Introduction to randomized quasi-Monte Carlo methods in simulation

Part B

Pierre L'Ecuyer

Université de Montréal, Canada

ETICS, Saissac, September 2024

QMC/RQMC Theory

Short Recap

We want to estimate $\mu = \int_{[0,1)^s} f(\boldsymbol{u})\mathrm{d}\boldsymbol{u}$ where $\boldsymbol{U} = (U_0, \ldots, U_{s-1})$ and the U_j are i.i.d. $U(0,1)$.

Monte Carlo (MC) estimator:

$$\hat{\mu}_n = \frac{1}{n}\sum_{i=0}^{n-1} f(\boldsymbol{U}_i)$$

where $\boldsymbol{U}_0, \ldots, \boldsymbol{U}_{n-1}$ i.i.d. uniform over $[0,1)^s$.
We have $\mathbb{E}[\hat{\mu}_n] = \mu$ and $\mathrm{Var}[\hat{\mu}_n] = \sigma^2/n = \mathrm{Var}[f(\boldsymbol{U})]/n$.

Quasi-Monte Carlo (QMC) approximation:

$$\bar{\mu}_n = \frac{1}{n}\sum_{i=0}^{n-1} f(\boldsymbol{u}_i)$$

where $P_n = \{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{n-1}\}$ are deterministic points that cover $[0,1)^s$ very evenly.

Randomized Quasi-Monte Carlo (RQMC) estimator:

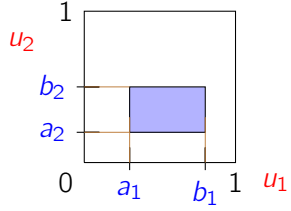
$$\hat{\mu}_{n,\mathrm{rqmc}} = \frac{1}{n}\sum_{i=0}^{n-1} f(\boldsymbol{U}_i),$$

where $\{\boldsymbol{U}_0, \ldots, \boldsymbol{U}_{n-1}\} \subset (0,1)^s$ are randomized versions of the points of P_n so that still cover the space evenly while each $\boldsymbol{U}_i$ is uniform over $[0,1)^s$.

How should we measure the uniformity of P_n?

We want a measure that can translate into error or variance bounds.

For two arbitrary points $\boldsymbol{a} < \boldsymbol{b} \in (0,1)^s$, the rectangular box $B = [\boldsymbol{a}, \boldsymbol{b})$ with corners at $\boldsymbol{a}$ and $\boldsymbol{b}$ is an s-dimensional interval. Let $\mathrm{vol}[\boldsymbol{a}, \boldsymbol{b})$ denote the volume of this box and $|P_n \cap [\boldsymbol{a}, \boldsymbol{b})|/n$ be the proportion of the points that it contains. We call the absolute difference $\mathrm{disc}([\boldsymbol{a}, \boldsymbol{b}), P_n) = |\mathrm{vol}[\boldsymbol{a}, \boldsymbol{b}) - |P_n \cap [\boldsymbol{a}, \boldsymbol{b})|/n|$ the local discrepancy for that interval.



To measure the discrepancy between the distribution of P_n and the uniform distribution, we can take the sup or average of $\mathrm{disc}([\boldsymbol{a}, \boldsymbol{b}), P_n)$ over the intervals B.

Classical discrepancies

Extreme discrepancy of P_n: $\mathcal{D}(P_n) = \sup_{[\boldsymbol{a},\boldsymbol{b})\subseteq[0,1)^s} \text{disc}([\boldsymbol{a},\boldsymbol{b}), P_n)$.

Star discrepancy of P_n: $\mathcal{D}^*(P_n) = \sup_{\boldsymbol{b}\in[0,1)^s} \text{disc}([\boldsymbol{0},\boldsymbol{b}], P_n)$.

These two discrepancies behave in the same way when $n \to \infty$:
Theorem. $\mathcal{D}^*(P_n) \le \mathcal{D}(P_n) \le 2^s \mathcal{D}^*(P_n)$.

The sup can also be replaced by a more general L_p norm, for $1 \le p < \infty$.
$\mathcal{L}_p$ star discrepancy of P_n:

$$\mathcal{D}^*_{(p)}(P_n) = \left(\int_{[0,1)^s} (\text{disc}([\boldsymbol{0},\boldsymbol{u}), P_n))^p \mathrm{d}\boldsymbol{u}\right)^{1/p} \le \mathcal{D}^*(P_n) \le c_{s,p}[\mathcal{D}^*_{(p)}(P_n)]^{p/(p+s)},$$

and similarly for the $\mathcal{L}_p$ extreme discrepancy. They behave as $\mathcal{D}^*(P_n)$ when n increases.

For large s and n, $\mathcal{D}^*(P_n)$ is too hard to compute, practice, but $\mathcal{D}^*_{(2)}(P_n)$ can be computed in $\mathcal{O}(sn^2)$ operations for an arbitrary P_n via a formula from Warnok.

Is $D^*(P_n)$ really meaningful?

Let $P_\infty = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \dots\}$ be an infinite (deterministic) sequence of points and P_n denote the first n points of this sequence.
P_∞ is called uniformly distributed if $\mathcal{D}^*(P_n) \to 0$ as $n \to \infty$.

Theorem (Weyl 1916). When $n \to \infty$, $|E_n| \to 0$ for all bounded Riemann-integrable functions f if and only if $\mathcal{D}^*(P_n) \to 0$.

Is $D^*(P_n)$ really meaningful?

Let $P_\infty = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \dots\}$ be an infinite (deterministic) sequence of points and P_n denote the first n points of this sequence.
P_∞ is called uniformly distributed if $\mathcal{D}^*(P_n) \to 0$ as $n \to \infty$.

Theorem (Weyl 1916). When $n \to \infty$, $|E_n| \to 0$ for all bounded Riemann-integrable functions f if and only if $\mathcal{D}^*(P_n) \to 0$.

Classical **Koksma-Hlawka inequality** (Koksma(1942) for $s = 1$, Hlawka(1961) for $s > 1$):

$$|E_n| \le \mathcal{V}_{\mathrm{hk}}(f) \cdot \mathcal{D}^*(P_n)$$

where $\mathcal{V}_{\mathrm{hk}}(f)$ denotes the total variation of f in the sense of Hardy and Krause;
see Kuipers and Niederreiter (1974) for a definition.
$\mathcal{V}_{\mathrm{hk}}(f) < \infty$ implies no discontinuity not aligned with the axes.

Erdős-Turán-Koksma Inequality. For each s, there is a constant c_s such that for all P_n,

$$\mathcal{D}^*(P_n) \le c_s \left(\frac{1}{m+1} + \sum_{\boldsymbol{h} \in \mathbb{Z}^s(m)} \prod_{j \neq 0} \frac{1}{|h_j|} \left| \frac{1}{n} \sum_{i=0}^{n-1} e^{2\pi \mathrm{i} \boldsymbol{h}^t \boldsymbol{u}_i} \right| \right)$$

for all integers $m \ge 1$, where $\mathbb{Z}^s(m) = \{\boldsymbol{h} = (h_1, \ldots, h_s) \in \{-m, \ldots, m\}^s\}$ and $\mathrm{i} = \sqrt{-1}$.

This inequality permits one to construct explicit low-discrepancy sequences by bounding the exponential sum. It is a convenient tool.

It was used for example to prove that the digital sequence constructions by Sobol', Faure, Niederreiter, etc., have a discrepancy that converges as $\mathcal{O}(n^{-1}(\log n)^s)$. This implies that with these point sets, if $\mathcal{V}_{\mathrm{hk}}(f) < \infty$, then

$$|E_n| \le \mathcal{V}_{\mathrm{hk}}(f) \cdot \mathcal{D}^*(P_n) = \mathcal{O}(n^{-1}(\log n)^s) .$$

Erdős-Turán-Koksma Inequality. For each s, there is a constant c_s such that for all P_n,

$$\mathcal{D}^*(P_n) \le c_s \left(\frac{1}{m+1} + \sum_{\boldsymbol{h} \in \mathbb{Z}^s(m)} \prod_{j \ne 0} \frac{1}{|h_j|} \left| \frac{1}{n} \sum_{i=0}^{n-1} e^{2\pi \mathrm{i} \boldsymbol{h}^{\mathrm{t}} \boldsymbol{u}_i} \right| \right)$$

for all integers $m \ge 1$, where $\mathbb{Z}^s(m) = \{\boldsymbol{h} = (h_1, \ldots, h_s) \in \{-m, \ldots, m\}^s\}$ and $\mathrm{i} = \sqrt{-1}$.

This inequality permits one to construct explicit low-discrepancy sequences by bounding the exponential sum. It is a convenient tool.

It was used for example to prove that the digital sequence constructions by Sobol', Faure, Niederreiter, etc., have a discrepancy that converges as $\mathcal{O}(n^{-1}(\log n)^s)$. This implies that with these point sets, if $\mathcal{V}_{\mathrm{hk}}(f) < \infty$, then

$$|E_n| \le \mathcal{V}_{\mathrm{hk}}(f) \cdot \mathcal{D}^*(P_n) = \mathcal{O}(n^{-1}(\log n)^s) .$$

For lattice rules and digital nets, we will rather use other variants of this inequality that do not involve $\mathcal{D}^*(P_n)$, but other discrepancies that are much easier to compute.

Error and variance expressions for lattice rules

Suppose f has Fourier expansion

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

with Fourier coefficients

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

Note that

$$e^{2\pi \mathrm{i} \boldsymbol{h}^{\mathrm{t}}\boldsymbol{u}} = \prod_{j=1}^{s} e^{2\pi \mathrm{i} h_j u_j} = \prod_{j=1}^{s} [\cos(2\pi h_j u_j) + \mathrm{i}\sin(2\pi h_j u_j)] \qquad \text{(by Euler formula)},$$

so this expansion is in terms of one-periodic trigonometric functions over the real space. It is defined everywhere, just by making copies of its definition over $[0,1)^s$.

When using standard MC to estimate $\mu = \mathbb{E}[f(\boldsymbol{U})]$, the MC variance is

$$\mathrm{Var}[f(\boldsymbol{U})] = \sum_{\boldsymbol{h}\in\mathbb{Z}^s} |\hat{f}(\boldsymbol{h})|^2 \mathrm{Var}[e^{2\pi \mathrm{i} \boldsymbol{h}^{\mathrm{t}}\boldsymbol{U}}] = \sum_{\boldsymbol{h}\in\mathbb{Z}^s} |\hat{f}(\boldsymbol{h})|^2.$$

Suppose now that we estimate $\mu = \mathbb{E}[f(\boldsymbol{U})]$ using a lattice rule with point set $P_n = \{\boldsymbol{u}_i, i = 0, \ldots, n-1\}$. The dual lattice to L_s is

$$L_s^* = \{\boldsymbol{h} \in \mathbb{R}^s : \boldsymbol{h}^t \boldsymbol{v} \in \mathbb{Z} \text{ for all } \boldsymbol{v} \in L_s\} \subseteq \mathbb{Z}^s.$$

One can show that in this case,

$$\frac{1}{n} \sum_{i=0}^{n-1} e^{2\pi i \boldsymbol{h}^t \boldsymbol{u}_i} = \begin{cases} 1 & \text{if } \boldsymbol{h} \in L_s^*; \\ 0 & \text{otherwise.} \end{cases}$$

For a deterministic lattice rule, if $\sum_{\boldsymbol{h} \in \mathbb{Z}^s} |\hat{f}(\boldsymbol{h})| < \infty$, then $E_n = \sum_{\boldsymbol{0} \neq \boldsymbol{h} \in L_s^*} \hat{f}(\boldsymbol{h})$.

Suppose now that we estimate $\mu = \mathbb{E}[f(\boldsymbol{U})]$ using a lattice rule with point set $P_n = \{\boldsymbol{u}_i, i = 0, \ldots, n-1\}$. The dual lattice to L_s is

$$L_s^* = \{\boldsymbol{h} \in \mathbb{R}^s : \boldsymbol{h}^t \boldsymbol{v} \in \mathbb{Z} \text{ for all } \boldsymbol{v} \in L_s\} \subseteq \mathbb{Z}^s.$$

One can show that in this case,

$$\frac{1}{n} \sum_{i=0}^{n-1} e^{2\pi i \boldsymbol{h}^t \boldsymbol{u}_i} = \begin{cases} 1 & \text{if } \boldsymbol{h} \in L_s^*; \\ 0 & \text{otherwise.} \end{cases}$$

For a deterministic lattice rule, if $\sum_{\boldsymbol{h} \in \mathbb{Z}^s} |\hat{f}(\boldsymbol{h})| < \infty$, then $E_n = \sum_{\boldsymbol{0} \neq \boldsymbol{h} \in L_s^*} \hat{f}(\boldsymbol{h})$.

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

From the viewpoint of variance reduction, an optimal lattice for f minimizes $\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}]$. Roughly, the dual lattice should not contain the large (absolute) Fourier coefficients.

For a integer $\alpha > 0$, a function f has **smoothness** α if it has square-integrable mixed partial derivatives up to order α. If f has smoothness α and the periodic continuation of its derivatives up to order $\alpha - 1$ is **continuous across the unit cube boundaries**, then

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

Moreover, there always exists a rank-1 lattice (a vector $\boldsymbol{v}_1 = \boldsymbol{v}_1(n) = \boldsymbol{a}(n)/n$) such that

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

which gives $\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}] = \mathcal{O}(n^{-2\alpha+\epsilon})$, for any $\epsilon > 0$.

$\mathcal{P}_\alpha$ is the error and $\mathcal{P}_{2\alpha}$ is the variance for a worst-case f for which

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

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

When α is a positive integer, this worst-case f is proportional to

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

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

$\mathcal{P}_\alpha$ or $\mathcal{P}_{2\alpha}$ has been proposed long ago as a figure of merit.
Beware: in many "older" papers, α is replaced by $\alpha/2$.

When α is a positive integer, this worst-case f is proportional to

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

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

$\mathcal{P}_\alpha$ or $\mathcal{P}_{2\alpha}$ has been proposed long ago as a figure of merit.
Beware: in many "older" papers, α is replaced by $\alpha/2$.

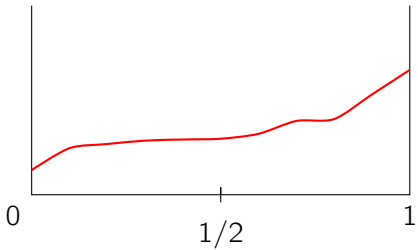
Worst-case function may not be representative of what happens in applications.
Also, the hidden factor in $\mathcal{O}$ increases quickly with s.

To get a bound that is uniform in s, the Fourier coefficients must decrease fast enough with the dimension and "size" of vectors $\boldsymbol{h}$. This is typically what happens in applications for which RQMC is effective!

Baker's (or tent) transformation

In applications, the periodic continuation of f is often not continuous at the boundary of $[0,1)^s$ (i.e., over the torus). A simple transformation can make it continuous:

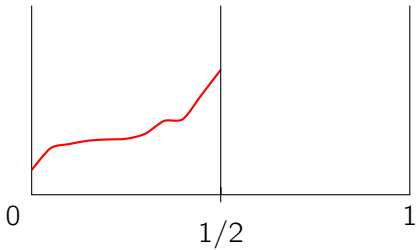
If $f(0) \neq f(1)$, define $\tilde{f}$ by $\tilde{f}(1-u) = \tilde{f}(u) = f(2u)$ for $0 \leq u \leq 1/2$.
This $\tilde{f}$ has the same integral as f and $\tilde{f}(0) = \tilde{f}(1)$.



Baker's (or tent) transformation

In applications, the periodic continuation of f is often not continuous at the boundary of $[0,1)^s$ (i.e., over the torus). A simple transformation can make it continuous:

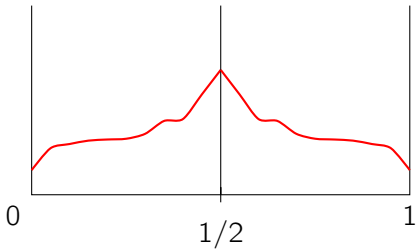
If $f(0) \neq f(1)$, define $\tilde{f}$ by $\tilde{f}(1-u) = \tilde{f}(u) = f(2u)$ for $0 \leq u \leq 1/2$.
This $\tilde{f}$ has the same integral as f and $\tilde{f}(0) = \tilde{f}(1)$.



Baker's (or tent) transformation

In applications, the periodic continuation of f is often not continuous at the boundary of $[0,1)^s$ (i.e., over the torus). A simple transformation can make it continuous:

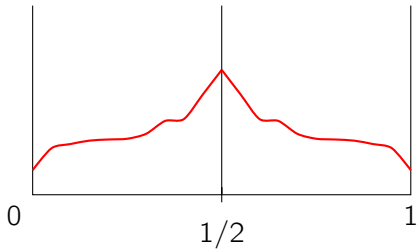
If $f(0) \neq f(1)$, define $\tilde{f}$ by $\tilde{f}(1-u) = \tilde{f}(u) = f(2u)$ for $0 \leq u \leq 1/2$.
This $\tilde{f}$ has the same integral as f and $\tilde{f}(0) = \tilde{f}(1)$.



Baker's (or tent) transformation

In applications, the periodic continuation of f is often not continuous at the boundary of $[0,1)^s$ (i.e., over the torus). A simple transformation can make it 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 from $\mathcal{O}(n^{-2+\epsilon})$ to $\mathcal{O}(n^{-4+\epsilon})$ (Hickernell 2002).
The resulting $\tilde{f}$ is symmetric with respect to $u = 1/2$.

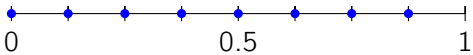
In practice, we transform the points $\boldsymbol{U}_i$ instead of f.

One-dimensional case

Random shift mod 1 followed by baker's transformation.

Along each coordinate, stretch everything by a factor of 2 and fold back.

Same as replacing U_j by $\min[2U_j, 2(1-U_j)]$.

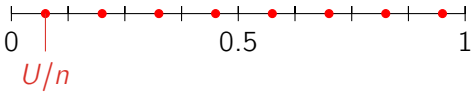


One-dimensional case

Random shift mod 1 followed by baker's transformation.

Along each coordinate, stretch everything by a factor of 2 and fold back.

Same as replacing U_j by $\min[2U_j, 2(1-U_j)]$.

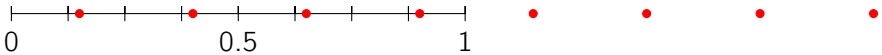


One-dimensional case

Random shift mod 1 followed by baker's transformation.

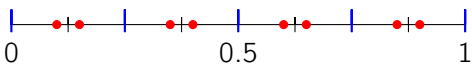
Along each coordinate, stretch everything by a factor of 2 and fold back.

Same as replacing U_j by $\min[2U_j, 2(1-U_j)]$.



One-dimensional case

Random shift mod 1 followed by baker's transformation.
Along each coordinate, stretch everything by a factor of 2 and fold back.
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.

High dimension: ANOVA decomposition

For large s, covering the hypercube $[0,1)^s$ evenly requires an astronomical number of points. Just to have one point in each "quadrant", we need 2^s points, which is unrealistic.

High dimension: ANOVA decomposition

For large s, covering the hypercube $[0,1)^s$ evenly requires an astronomical number of points. Just to have one point in each "quadrant", we need 2^s points, which is unrealistic.

Fortunately, RQMC can still be effective if f is well approximated by a sum of low-dimensional functions, because it suffices to have low error or variance for these low-dimensional functions.

High dimension: ANOVA decomposition

For large s, covering the hypercube $[0,1)^s$ evenly requires an astronomical number of points. Just to have one point in each "quadrant", we need 2^s points, which is unrealistic.

Fortunately, RQMC can still be effective if f is well approximated by a sum of low-dimensional functions, because it suffices to have low error or variance for these low-dimensional functions.

In general, we can write

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

in a way that the Monte Carlo variance decomposes as

$$\mathrm{Var}[f(\boldsymbol{U})] = \sigma^2 = \sum_{\mathfrak{u}\subseteq\{1,\ldots,s\}} \sigma_{\mathfrak{u}}^2 = \sum_{\mathfrak{u}\subseteq\{1,\ldots,s\}} \mathrm{Var}[f_{\mathfrak{u}}(\boldsymbol{U})].$$

The $\sigma_{\mathfrak{u}}^2$'s can be estimated by MC or RQMC (I skip the details here). Idea: Make sure the projections $P_n(\mathfrak{u})$ of P_n over $\mathfrak{u}$ are very uniform for the important subsets $\mathfrak{u}$ (those with a large $\sigma_{\mathfrak{u}}^2$). When they are mostly lower-dim. projections, then it is much easier to make RQMC effective.

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

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

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

For α integer > 0, with $\boldsymbol{u}_i = (u_{i,1}, \ldots, u_{i,s}) = i\boldsymbol{v}_1 \bmod 1$,

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

and the corresponding variation is

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

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

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

This $\mathcal{P}_{\gamma,2\alpha}$ is the RQMC variance for the worst-case function with $\mathcal{V}^2_{\gamma,\alpha}(f) \le 1$:

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

whose Fourier coefficients are

$$\hat{f}^*(\boldsymbol{h}) = \gamma_{\mathfrak{u}(\boldsymbol{h})} (\max(1,|h_1|)\cdots\max(1,|h_s|))^{-\alpha}.$$

For this worst-case function, we have

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

For $\alpha = 1$, we should take $\gamma_{\mathfrak{u}} = (3/\pi^2)^{|\mathfrak{u}|}\sigma^2_{\mathfrak{u}} \approx (0.30396)^{|\mathfrak{u}|}\sigma^2_{\mathfrak{u}}$.
For $\alpha = 2$, we should take $\gamma_{\mathfrak{u}} = [45/\pi^4]^{|\mathfrak{u}|}\sigma^2_{\mathfrak{u}} \approx (0.46197)^{|\mathfrak{u}|}\sigma^2_{\mathfrak{u}}$.
For $\alpha \to \infty$, we have $\gamma_{\mathfrak{u}} \to (0.5)^{|\mathfrak{u}|}\sigma^2_{\mathfrak{u}}$.

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

Heuristics for choosing the weights

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

Heuristics for choosing the weights

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

For f^*, take $\gamma_{\mathfrak{u}} = \rho^{|\mathfrak{u}|}\sigma_{\mathfrak{u}}^2$ for a constant ρ, but there are $2^s - 1$ subsets $\mathfrak{u}$ to consider!

Heuristics for choosing the weights

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

For f^*, take $\gamma_{\mathfrak{u}} = \rho^{|\mathfrak{u}|}\sigma_{\mathfrak{u}}^2$ for a constant ρ, but there are $2^s - 1$ subsets $\mathfrak{u}$ to consider!

One could define a simple parametric model and estimate the parameters by matching the ANOVA variances $\sigma_{\mathfrak{u}}^2$. Some popular model types for the weights:

Order-dependent weights: $\gamma_{\mathfrak{u}} = \Gamma_{|\mathfrak{u}|}$ depends only on $|\mathfrak{u}|$.
It suffices to select $\Gamma_1, \Gamma_2, \Gamma_3$, etc.
Often, one would take $\Gamma_j = 0$ for $j > k$ for some integer k (k parameters to select).
Or $\Gamma_j = \rho^{j-1}$ for all j (a single parameter to select).

Product weights: $\gamma_{\mathfrak{u}} = \prod_{j \in \mathfrak{u}} \gamma_j$ where $\gamma_j \geq 0$ is the weight for coordinate j.

Mixture: POD weights $\gamma_{\mathfrak{u}} = \Gamma_{|\mathfrak{u}|} \prod_{j \in \mathfrak{u}} \gamma_j$, SPOD weights, etc.

Notion of effective dimension

(Caflisch, Morokoff, and Owen 1997).
A function f has effective dimension d in proportion ρ in the superposition sense if

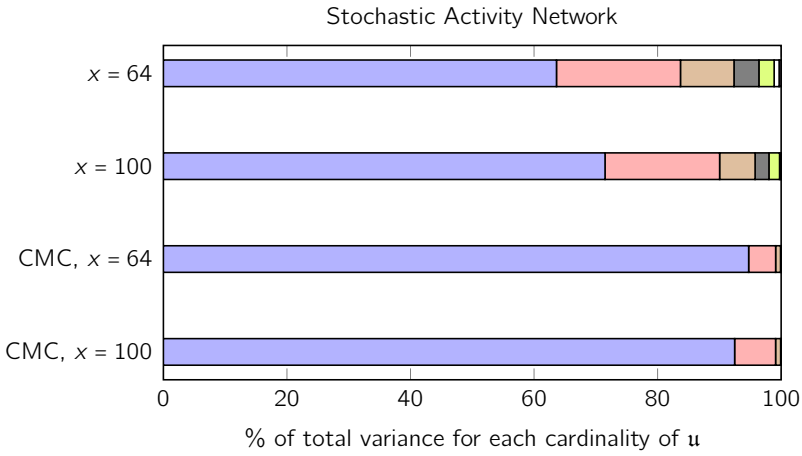
$$\sum_{|\mathfrak{u}| \le d} \sigma_{\mathfrak{u}}^2 \ge \rho \sigma^2.$$

It has effective dimension d in proportion ρ in the truncation sense if

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

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

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



Searching for good parameters

Korobov lattices. Search over all admissible a, for $\boldsymbol{a} = (1, a, a^2, \ldots, \ldots)$.
Random Korobov. Try r random values of a.

Searching for good parameters

Korobov lattices. Search over all admissible a, for $\boldsymbol{a} = (1, a, a^2, \ldots, \ldots)$.
Random Korobov. Try r random values of a.

Rank 1, exhaustive search. Try all possibilities for $\boldsymbol{a} = (1, a_2, \ldots, a_s)$.
Pure random search. Try admissible vectors $\boldsymbol{a}$ at random.

Searching for good parameters

Korobov lattices. Search over all admissible a, for $\boldsymbol{a} = (1, a, a^2, \ldots, \ldots)$.
Random Korobov. Try r random values of a.

Rank 1, exhaustive search. Try all possibilities for $\boldsymbol{a} = (1, a_2, \ldots, a_s)$.
Pure random search. Try admissible vectors $\boldsymbol{a}$ 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).

Searching for good parameters

Korobov lattices. Search over all admissible a, for $\boldsymbol{a} = (1, a, a^2, \ldots, \ldots)$.
Random Korobov. Try r random values of a.

Rank 1, exhaustive search. Try all possibilities for $\boldsymbol{a} = (1, a_2, \ldots, a_s)$.
Pure random search. Try admissible vectors $\boldsymbol{a}$ 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).

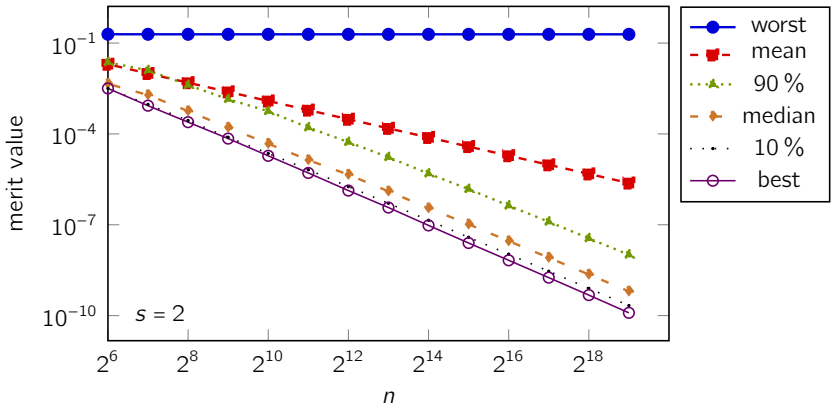
Randomized CBC construction.
Let $a_1 = 1$;
For $j = 2, \ldots, s$, try r random $z \in \{1, \ldots, n-1\}$, $\gcd(z, n) = 1$, and retain $(a_1, a_2, \ldots, a_j = z)$ that minimizes $\mathcal{D}(P_n(\{1, \ldots, j\}))$.

Quantiles of figure of merit

We computed $\mathcal{P}_{\gamma,2}$ for order-dependent weights with $\Gamma_j = 0.3^{j/2}$ for all j, for $n = 2^6, \ldots, 2^{19}$, for all admissible vectors $\boldsymbol{a} = (1, a_2, \ldots, a_s)$ with odd a_j.

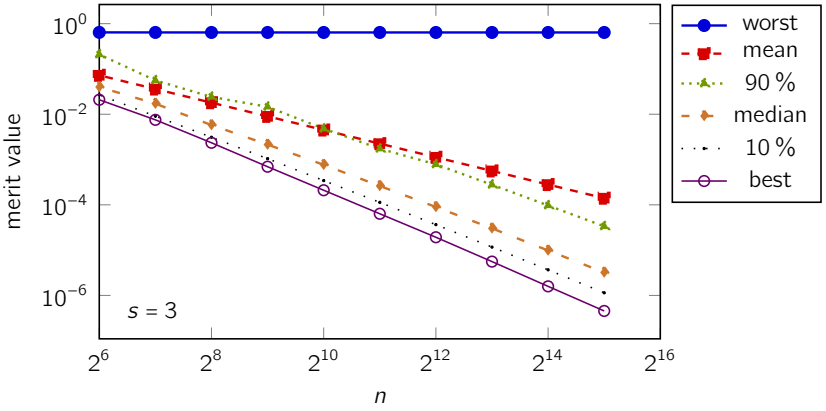
For $s = 2$, a linear regression of $\log \mathcal{P}_{\gamma,2}$ vs $\log n$ for $2^{12} \le n \le 2^{19}$ gives a decreasing rate near $\mathcal{O}(n^{-1.9})$ for the best and $\mathcal{O}(n^{-1})$ for the mean. The worst value is with $\boldsymbol{a} = (1, 1)$ for all n.

The median is close to the best, which means that the majority of vectors a are quite good!



For $s = 3$, a linear regression $\log \mathcal{P}_{\gamma,2}$ vs $\log n$ for $2^{10} \le n \le 2^{15}$ gives decreasing rates near $\mathcal{O}(n^{-1.75})$ for the best and $\mathcal{O}(n^{-1})$ for the mean.

Again, the median is better than the mean.



Example: Playing with the Weights

To see the effect of weights selection on RQMC variance, when choosing a lattice rule, we shall integrate the worst-case function

$$f_\alpha^*(\boldsymbol{u}) = \sum_{\mathfrak{u}\subseteq\{1,\ldots,s\}} \sqrt{v_\mathfrak{u}} \prod_{j\in\mathfrak{u}} \frac{(2\pi)^\alpha}{(\alpha)!} B_\alpha(u_j),$$

whose RQMC variance is $\mathcal{P}_{\gamma,2\alpha}$.

The ideal weights are $\gamma_\mathfrak{u} = v_\mathfrak{u}$. In our experiments, we measure the inflation factor of RQMC variance when we use a lattice rule constructed via fast CBC with different weights $\gamma_\mathfrak{u} \neq v_\mathfrak{u}$.

We start with $s = 10$ and $v_\mathfrak{u} = \Gamma^{|\mathfrak{u}|}$ for $|\mathfrak{u}| \leq k$, for a given integer k and a given constant $\Gamma > 0$. We select the weights as $\gamma_\mathfrak{u} = \tilde{\Gamma}^{|\mathfrak{u}|}$ for $|\mathfrak{u}| \leq \tilde{k}$, where $\tilde{\Gamma}$ and $\tilde{k}$ may differ from Γ and k.

Ratio of RQMC variances for modified vs ideal weights

n	A1	A2	B1	B2	C1	C2
2^8	1.11	1.21	1.13	4.08	3.82	6.80
2^9	1.21	1.10	1.42	10.5	2.93	7.25
2^{10}	1.36	1.38	2.04	4.64	2.86	5.94
2^{11}	1.24	1.43	2.40	6.18	2.15	5.14
2^{12}	1.42	1.66	3.79	13.2	2.47	5.94
2^{13}	1.30	2.38	5.51	9.09	2.66	5.97
2^{14}	1.51	2.54	**30.5**	8.66	**9.11**	**29.1**
2^{15}	1.46	1.93	25.6	**13.3**	3.52	9.71
2^{16}	**1.80**	**2.55**	3.13	12.9	2.73	10.2

A1: $k = \tilde{k} = s$, $\Gamma^2 = 0.1$ and $\tilde{\Gamma}^2 = 0.001$. Not too bad.

A2: $k = \tilde{k} = s$, $\Gamma^2 = 0.001$ and $\tilde{\Gamma}^2 = 0.1$. More impact.

B1: $\Gamma^2 = \tilde{\Gamma}^2 = 0.1$, $k = 4$ and $\tilde{k} = 2$. Criterion is blind on projections of order 3 and 4. This has unpredictable (sometimes dramatic) impact on RQMC variance.

B2: $\Gamma^2 = \tilde{\Gamma}^2 = 0.5$, $k = 2$ and $\tilde{k} = 4$. Gives weight to irrelevant projections. Stronger degradation on average.

C1: $\Gamma^2 = \tilde{\Gamma}^2 = 0.1$ and $k = \tilde{k} = 4$, but increase the variation of f by replacing $v_{\mathfrak{u}}^2$ with $v_{\mathfrak{u}}^2 + \tilde{v}_{\mathfrak{u}}^2$, with
$\tilde{v}_{\mathfrak{u}}^2 = 1.0$ for $\mathfrak{u} = \{1,3\}, \{3,5\}, \{5,7\}, \{7,9\}$,
$\tilde{v}_{\mathfrak{u}}^2 = 0.5$ for $\mathfrak{u} = \{2,3,4\}, \{4,5,6\}, \{6,7,8\}, \{8,9,10\}$,
$\tilde{v}_{\mathfrak{u}}^2 = 0.25$ for $\mathfrak{u} = \{1,2,3,4\}, \{4,5,6,7\}, \{7,8,9,10\}$,
and $\tilde{v}_{\mathfrak{u}} = 0$ for all other $\mathfrak{u}$.
Important projections are not given enough weight relative to others.

C2: Like C1, but $v_{\mathfrak{u}}^2$ is replaced with only $\tilde{v}_{\mathfrak{u}}^2$, as defined above. Many irrelevant projections, with $\tilde{v}_{\mathfrak{u}} = 0$, now have nonzero weights.

In all cases, the variance ratio increases (non-monotonically) with n.

The worst degradations are usually observed when we give too much weight to irrelevant projections!

Discrepancies and Bounds with Projection-Dependent Weights

There are many other ways of defining the variation and the discrepancy. This is usually done by selecting a Reproducing-Kernel Hilbert Space of functions. The choice of kernel determines $\mathcal{D}$ and $\mathcal{V}$.

In general, we can obtain RQMC variance bounds via Hölder-type inequality:

$$\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}] \le (\mathcal{D}(P_n) \cdot \mathcal{V}(f))^2 \qquad \text{where}$$

$$\mathcal{V}(f) = \left(\sum_{\emptyset \ne \mathfrak{u} \subseteq \{1,2,\ldots,s\}} (\mathcal{V}(f_\mathfrak{u})/\gamma_\mathfrak{u})^p \right)^{1/p}, \qquad \mathcal{D}(P_n) = \left(\sum_{\emptyset \ne \mathfrak{u} \subseteq \{1,2,\ldots,s\}} (\gamma_\mathfrak{u} \mathcal{D}_\mathfrak{u}(P_n))^q \right)^{1/q},$$

$1/p + 1/q = 1$, $\gamma_\mathfrak{u} \ge 0$ is a weight assigned to the subset $\mathfrak{u}$, $\mathcal{V}(f_\mathfrak{u})$ is the variation of $f_\mathfrak{u}$, and $\mathcal{D}_\mathfrak{u}(P_n)$ is the discrepancy of the projection of $P_n(\mathfrak{u})$.
Common choice: $p = q = 2$. Sometimes $q = \infty$ with $p = 1$.

Form of discrepancies commonly used in constructions

Many discrepancies that are typically used to search for good parameters efficiently (e.g., in LatNet Builder) are as in the previous slide, with

$$\mathcal{D}_{\mathfrak{u}}^{q}(P_n) = \frac{1}{n}\sum_{i=0}^{n-1}\prod_{j\in\mathfrak{u}}\phi(u_{i,j})$$

for some function $\phi : [0,1) \to \mathbb{R}$, usually with $q = 2$. When computing the products for all subsets $\mathfrak{u}$, we can start with the smaller subsets $\mathfrak{u}$, then save and re-use the partial products.

For the weighted $\mathcal{P}_{2\alpha}$, we take $q = 2$ and

$$\phi(u_{i,j}) = -(-4\pi^2)^{\alpha} B_{2\alpha}(u_{i,j})/(2\alpha)!$$

For digital nets, we have other choices of ϕ.

We need software to support (1) search for good point set parameters and (2) to generate and randomize the points, for these various constructions.

LatNet Builder Software

A C++ library offering tools to search for good parameters for lattice rules, polynomial lattice rules, and digital nets.

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 techniques to reduce computations and improve speed.

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

Also a graphical interface implemented in Python.

Available for download on GitHub, with source code and documentation.

Show graphical interface

Error and variance expressions for digital nets

Similar development as for lattice rules, but instead of using trigonometric Fourier series, we use **Walsh series** for the expansion of f.

Error and variance expressions for digital nets

Similar development as for lattice rules, but instead of using trigonometric Fourier series, we use **Walsh series** for the expansion of f.

Let $\mathbb{N}_0 = \{0, 1, 2, \dots\}$. For $\boldsymbol{k} = (k_1, \dots, k_s) \in \mathbb{N}_0^s$ and $\boldsymbol{u} = (u_1, \dots, u_s) \in [0,1)^s$, where

$$k_j = \sum_{\ell=1}^{\ell_j} k_{j,\ell} b^{\ell-1} \in \mathbb{N}_0 \quad \text{and} \quad u_j = \sum_{\ell \geq 1} u_{j,\ell} b^{-\ell} \in [0,1), \text{ define } \langle \boldsymbol{k}, \boldsymbol{u} \rangle = \sum_{j=1}^{s} \sum_{\ell=1}^{\ell_j} k_{j,\ell} u_{j,\ell} \bmod b.$$

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

$$f(\boldsymbol{u}) = \sum_{\boldsymbol{k} \in \mathbb{N}_0^s} \tilde{f}(\boldsymbol{k}) e^{2\pi \mathrm{i} \langle \boldsymbol{k}, \boldsymbol{u} \rangle / b},$$

with Walsh coefficients

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

Error and variance expressions for digital nets

Similar development as for lattice rules, but instead of using trigonometric Fourier series, we use **Walsh series** for the expansion of f.

Let $\mathbb{N}_0 = \{0, 1, 2, \dots\}$. For $\boldsymbol{k} = (k_1, \dots, k_s) \in \mathbb{N}_0^s$ and $\boldsymbol{u} = (u_1, \dots, u_s) \in [0,1)^s$, where

$$k_j = \sum_{\ell=1}^{\ell_j} k_{j,\ell} b^{\ell-1} \in \mathbb{N}_0 \quad \text{and} \quad u_j = \sum_{\ell \geq 1} u_{j,\ell} b^{-\ell} \in [0,1), \text{ define } \langle \boldsymbol{k}, \boldsymbol{u} \rangle = \sum_{j=1}^{s} \sum_{\ell=1}^{\ell_j} k_{j,\ell} u_{j,\ell} \bmod b.$$

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

$$f(\boldsymbol{u}) = \sum_{\boldsymbol{k} \in \mathbb{N}_0^s} \tilde{f}(\boldsymbol{k}) e^{2\pi \mathrm{i} \langle \boldsymbol{k}, \boldsymbol{u} \rangle / b},$$

with Walsh coefficients

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

For $b = 2$, $e^{-2\pi \mathrm{i} \langle \boldsymbol{k}, \boldsymbol{u} \rangle / b}$ is either $e^0 = 1$ or $e^{-\pi \mathrm{i}} = \cos \pi = -1$.

The Walsh functions are square waves that take values only in $\{-1, 1\}$.

Dual net

Suppose we estimate $\mu = \mathbb{E}[f(\boldsymbol{U})]$ using a digital net in base b with point set $P_n = \{\boldsymbol{u}_i, i = 0, \ldots, n-1\}$. The dual net is

$$\mathcal{C}_s^* = \{\boldsymbol{k} \in \mathbb{N}_0^s : \langle \boldsymbol{k}, \boldsymbol{u}_i \rangle = 0 \text{ for all } \boldsymbol{u}_i \in P_n\}.$$

One can show that

$$\frac{1}{n} \sum_{i=0}^{n-1} e^{2\pi \mathrm{i} \langle \boldsymbol{k}, \boldsymbol{u}_i \rangle / b} = \begin{cases} 1 & \text{if } \boldsymbol{k} \in \mathcal{C}_s^*; \\ 0 & \text{otherwise.} \end{cases}$$

The nonzero digits $k_{j,\ell}$ of a vector $\boldsymbol{k}$ select a specific set of digits $u_{i,j,\ell}$ of the points. The points are equidistributed for those specific digits iff the above sum is 0 for this $\boldsymbol{k}$, iff $\boldsymbol{k} \notin \mathcal{C}_s^*$.

Error and variance expressions

For a deterministic digital net, if $\sum_{\boldsymbol{k} \in \mathbb{N}_0^s} |\tilde{f}(\boldsymbol{k})| < \infty$, then $E_n = \sum_{\boldsymbol{0} \neq \boldsymbol{k} \in \mathcal{C}_s^*} \tilde{f}(\boldsymbol{k})$.

Error and variance expressions

For a deterministic digital net, if $\sum_{\boldsymbol{k} \in \mathbb{N}_0^s} |\tilde{f}(\boldsymbol{k})| < \infty$, then $E_n = \sum_{\boldsymbol{0} \neq \boldsymbol{k} \in \mathcal{C}_s^*} \tilde{f}(\boldsymbol{k})$.

For a digital net with a random digital shift, $\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}] = \sum_{\boldsymbol{0} \neq \boldsymbol{k} \in \mathcal{C}_s^*} |\tilde{f}(\boldsymbol{k})|^2$.

From a variance reduction viewpoint, an optimal digital net for f minimizes this $\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}]$. Roughly, the dual net should not contain vectors $\boldsymbol{k}$ that correspond to large square Walsh coefficients.

Bounds on Walsh coefficients

In the following, we assume for simplicity that $b = 2$.

For a integer $\alpha > 0$, a function f has smoothness α (roughly) if it has square-integrable mixed partial derivatives up to order α.

For an integer $k = 2^{a_1} + \cdots + 2^{a_\nu} > 0$ with $a_1 > \cdots > a_\nu$, we put $\mu_\alpha(k) = (a_1 + 1) + \cdots + (a_{\min(\alpha,\nu)} + 1)$, and $\mu_\alpha(0) = 0$.
Equivalently, if $k_j = \sum_{\ell=1}^{\ell_j} k_{j,\ell} 2^{\ell-1}$, $\mu_\alpha(k_j) = \sum_{\ell=1}^{\min(\alpha,\ell_j)} \ell \mathbb{I}[k_{j,\ell} > 0]$.
For a vector $\boldsymbol{k} = (k_1, \ldots, k_s)$, let $\mu_\alpha(\boldsymbol{k}) = \mu_\alpha(k_1) + \cdots + \mu_\alpha(k_s)$.

Dick (2007, 2008) has shown that if f has smoothness α, then $\left|\hat{f}(\boldsymbol{k})\right| \leq K \tilde{\mathcal{V}}_\alpha^2(f) 2^{-\mu_\alpha(\boldsymbol{k})}$ approximately, where K is a constant and $\tilde{\mathcal{V}}_\alpha(f)$ is the variation of f, which depends on the integrals of the mixed partial derivatives of f of order up to α.

Bounds on Walsh coefficients

In the following, we assume for simplicity that $b = 2$.

For a integer $\alpha > 0$, a function f has smoothness α (roughly) if it has square-integrable mixed partial derivatives up to order α.

For an integer $k = 2^{a_1} + \cdots + 2^{a_\nu} > 0$ with $a_1 > \cdots > a_\nu$, we put $\mu_\alpha(k) = (a_1 + 1) + \cdots + (a_{\min(\alpha,\nu)} + 1)$, and $\mu_\alpha(0) = 0$.
Equivalently, if $k_j = \sum_{\ell=1}^{\ell_j} k_{j,\ell} 2^{\ell-1}$, $\mu_\alpha(k_j) = \sum_{\ell=1}^{\min(\alpha,\ell_j)} \ell \mathbb{I}[k_{j,\ell} > 0]$.
For a vector $\boldsymbol{k} = (k_1, \ldots, k_s)$, let $\mu_\alpha(\boldsymbol{k}) = \mu_\alpha(k_1) + \cdots + \mu_\alpha(k_s)$.
Dick (2007, 2008) has shown that if f has smoothness α, then $\left|\hat{f}(\boldsymbol{k})\right| \le K \tilde{\mathcal{V}}_\alpha^2(f) 2^{-\mu_\alpha(\boldsymbol{k})}$ approximately, where K is a constant and $\tilde{\mathcal{V}}_\alpha(f)$ is the variation of f, which depends on the integrals of the mixed partial derivatives of f of order up to α.

This suggests using the FOM $\sum_{\mathbf{0} \ne \boldsymbol{k} \in \mathcal{C}_s^*} 2^{-\mu_\alpha(\boldsymbol{k})}$ for QMC and $\sum_{\mathbf{0} \ne \boldsymbol{k} \in \mathcal{C}_s^*} 2^{-2\mu_\alpha(\boldsymbol{k})}$ for RQMC, for a given α. But these sums have an infinite number of terms!

Dick replaced the sum by a max and showed how to construct digital nets for which $\max_{\mathbf{0} \ne \boldsymbol{k} \in \mathcal{C}_s^*} 2^{-\mu_\alpha(\boldsymbol{k})} = \mathcal{O}(n^{-\alpha}(\log n)^{\alpha s})$, for any integer $\alpha > 0$, using an interlacing technique.

Figures of Merit for Digital Nets

A variety of discrepancies of the form

$$\mathcal{D}(P_n) = \left(\sum_{\emptyset \neq \mathfrak{u} \subseteq \{1,2,\ldots,s\}} (\gamma_{\mathfrak{u}} \mathcal{D}_{\mathfrak{u}}(P_n))^q \right)^{1/q} \qquad \text{with} \qquad (\mathcal{D}_{\mathfrak{u}}(P_n))^q = \frac{1}{n} \sum_{i=0}^{n-1} \prod_{j \in \mathfrak{u}} \phi(u_{i,j})$$

for some function $\phi : [0,1) \to \mathbb{R}$ have been defined recently for digital nets, based on various assumptions on the rate of decrease of the Walsh coefficients, which often follow from an assumption on the smoothness level α of f.

Some of them can be seen as counterparts of the $\mathcal{P}_\alpha$ used for lattices.
They can be used in the same way to make CBC constructions

One popular choice of ϕ for $b = 2$ and $\alpha = 1$:

$$\phi(u) \quad = \quad 2 - \mathbb{I}[u > 0] \cdot 6 \cdot 2^{\lfloor \log_2(u) \rfloor}$$

where $-\lfloor \log_2 u \rfloor$ is the index of the first nonzero digit in the binary expansion of u.

Latnet Builder can make searches for good point sets based on such discrepancies.

Polynomial lattice rules (in base $b = 2$)

Similar to lattice rules, except that:
Replace $\mathbb{Z}$ by $\mathbb{F}_2[z]$, the ring of polynomials over finite field $\mathbb{F}_2 \equiv \{0, 1\}$;
Replace $\mathbb{R}$ by $\mathbb{L}_2 = \mathbb{F}_2((z^{-1}))$, the field of formal Laurent series over $\mathbb{F}_2$, of the form $\sum_{\ell=\omega}^{\infty} x_\ell z^{-\ell}$, where $x_\ell \in \mathbb{F}_2$. Define $\varphi : \mathbb{L} \to \mathbb{R}$ by

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

We select a polynomial modulus $Q(z) \in \mathbb{Z}_2[z]$ of degree k and a generating vector $\boldsymbol{a}(z) = (a_1(z), \ldots, a_s(z)) \in \mathbb{Z}_2[z]^s$, whose coordinates are polynomials of degrees less than k having no common factor with $Q(z)$. The point set of cardinality $n = 2^k$ is

$$P_n = \left\{\left(\varphi\left(\frac{h(z)a_1(z)}{Q(z)}\right), \ldots, \varphi\left(\frac{h(z)a_s(z)}{Q(z)}\right)\right) : h(z) \in \mathbb{Z}_2[z] \text{ and } \deg(h(z)) < k\right\}.$$

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.

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.

A PLR is actually a **special case of a digital net in base** b. 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 $b = 2$.

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.

A PLR is actually a **special case of a digital net in base** b. 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.

More Examples

Example: Pricing a financial derivative.

Market price of some asset (e.g., one share of a stock) evolves in time as stochastic process $\{S(t),\ t \geq 0\}$ with (supposedly) known probability law (estimated from data).

A financial contract gives owner net payoff $g(S(t_1), \ldots, S(t_d))$ at time $T = t_d$, where $g : \mathbb{R}^d \to \mathbb{R}$, and $0 \leq t_1 < \cdots < t_d$ are fixed observation times.

Under a no-arbitrage assumption, present value of contract at time 0, when $S(0) = s_0$, is

$$v(s_0, T) = \mathbb{E}^* \left[e^{-rT} g(S(t_1), \ldots, S(t_d)) \right],$$

where $\mathbb{E}^*$ is under a risk-neutral measure and e^{-rT} is the discount factor.

This expectation can be written as an integral over $[0,1)^s$ for some s, and estimated by the average of n i.i.d. replicates of $X = e^{-rT} g(S(t_1), \ldots, S(t_d))$.

A simple model for S: geometric Brownian motion (GBM):

$$S(t) = s_0 e^{X(t)} \qquad \text{where} \qquad X(t) = (r - \sigma^2/2)t + \sigma B(t),$$

r is the interest rate, σ is the volatility, and $B(\cdot)$ is a standard Brownian motion: for any $t_2 > t_1 \geq 0$, $B(t_2) - B(t_1) \sim \mathsf{N}(0, t_2 - t_1)$, and the increments over disjoint intervals are independent.

A simple model for S: geometric Brownian motion (GBM):

$$S(t) = s_0 e^{X(t)} \qquad \text{where} \qquad X(t) = (r - \sigma^2/2)t + \sigma B(t),$$

r is the interest rate, σ is the volatility, and $B(\cdot)$ is a standard Brownian motion: for any $t_2 > t_1 \geq 0$, $B(t_2) - B(t_1) \sim \mathsf{N}(0, t_2 - t_1)$, and the increments over disjoint intervals are independent.

Here, the vector $\boldsymbol{Y} = (X(t_1), \ldots, X(t_d))$ has a multivariate normal distribution with some mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

To generate $\boldsymbol{Y}$, we decompose $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^{\mathsf{t}}$, and put $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{Z}$ where $\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

Possible decompositions: Cholesky, Brownian bridge, PCA, etc.
Choice does not matter for MC but does matter for RQMC.

A simple model for S: geometric Brownian motion (GBM):

$$S(t) = s_0 e^{X(t)} \qquad \text{where} \qquad X(t) = (r - \sigma^2/2)t + \sigma B(t),$$

r is the interest rate, σ is the volatility, and $B(\cdot)$ is a standard Brownian motion: for any $t_2 > t_1 \geq 0$, $B(t_2) - B(t_1) \sim \mathsf{N}(0, t_2 - t_1)$, and the increments over disjoint intervals are independent.

Here, the vector $\boldsymbol{Y} = (X(t_1), \ldots, X(t_d))$ has a multivariate normal distribution with some mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

To generate $\boldsymbol{Y}$, we decompose $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^{\mathsf{t}}$, and put $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{Z}$ where $\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

Possible decompositions: Cholesky, Brownian bridge, PCA, etc.
Choice does not matter for MC but does matter for RQMC.

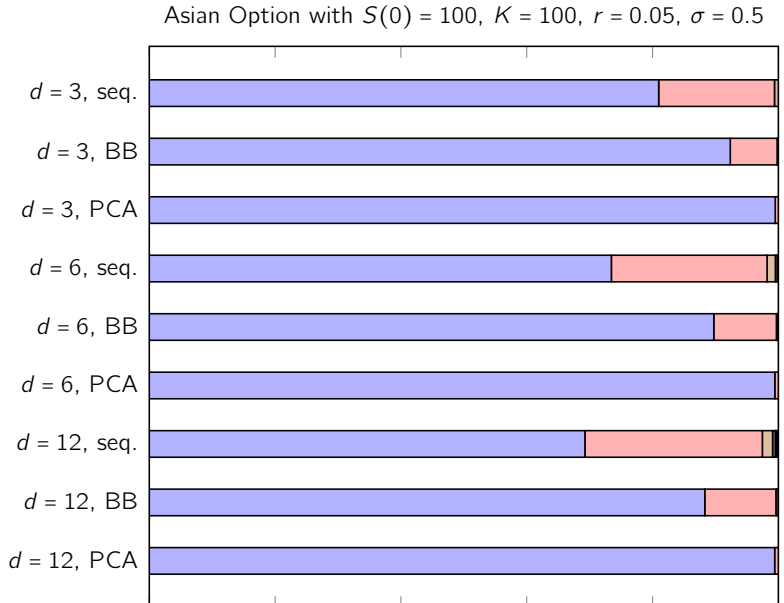
More general: $X(t)$ can also be a vector in c dimensions, so $s = cd$.
This can apply more generally to a function of a multinormal vector, in many areas.

Example of contract: Discretely-monitored **Asian call option**:

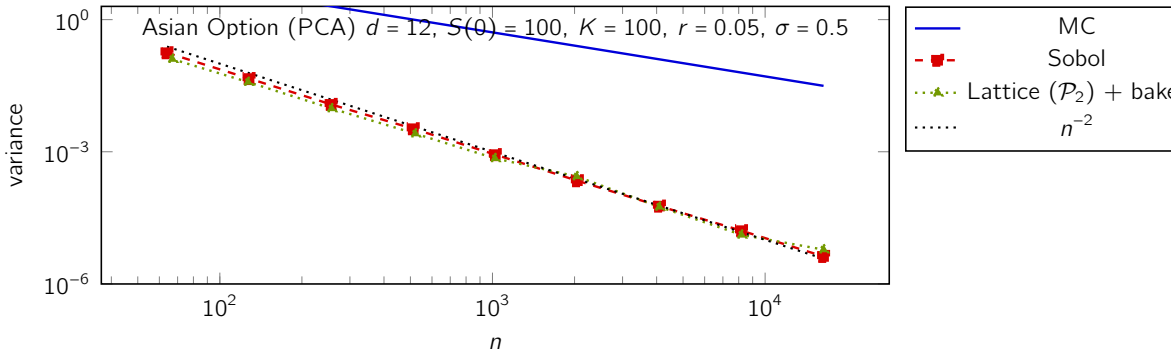
$$g(S(t_1), \ldots, S(t_d)) = \max\left(0, \frac{1}{d}\sum_{j=1}^{d} S(t_j) - K\right).$$

Numerical illustration: $s = d = 12$, $T = 1$ (one year), $t_j = j/12$ for $j = 0, \ldots, 12$, $K = 100$, $s_0 = 100$, $r = 0.05$, $\sigma = 0.5$.

ANOVA Variances for ordinary Asian Option



Variance with good lattices rules and Sobol points



Example: Pricing an Asian basket financial option

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

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

Example: Pricing an Asian basket financial option

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

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

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

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

This gives a 250-dimensional integration problem.

Let $\rho_{i,j} = 0.4$ for all $i \neq j$ (correlations between assets), $T = 1$, $\sigma_i = 0.1 + 0.4(i-1)/9$ for all i, $r = 0.04$, $S(0) = 100$, and $K = 100$. (Imai and Tan 2002; see L. 2009 for more details).

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

This gives a 250-dimensional integration problem.

Let $\rho_{i,j} = 0.4$ for all $i \neq j$ (correlations between assets), $T = 1$, $\sigma_i = 0.1 + 0.4(i-1)/9$ for all i, $r = 0.04$, $S(0) = 100$, and $K = 100$. (Imai and Tan 2002; see L. 2009 for more details).

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

Korobov Lattice Rules

	$n = 16381$		$n = 65521$		$n = 262139$	
	$a = 5693$		$a = 944$		$a = 21876$	
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. So Koksma-Hlawka does not apply.

Many other types of successful applications of RQMC, for example:

Statistics: estimate and optimize the likelihood function for given data.

Computer graphics: estimate the color of a pixel.

Service systems with random arrival rates.

Stochastic differential equations (e.g., in finance).

PDEs with random coefficients.

Etc.

RQMC for Markov chains

Array-RQMC for Markov Chains

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

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

where the $\boldsymbol{U}_j$ 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 for Markov Chains

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

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

where the $\boldsymbol{U}_j$ 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.

Array-RQMC insight: To simplify, suppose $X_j \sim U(0,1)^\ell$. We estimate

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

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

This is (roughly) RQMC with the point set $Q_n = \{(X_{i,j-1}, \boldsymbol{U}_{i,j}),\ 0 \le i < n\}$.

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

Array-RQMC insight: To simplify, suppose $X_j \sim U(0,1)^\ell$. We estimate

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

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

This is (roughly) RQMC with the point set $Q_n = \{(X_{i,j-1}, \boldsymbol{U}_{i,j}),\ 0 \le i < n\}$.

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

However, we do not choose the $X_{i,j-1}$'s in Q_n: they come from the simulation.

Array-RQMC insight: To simplify, suppose $X_j \sim U(0,1)^\ell$. We estimate

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

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

This is (roughly) RQMC with the point set $Q_n = \{(X_{i,j-1}, \boldsymbol{U}_{i,j}),\ 0 \le i < n\}$.
We want Q_n to have low discrepancy (LD) over $[0,1)^{\ell+d}$.
However, we do not choose the $X_{i,j-1}$'s in Q_n: they come from the simulation.

Idea: select RQMC point set $\tilde{Q}_n = \{(\boldsymbol{w}_0, \boldsymbol{U}_{0,j}), \ldots, (\boldsymbol{w}_{n-1}, \boldsymbol{U}_{n-1,j})\}$, where the $\boldsymbol{w}_i \in [0,1)^\ell$ are fixed and each $\boldsymbol{U}_{i,j} \sim U(0,1)^d$.
Permute the states $X_{i,j-1}$ so that $X_{\pi_j(i),j-1}$ is "close" to $\boldsymbol{w}_i$ for each i and compute $X_{i,j} = \varphi_j(X_{\pi_j(i),j-1}, \boldsymbol{U}_{i,j})$ for each i.

Example: If $\ell = 1$, can take $\boldsymbol{w}_i = (i + 0.5)/n$ 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 $\{\boldsymbol{U}_{0,j}, \ldots, \boldsymbol{U}_{n-1,j}\}$ in $\tilde{Q}_n$;
  $X_{i,j} = \varphi_j(X_{\pi_j(i),j-1}, \boldsymbol{U}_{i,j})$, 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

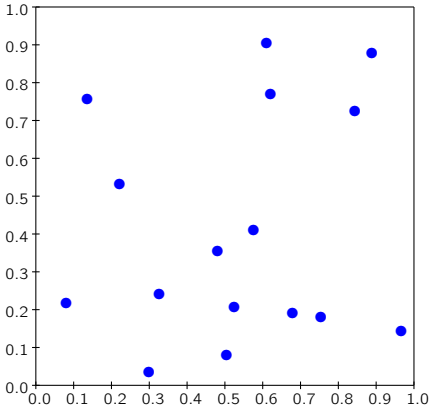
$X_{i,0} \leftarrow x_0$ (or $X_{i,0} \leftarrow x_{i,0}$) for $i = 0, \ldots, n-1$;
for $j = 1, 2, \ldots, \tau$ **do**
  Compute the permutation π_j of the states (for matching);
  Randomize afresh $\{\boldsymbol{U}_{0,j}, \ldots, \boldsymbol{U}_{n-1,j}\}$ in $\tilde{Q}_n$;
  $X_{i,j} = \varphi_j(X_{\pi_j(i),j-1}, \boldsymbol{U}_{i,j})$, 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 $\bar{Y}_n = \hat{\mu}_{\text{arqmc},n} = \sum_{j=1}^{\tau} \hat{\mu}_{\text{arqmc},j,n}$.

Proposition: The average $\bar{Y}_n$ is an unbiased estimator of μ.

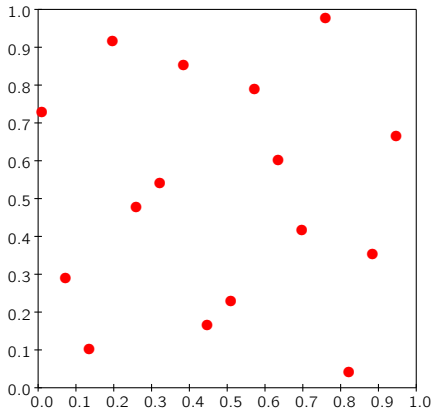
Can estimate $\mathrm{Var}[\bar{Y}_n]$ by making m independent replications.

Example of 2-dim batch sort: a (4,4) mapping

States of the chains

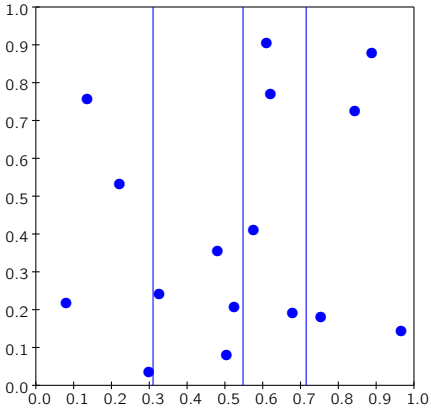


Sobol' net in 2 dimensions after random digital shift

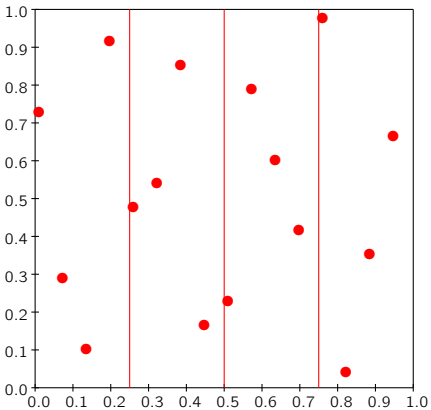


Example of 2-dim batch sort: a (4,4) mapping

States of the chains

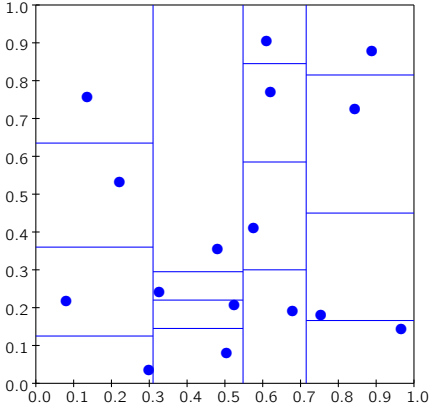


Sobol' net in 2 dimensions after random digital shift

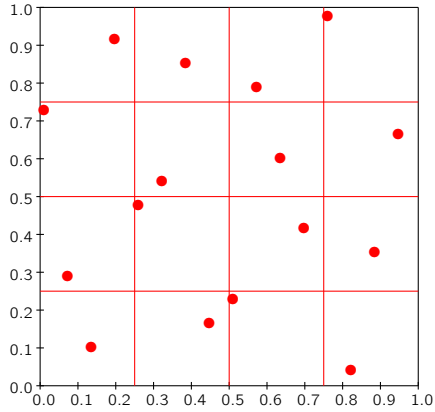


Example of 2-dim batch sort: a (4,4) mapping

States of the chains

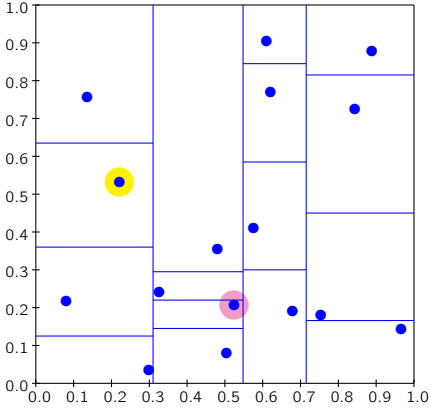


Sobol' net in 2 dimensions after random digital shift

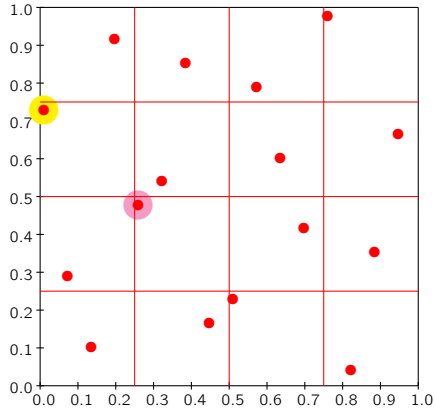


Example of 2-dim batch sort: a (4,4) mapping

States of the chains



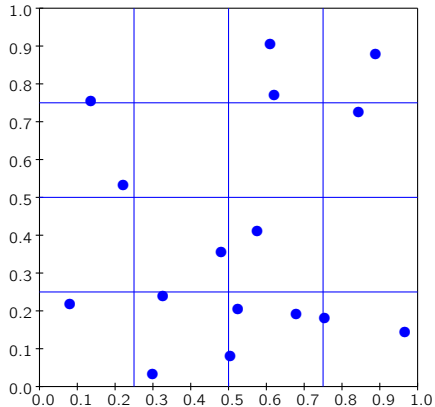
Sobol' net in 2 dimensions after random digital shift



Hilbert curve sort

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

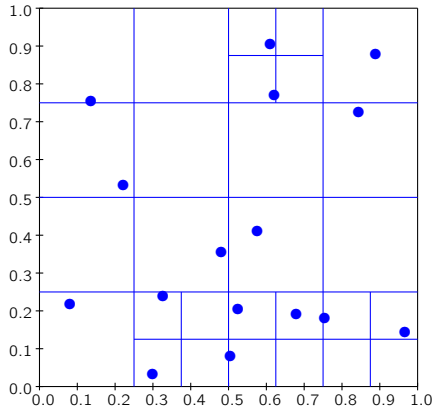
States of the chains



Hilbert curve sort

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

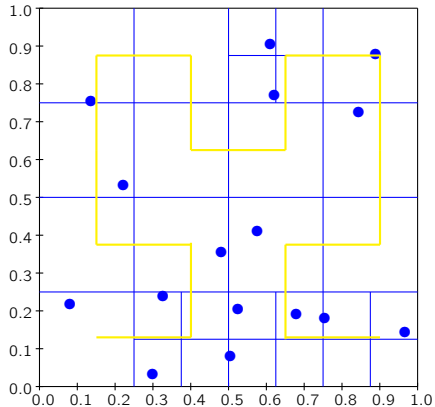
States of the chains



Hilbert curve sort

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

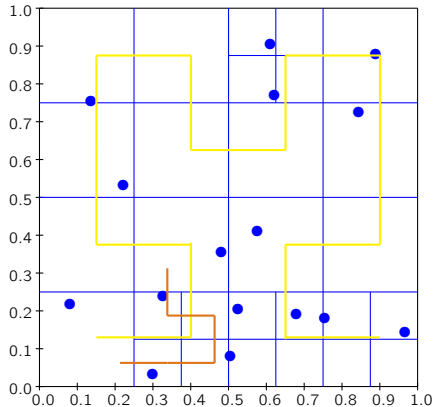
States of the chains



Hilbert curve sort

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

States of the chains



Convergence results and proofs

For $\ell = 1$, $\mathcal{O}(n^{-3/2})$ variance has been proved under some conditions.

For $\ell > 1$, worst-case error of $\mathcal{O}(n^{-1/(\ell+1)})$ has been proved in deterministic settings under strong conditions on φ_j, using a batch sort (El Haddad, Lécot, L'Ecuyer 2008, 2010).

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

A very simple example, 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 $X_{0,j}, \ldots, X_{n-1,j}$.

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

We have

$$\mathbb{E}[\mathcal{D}_j^2] \le \frac{n^{-3/2}}{4(1-\rho)} = \frac{1-\theta}{4(1-2\theta)} n^{-3/2}.$$

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 $\log_2 \mathrm{Var}[\bar{Y}_{n,j}]$ vs $\log_2 n$, for various g_j

We also looked at $\mathbb{E}[\mathcal{D}_j^2]$ and $\mathbb{E}[\mathcal{P}_\alpha]$ 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 $\log_2 n$	$\log_2 \mathbb{E}[\mathcal{D}_j^2]$	$\log_2 \mathrm{Var}[\bar{Y}_{n,j}]$			
		$X_j - \frac{1}{2}$	$X_j^2 - \frac{1}{3}$	$(X_j - \frac{1}{2})^+ - \frac{1}{8}$	$\mathbb{I}[X_j \le \frac{1}{3}] - \frac{1}{3}$
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

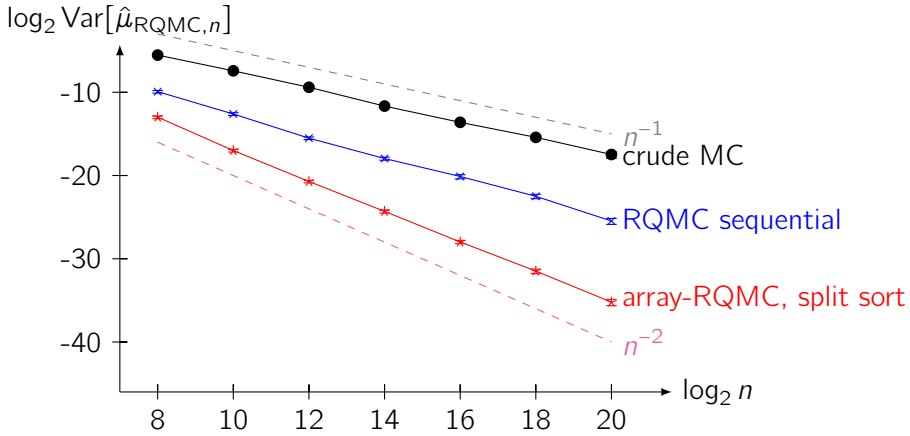
	$-\log_{10} \mathrm{Var}[\bar{Y}_{n,j}]$ for $n = 2^{21}$			CPU time (sec)
	$X_j^2 - \frac{1}{3}$	$(X_j - \frac{1}{2})^+ - \frac{1}{8}$	$\mathbb{I}[X_j \le \frac{1}{3}] - \frac{1}{3}$	
MC	7.35	7.86	6.98	270
LHS	8.82	8.93	7.61	992
SS	13.73	14.10	10.20	2334
SSA	18.12	17.41	10.38	1576
Sobol	19.86	17.51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

Example: Pricing an 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

$\mathrm{Var}[\bar{Y}_{n,j}] \approx \mathcal{O}(n^{\alpha})$.
VRF compared with MC, for $n = 2^{20}$.
CPU time for $m = 100$ replications.

Sort	RQMC points	α	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

Array-RQMC for a system of biological and chemical reactions

[Puchhammer, Ben Abdellah, and L. 2022]

We have d types of molecules and K types of reactions:

$$\alpha_{1,k} S_1 + \cdots + \alpha_{d,k} S_d \xrightarrow{c_k} \beta_{1,k} S_1 + \cdots + \beta_{d,k} S_d, \quad \alpha_{i,k}, \beta_{i,k} \in \mathbb{N}_0, \ k = 1, \ldots, K.$$

Let $\boldsymbol{X}(t) = (X_1(t), \ldots, X_d(t))$ where $X_k(t)$ is the number of molecules of type k at time t. The process $\{\boldsymbol{X}(t), t \geq 0\}$ is modeled as a continuous-time Marlov chain (CTMC).

The propensity (or occurrence rate) of reaction k at time t is $a_k(\boldsymbol{X}(t)) = c_k h_k(\boldsymbol{X}(t))$.

Array-RQMC for a system of biological and chemical reactions

[Puchhammer, Ben Abdellah, and L. 2022]

We have d types of molecules and K types of reactions:

$$\alpha_{1,k} S_1 + \cdots + \alpha_{d,k} S_d \xrightarrow{c_k} \beta_{1,k} S_1 + \cdots + \beta_{d,k} S_d, \quad \alpha_{i,k}, \beta_{i,k} \in \mathbb{N}_0, \ k = 1, \ldots, K.$$

Let $\boldsymbol{X}(t) = (X_1(t), \ldots, X_d(t))$ where $X_k(t)$ is the number of molecules of type k at time t. The process $\{\boldsymbol{X}(t), t \geq 0\}$ is modeled as a continuous-time Marlov chain (CTMC).

The propensity (or occurrence rate) of reaction k at time t is $a_k(\boldsymbol{X}(t)) = c_k h_k(\boldsymbol{X}(t))$.

We discretize the time and simulate the discrete-time chain to estimate say $\mathbb{E}[g(\boldsymbol{X}(T)]$ for a given time T.

For Array-RQMC, finding good sorting methods is an important issue.

Example with 6 molecule types and 6 reactions types

$$\text{PKA} + 2\text{cAMP} \underset{c_2}{\overset{c_1}{\rightleftarrows}} \text{PKA-cAMP}_2,$$

$$\text{PKA-cAMP}_2 + 2\text{cAMP} \underset{c_4}{\overset{c_3}{\rightleftarrows}} \text{PKA-cAMP}_4,$$

$$\text{PKA-cAMP}_4 \underset{c_6}{\overset{c_5}{\rightleftarrows}} \text{PKAr} + 2\text{PKAc}.$$

We simulate 256 time steps.
The state has 6 dimensions, and we need 6 random numbers per step.

Classical RQMC requires $256 \times 6 =$ **1536**-dimensional points.
Array-RQMC requires **7** or **12**-dimensional RQMC points.

Example with 6 molecule types and 6 reactions types

Here we estimate the expected number of PKAr molecules at time T.

Estimated convergence rate $\hat{\beta}$ and variance reduction factor for $n = 2^{19}$ points:

Sort	Sampling	$\hat{\beta}$	vrf19	eif19
	MC	1.03	1	1
	RQMC	1.17	39	45
OSLAIF	Latice+shift	1.42	3634	1550
	Sobol+LMS+ds	1.47	2062	1059
Batch-sort-6	Latice+shift	1.62	3026	1560
	Sobol+LMS+ds	1.46	2753	1749

Array-RQMC: 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.

Array-RQMC: Convergence results

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.

Array-RQMC: Some applications and examples

L., Lécot, and Tuffin [2008]: Several examples.

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.

Ben Abdellah, L., Puchhammer [2019]: Option Pricing Under Stochastic Volatility Models.

Puchhammer, Ben Abdellah, L. [2020]: Simulation of Biological and Chemical Reactions.

Arnold, Chen, Sha [2021]: Policy Gradient Methods for Reinforcement Learning.

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 ...
- **Software support still very limited in general!**