j \notin \mathcal{L}\}].$$

This makes the integrand continuous in the U_j's.

To compute X_e: for each $l \in \mathcal{L}$, say from a_l to b_l, compute the length α_l of the longest path from 1 to a_l, and the length β_l of the longest path from b_l to the destination.

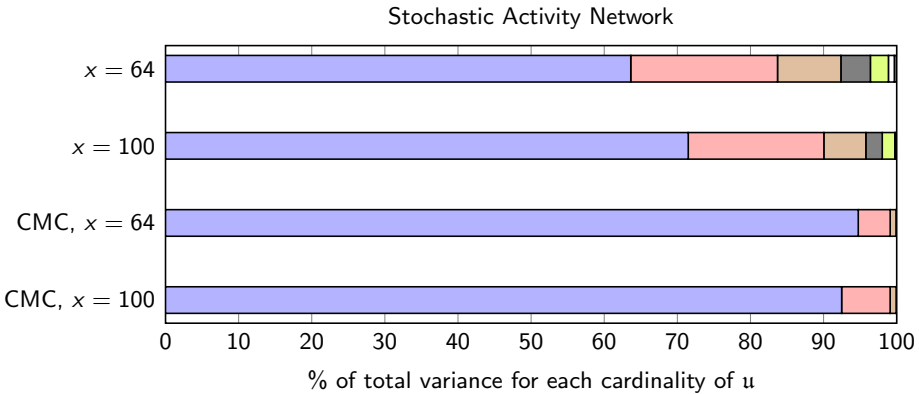
The longest path that passes through link l does not exceed x iff $\alpha_l + Y_l + \beta_l \leq x$, which occurs with probability $\mathbb{P}[Y_l \leq x - \alpha_l - \beta_l] = F_l[x - \alpha_l - \beta_l]$.

Since the Y_l are independent, we obtain

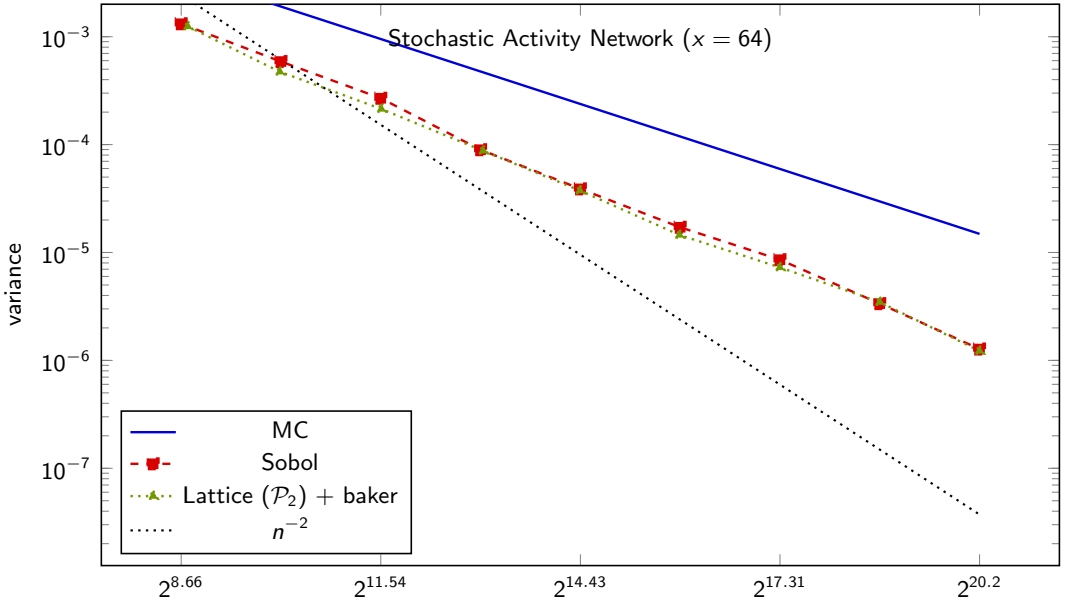
$$X_e = 1 - \prod_{l \in \mathcal{L}} F_l[x - \alpha_l - \beta_l].$$

Can be faster to compute than X, and always has less variance.

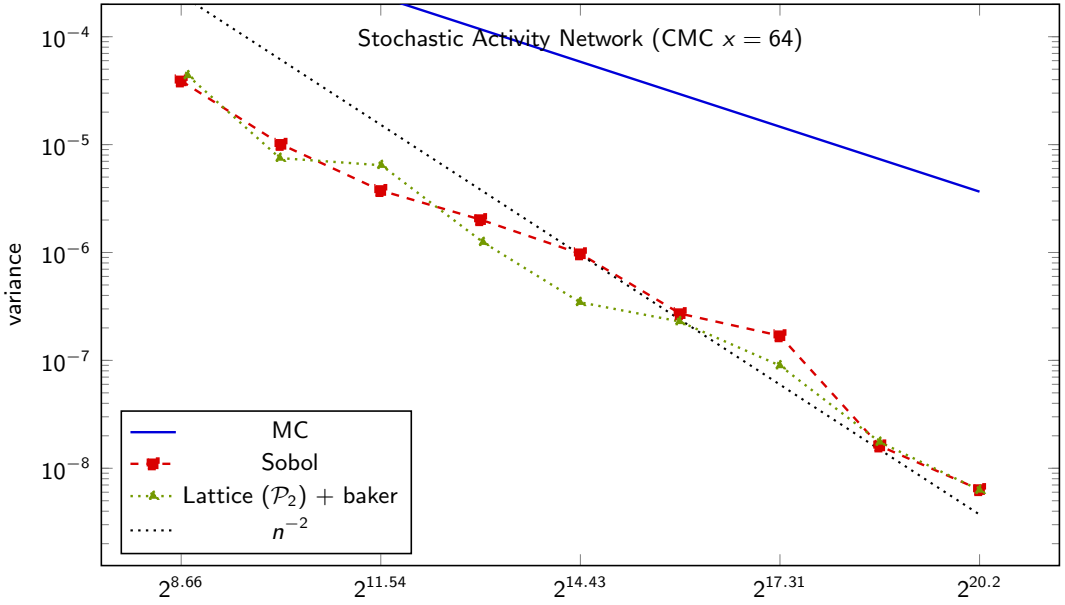
ANOVA Variances for estimator of $\mathbb{P}[T > x]$ in Stochastic Activity Network



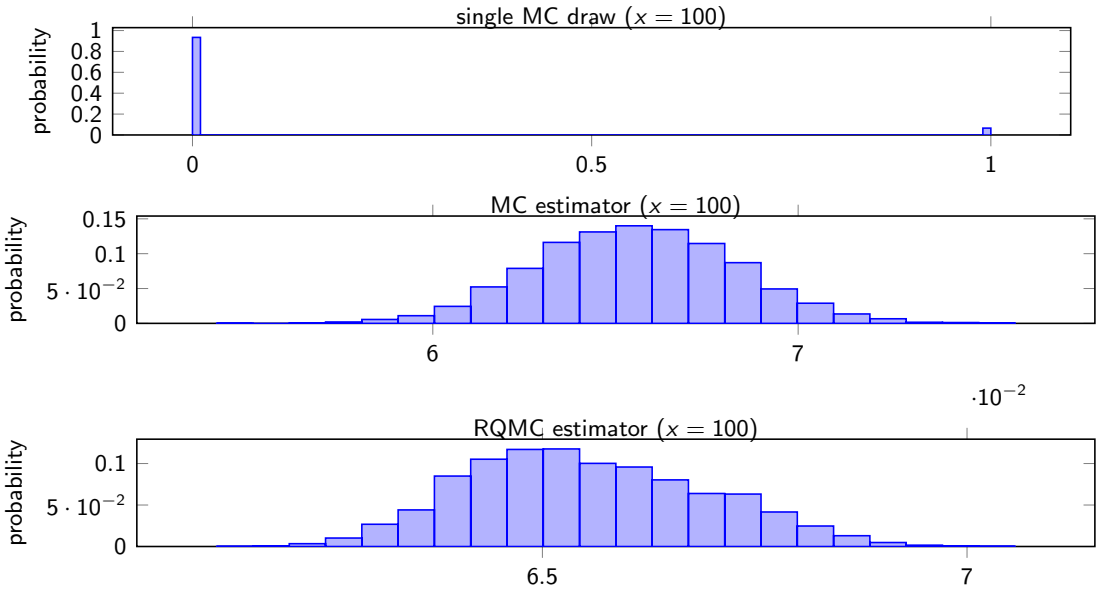
Variance for estimator of $\mathbb{P}[T > x]$ for SAN



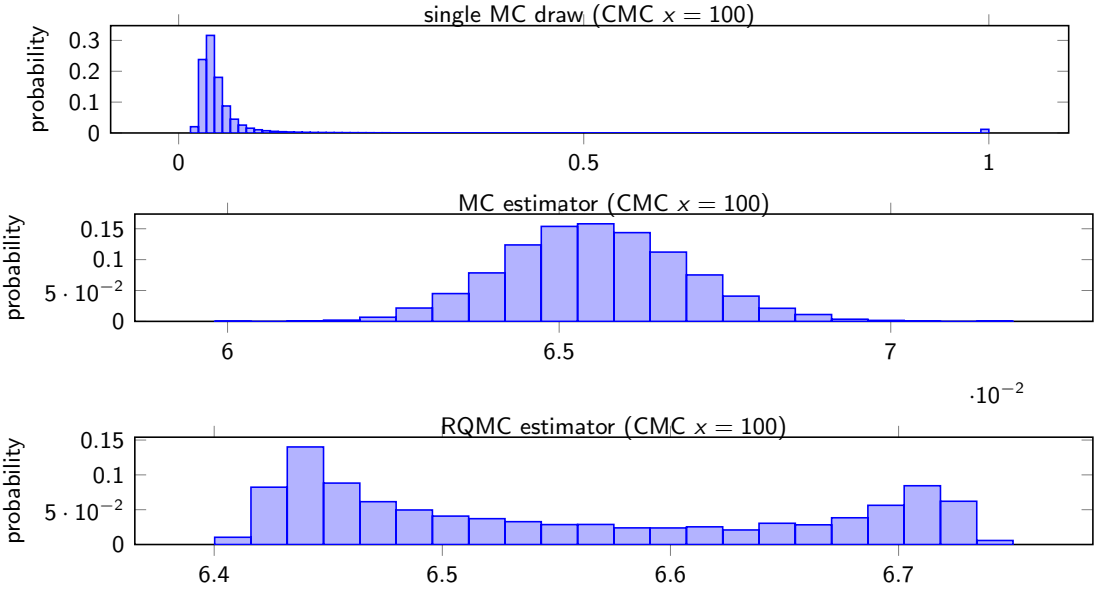
Variance for estimator of $\mathbb{P}[T > x]$ with CMC



Histograms, with $n = 8191$ and $m = 10,000$



Histograms, with $n = 8191$ and $m = 10,000$



Effective dimension

(Caflisch, Morokoff, and Owen 1997).

A function f has effective dimension d in proportion ρ in the superposition sense if

$$\sum_{|\mathrm{u}|\le d} \sigma_{\mathrm{u}}^2 \ge \rho\sigma^2.$$

It has effective dimension d in the truncation sense if

$$\sum_{\mathrm{u}\subseteq\{1,\dots,d\}} \sigma_{\mathrm{u}}^2 \ge \rho\sigma^2.$$

High-dimensional functions with low effective dimension are frequent.
One may change f to make this happen.

Example: Function of a Multinormal vector

Let $\mu = E[f(\mathbf{U})] = E[g(\mathbf{Y})]$ where $\mathbf{Y} = (Y_1, \ldots, Y_s) \sim N(\mathbf{0}, \mathbf{\Sigma})$.

Example: Function of a Multinormal vector

Let $\mu = E[f(\mathbf{U})] = E[g(\mathbf{Y})]$ where $\mathbf{Y} = (Y_1, \ldots, Y_s) \sim N(\mathbf{0}, \mathbf{\Sigma})$.

For example, if the payoff of a financial derivative is a function of the values taken by a c-dimensional geometric Brownian motion (GMB) at d observations times $0 < t_1 < \cdots < t_d = T$, then we have $s = cd$.

Example: Function of a Multinormal vector

Let $\mu = E[f(\mathbf{U})] = E[g(\mathbf{Y})]$ where $\mathbf{Y} = (Y_1, \ldots, Y_s) \sim N(\mathbf{0}, \mathbf{\Sigma})$.

For example, if the payoff of a financial derivative is a function of the values taken by a c-dimensional geometric Brownian motion (GMB) at d observations times $0 < t_1 < \cdots < t_d = T$, then we have $s = cd$.

To generate $\mathbf{Y}$: Decompose $\mathbf{\Sigma} = \mathbf{A}\mathbf{A}^t$, generate $\mathbf{Z} = (Z_1, \ldots, Z_s) \sim N(\mathbf{0}, \mathbf{I})$ where the (independent) Z_j's are generated by inversion: $Z_j = \Phi^{-1}(U_j)$, and return $\mathbf{Y} = \mathbf{A}\mathbf{Z}$.

Example: Function of a Multinormal vector

Let $\mu = E[f(\mathbf{U})] = E[g(\mathbf{Y})]$ where $\mathbf{Y} = (Y_1, \ldots, Y_s) \sim N(\mathbf{0}, \mathbf{\Sigma})$.

For example, if the payoff of a financial derivative is a function of the values taken by a c-dimensional geometric Brownian motion (GMB) at d observations times $0 < t_1 < \cdots < t_d = T$, then we have $s = cd$.

To generate $\mathbf{Y}$: Decompose $\mathbf{\Sigma} = \mathbf{A}\mathbf{A}^t$, generate $\mathbf{Z} = (Z_1, \ldots, Z_s) \sim N(\mathbf{0}, \mathbf{I})$ where the (independent) Z_j's are generated by inversion: $Z_j = \Phi^{-1}(U_j)$, and return $\mathbf{Y} = \mathbf{A}\mathbf{Z}$.

Choice of $\mathbf{A}$?

Example: Function of a Multinormal vector

Let $\mu = E[f(\mathbf{U})] = E[g(\mathbf{Y})]$ where $\mathbf{Y} = (Y_1, \ldots, Y_s) \sim N(\mathbf{0}, \mathbf{\Sigma})$.

For example, if the payoff of a financial derivative is a function of the values taken by a c-dimensional geometric Brownian motion (GMB) at d observations times $0 < t_1 < \cdots < t_d = T$, then we have $s = cd$.

To generate $\mathbf{Y}$: Decompose $\mathbf{\Sigma} = \mathbf{A}\mathbf{A}^t$, generate $\mathbf{Z} = (Z_1, \ldots, Z_s) \sim N(\mathbf{0}, \mathbf{I})$ where the (independent) Z_j's are generated by inversion: $Z_j = \Phi^{-1}(U_j)$, and return $\mathbf{Y} = \mathbf{A}\mathbf{Z}$.

Choice of $\mathbf{A}$?

Cholesky factorization: $\mathbf{A}$ is lower triangular.

Principal component decomposition (PCA) (Ackworth et al. 1998):
$\mathbf{A} = \mathbf{P}\mathbf{D}^{1/2}$ where $\mathbf{D} = \mathrm{diag}(\lambda_s, \ldots, \lambda_1)$ (eigenvalues of $\mathbf{\Sigma}$ in decreasing order) and the columns of $\mathbf{P}$ are the corresponding unit-length eigenvectors.

Principal component decomposition (PCA) (Ackworth et al. 1998):
$\mathbf{A} = \mathbf{P}\mathbf{D}^{1/2}$ where $\mathbf{D} = \mathrm{diag}(\lambda_s, \ldots, \lambda_1)$ (eigenvalues of $\mathbf{\Sigma}$ in decreasing order) and the columns of $\mathbf{P}$ are the corresponding unit-length eigenvectors. With this $\mathbf{A}$, Z_1 accounts for the max amount of variance of $\mathbf{Y}$, then Z_2 the max amount of variance cond. on Z_1, etc.

Principal component decomposition (PCA) (Ackworth et al. 1998):
$\mathbf{A} = \mathbf{P}\mathbf{D}^{1/2}$ where $\mathbf{D} = \mathrm{diag}(\lambda_s, \ldots, \lambda_1)$ (eigenvalues of $\mathbf{\Sigma}$ in decreasing order) and the columns of $\mathbf{P}$ are the corresponding unit-length eigenvectors. With this $\mathbf{A}$, Z_1 accounts for the max amount of variance of $\mathbf{Y}$, then Z_2 the max amount of variance cond. on Z_1, etc.

Function of a Brownian motion (or other Lévy process):
Payoff depends on c-dimensional Brownian motion $\{\mathbf{X}(t),\ t \geq 0\}$ observed at times $0 = t_0 < t_1 < \cdots < t_d = T$.

Principal component decomposition (PCA) (Ackworth et al. 1998):
$\mathbf{A} = \mathbf{P}\mathbf{D}^{1/2}$ where $\mathbf{D} = \mathrm{diag}(\lambda_s, \ldots, \lambda_1)$ (eigenvalues of $\mathbf{\Sigma}$ in decreasing order) and the columns of $\mathbf{P}$ are the corresponding unit-length eigenvectors. With this $\mathbf{A}$, Z_1 accounts for the max amount of variance of $\mathbf{Y}$, then Z_2 the max amount of variance cond. on Z_1, etc.

Function of a Brownian motion (or other Lévy process):
Payoff depends on c-dimensional Brownian motion $\{\mathbf{X}(t),\ t \geq 0\}$ observed at times $0 = t_0 < t_1 < \cdots < t_d = T$.

Sequential (or random walk) method: generate $\mathbf{X}(t_1)$, then $\mathbf{X}(t_2) - \mathbf{X}(t_1)$, then $\mathbf{X}(t_3) - \mathbf{X}(t_2)$, etc.

Principal component decomposition (PCA) (Ackworth et al. 1998):
$\mathbf{A} = \mathbf{P}\mathbf{D}^{1/2}$ where $\mathbf{D} = \mathrm{diag}(\lambda_s, \ldots, \lambda_1)$ (eigenvalues of $\mathbf{\Sigma}$ in decreasing order) and the columns of $\mathbf{P}$ are the corresponding unit-length eigenvectors. With this $\mathbf{A}$, Z_1 accounts for the max amount of variance of $\mathbf{Y}$, then Z_2 the max amount of variance cond. on Z_1, etc.

Function of a Brownian motion (or other Lévy process):
Payoff depends on c-dimensional Brownian motion $\{\mathbf{X}(t),\ t \geq 0\}$ observed at times $0 = t_0 < t_1 < \cdots < t_d = T$.

Sequential (or random walk) method: generate $\mathbf{X}(t_1)$, then $\mathbf{X}(t_2) - \mathbf{X}(t_1)$, then $\mathbf{X}(t_3) - \mathbf{X}(t_2)$, etc.

Bridge sampling (Moskowitz and Caflisch 1996). Suppose $d = 2^m$.
generate $\mathbf{X}(t_d)$, then $\mathbf{X}(t_{d/2})$ conditional on $(\mathbf{X}(0), \mathbf{X}(t_d))$,

Principal component decomposition (PCA) (Ackworth et al. 1998):
$\mathbf{A} = \mathbf{P}\mathbf{D}^{1/2}$ where $\mathbf{D} = \mathrm{diag}(\lambda_s, \ldots, \lambda_1)$ (eigenvalues of $\mathbf{\Sigma}$ in decreasing order) and the columns of $\mathbf{P}$ are the corresponding unit-length eigenvectors. With this $\mathbf{A}$, Z_1 accounts for the max amount of variance of $\mathbf{Y}$, then Z_2 the max amount of variance cond. on Z_1, etc.

Function of a Brownian motion (or other Lévy process):
Payoff depends on c-dimensional Brownian motion $\{\mathbf{X}(t),\ t \geq 0\}$ observed at times $0 = t_0 < t_1 < \cdots < t_d = T$.

Sequential (or random walk) method: generate $\mathbf{X}(t_1)$, then $\mathbf{X}(t_2) - \mathbf{X}(t_1)$, then $\mathbf{X}(t_3) - \mathbf{X}(t_2)$, etc.

Bridge sampling (Moskowitz and Caflisch 1996). Suppose $d = 2^m$.
generate $\mathbf{X}(t_d)$, then $\mathbf{X}(t_{d/2})$ conditional on $(\mathbf{X}(0), \mathbf{X}(t_d))$,
then $\mathbf{X}(t_{d/4})$ conditional on $(\mathbf{X}(0), \mathbf{X}(t_{d/2}))$, and so on.

The first few $N(0,1)$ r.v.'s already sketch the path trajectory.

Principal component decomposition (PCA) (Ackworth et al. 1998):
$\mathbf{A} = \mathbf{P}\mathbf{D}^{1/2}$ where $\mathbf{D} = \mathrm{diag}(\lambda_s, \ldots, \lambda_1)$ (eigenvalues of $\mathbf{\Sigma}$ in decreasing order) and the columns of $\mathbf{P}$ are the corresponding unit-length eigenvectors. With this $\mathbf{A}$, Z_1 accounts for the max amount of variance of $\mathbf{Y}$, then Z_2 the max amount of variance cond. on Z_1, etc.

Function of a Brownian motion (or other Lévy process):
Payoff depends on c-dimensional Brownian motion $\{\mathbf{X}(t),\ t \geq 0\}$ observed at times $0 = t_0 < t_1 < \cdots < t_d = T$.

Sequential (or random walk) method: generate $\mathbf{X}(t_1)$, then $\mathbf{X}(t_2) - \mathbf{X}(t_1)$, then $\mathbf{X}(t_3) - \mathbf{X}(t_2)$, etc.

Bridge sampling (Moskowitz and Caflisch 1996). Suppose $d = 2^m$.
generate $\mathbf{X}(t_d)$, then $\mathbf{X}(t_{d/2})$ conditional on $(\mathbf{X}(0), \mathbf{X}(t_d))$,
then $\mathbf{X}(t_{d/4})$ conditional on $(\mathbf{X}(0), \mathbf{X}(t_{d/2}))$, and so on.

The first few $N(0,1)$ r.v.'s already sketch the path trajectory.

Each of these methods corresponds to some matrix $\mathbf{A}$.
Choice has a large impact on the ANOVA decomposition of f.

Example: Pricing an Asian basket option

We have c assets, d observation times. Want to estimate $\mathbb{E}[f(\mathbf{U})]$, where

$$f(\mathbf{U}) = e^{-rT} \max\left[0,\ \frac{1}{cd}\sum_{i=1}^{c}\sum_{j=1}^{d} S_i(t_j) - K\right]$$

is the net discounted payoff and $S_i(t_j)$ is the price of asset i at time t_j.

Example: Pricing an Asian basket option

We have c assets, d observation times. Want to estimate $\mathbb{E}[f(\mathbf{U})]$, where

$$f(\mathbf{U}) = e^{-rT} \max\left[0,\ \frac{1}{cd}\sum_{i=1}^{c}\sum_{j=1}^{d} S_i(t_j) - K\right]$$

is the net discounted payoff and $S_i(t_j)$ is the price of asset i at time t_j.

Suppose $(S_1(t), \ldots, S_c(t))$ obeys a geometric Brownian motion.
Then, $f(\mathbf{U}) = g(\mathbf{Y})$ where $\mathbf{Y} = (Y_1, \ldots, Y_s) \sim N(\mathbf{0}, \mathbf{\Sigma})$ and $s = cd$.

Example: Pricing an Asian basket option

We have c assets, d observation times. Want to estimate $\mathbb{E}[f(\mathbf{U})]$, where

$$f(\mathbf{U}) = e^{-rT} \max\left[0,\ \frac{1}{cd}\sum_{i=1}^{c}\sum_{j=1}^{d} S_i(t_j) - K\right]$$

is the net discounted payoff and $S_i(t_j)$ is the price of asset i at time t_j.

Suppose $(S_1(t), \ldots, S_c(t))$ obeys a geometric Brownian motion.
Then, $f(\mathbf{U}) = g(\mathbf{Y})$ where $\mathbf{Y} = (Y_1, \ldots, Y_s) \sim N(\mathbf{0}, \mathbf{\Sigma})$ and $s = cd$.

Even with Cholesky decompositions of $\mathbf{\Sigma}$, the two-dimensional projections often account for more than 99% of the variance: low effective dimension in the superposition sense.

With PCA or bridge sampling, we get low effective dimension in the truncation sense. In realistic examples, the first two coordinates Z_1 and Z_2 often account for more than 99.99% of the variance!

Numerical experiment with $c = 10$ and $d = 25$

This gives a 250-dimensional integration problem.

Let $\rho_{i,j} = 0.4$ for all $i \neq j$, $T = 1$, $\sigma_i = 0.1 + 0.4(i-1)/9$ for all i, $r = 0.04$, $S(0) = 100$, and $K = 100$. (Imai and Tan 2002).

Numerical experiment with $c = 10$ and $d = 25$

This gives a 250-dimensional integration problem.

Let $\rho_{i,j} = 0.4$ for all $i \neq j$, $T = 1$, $\sigma_i = 0.1 + 0.4(i-1)/9$ for all i, $r = 0.04$, $S(0) = 100$, and $K = 100$. (Imai and Tan 2002).

Variance reduction factors for Cholesky (left) and PCA (right) (experiment from 2003):

Korobov Lattice Rules

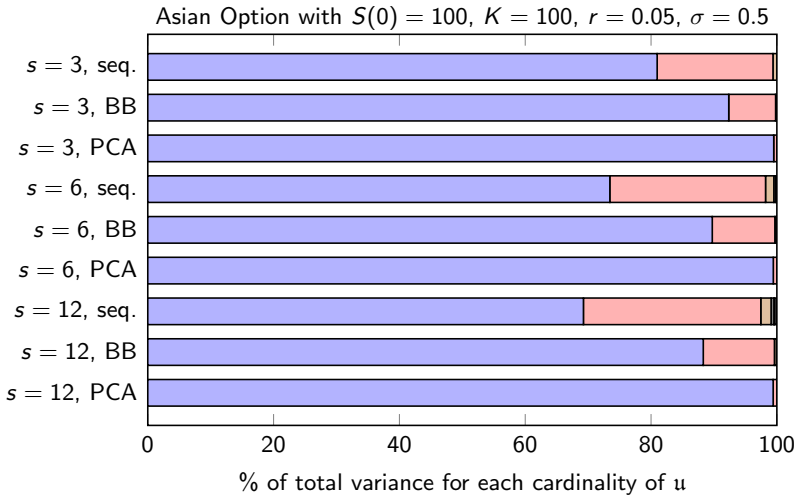
	$n = 16381$ $a = 5693$		$n = 65521$ $a = 944$		$n = 262139$ $a = 21876$	
Lattice+shift	18	878	18	1504	9	2643
Lattice+shift+baker	50	4553	46	3657	43	7553

Sobol' Nets

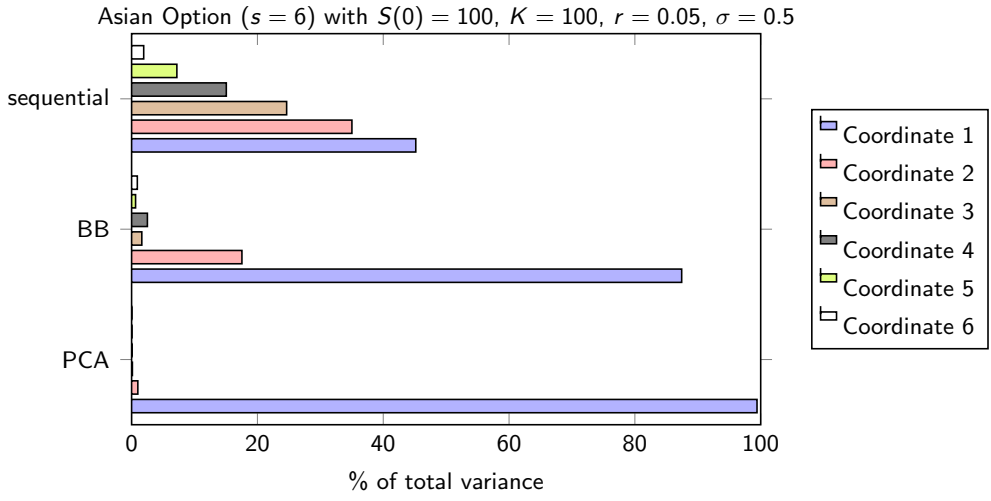
	$n = 2^{14}$		$n = 2^{16}$		$n = 2^{18}$	
Sobol+Shift	10	1299	17	3184	32	6046
Sobol+LMS+Shift	6	4232	4	9219	35	16557

Note: The payoff function is not smooth and also unbounded!

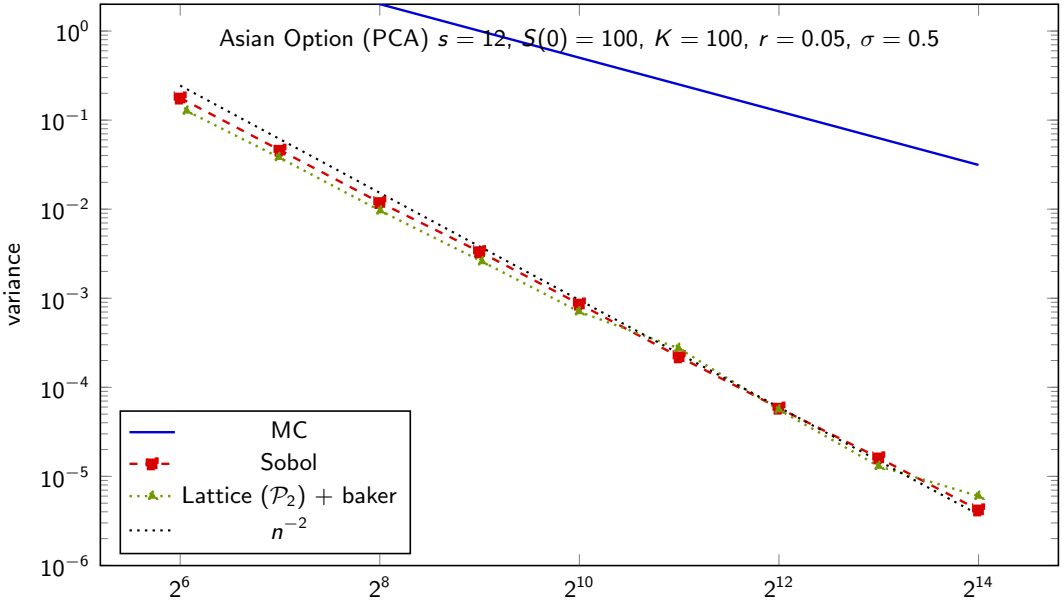
ANOVA Variances for ordinary Asian Option



Total Variance per Coordinate for the Asian Option



Variance with good lattices rules and Sobol points



Polynomial lattice rules

Integers and real numbers are replaced by polynomials and formal series, respectively.

Select prime base $b \geq 2$. Usually $b = 2$.
Replace $\mathbb{Z}$ by $\mathbb{F}_b[z]$, the ring of polynomials over finite field $\mathbb{F}_b \equiv \mathbb{Z}_b$;
Replace $\mathbb{R}$ by $\mathbb{L}_b = \mathbb{F}_b((z^{-1}))$, the field of formal Laurent series over $\mathbb{F}_b$, of the form $\sum_{\ell=\omega}^{\infty} x_\ell z^{-\ell}$, where $x_\ell \in \mathbb{F}_b$.

Polynomial lattice

$$\mathcal{L}_s = \left\{ \mathbf{v}(z) = \sum_{j=1}^{s} q_j(z)\mathbf{v}_j(z) \text{ such that each } q_j(z) \in \mathbb{F}_b[z] \right\},$$

where $\mathbf{v}_1(z), \ldots, \mathbf{v}_s(z)$ are independent vectors in $\mathbb{L}_b^s$, of the form $\mathbf{v}_j(\mathbf{z}) = \mathbf{a}_j(z)/P(z)$, where $P(z) = z^k + \alpha_1 z^{k-1} + \cdots + \alpha_k \in \mathbb{Z}_b[z]$ and each $\mathbf{a}_j(z)$ is a vector of polynomials of degree less than k. Note that $(\mathbb{Z}_b[z])^s \subseteq \mathcal{L}_s$ (integration lattice) and $\mathcal{L}_s$ mod $\mathbb{F}_b[z]$ contains exactly b^k points in $\mathbb{L}_b^s$.

Polynomial lattice rules

Integers and real numbers are replaced by polynomials and formal series, respectively.

Select prime base $b \geq 2$. Usually $b = 2$.
Replace $\mathbb{Z}$ by $\mathbb{F}_b[z]$, the ring of polynomials over finite field $\mathbb{F}_b \equiv \mathbb{Z}_b$;
Replace $\mathbb{R}$ by $\mathbb{L}_b = \mathbb{F}_b((z^{-1}))$, the field of formal Laurent series over $\mathbb{F}_b$, of the form $\sum_{\ell=\omega}^{\infty} x_\ell z^{-\ell}$, where $x_\ell \in \mathbb{F}_b$.

Polynomial lattice

$$\mathcal{L}_s = \left\{ \mathbf{v}(z) = \sum_{j=1}^{s} q_j(z)\mathbf{v}_j(z) \text{ such that each } q_j(z) \in \mathbb{F}_b[z] \right\},$$

where $\mathbf{v}_1(z), \ldots, \mathbf{v}_s(z)$ are independent vectors in $\mathbb{L}_b^s$, of the form $\mathbf{v}_j(\mathbf{z}) = \mathbf{a}_j(z)/P(z)$, where $P(z) = z^k + \alpha_1 z^{k-1} + \cdots + \alpha_k \in \mathbb{Z}_b[z]$ and each $\mathbf{a}_j(z)$ is a vector of polynomials of degree less than k. Note that $(\mathbb{Z}_b[z])^s \subseteq \mathcal{L}_s$ (integration lattice) and $\mathcal{L}_s$ mod $\mathbb{F}_b[z]$ contains exactly b^k points in $\mathbb{L}_b^s$.

For a rule of rank 1, $\mathbf{v}_2(z), \ldots, \mathbf{v}_s(z)$ are the unit vectors.

Define $\varphi : \mathbb{L} \to \mathbb{R}$ by

$$\varphi\left(\sum_{\ell=\omega}^{\infty} x_\ell z^{-\ell}\right) = \sum_{\ell=\omega}^{\infty} x_\ell b^{-\ell}.$$

The polynomial lattice rule (PLR) uses the node set $P_n = \varphi(\mathcal{L}_s) \cap [0,1)^s = \varphi(\mathcal{L}_s \bmod \mathbb{F}_b[z])$.

Define $\varphi : \mathbb{L} \to \mathbb{R}$ by

$$\varphi\left(\sum_{\ell=\omega}^{\infty} x_\ell z^{-\ell}\right) = \sum_{\ell=\omega}^{\infty} x_\ell b^{-\ell}.$$

The polynomial lattice rule (PLR) uses the node set $P_n = \varphi(\mathcal{L}_s) \cap [0,1)^s = \varphi(\mathcal{L}_s \bmod \mathbb{F}_b[z])$.

PLRs were first studied by Niederreiter, Larcher, Tezuka (circa 1990), with rank 1. They were generalized and further studied by Lemieux and L'Ecuyer (circa 2000), then by Dick, Pillischammer, Nuyens, Goda, and others. Most of the properties of ordinary lattice rules have counterparts for the polynomial rules.

The Fourier expansion is replaced by a Walsh expansion, the weighted $\mathcal{P}_{\gamma,\alpha}$ has a counterpart $\mathcal{P}_{\gamma,\alpha,\mathrm{PRL}}$, CBC constructions can provide good parameters, fast CBC also works, etc.

Walsh expansion

For $\mathbf{h} \equiv \mathbf{h}(z) = (h_1(z), \ldots, h_s(z)) \in (\mathbb{F}_b[z])^s$ and $\mathbf{u} = (u_1, \ldots, u_s) \in [0,1)^s$, where

$$h_i(z) = \sum_{j=1}^{\ell_i} h_{i,j} z^{j-1} \quad \text{and} \quad u_i = \sum_{j \geq 1} u_{i,j} b^{-j} \in [0,1), \text{ define } \langle \mathbf{h}, \mathbf{u} \rangle = \sum_{i=1}^{s} \sum_{j=1}^{\ell_i} h_{i,j} u_{i,j} \quad \text{in } \mathbb{F}_b.$$

The Walsh expansion in $\mathbb{F}_b$ of $f : [0,1)^s \to \mathbb{R}$ is

$$f(\mathbf{u}) = \sum_{\mathbf{h} \in (\mathbb{F}_b[z])^s} \tilde{f}(\mathbf{h}) e^{2\pi \mathrm{i} \langle \mathbf{h}, \mathbf{u} \rangle / b},$$

with Walsh coefficients

$$\tilde{f}(\mathbf{h}) = \int_{[0,1)^s} f(\mathbf{u}) e^{-2\pi \mathrm{i} \langle \mathbf{h}, \mathbf{u} \rangle / b} d\mathbf{u}.$$

Theorem: For a PLR with a random digital shift, $\mathrm{Var}[Q_n] = \sum_{\mathbf{0} \neq \mathbf{h} \in \mathcal{L}_s^*} |\tilde{f}(\mathbf{h})|^2$.

Again, we want to kick out of the dual lattice the $\mathbf{h}$'s for which $|\tilde{f}(\mathbf{h})|^2$ is large. For smooth f, the small $\mathbf{h}$ are the most important.

Version of $\mathcal{P}_{\gamma,\alpha}$ for PLRs

A similar reasoning as for ordinary lattice rules leads to

$$\begin{aligned}\mathcal{P}_{\gamma,\alpha,\mathrm{PLR}} &= \sum_{\mathfrak{u}\subseteq\{1,\ldots,s\}} \gamma_{\mathfrak{u}} \prod_{j\in\mathfrak{u},\, h_j\neq 0} 2^{\alpha\lfloor \log_2 h_j\rfloor} \\ &= \sum_{\mathfrak{u}\subseteq\{1,\ldots,s\}} \gamma_{\mathfrak{u}} \frac{1}{n} \sum i = 0^{n-1} \prod_{j\in\mathfrak{u}} \left(\mu(\alpha) - 2^{(1+\lfloor \log_2(x_{i,j})\rfloor)(\alpha-1)}(\mu(\alpha)+1)\right).\end{aligned}$$

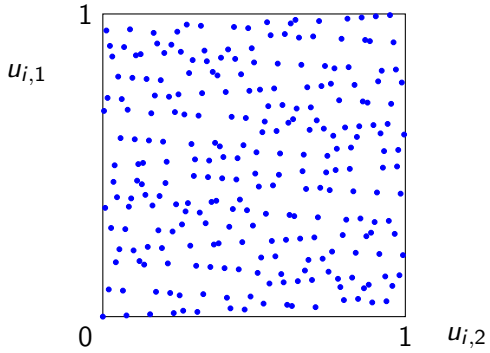
where $\mu(\alpha) = (1-2^{1-\alpha})^{-1}$.

For $\alpha = 2$, this simplifies to $\mu(2) = 2$ and

$$\mathcal{P}_{\gamma,2,\mathrm{PLR}} = \sum_{\mathfrak{u}\subseteq\{1,\ldots,s\}} \gamma_{\mathfrak{u}} \frac{1}{n} \sum_{i=0}^{n-1} \prod_{j\in\mathfrak{u}} \left(2 - 6\cdot 2^{\lfloor \log_2(x_{i,j})\rfloor}\right).$$

Example in $s = 2$ dimensions

Base $b = 2$, $k = 8$, $n = 2^8 = 256$,
$P(z) = 1 + z + z^3 + z^5 + z^8 \equiv [110101001]$,
$q_1(z) = 1$, $q_2(z) = 1 + z + z^2 + z^3 + z^5 + z^7 \equiv [11110101]$.



A PLR is also a special case of a digital net in base b, and this can be used to generate the points efficiently: compute the generating matrices and use the digital net implementation. This is particularly fast in base $b = 2$.

Random shift in space of formal series: equivalent to a random digital shift in base b, applied to all the points. It preserves equidistribution.

Random digital shift for digital net

Equidistribution in digital boxes is lost with random shift modulo 1, but can be kept with a random digital shift in base b.

In base 2: Generate $\mathbf{U} \sim U(0,1)^s$ and XOR it bitwise with each $\mathbf{u}_i$.

Example for $s = 2$:

$$\begin{aligned} \mathbf{u}_i &= (0.01100100\ldots,\ 0.10011000\ldots)_2 \\ \mathbf{U} &= (0.01001010\ldots,\ 0.11101001\ldots)_2 \\ \mathbf{u}_i \oplus \mathbf{U} &= (0.00101110\ldots,\ 0.01110001\ldots)_2. \end{aligned}$$

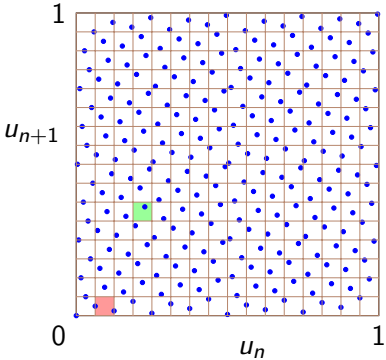
Each point has $U(0,1)$ distribution.
Preservation of the equidistribution ($k_1 = 3$, $k_2 = 5$):

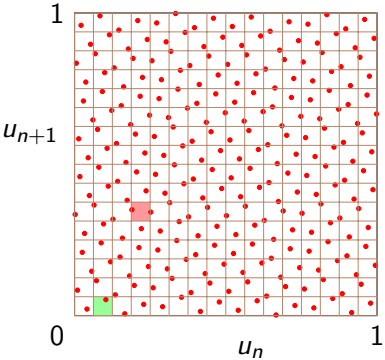
$$\begin{aligned} \mathbf{u}_i &= (0.***,\ 0.*****) \\ \mathbf{U} &= (0.010,\ 0.11101)_2 \\ \mathbf{u}_i \oplus \mathbf{U} &= (0.***,\ 0.*****) \end{aligned}$$

Example with

$$\begin{aligned}\mathbf{U} &= (0.1270111220,\ 0.3185275653)_{10} \\ &= (0.\,0010\,0000100000111100,\ 0.\,0101\,0001100010110000)_2.\end{aligned}$$

Changes the bits 3, 9, 15, 16, 17, 18 of $u_{i,1}$
and the bits 2, 4, 8, 9, 13, 15, 16 of $u_{i,2}$.





Red and green squares are permuted ($k_1 = k_2 = 4$, first 4 bits of $\mathbf{U}$).

Array-RQMC for Markov Chains

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

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

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

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

Ordinary MC: n i.i.d. realizations of Y. Requires $s = \tau d$ uniforms.

Array-RQMC: L., Lécot, Tuffin, et al. [2004, 2006, 2008, etc.]
Simulate an "array" (or population) of n chains in "parallel."
Goal: Want small discrepancy between empirical distribution of states $S_{n,j} = \{X_{0,j}, \ldots, X_{n-1,j}\}$ and theoretical distribution of X_j, at each step j.
At each step, use RQMC point set to advance all the chains by one step.

Some RQMC insight: To simplify, suppose $X_j \sim U(0,1)^\ell$.

We estimate

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

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 $Q_n = \{(X_{i,j-1}, \mathbf{U}_{i,j}),\ 0 \le i < n\}$.

We want Q_n to have low discrepancy (LD) over $[0,1)^{\ell+d}$.

Some RQMC insight: To simplify, suppose $X_j \sim U(0,1)^\ell$.

We estimate

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

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 $Q_n = \{(X_{i,j-1}, \mathbf{U}_{i,j}),\ 0 \le i < n\}$.

We want Q_n to have low discrepancy (LD) over $[0,1)^{\ell+d}$.

We do not choose the $X_{i,j-1}$'s in Q_n: they come from the simulation.
We select a LD point set

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

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 $i = 0, \ldots, n-1$;
for $j = 1, 2, \ldots, \tau$ **do**
  Compute the permutation π_j of the states (for matching);
  Randomize afresh $\{\mathbf{U}_{0,j}, \ldots, \mathbf{U}_{n-1,j}\}$ in $\tilde{Q}_n$;
  $X_{i,j} = \varphi_j(X_{\pi_j(i),j-1}, \mathbf{U}_{i,j})$, for $i = 0, \ldots, n-1$;
  $\hat{\mu}_{\text{arqmc},j,n} = \bar{Y}_{n,j} = \frac{1}{n}\sum_{i=0}^{n-1} g(X_{i,j})$;
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 $i = 0, \ldots, n-1$;
for $j = 1, 2, \ldots, \tau$ **do**
  Compute the permutation π_j of the states (for matching);
  Randomize afresh $\{\mathbf{U}_{0,j}, \ldots, \mathbf{U}_{n-1,j}\}$ in $\tilde{Q}_n$;
  $X_{i,j} = \varphi_j(X_{\pi_j(i),j-1}, \mathbf{U}_{i,j})$, for $i = 0, \ldots, n-1$;
  $\hat{\mu}_{\mathrm{arqmc},j,n} = \bar{Y}_{n,j} = \frac{1}{n}\sum_{i=0}^{n-1} g(X_{i,j})$;
Estimate μ by the average $\bar{Y}_n = \hat{\mu}_{\mathrm{arqmc},n} = \sum_{j=1}^{\tau} \hat{\mu}_{\mathrm{arqmc},j,n}$.

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 $\mathrm{Var}[\bar{Y}_n]$.

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. $\mathcal{O}(n^{-3/2})$ variance.

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

El Haddad, Lécot, L. [2008, 2010]: Deterministic, multidimensional. $\mathcal{O}(n^{-1/(\ell+1)})$ worst-case error under some conditions.

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

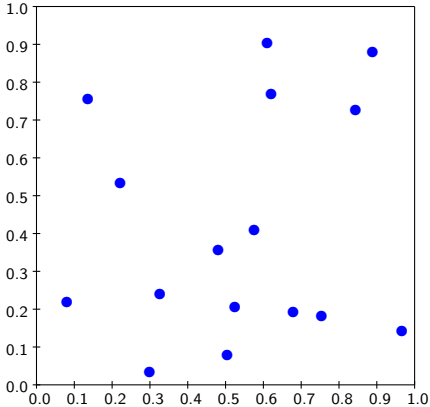
L., Lécot, Munger, and Tuffin [2016]: Survey, comparing sorts, and further examples, some with $\mathcal{O}(n^{-3})$ empirical variance.

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

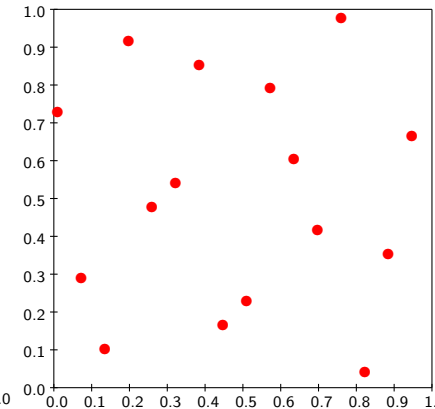
Gerber and Chopin [2015]: Sequential QMC (particle filters), Owen nested scrambling and Hilbert sort. $o(n^{-1})$ variance.

A (4,4) mapping

States of the chains

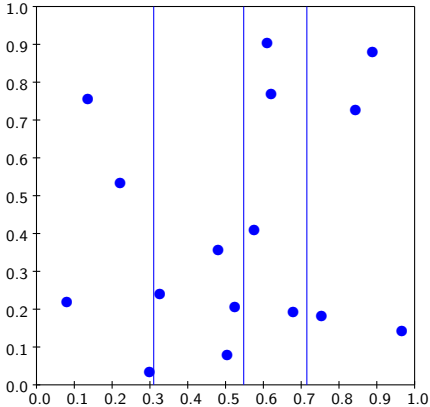


Sobol' net in 2 dimensions after random digital shift

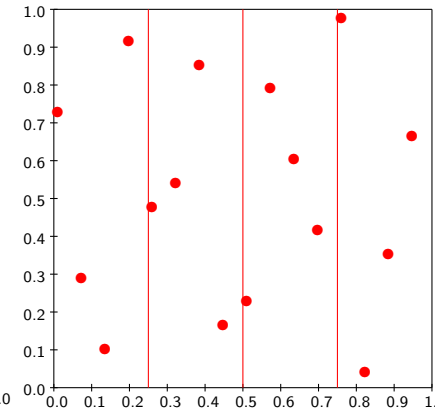


A (4,4) mapping

States of the chains

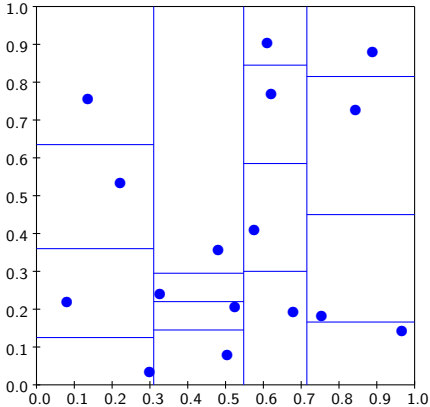


Sobol' net in 2 dimensions after random digital shift

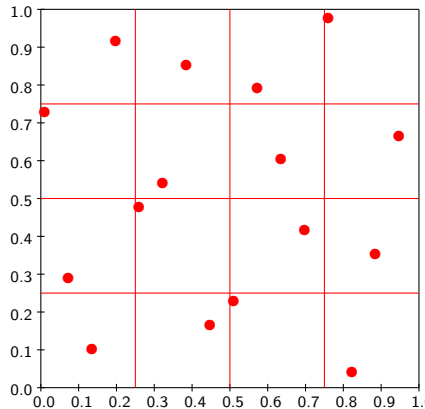


A (4,4) mapping

States of the chains

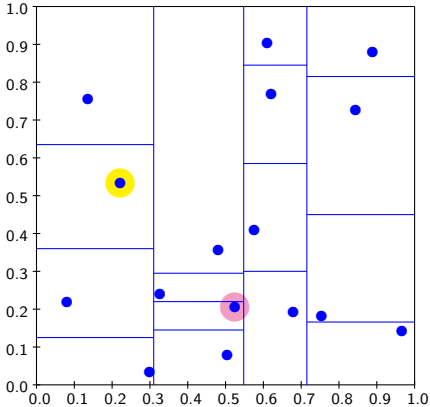


Sobol' net in 2 dimensions after random digital shift

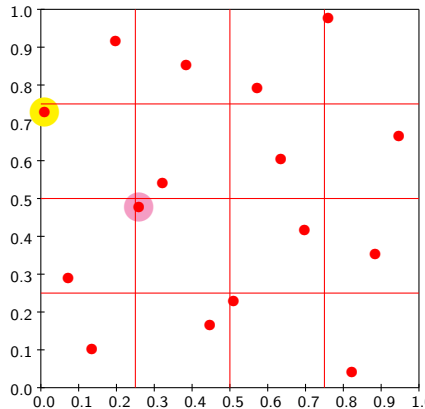


A (4,4) mapping

States of the chains



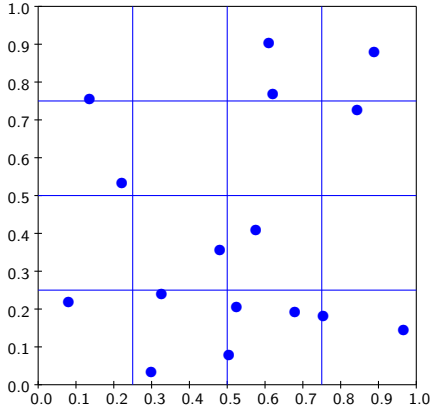
Sobol' net in 2 dimensions after random digital shift



Hilbert curve sort

Map the states to $[0, 1]$, then sort.

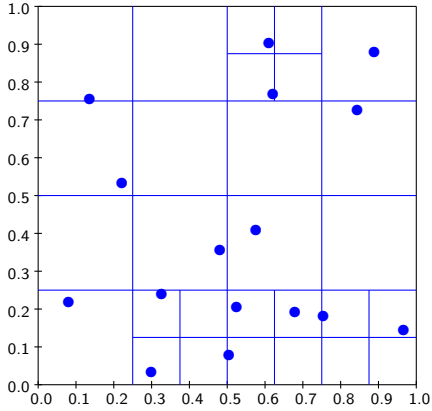
States of the chains



Hilbert curve sort

Map the states to $[0, 1]$, then sort.

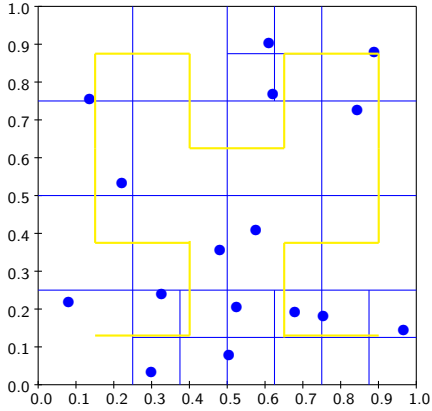
States of the chains



Hilbert curve sort

Map the states to $[0, 1]$, then sort.

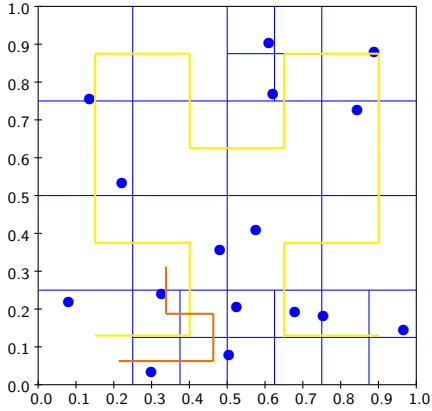
States of the chains



Hilbert curve sort

Map the states to $[0, 1]$, then sort.

States of the chains

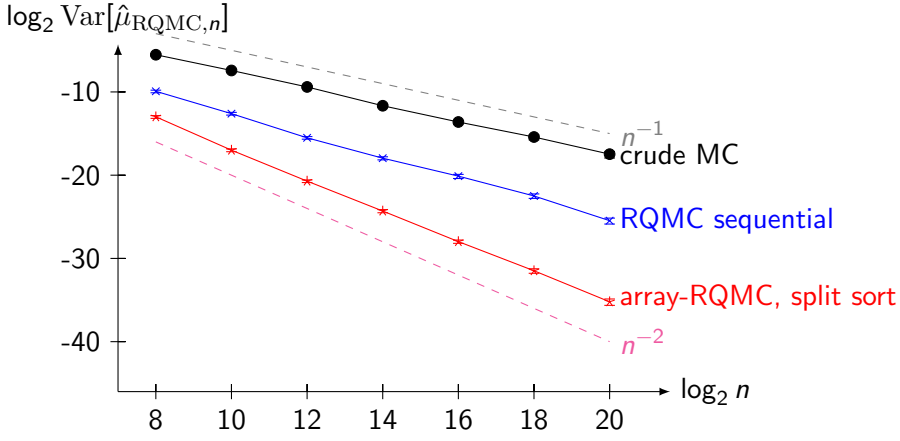


Example: Asian Call Option

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

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

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



Example: Asian Call Option

Sort	RQMC points	$\frac{\log_2 \mathrm{Var}[\bar{Y}_{n,j}]}{\log_2 n}$	VRF	CPU (sec)
Batch sort	SS	-1.38	2.0×10^2	744
($n_1 = n_2$)	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 sort	SS	-1.55	2.4×10^3	840
(logistic map)	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, discussion, etc.

- RQMC can improve the accuracy of estimators considerably in some applications.
- Cleverly modifying the function f can often bring huge statistical efficiency improvements in simulations with RQMC.
- There are often many possibilities for how to change f to make it smoother, periodic, and reduce its effective dimension.
- Point set constructions should be based on discrepancies that take that into account.
- Nonlinear functions of expectations: RQMC also reduces the bias.
- RQMC for density estimation.
- RQMC for optimization.
- Array-RQMC and other QMC methods for Markov chains. Sequential RQMC.
- Still a lot to learn and do ...

Some references on QMC, RQMC, and lattice rules:

- *Monte Carlo and Quasi-Monte Carlo Methods 2014, 2012, 2010, ...* Springer-Verlag, Berlin, 2016, 2014, 2012, ...
- J. Dick and F. Pillichshammer. *Digital Nets and Sequences: Discrepancy Theory and Quasi-Monte Carlo Integration*. Cambridge University Press, Cambridge, U.K., 2010.
- F. J. Hickernell. Lattice rules: How well do they measure up? In P. Hellekalek and G. Larcher, editors, *Random and Quasi-Random Point Sets*, volume 138 of *Lecture Notes in Statistics*, pages 109–166. Springer-Verlag, New York, 1998.
- F. J. Hickernell, H. S. Hong, P. L'Ecuyer, and C. Lemieux. Extensible lattice sequences for quasi-Monte Carlo quadrature. *SIAM Journal on Scientific Computing*, 22(3):1117–1138, 2001.
- J. Imai and K. S. Tan. A general dimension reduction technique for derivative pricing. *Journal of Computational Finance*, 10(2):129–155, 2006.
- P. L'Ecuyer. Polynomial integration lattices. In H. Niederreiter, editor, *Monte Carlo and Quasi-Monte Carlo Methods 2002*, pages 73–98, Berlin, 2004. Springer-Verlag.
- P. L'Ecuyer. Quasi-Monte Carlo methods with applications in finance. *Finance and Stochastics*, 13(3):307–349, 2009.

- P. L'Ecuyer. Randomized quasi-monte carlo: An introduction for practitioners. In P. W. Glynn and A. B. Owen, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2016*, 2017.
- P. L'Ecuyer and C. Lemieux. Variance reduction via lattice rules. *Management Science*, 46(9):1214–1235, 2000.
- P. L'Ecuyer and D. Munger. On figures of merit for randomly-shifted lattice rules. In H. Woźniakowski and L. Plaskota, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2010*, pages 133–159, Berlin, 2012. Springer-Verlag.
- P. L'Ecuyer and D. Munger. Algorithm 958: Lattice builder: A general software tool for constructing rank-1 lattice rules. *ACM Trans. on Mathematical Software*, 42(2):Article 15, 2016.
- C. Lemieux. *Monte Carlo and Quasi-Monte Carlo Sampling*. Springer-Verlag, New York, NY, 2009.
- C. Lemieux and P. L'Ecuyer. Randomized polynomial lattice rules for multivariate integration and simulation. *SIAM Journal on Scientific Computing*, 24(5):1768–1789, 2003.
- H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*, volume 63 of *SIAM CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, PA, 1992.

- D. Nuyens. The construction of good lattice rules and polynomial lattice rules. In Peter Kritzer, Harald Niederreiter, Friedrich Pillichshammer, and Arne Winterhof, editors, *Uniform Distribution and Quasi-Monte Carlo Methods: Discrepancy, Integration and Applications*, pages 223–255. De Gruyter, 2014.
- D. Nuyens and R. Cools. Fast algorithms for component-by-component construction of rank-1 lattice rules in shift-invariant reproducing kernel Hilbert spaces. *Mathematics of Computation*, 75:903–920, 2006.
- I. H. Sloan and S. Joe. *Lattice Methods for Multiple Integration*. Clarendon Press, Oxford, 1994.

Some references on Array-RQMC:

- 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*, 2016. `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.