Simulation de chaines de Markov: briser le mur de la convergence en $n^{-1/2}$

Pierre L'Ecuyer

Université de Montréal — DIRO — GERAD GROUP FOR RESEARCH IN DECISION ANALYSIS — CIRRELT — Inria informatics mathematics

En collaboration avec: Amal Ben Abdellah, Christian Lécot, David Munger, Art B. Owen, et Bruno Tuffin

DIRO, Université de Montréal, Mars 2017

Markov chain setting

A Markov chain with state space $\mathcal{X}$ 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 $(0,1)^d$.

Payoff (or cost) at step $j = \tau$:

$$Y = g(X_\tau).$$

for some fixed time step τ.

Markov chain setting

A Markov chain with state space $\mathcal{X}$ 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 $(0,1)^d$.

Payoff (or cost) at step $j = \tau$:

$$Y = g(X_\tau).$$

for some fixed time step τ.

We may want to estimate

$$\mu = \mathbb{E}[Y],$$

or some other functional of Y, or perhaps the entire distribution of Y.

Baby example: a small finite Markov chain

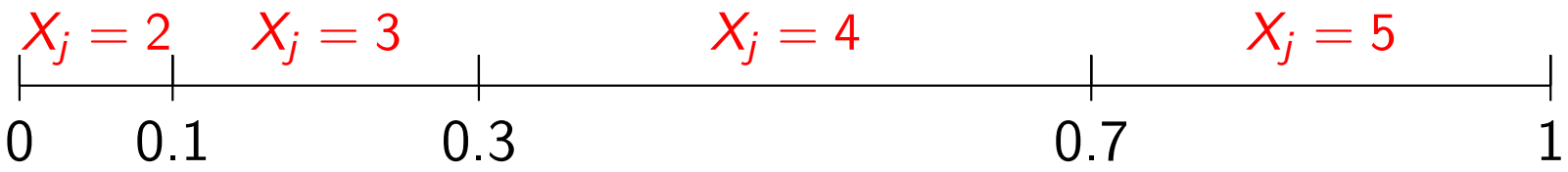
State space $\mathcal{X} = \{0, 1, \ldots, k-1\}$, $X_0 = 0$,
transition probability matrix $\mathbf{P} = (p_{x,y})$,
$p_{x,y} = \mathbb{P}[X_j = y \mid X_{j-1} = x]$ for $0 \leq x, y < k$.

Baby example: a small finite Markov chain

State space $\mathcal{X} = \{0, 1, \ldots, k-1\}$, $X_0 = 0$,
transition probability matrix $\mathbf{P} = (p_{x,y})$,
$p_{x,y} = \mathbb{P}[X_j = y \mid X_{j-1} = x]$ for $0 \le x, y < k$.

Exemple: $k = 6$ and $\mathbf{P} = \begin{pmatrix} 0.1 & 0.2 & 0.4 & 0.1 & 0.2 & 0.0 \\ 0.2 & 0.1 & 0.2 & 0.3 & 0.0 & 0.2 \\ 0.0 & 0.0 & 0.1 & 0.2 & 0.4 & 0.3 \\ 0.2 & 0.3 & 0.1 & 0.1 & 0.2 & 0.1 \\ 0.0 & 0.2 & 0.4 & 0.2 & 0.2 & 0.0 \\ 0.0 & 0.2 & 0.1 & 0.1 & 0.2 & 0.4 \end{pmatrix}$.

To simulate, e.g., if $X_{j-1} = 2$, generate $U \sim U(0,1)$ and find X_j:

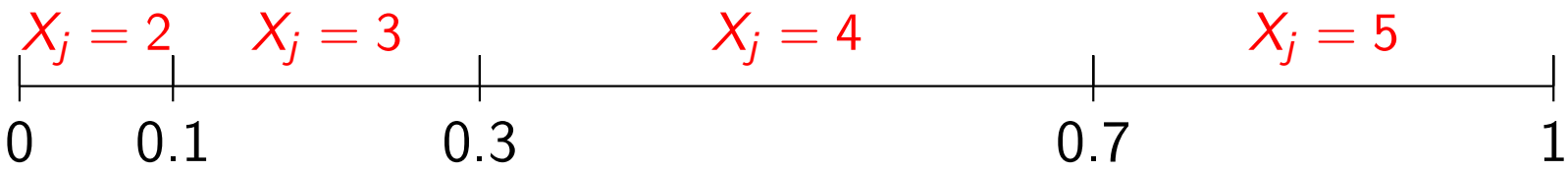


Baby example: a small finite Markov chain

State space $\mathcal{X} = \{0, 1, \ldots, k-1\}$, $X_0 = 0$,
transition probability matrix $\mathbf{P} = (p_{x,y})$,
$p_{x,y} = \mathbb{P}[X_j = y \mid X_{j-1} = x]$ for $0 \le x, y < k$.

Exemple: $k = 6$ and $\mathbf{P} = \begin{pmatrix} 0.1 & 0.2 & 0.4 & 0.1 & 0.2 & 0.0 \\ 0.2 & 0.1 & 0.2 & 0.3 & 0.0 & 0.2 \\ 0.0 & 0.0 & 0.1 & 0.2 & 0.4 & 0.3 \\ 0.2 & 0.3 & 0.1 & 0.1 & 0.2 & 0.1 \\ 0.0 & 0.2 & 0.4 & 0.2 & 0.2 & 0.0 \\ 0.0 & 0.2 & 0.1 & 0.1 & 0.2 & 0.4 \end{pmatrix}$.

To simulate, e.g., if $X_{j-1} = 2$, generate $U \sim U(0,1)$ and find X_j:



We can simulate the chain for τ steps, repeat n times, and estimate $\pi_x = \mathbb{P}[X_\tau = x]$ by $\hat{\pi}_x$, the proportion of times where $X_\tau = x$.

Ordinary Monte Carlo simulation

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

$$\hat{\mu}_n = \frac{1}{n}\sum_{i=0}^{n-1} Y_i \qquad \text{where } Y_i = g(X_{i,\tau}).$$

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

The width of a confidence interval on μ converges as $\mathcal{O}(n^{-1/2})$.
That is, for each additional digit of accuracy, one must multiply n by 100.

Ordinary Monte Carlo simulation

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

$$\hat{\mu}_n = \frac{1}{n}\sum_{i=0}^{n-1} Y_i \qquad \text{where } Y_i = g(X_{i,\tau}).$$

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

The width of a confidence interval on μ converges as $\mathcal{O}(n^{-1/2})$.
That is, for each additional digit of accuracy, one must multiply n by 100.

Can also estimate the distribution (density) of Y by the empirical distribution of $Y_0, \ldots, Y_{n-1}$, or by an histogram (perhaps smoothed), or by a kernel density estimator. The mean integrated square error (MISE) for the density typically converges as $\mathcal{O}(n^{-2/3})$ for an histogram and $\mathcal{O}(n^{-4/5})$ for the best density estimators (e.g., ASH, KDE, ...).

Ordinary Monte Carlo simulation

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

$$\hat{\mu}_n = \frac{1}{n}\sum_{i=0}^{n-1} Y_i \qquad \text{where } Y_i = g(X_{i,\tau}).$$

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

The width of a confidence interval on μ converges as $\mathcal{O}(n^{-1/2})$.
That is, for each additional digit of accuracy, one must multiply n by 100.

Can also estimate the distribution (density) of Y by the empirical distribution of $Y_0, \ldots, Y_{n-1}$, or by an histogram (perhaps smoothed), or by a kernel density estimator. The mean integrated square error (MISE) for the density typically converges as $\mathcal{O}(n^{-2/3})$ for an histogram and $\mathcal{O}(n^{-4/5})$ for the best density estimators (e.g., ASH, KDE, ...).

Can we do better than those rates?

Plenty of applications fit this setting:

Finance

Queueing systems

Inventory, distribution, logistic systems

Reliability models

MCMC in Bayesian statistics

Many many more...

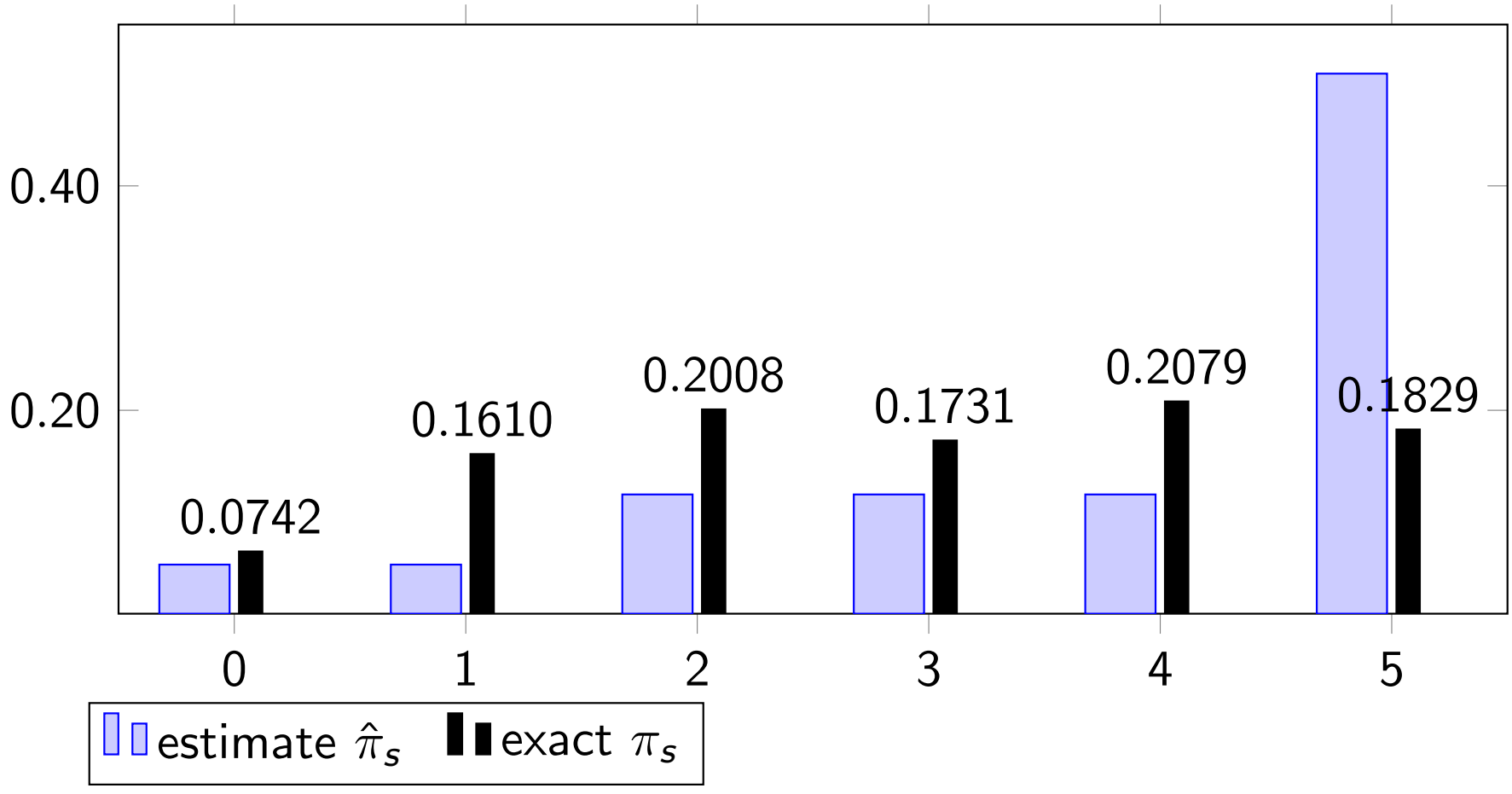
Baby example: a small finite Markov chain

Take $k = 6$, $X_0 = 0$, and $\mathbf{P} = \begin{pmatrix} 0.1 & 0.2 & 0.4 & 0.1 & 0.2 & 0.0 \\ 0.2 & 0.1 & 0.2 & 0.3 & 0.0 & 0.2 \\ 0.0 & 0.0 & 0.1 & 0.2 & 0.4 & 0.3 \\ 0.2 & 0.3 & 0.1 & 0.1 & 0.2 & 0.1 \\ 0.0 & 0.2 & 0.4 & 0.2 & 0.2 & 0.0 \\ 0.0 & 0.2 & 0.1 & 0.1 & 0.2 & 0.4 \end{pmatrix}$

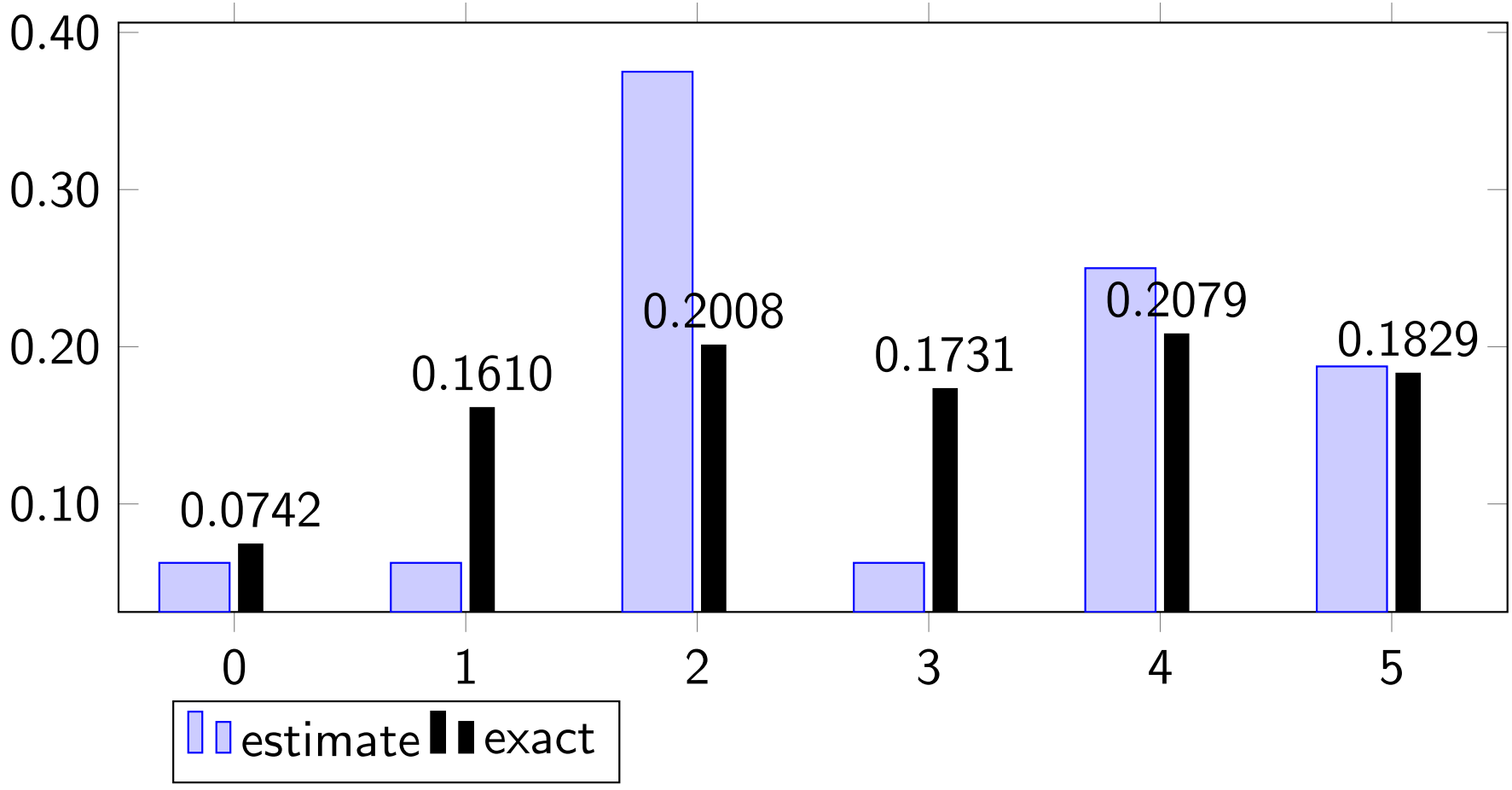
Suppose we want to estimate $\boldsymbol{\pi} = (\pi_0, \ldots, \pi_5)^{\mathrm{t}}$ where $\pi_x = \mathbb{P}[X_\tau = x]$.
We know $\boldsymbol{\pi} = \mathbf{P}^\tau \mathbf{e}_1$, but let us pretend we do not know.
We simulate the chain for τ steps, repeat n times, and estimate π_x by $\hat{\pi}_x$, the proportion of times where $X_\tau = x$.

For $\tau = 25$ steps, $\boldsymbol{\pi} = (0.0742, 0.1610, 0.2008, 0.1731, 0.2079, 0.1829)$.

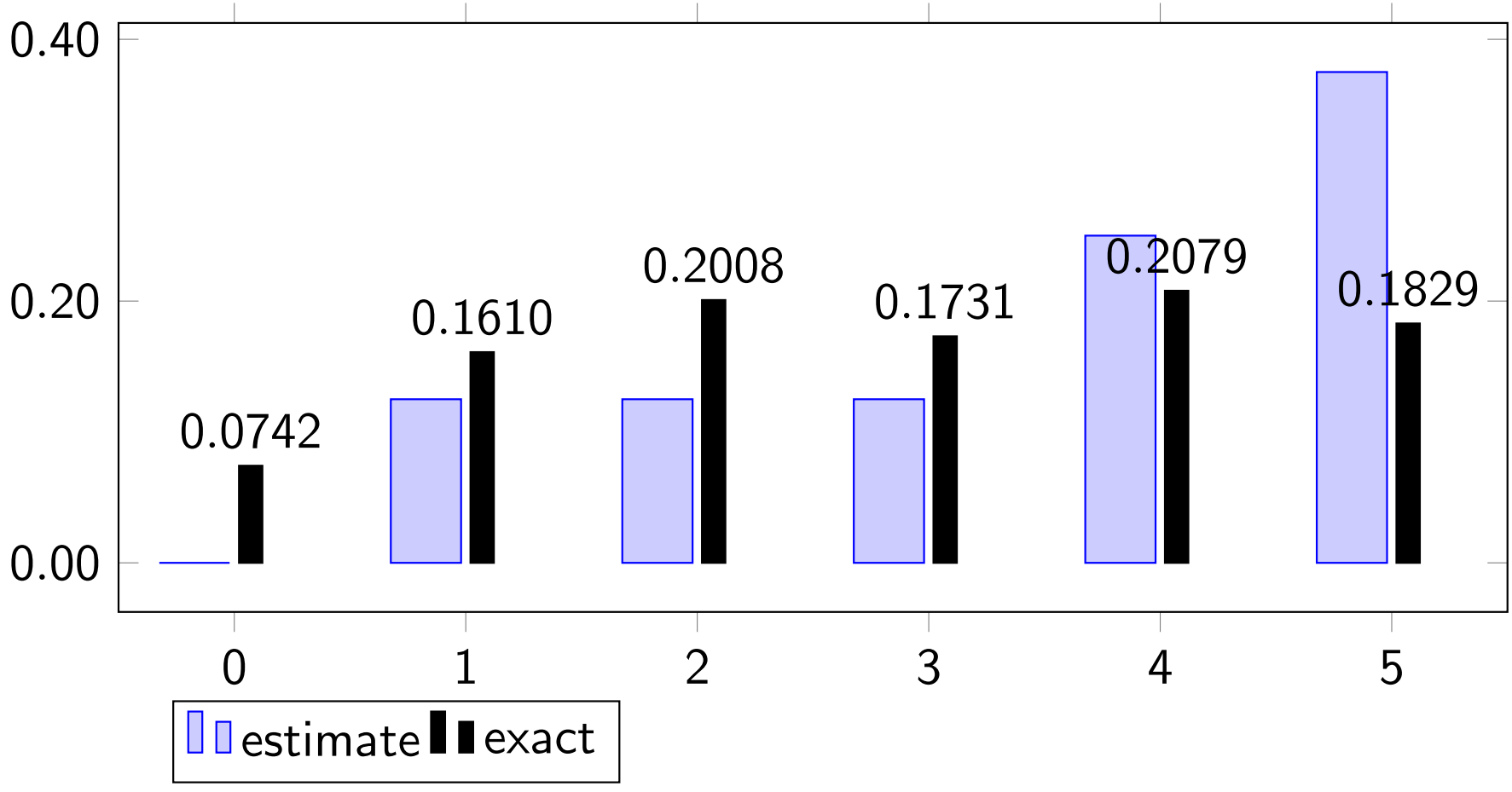
Monte Carlo simulation with $n = 16$, state after $\tau = 25$ steps:



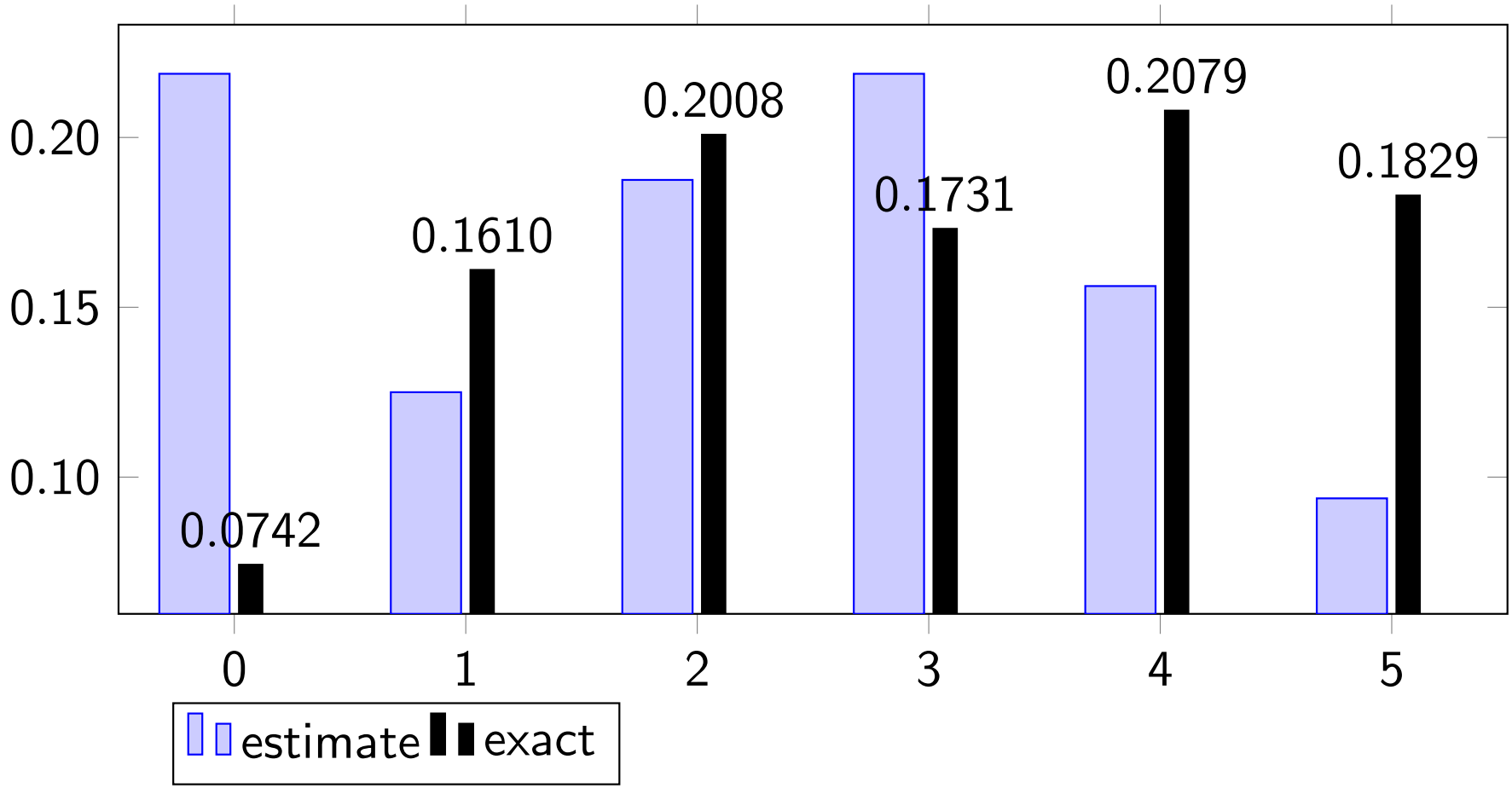
Monte Carlo simulation with $n = 16$, state after $\tau = 25$ steps:



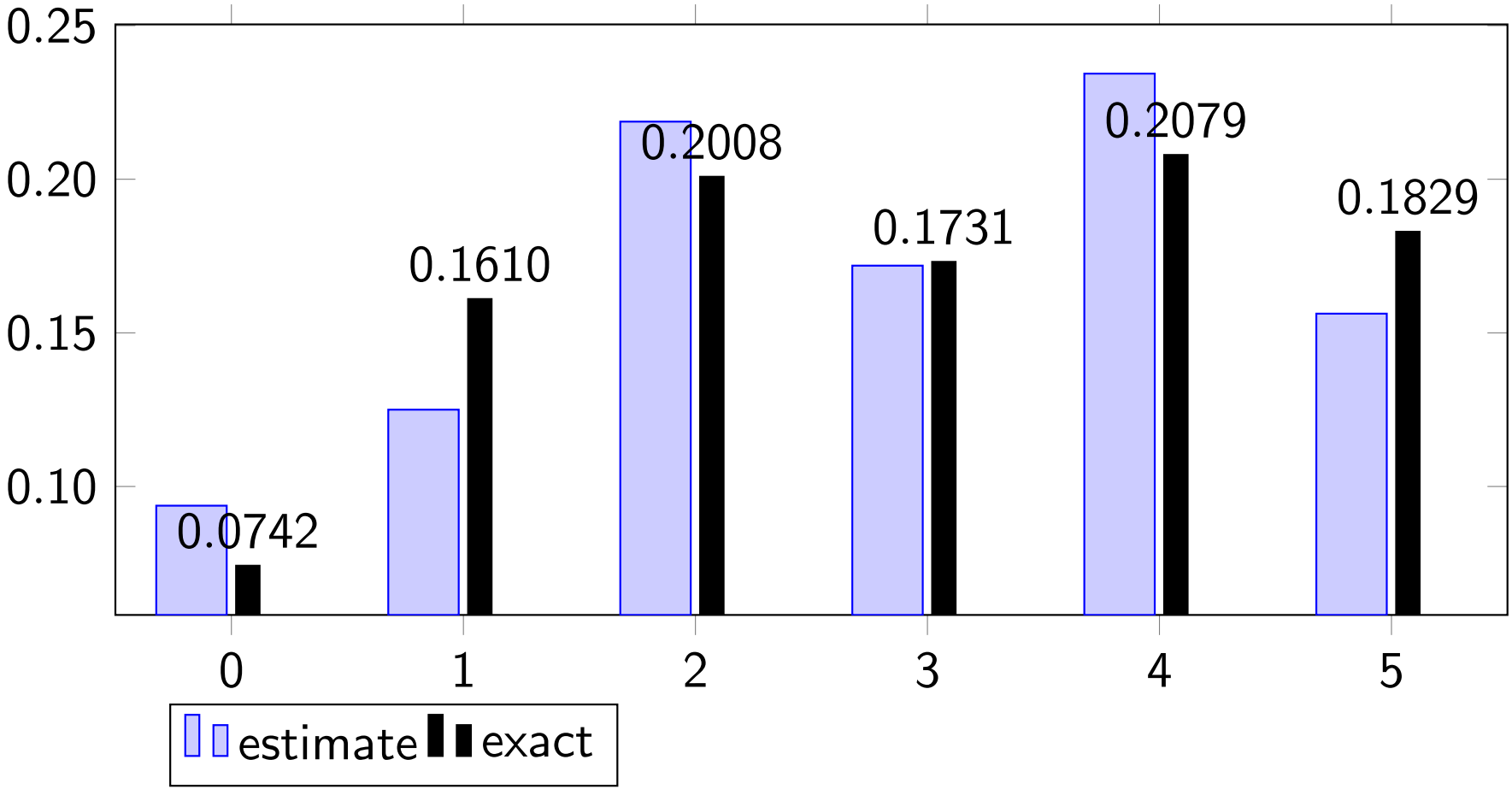
Monte Carlo simulation with $n = 16$, state after $\tau = 25$ steps:



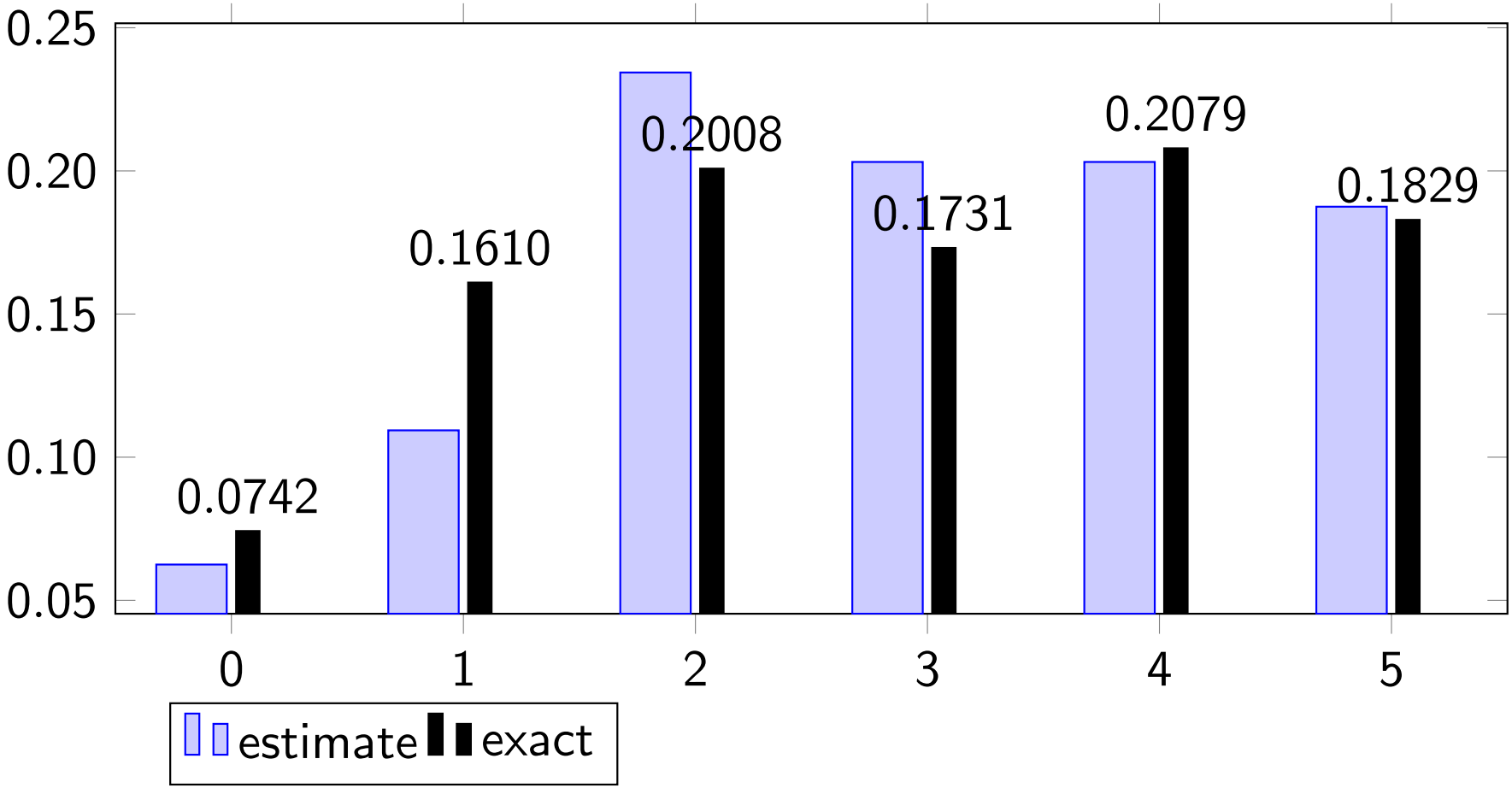
Monte Carlo simulation with $n = 32$, state after $\tau = 25$ steps:



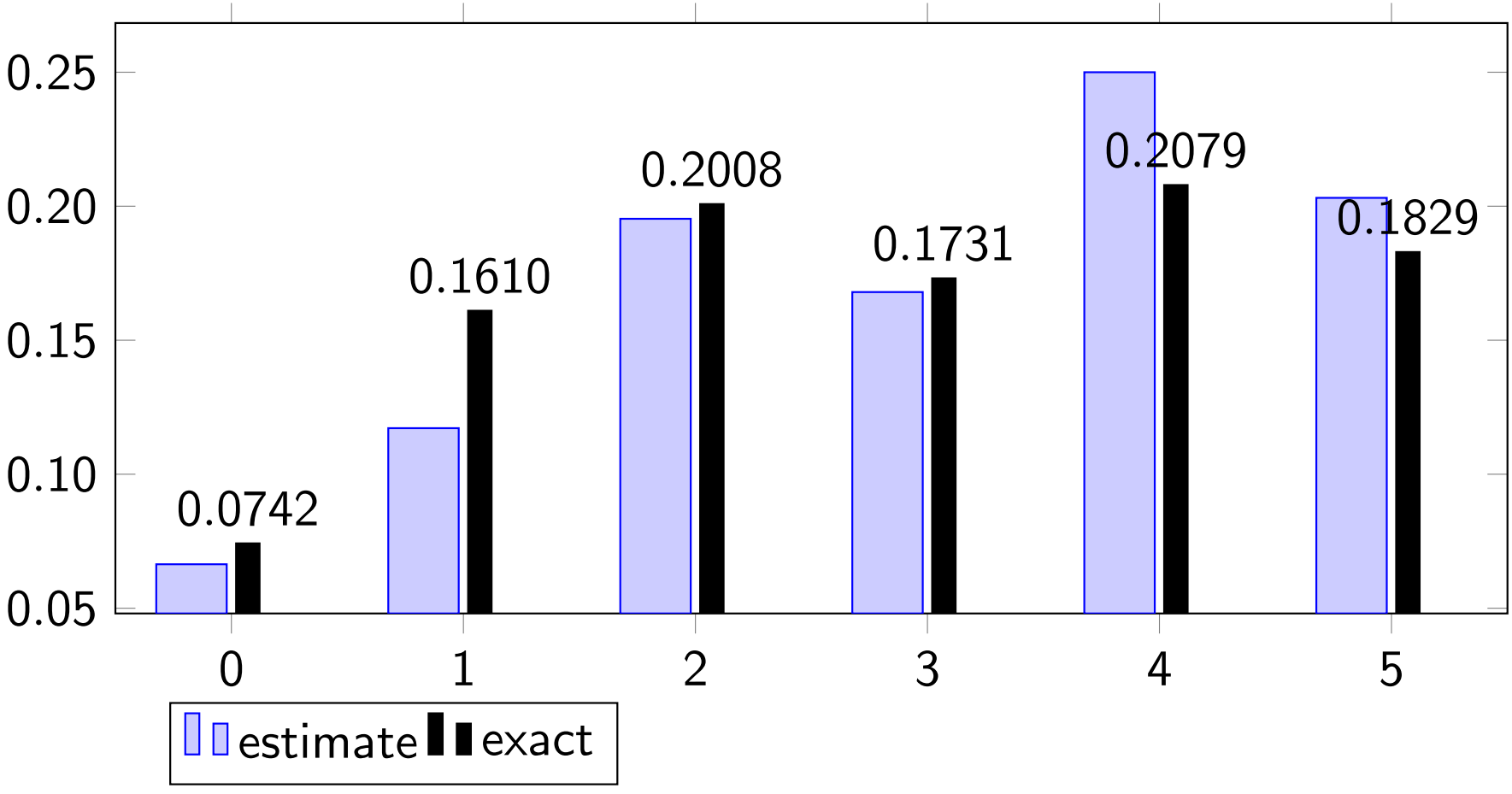
Monte Carlo simulation with $n = 64$, state after $\tau = 25$ steps:



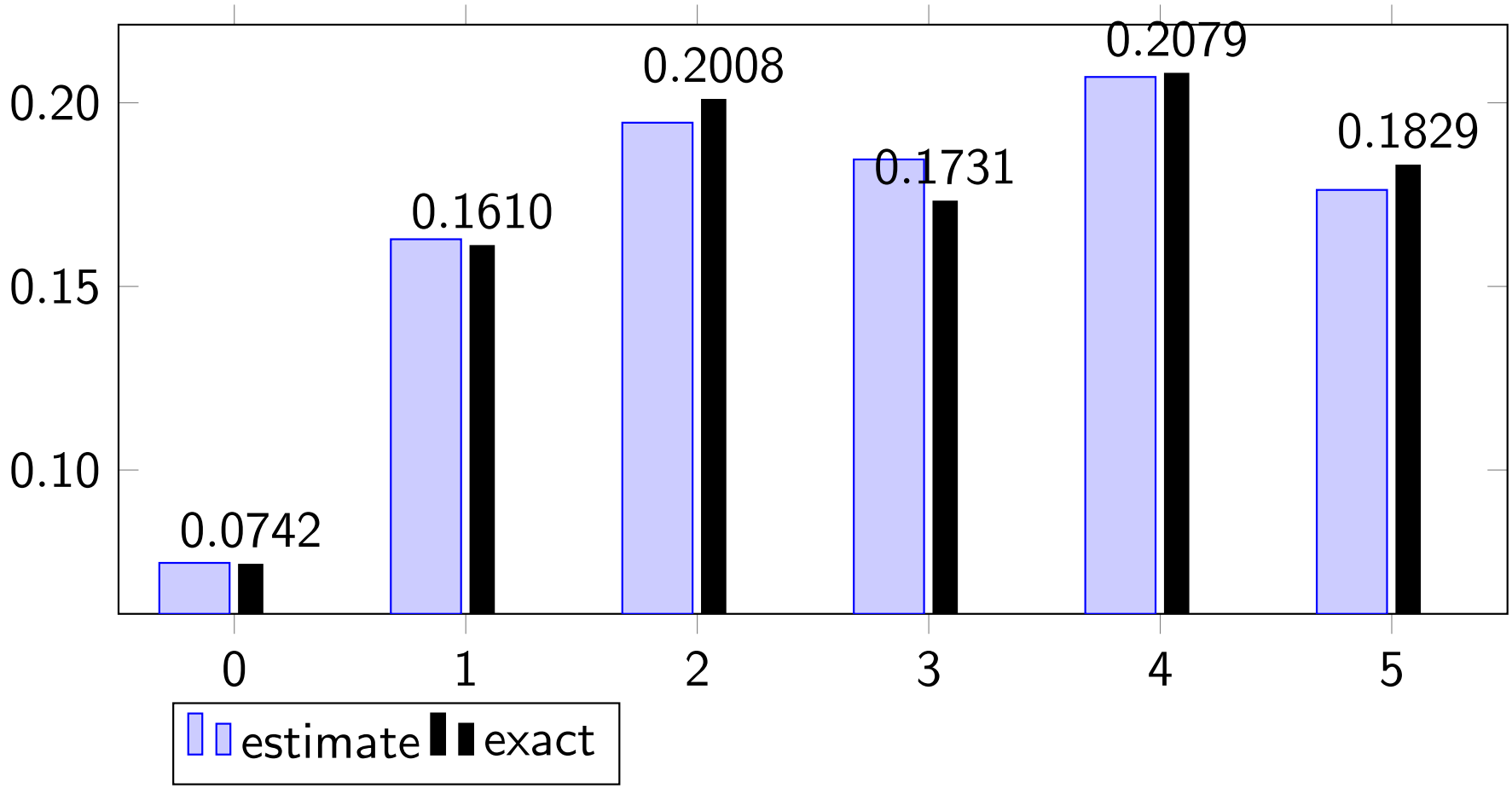
Monte Carlo simulation with $n = 128$, state after $\tau = 25$ steps:



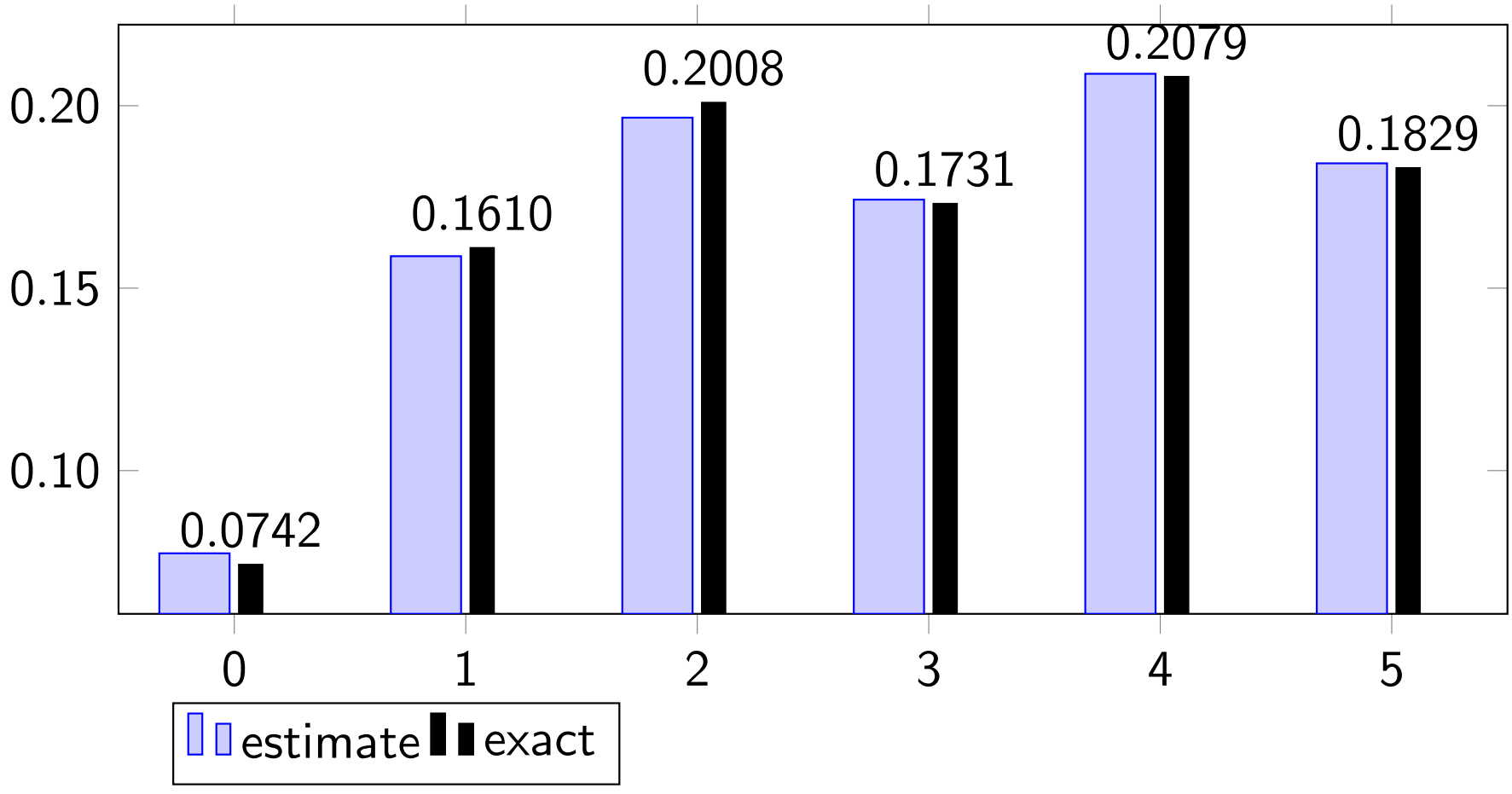
Monte Carlo simulation with $n = 256$, state after $\tau = 25$ steps:



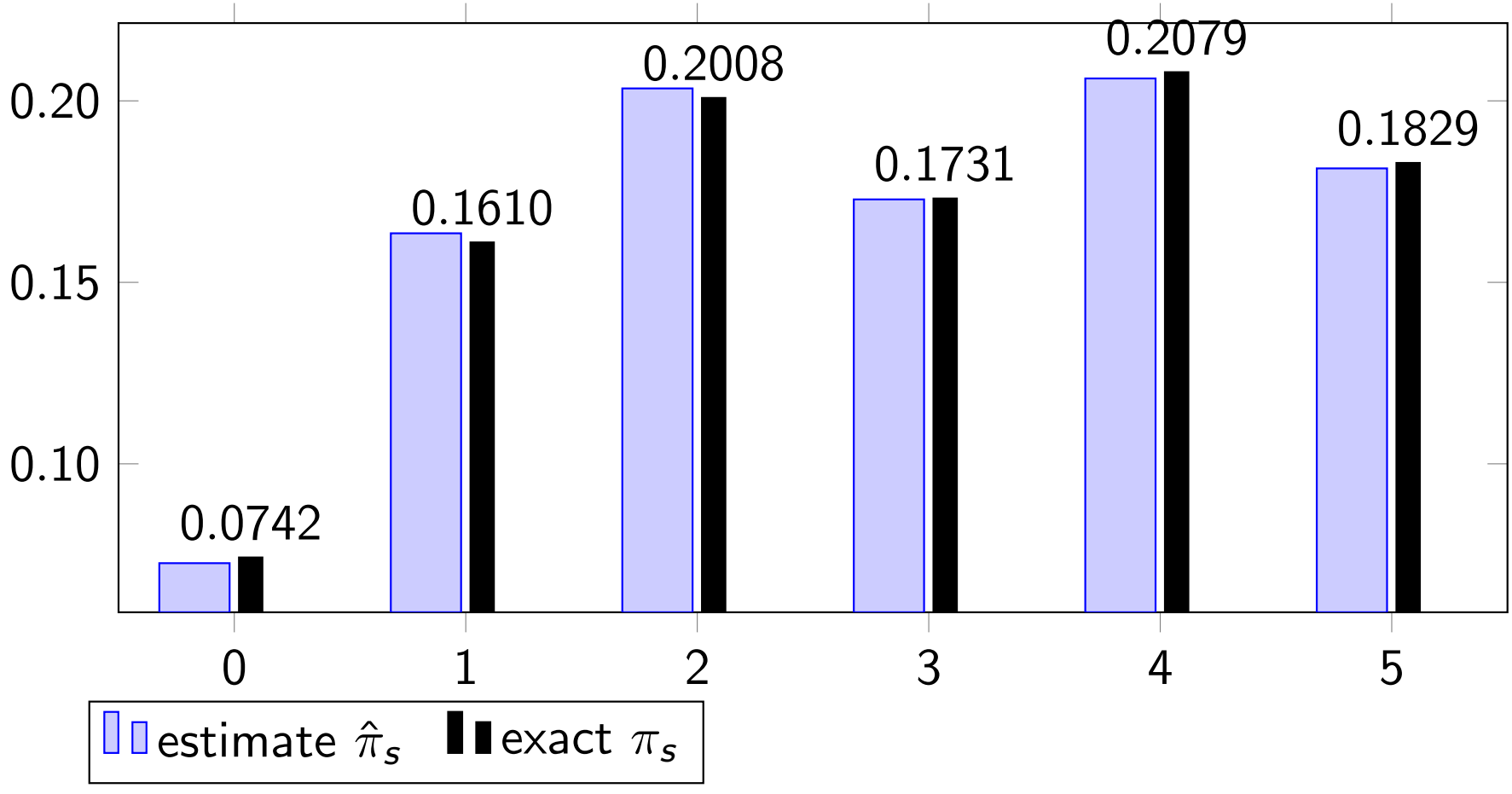
Monte Carlo simulation with $n = 4096$, state after $\tau = 25$ steps:



Monte Carlo simulation with $n = 16384$, state after $\tau = 25$ steps:



Monte Carlo simulation with $n = 16384$, state after $\tau = 25$ steps:



Mean integrated square error (MISE): $\frac{1}{6}\sum_{s=0}^{5}(\hat{\pi}_s - \pi_s)^2$.

With Monte Carlo, $\mathbb{E}[\mathrm{MISE}] = \mathcal{O}(1/n)$.

With Array-RQMC, $\mathbb{E}[\mathrm{MISE}] \approx \mathcal{O}(1/n^2)$.

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

Given $s_0 > 0$, $B(0) = 0$, and observation times $t_j = jh$ for $j = 1, \ldots, \tau$, let

$$\begin{aligned} B(t_j) &= B(t_{j-1}) + (r - \sigma^2/2)h + \sigma h^{1/2} Z_j, \\ S(t_j) &= s_0 \exp[B(t_j)], \qquad \text{(geometric Brownian motion)} \end{aligned}$$

where $U_j \sim U[0,1)$ and $Z_j = \Phi^{-1}(U_j) \sim N(0,1)$.

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

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

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

Transition:

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

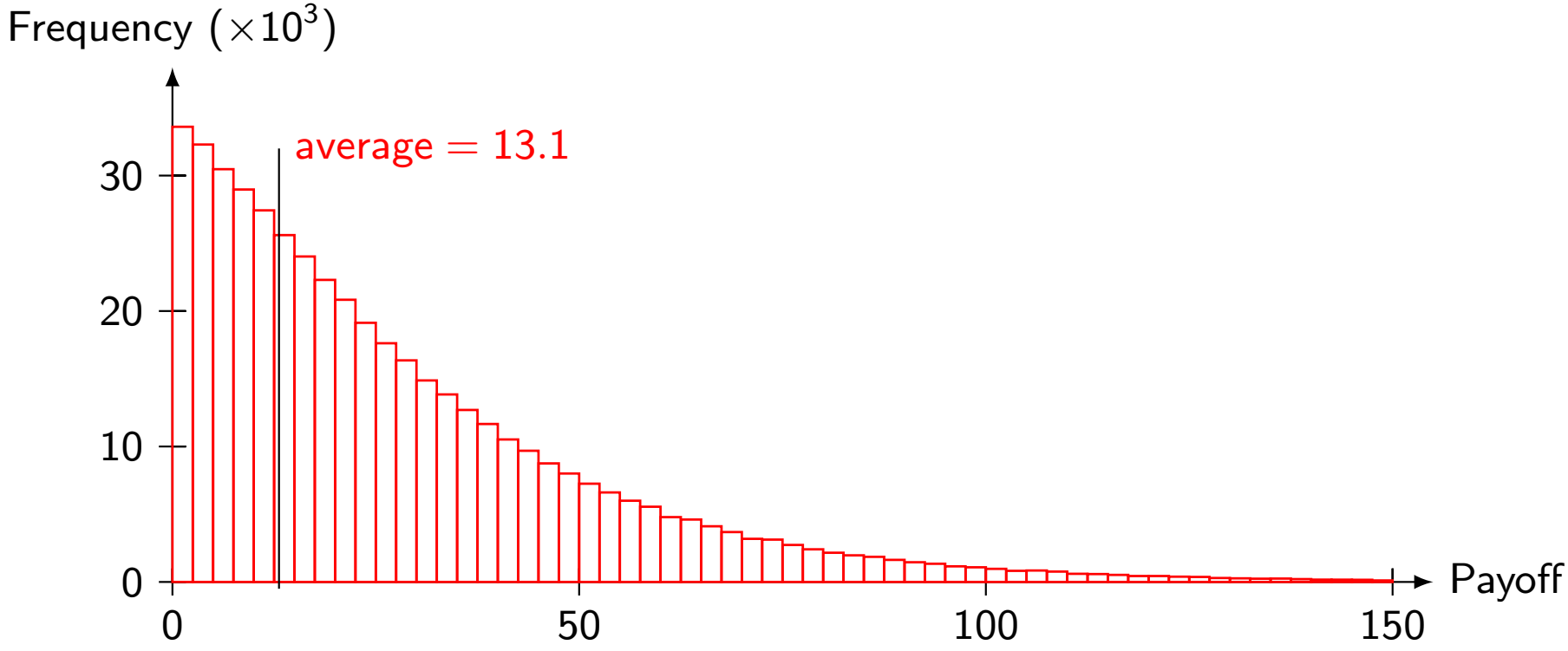
Want to estimate $\mathbb{E}[Y]$, or distribution of Y, etc.

Take $\tau = 12$, $T = 1$ (one year), $t_j = j/12$ for $j = 0, \ldots, 12$, $K = 100$, $s_0 = 100$, $r = 0.05$, $\sigma = 0.5$.

We make $n = 10^6$ independent runs. Mean: **13.1**. Max = 390.8
In 53.47% of cases, the payoff is 0.

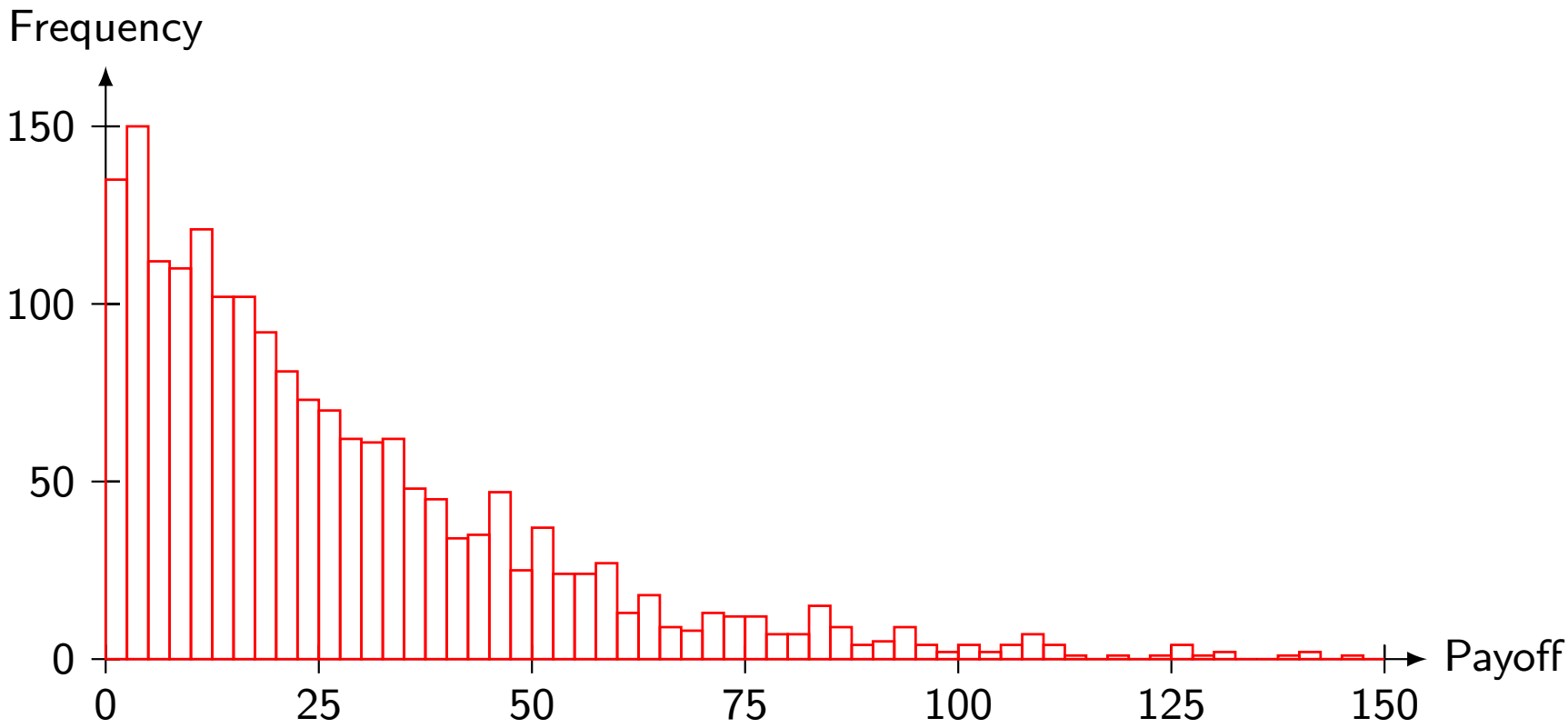
Take $\tau = 12$, $T = 1$ (one year), $t_j = j/12$ for $j = 0, \ldots, 12$, $K = 100$, $s_0 = 100$, $r = 0.05$, $\sigma = 0.5$.

We make $n = 10^6$ independent runs. Mean: **13.1**. Max = 390.8
In 53.47% of cases, the payoff is 0. Histogram of positive values:



Confidence interval on $\mathbb{E}[Y]$ converges as $\mathcal{O}(n^{-1/2})$. Can we do better?

Another histogram, with $n = 4096$ runs.



For histogram: MISE $= \mathcal{O}(n^{-2/3})$.

For polygonal interpolation, ASH, KDE: MISE $= \mathcal{O}(n^{-4/5})$. // Same with KDE.

Can we do better?

Randomized quasi-Monte Carlo (RQMC)

To estimate $\mu = \int_{(0,1)^s} f(\mathbf{u})d\mathbf{u}$, RQMC Estimator:

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

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.

$$\begin{aligned}
\mathbb{E}[\hat{\mu}_{n,\mathrm{rqmc}}] &= \mu \qquad \text{(unbiased)}, \\
\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}] &= \frac{\mathrm{Var}[f(\mathbf{U}_i)]}{n} + \frac{2}{n^2}\sum_{i<j}\mathrm{Cov}[f(\mathbf{U}_i), f(\mathbf{U}_j)].
\end{aligned}$$

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

Randomized quasi-Monte Carlo (RQMC)

To estimate $\mu = \int_{(0,1)^s} f(\mathbf{u})d\mathbf{u}$, RQMC Estimator:

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

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.

$$\begin{aligned}
\mathbb{E}[\hat{\mu}_{n,\mathrm{rqmc}}] &= \mu \qquad \text{(unbiased)}, \\
\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}] &= \frac{\mathrm{Var}[f(\mathbf{U}_i)]}{n} + \frac{2}{n^2}\sum_{i<j}\mathrm{Cov}[f(\mathbf{U}_i), f(\mathbf{U}_j)].
\end{aligned}$$

We want to make the last sum as negative as possible.
Weak attempts: antithetic variates ($n = 2$), Latin hypercube sampling,...

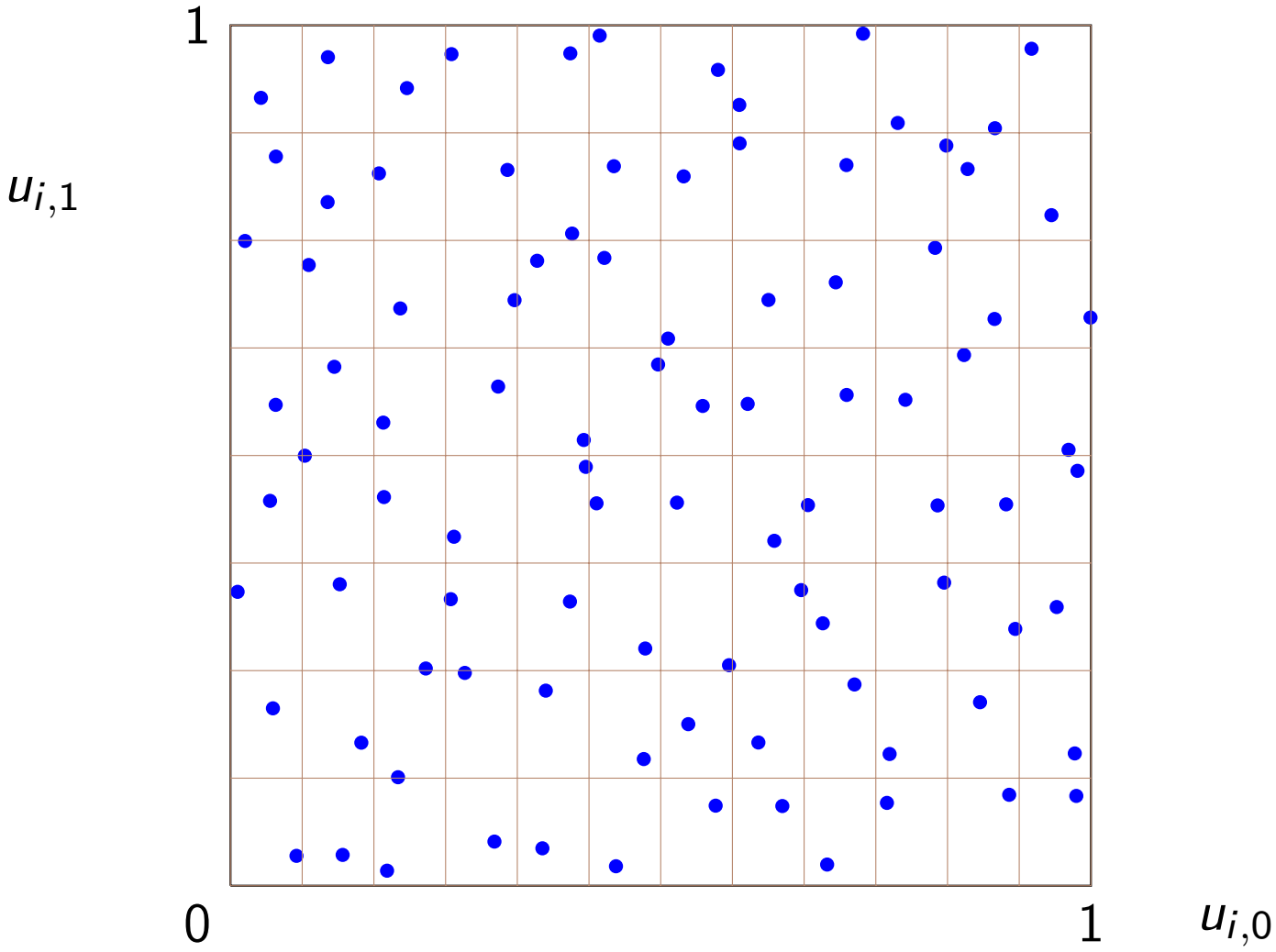
Variance estimation:

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.

Stratification of the unit hypercube

Partition axis j in $k_j \geq 1$ equal parts, for $j = 1, \ldots, s$.
Draw $n = k_1 \cdots k_s$ random points, one per box, independently.

Example, $s = 2$, $k_1 = 12$, $k_2 = 8$, $n = 12 \times 8 = 96$.



Stratified estimator:

$$X_{\mathrm{s},n} = \frac{1}{n}\sum_{j=0}^{n-1} f(\mathbf{U}_j).$$

The crude MC variance with n points can be decomposed as

$$\mathrm{Var}[\bar{X}_n] = \mathrm{Var}[X_{\mathrm{s},n}] + \frac{1}{n}\sum_{j=0}^{n-1}(\mu_j - \mu)^2$$

where μ_j is the mean over box j.

The more the μ_j differ, the more the variance is reduced.

Stratified estimator:

$$X_{\mathrm{s},n} = \frac{1}{n}\sum_{j=0}^{n-1} f(\mathbf{U}_j).$$

The crude MC variance with n points can be decomposed as

$$\mathrm{Var}[\bar{X}_n] = \mathrm{Var}[X_{\mathrm{s},n}] + \frac{1}{n}\sum_{j=0}^{n-1}(\mu_j - \mu)^2$$

where μ_j is the mean over box j.

The more the μ_j differ, the more the variance is reduced.
If f' is continuous and bounded, and all k_j are equal, then

$$\mathrm{Var}[X_{\mathrm{s},n}] = \mathcal{O}(n^{-1-2/s}).$$

Stratified estimator:

$$X_{\mathrm{s},n} = \frac{1}{n}\sum_{j=0}^{n-1} f(\mathbf{U}_j).$$

The crude MC variance with n points can be decomposed as

$$\mathrm{Var}[\bar{X}_n] = \mathrm{Var}[X_{\mathrm{s},n}] + \frac{1}{n}\sum_{j=0}^{n-1}(\mu_j - \mu)^2$$

where μ_j is the mean over box j.

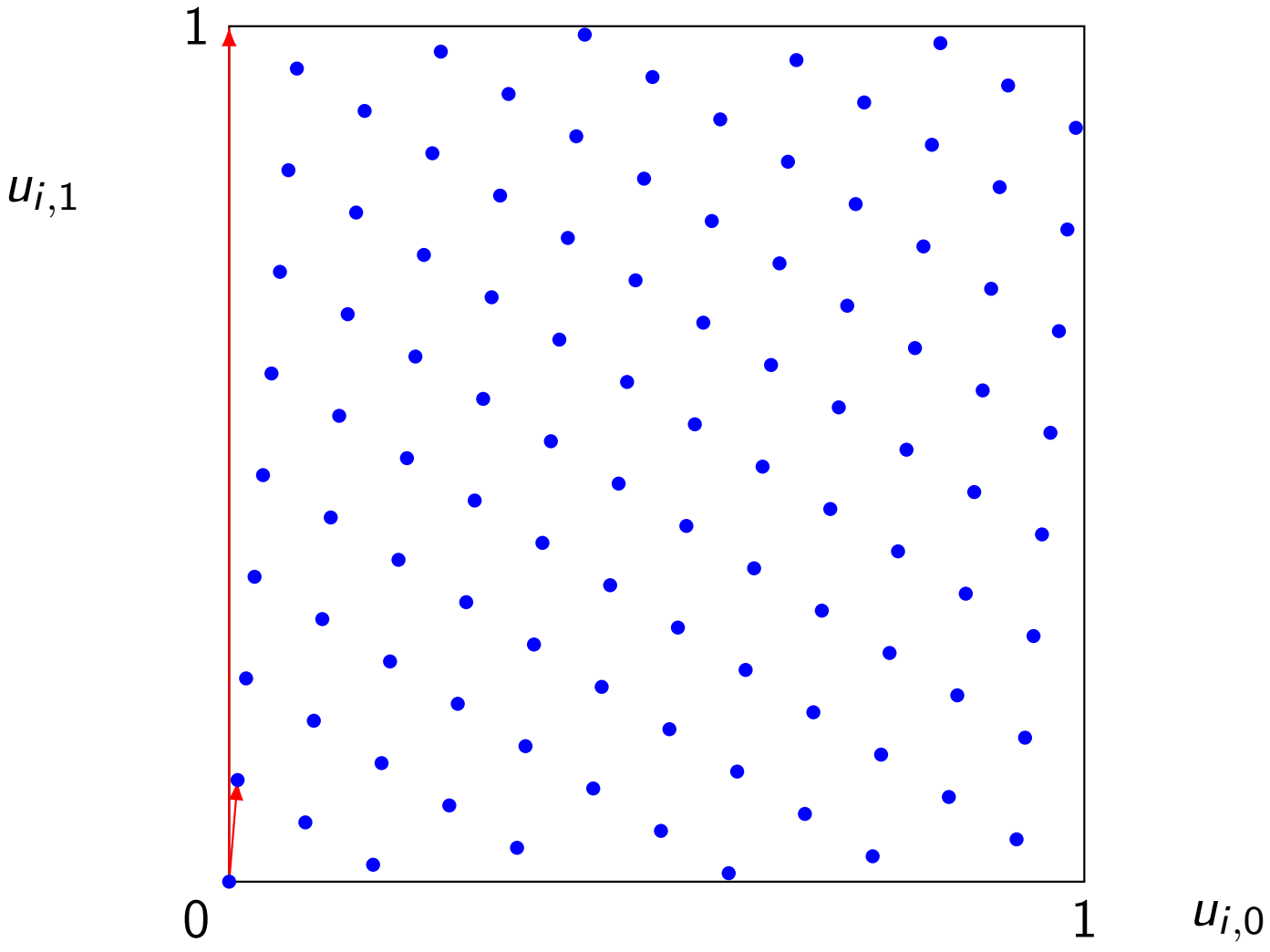
The more the μ_j differ, the more the variance is reduced.
If f' is continuous and bounded, and all k_j are equal, then

$$\mathrm{Var}[X_{\mathrm{s},n}] = \mathcal{O}(n^{-1-2/s}).$$

For large s, not practical. For small s, not really better than midpoint rule with a grid when f is smooth. But can still be applied to a few important random variables. Gives an unbiased estimator, and variance can be estimated by replicating $m \geq 2$ times.

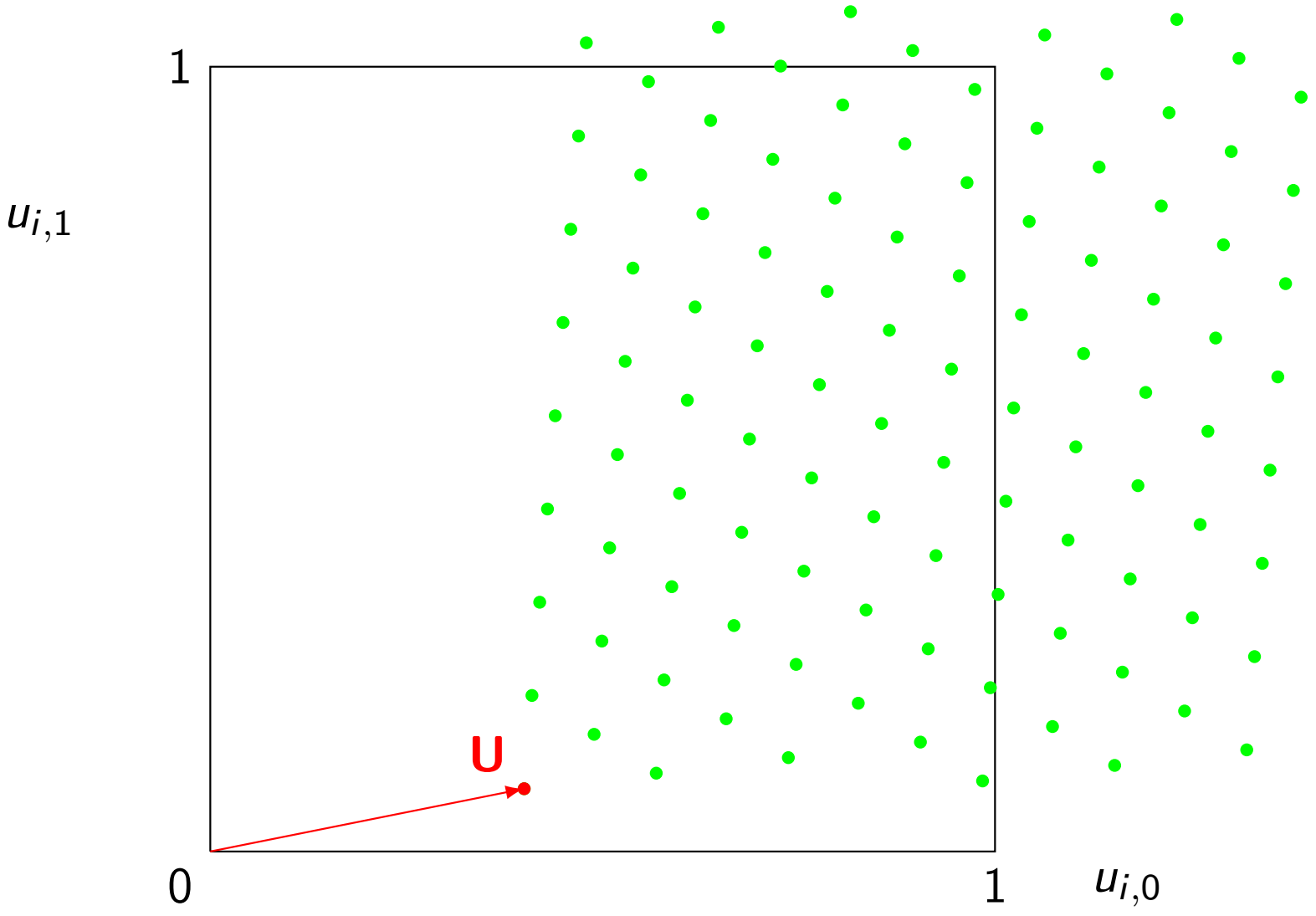
Randomly-Shifted Lattice

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



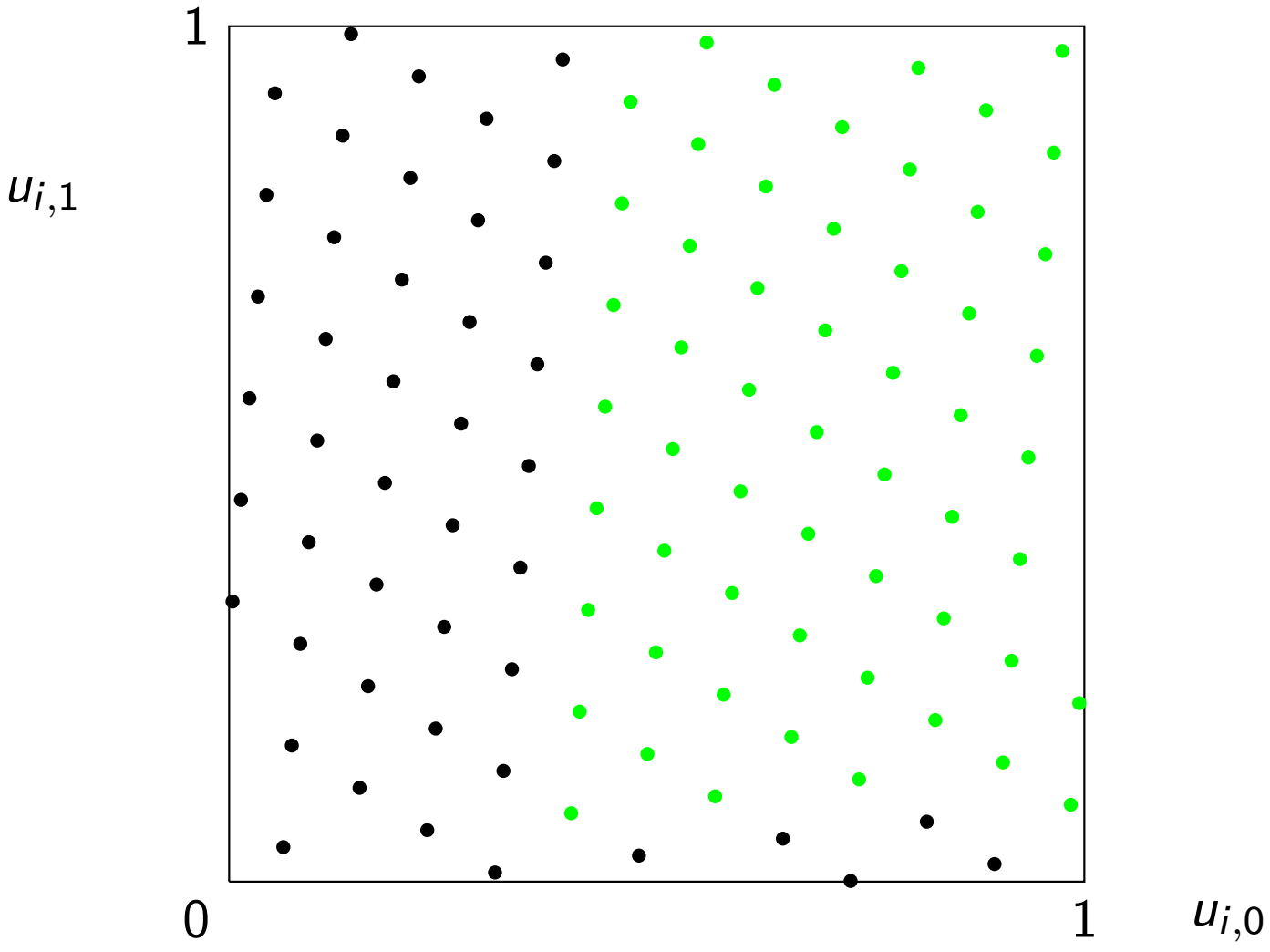
Randomly-Shifted Lattice

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



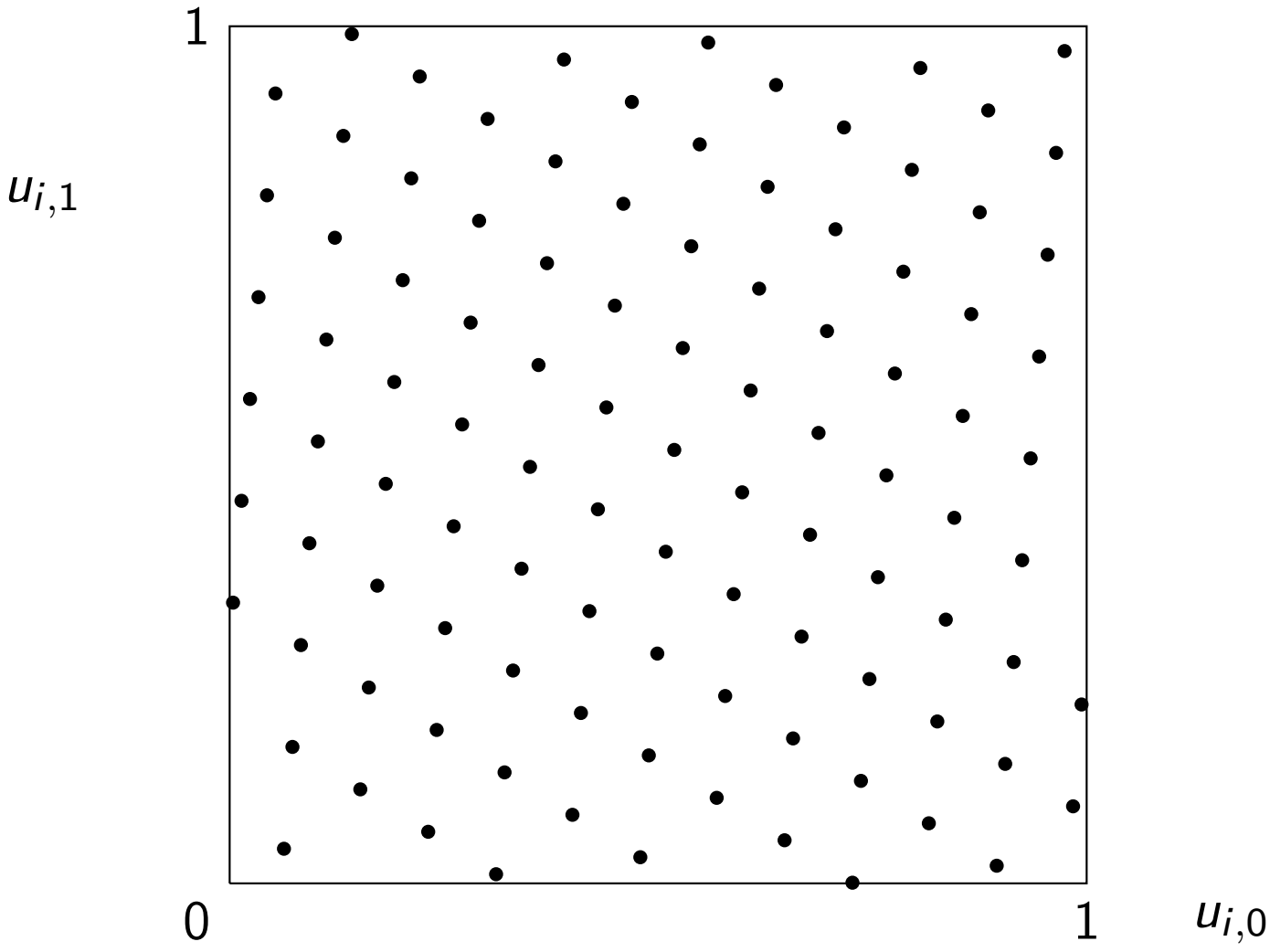
Randomly-Shifted Lattice

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



Randomly-Shifted Lattice

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



Variance bounds

We can obtain various Cauchy-Shwartz inequalities of the form

$$\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}] \;\leq\; V^2(f) \cdot D^2(P_n)$$

for all f in some Hilbert space or Banach space $\mathcal{H}$, where $V(f) = \|f - \mu\|_{\mathcal{H}}$ is the variation of f, and $D(P_n)$ is the discrepancy of P_n (defined by an expectation in the RQMC case).

Variance bounds

We can obtain various Cauchy-Shwartz inequalities of the form

$$\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}] \;\leq\; V^2(f) \cdot D^2(P_n)$$

for all f in some Hilbert space or Banach space $\mathcal{H}$, where $V(f) = \|f - \mu\|_{\mathcal{H}}$ is the variation of f, and $D(P_n)$ is the discrepancy of P_n (defined by an expectation in the RQMC case).

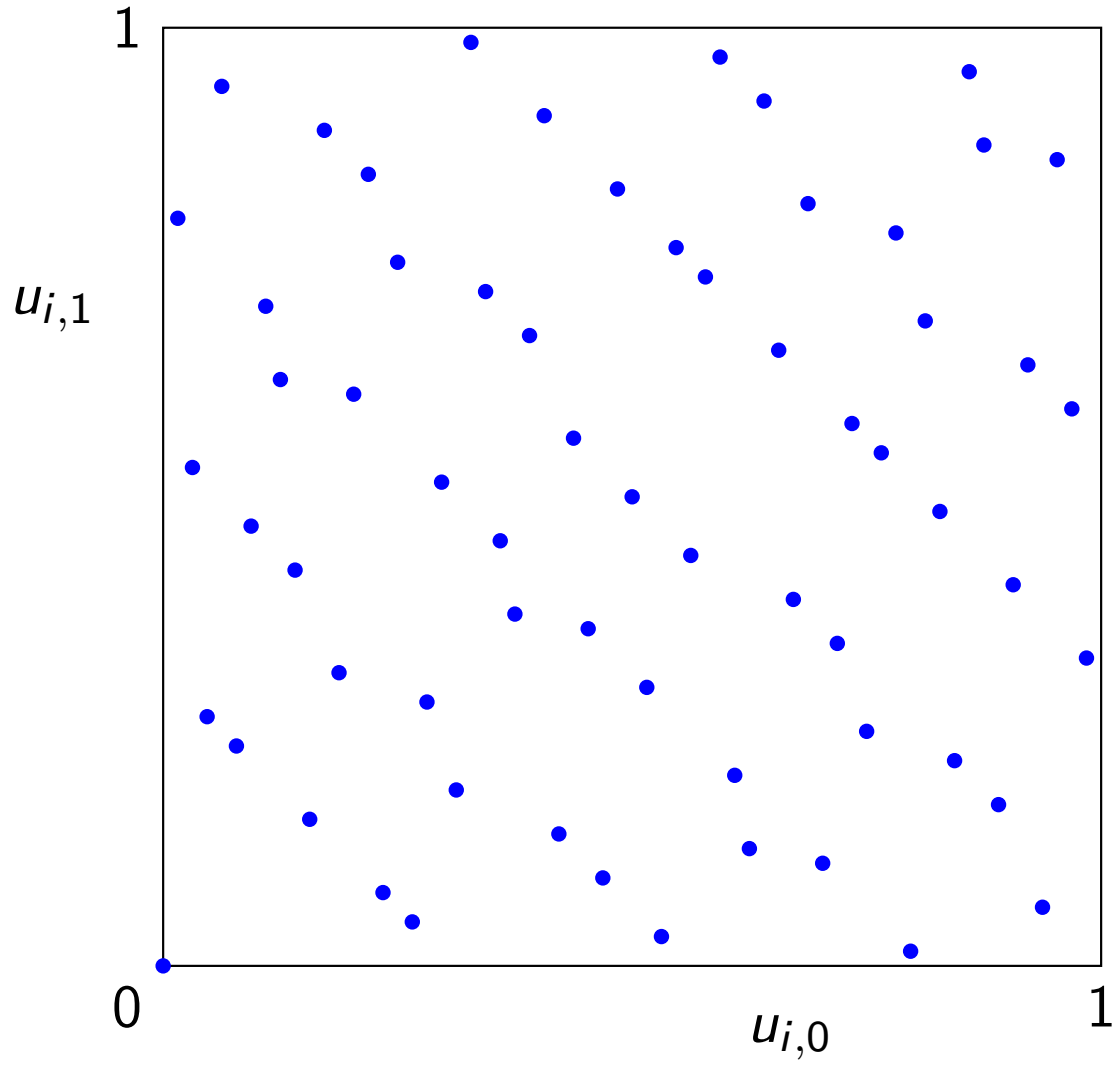
Lattice rules: For certain Hilbert spaces of smooth periodic functions f with square-integrable partial derivatives of order up to α:

$D(P_n) = \mathcal{O}(n^{-\alpha+\epsilon})$ for all $\epsilon > 0$.

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

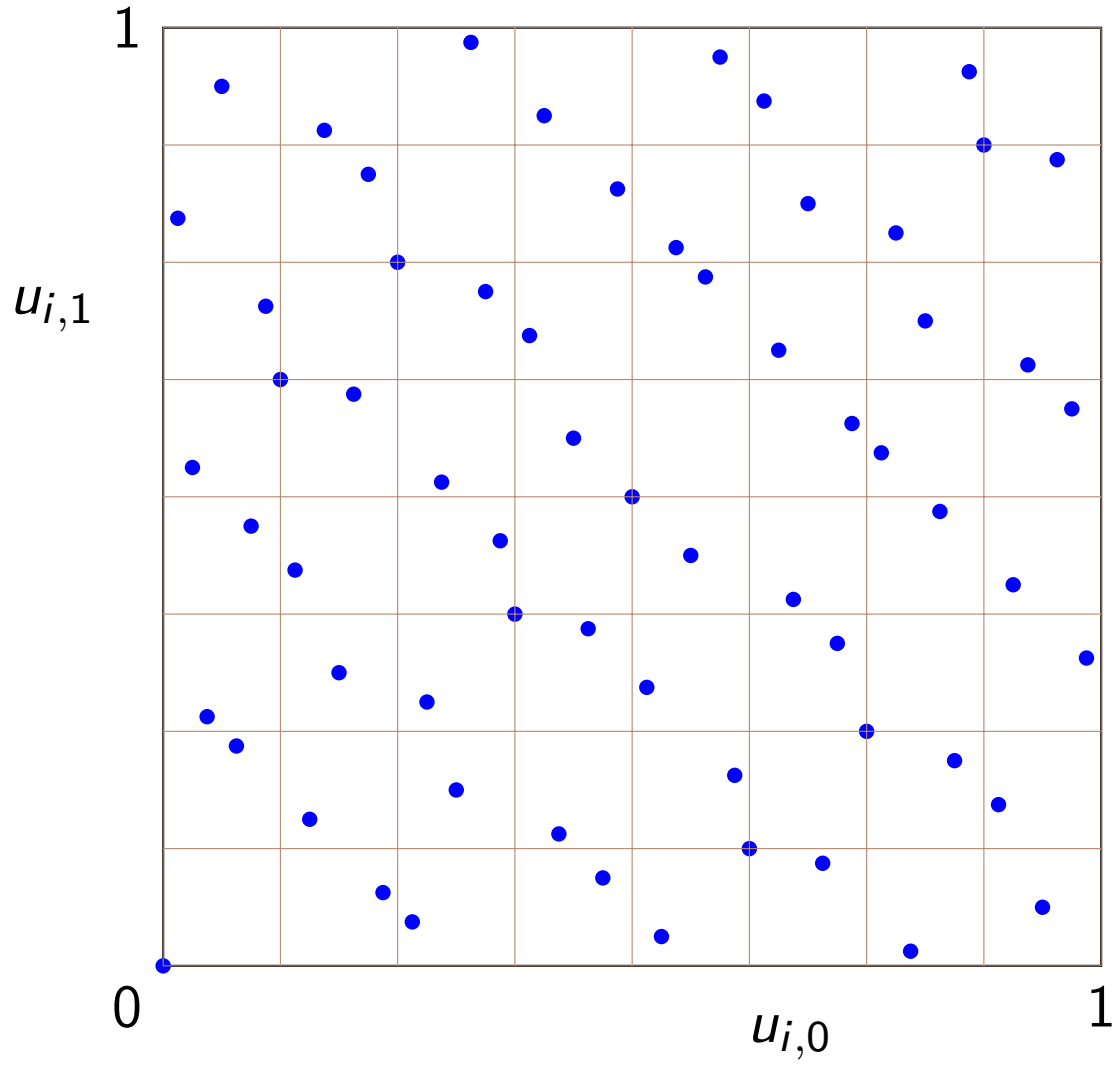
Non-periodic functions can be made periodic via a baker's transformation (easy).

Example of a **digital net in base 2**:
The first $n = 64 = 2^6$ Sobol points in $s = 2$ dimensions:



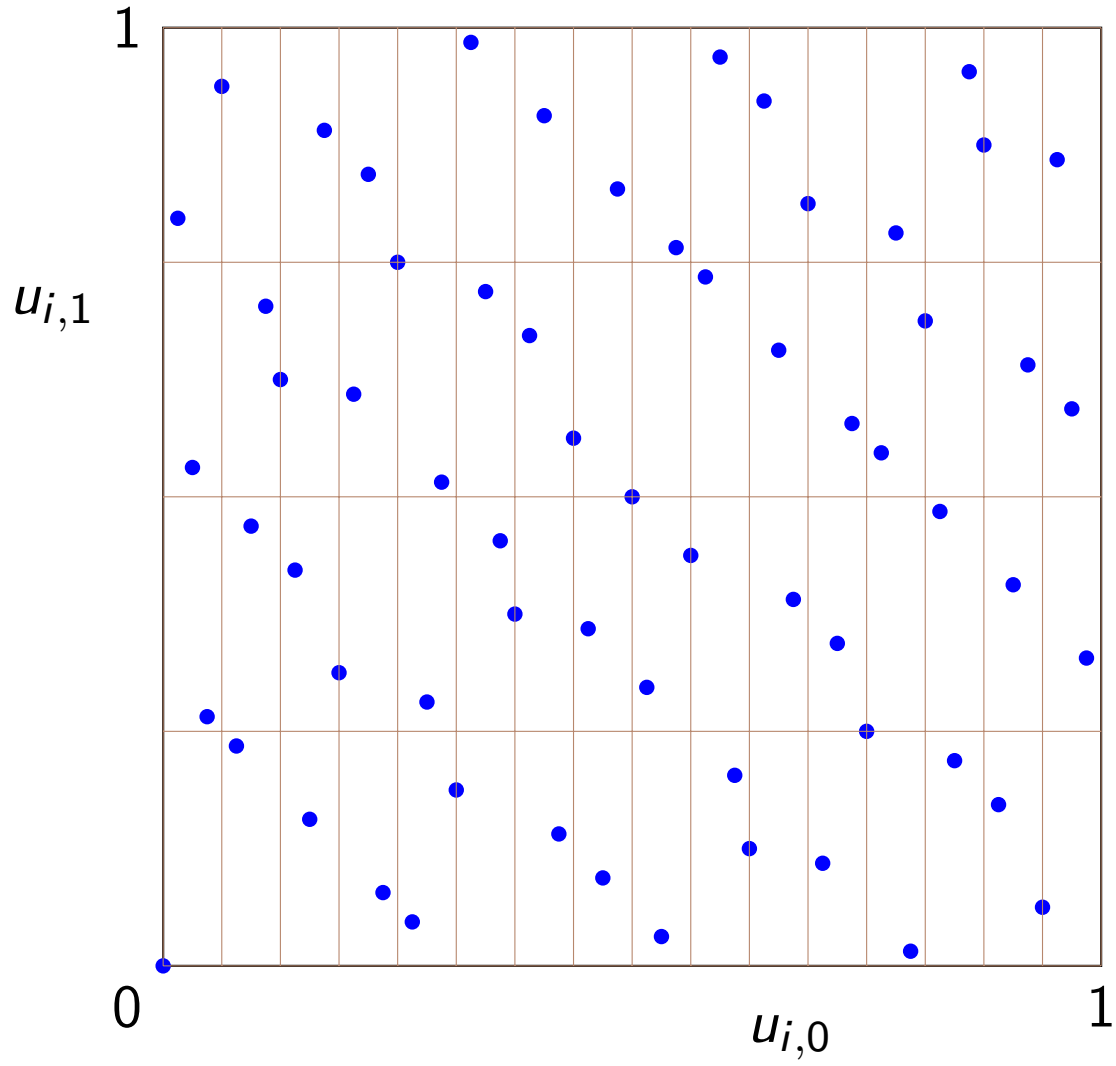
They form a $(0, 6, 2)$-net in two dimensions.

Example of a **digital net in base 2**:
The first $n = 64 = 2^6$ Sobol points in $s = 2$ dimensions:



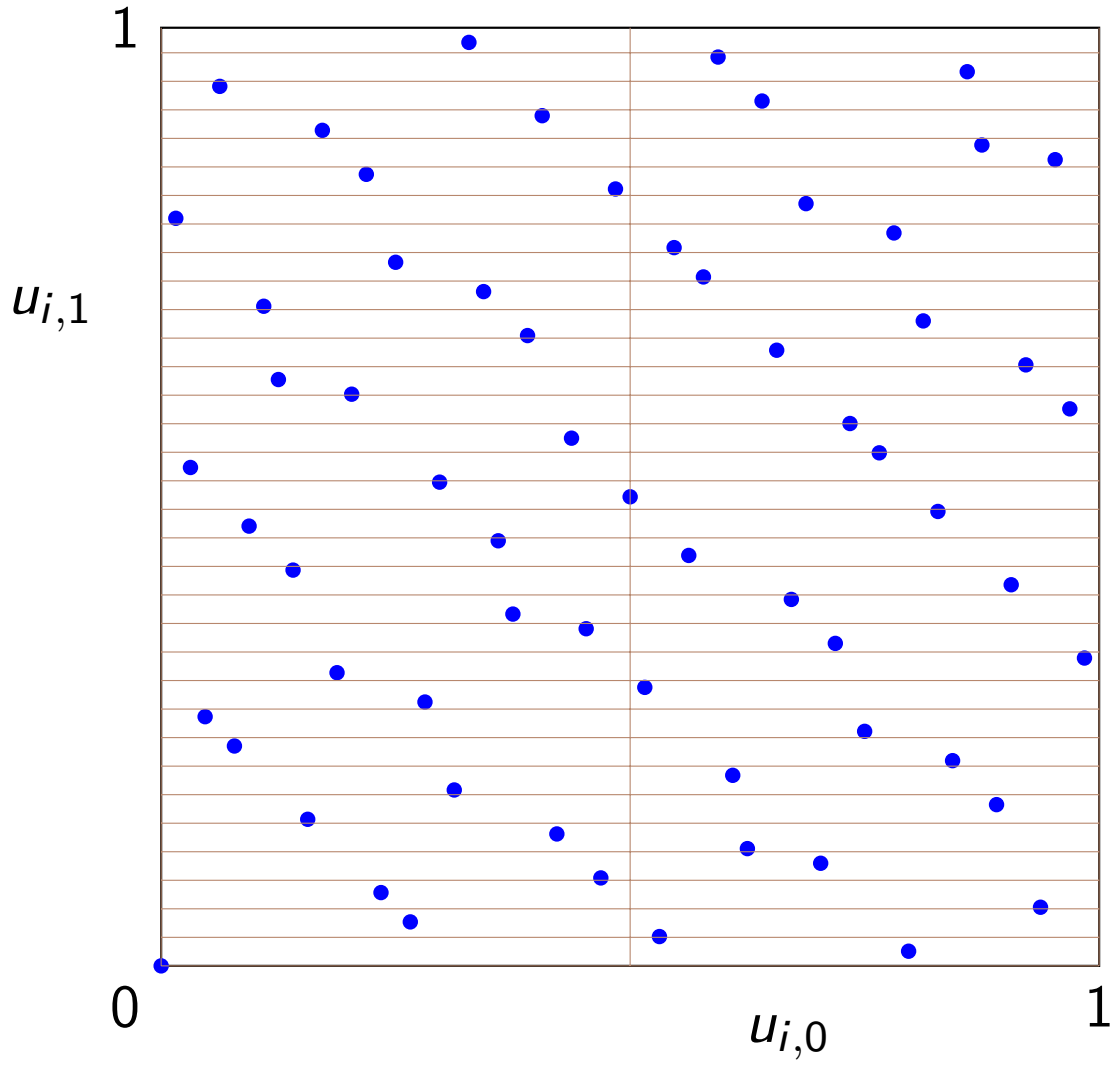
They form a $(0, 6, 2)$-net in two dimensions.

Example of a **digital net in base 2**:
The first $n = 64 = 2^6$ Sobol points in $s = 2$ dimensions:



They form a $(0, 6, 2)$-net in two dimensions.

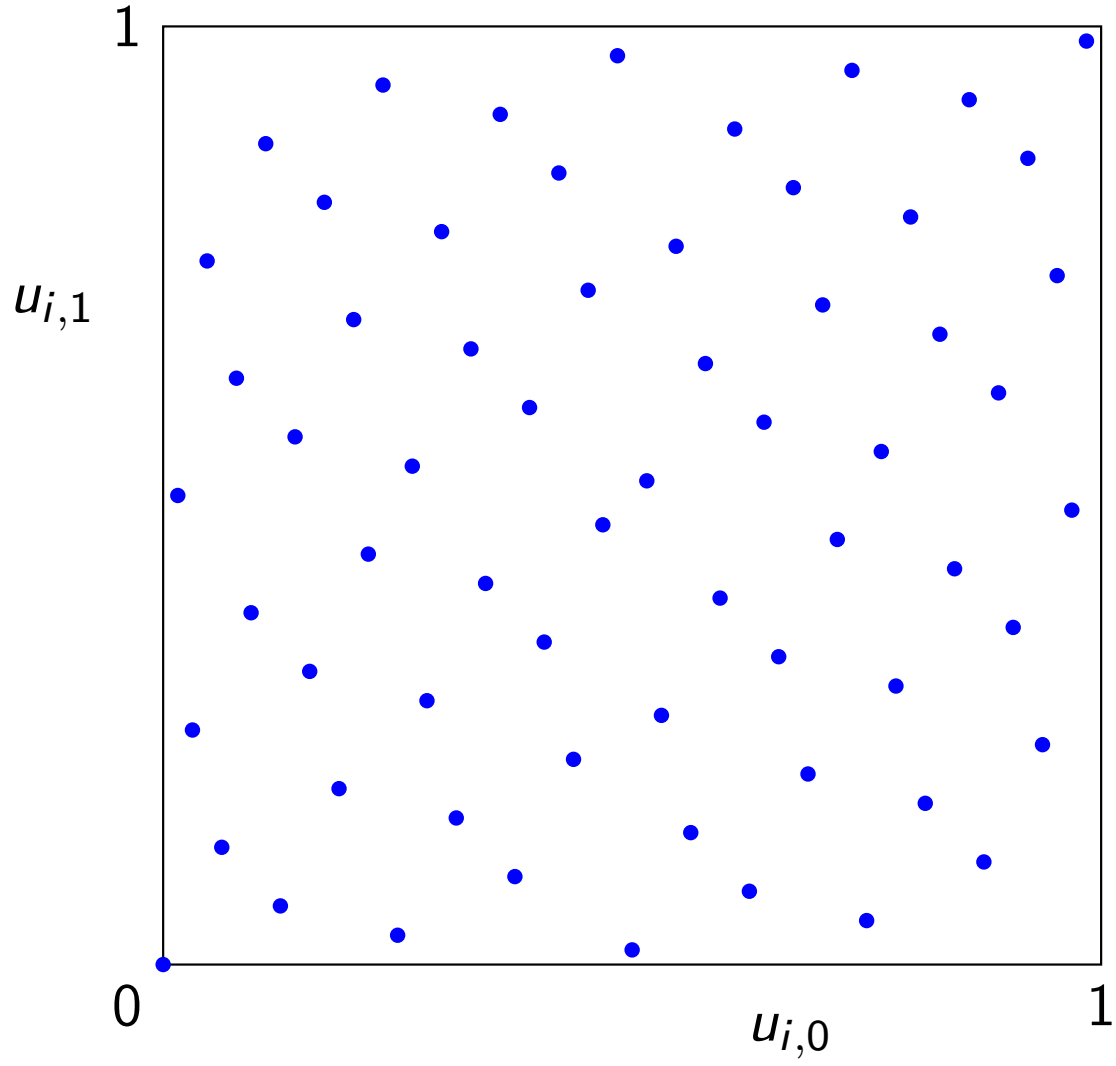
Example of a **digital net in base 2**:
The first $n = 64 = 2^6$ Sobol points in $s = 2$ dimensions:



They form a $(0, 6, 2)$-net in two dimensions.

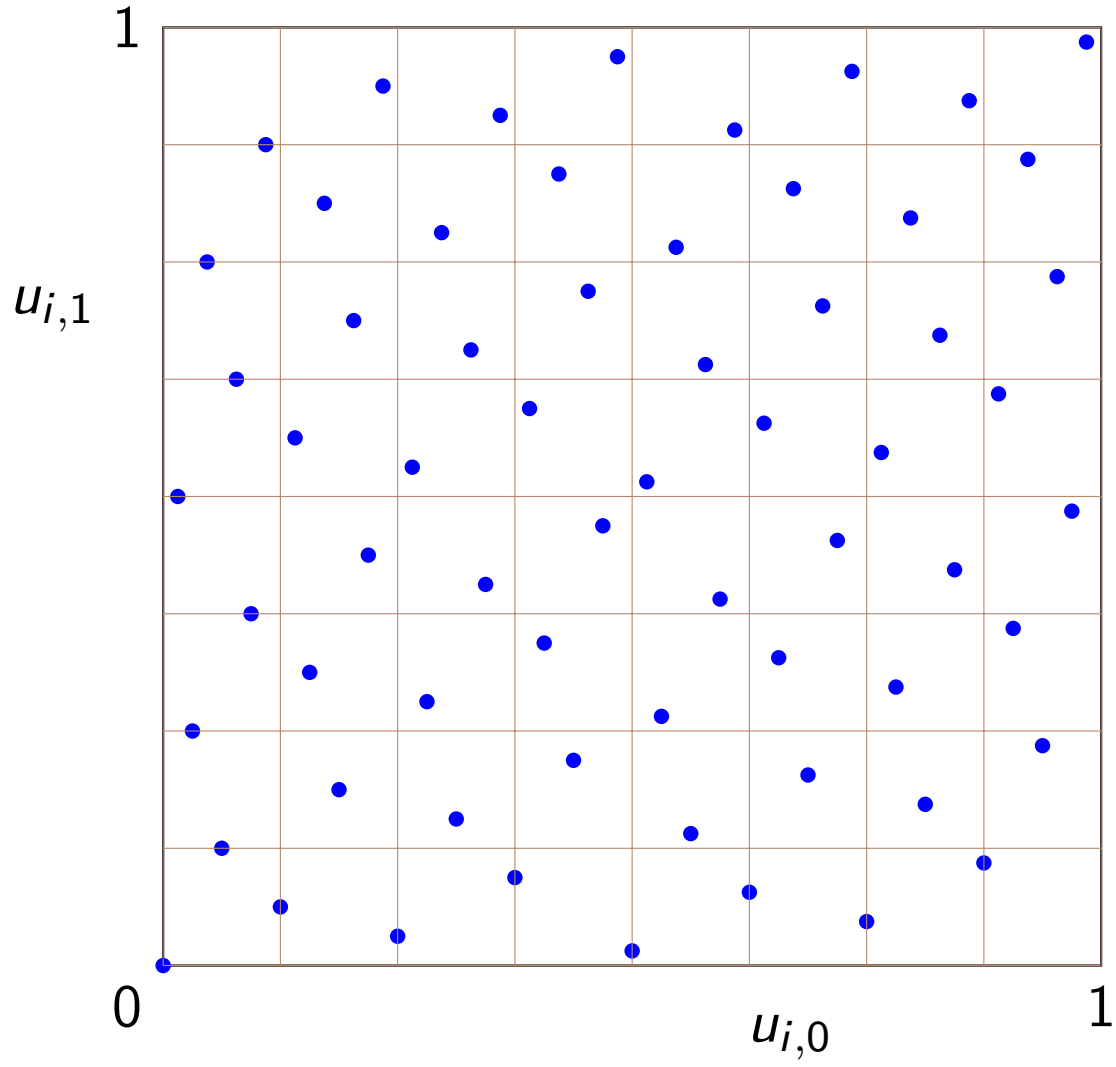
Example of a **digital net in base 2**:

Hammersley point set (or Sobol + 1 coord.), $n = 64$, $s = 2$.



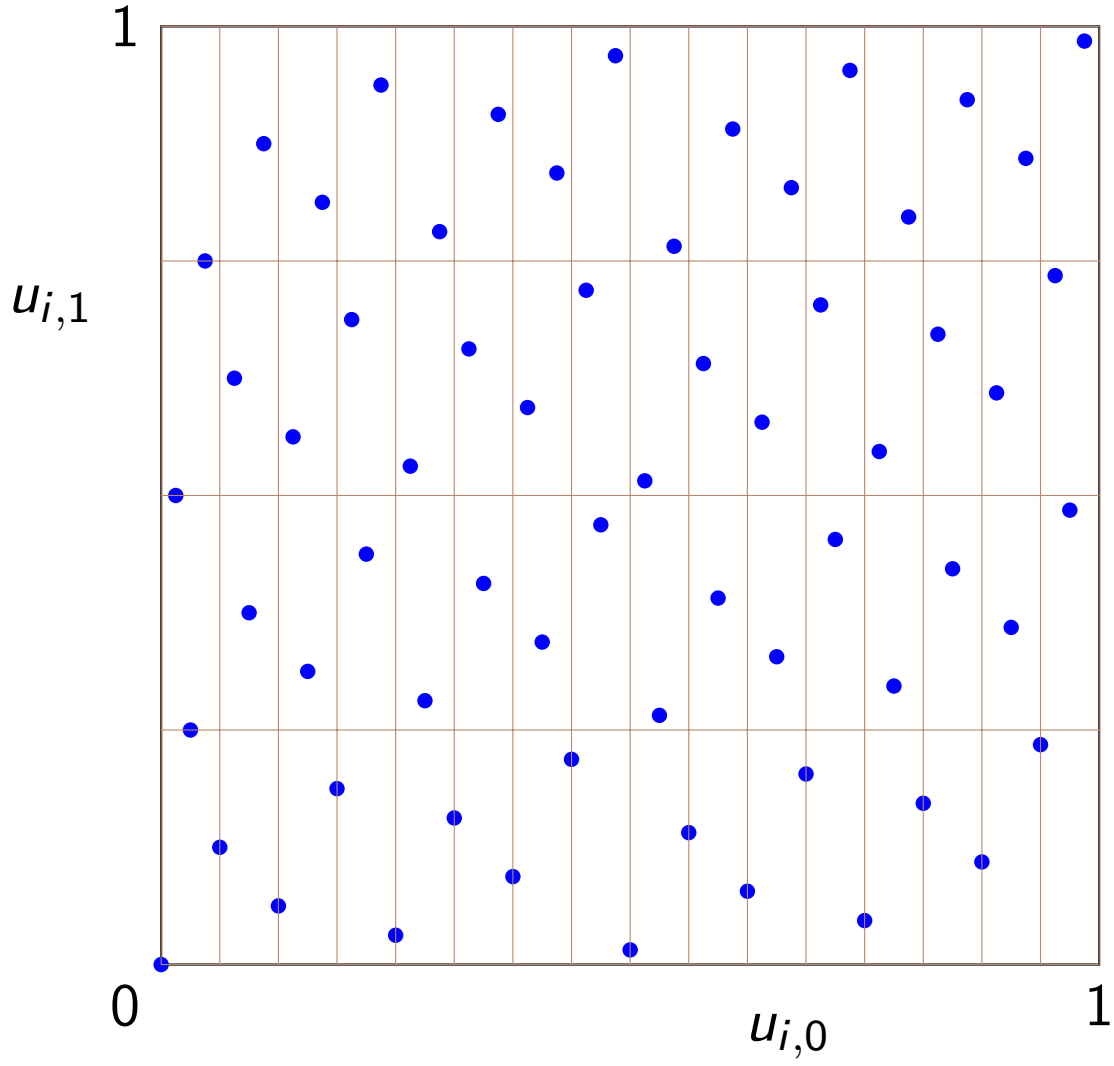
Also a $(0, 6, 2)$-net in two dimensions.

Example of a **digital net in base 2**:
Hammersley point set (or Sobol + 1 coord.), $n = 64$, $s = 2$.



Also a $(0, 6, 2)$-net in two dimensions.

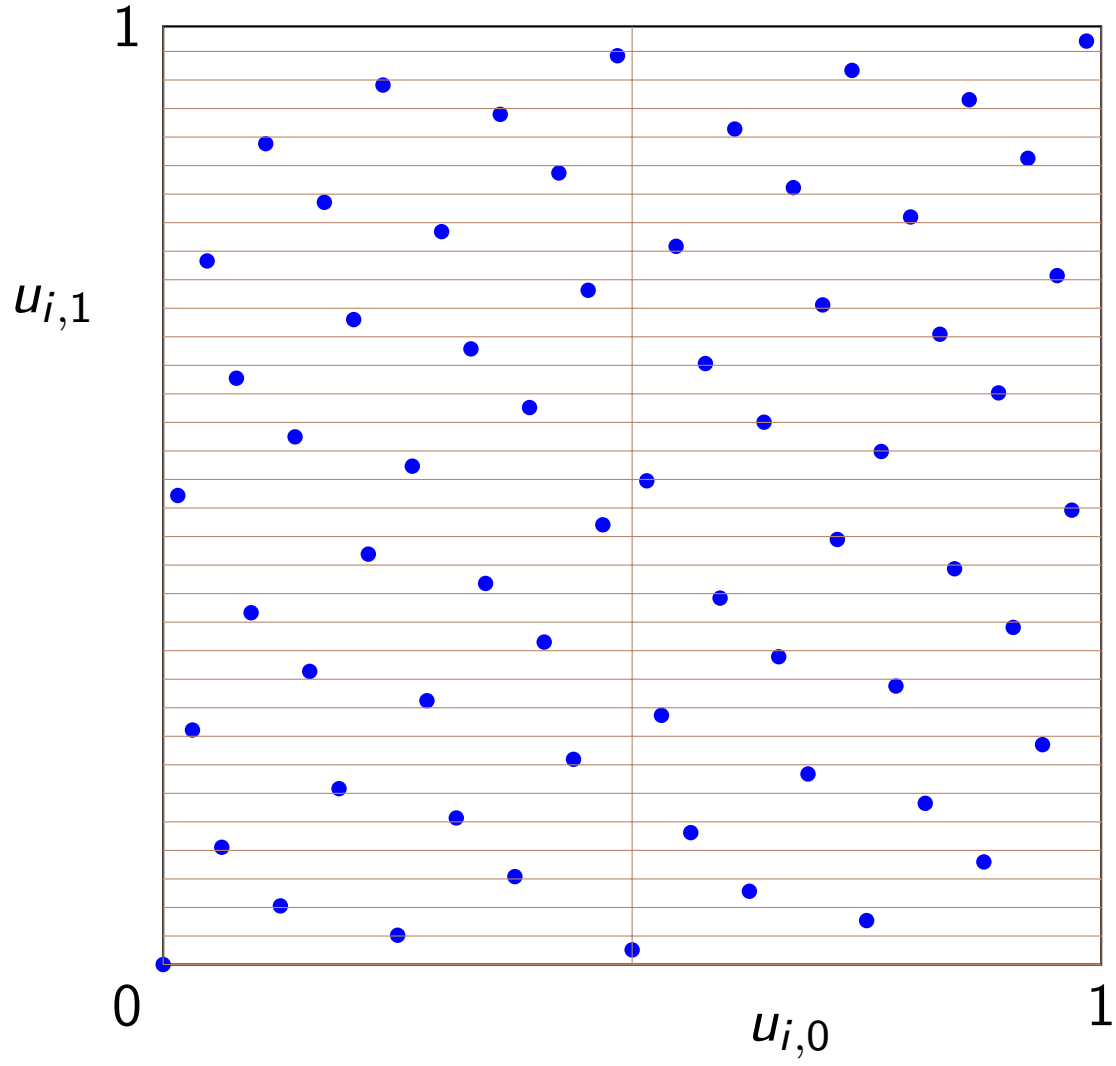
Example of a **digital net in base 2**:
Hammersley point set (or Sobol + 1 coord.), $n = 64$, $s = 2$.



Also a $(0, 6, 2)$-net in two dimensions.

Example of a **digital net in base 2**:

Hammersley point set (or Sobol + 1 coord.), $n = 64$, $s = 2$.



Also a $(0, 6, 2)$-net in two dimensions.

Digital net with random digital shift

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 $s = 2$:

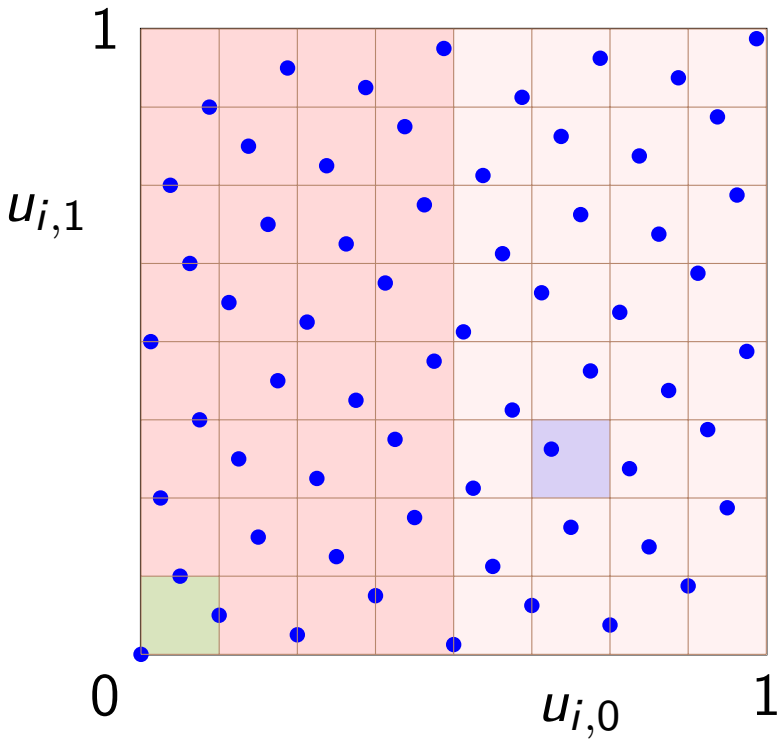
$$\begin{aligned} \mathbf{u}_i &= (0.01100100\ldots,\ 0.10011000\ldots)_2 \\ \mathbf{U} &= (0.01001010\ldots,\ 0.11101001\ldots)_2 \\ \mathbf{u}_i \oplus \mathbf{U} &= (0.00101110\ldots,\ 0.01110001\ldots)_2. \end{aligned}$$

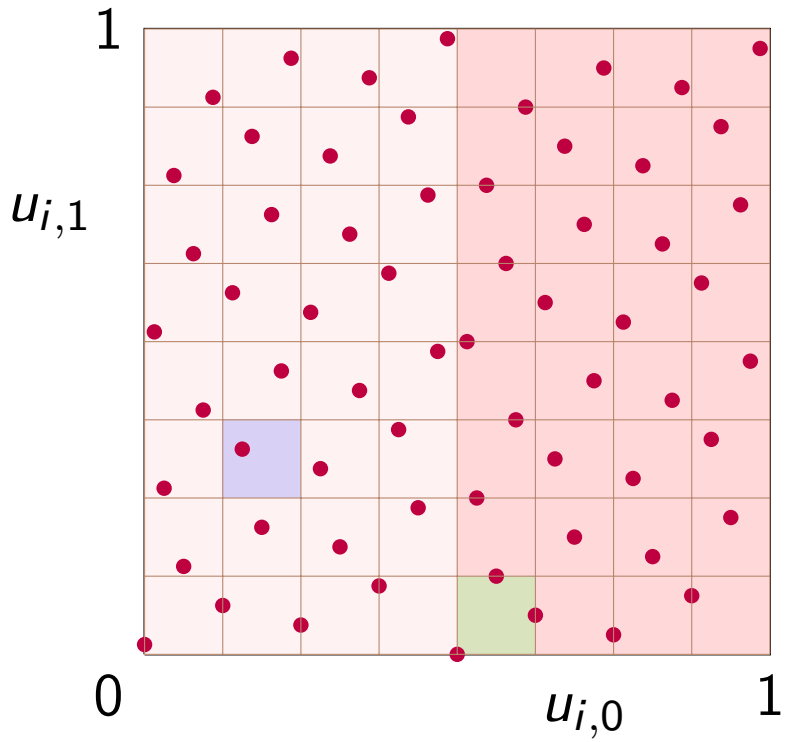
Each point has $U(0,1)$ distribution.
Preservation of the equidistribution ($k_1 = 3$, $k_2 = 5$):

$$\begin{aligned} \mathbf{u}_i &= (0.***,\ 0.*****) \\ \mathbf{U} &= (0.101,\ 0.01011)_2 \\ \mathbf{u}_i \oplus \mathbf{U} &= (0.\mathtt{C}*\mathtt{C},\ 0.*\mathtt{C}*\mathtt{CC}) \end{aligned}$$

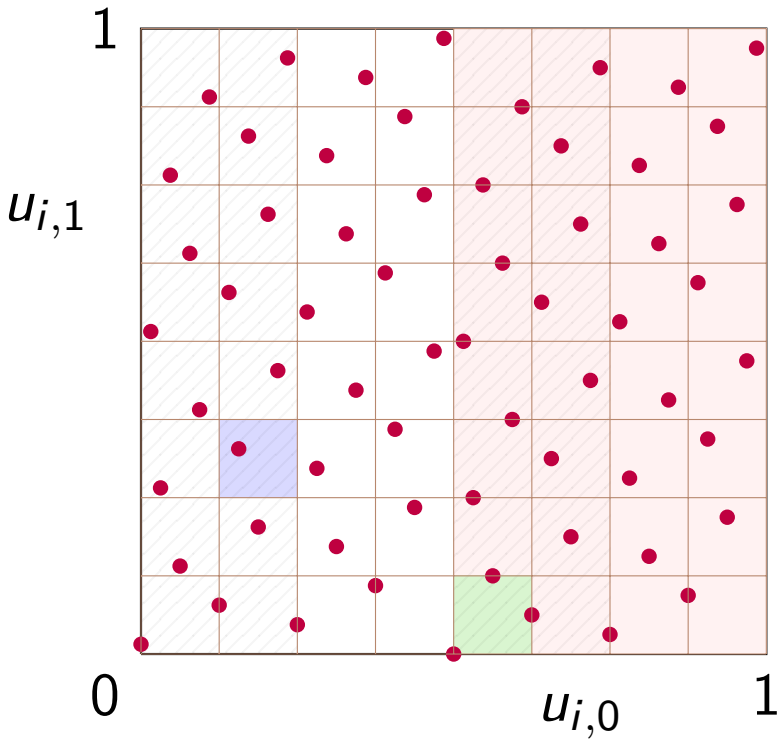
Hammersley points

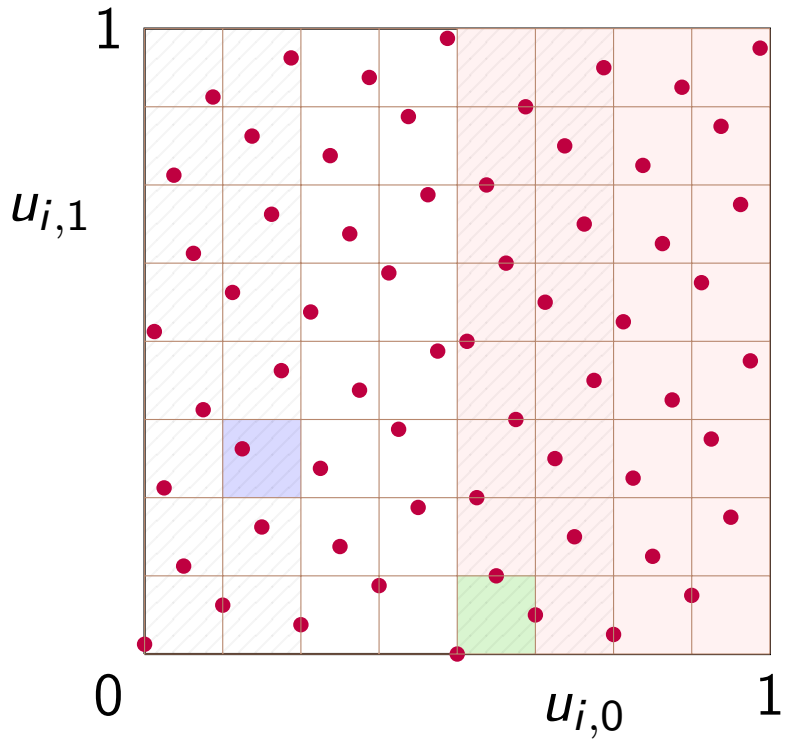
Digital shift with $\mathbf{U} = (0.110100101..., 0.0101100...)_2$, first bit.



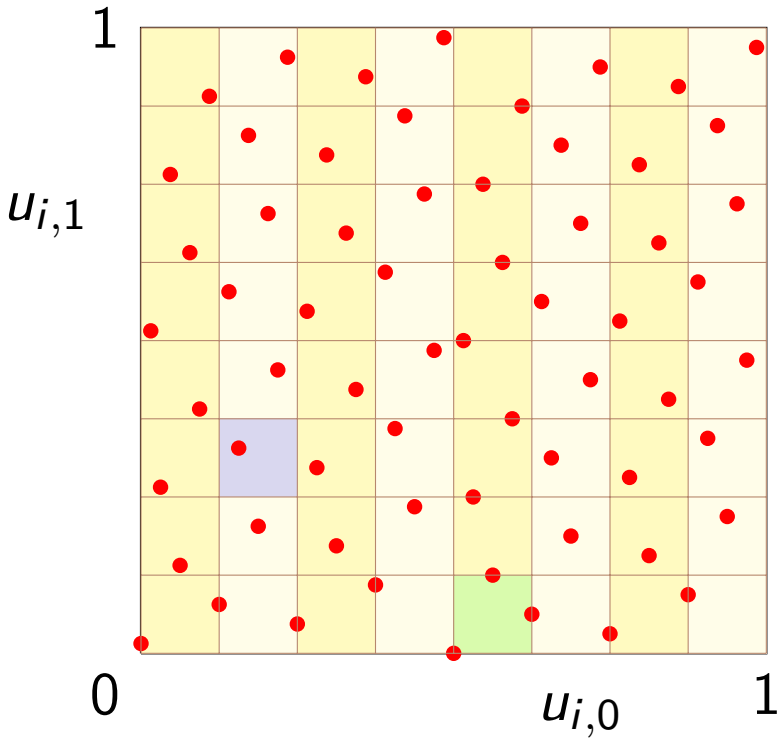


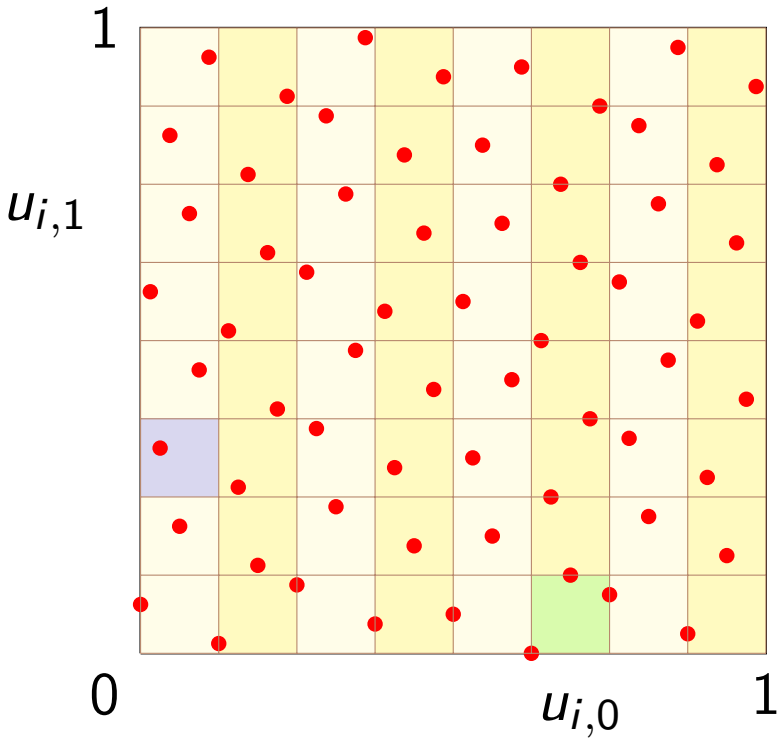
Digital shift with $\mathbf{U} = (0.\mathbf{1}0100101..., 0.0101100...)_2$, second bit.



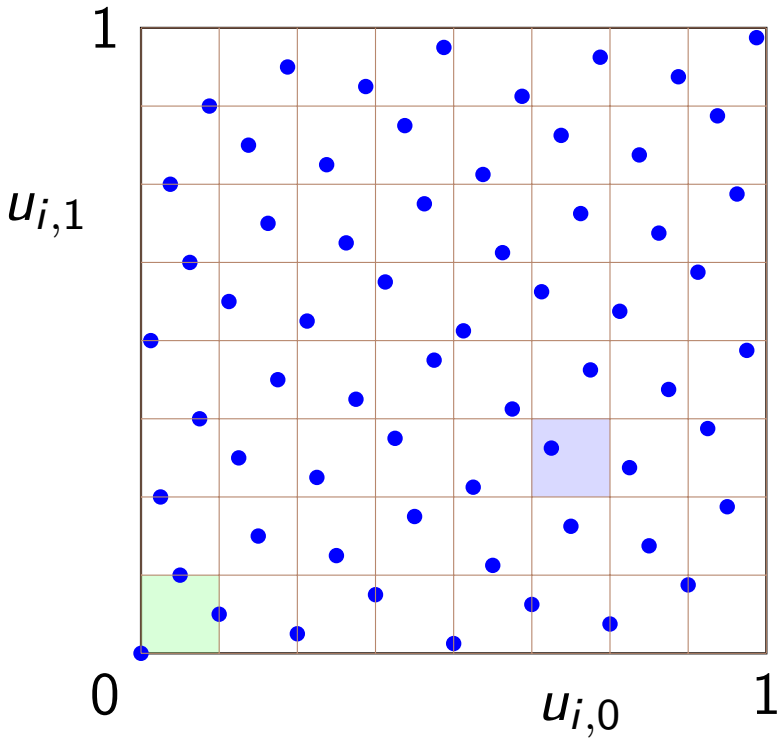


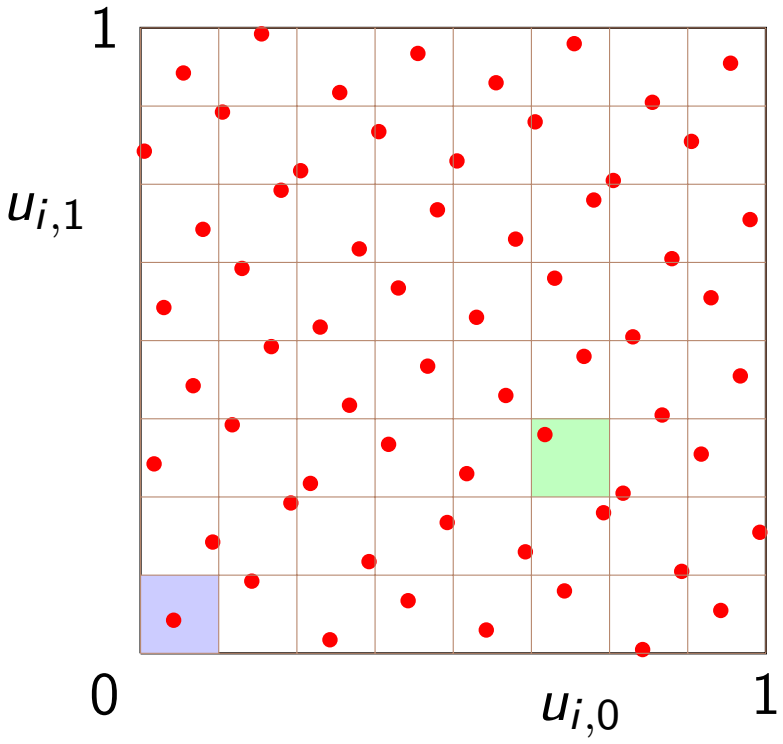
Digital shift with $\mathbf{U} = (0.10100101..., 0.0101100...)_2$, third bit.





Digital shift with $\mathbf{U} = (0.10100101..., 0.0101100...)_2$, all bits (final).





Variance bounds

Digital nets: "Classical" Koksma-Hlawka inequality for QMC: f must have finite variation in the sense of Hardy and Krause (implies no discontinuity not aligned with the axes). Popular constructions achieve

$$D(P_n) = O(n^{-1}(\ln n)^s) = O(n^{-1+\epsilon}) \text{ for all } \epsilon > 0.$$

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

More recent constructions (polynomial lattice rules) offer better rates for smooth functions.

Variance bounds

Digital nets: "Classical" Koksma-Hlawka inequality for QMC: f must have finite variation in the sense of Hardy and Krause (implies no discontinuity not aligned with the axes). Popular constructions achieve

$$D(P_n) = O(n^{-1}(\ln n)^s) = O(n^{-1+\epsilon}) \text{ for all } \epsilon > 0.$$

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

More recent constructions (polynomial lattice rules) offer better rates for smooth functions.

With nested uniform scrambling (NUS) by Owen, one has

$\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}] = \mathcal{O}(n^{-3+\epsilon})$ for all $\epsilon > 0$.

Bounds are conservative and too hard to compute in practice.
Hidden constant and variation often increase fast with dimension s.

But still often works very well empirically!

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

One RQMC point for each sample path.

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

$$\hat{\mu}_{\mathrm{rqmc},n} = \frac{1}{n} \sum_{i=0}^{n-1} g(X_{i,\tau})$$

where $P_n = \{\mathbf{V}_0, \ldots, \mathbf{V}_{n-1}\} \subset (0,1)^s$ is an RQMC point set:
(a) each point $\mathbf{V}_i$ has the uniform distribution over $(0,1)^s$;
(b) P_n covers $(0,1)^s$ very evenly (i.e., has low discrepancy).

The dimension $s = d\tau$ is often very large!

Array-RQMC for Markov Chains

L., Lécot, Tuffin, et al. [2004, 2006, 2008, etc.]
Earlier deterministic versions by Lécot et al.
Simulate an "array" of n chains in "parallel."
At each step, use an RQMC point set P_n to advance all the chains by one step. Seek global negative dependence across the chains.

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

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

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

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

(we take a single j here) by

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

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) (be highly uniform) over $[0,1)^{\ell+d}$.

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

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

(we take a single j here) by

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

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) (be highly uniform) over $[0,1)^{\ell+d}$.

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

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

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 $i = 0, \ldots, n-1$;
  $\hat{\mu}_{\mathrm{arqmc},j,n} = \bar{Y}_{n,j} = \frac{1}{n}\sum_{i=0}^{n-1} g(X_{i,j})$;
end for
Estimate μ by the average $\bar{Y}_n = \hat{\mu}_{\mathrm{arqmc},\tau,n}$.

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 $i = 0, \ldots, n-1$;
  $\hat{\mu}_{\mathrm{arqmc},j,n} = \bar{Y}_{n,j} = \frac{1}{n}\sum_{i=0}^{n-1} g(X_{i,j})$;
end for
Estimate μ by the average $\bar{Y}_n = \hat{\mu}_{\mathrm{arqmc},\tau,n}$.

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

Key issues:

1. How can we preserve LD of $S_{n,j} = \{X_{0,j}, \ldots, X_{n-1,j}\}$ as j increases?
2. Can we prove that $\mathrm{Var}[\hat{\mu}_{\mathrm{arqmc},\tau,n}] = \mathcal{O}(n^{-\alpha})$ for some $\alpha > 1$?
How? What α?
3. How does it behave empirically for moderate n?

Intuition: Write discrepancy measure of $S_{n,j}$ as the mean square integration error (or variance) when integrating some function $\psi : [0,1)^{\ell+d} \to \mathbb{R}$ using Q_n.
Use RQMC theory to show it is small if Q_n has LD. Then use induction.

Convergence results and applications

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

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

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

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

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

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

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.

Mapping chains to points when $\ell > 2$

1. Multivariate batch sort:

Sort the states (chains) by first coordinate, in n_1 packets of size n/n_1.

Sort each packet by second coordinate, in n_2 packets of size $n/n_1 n_2$.

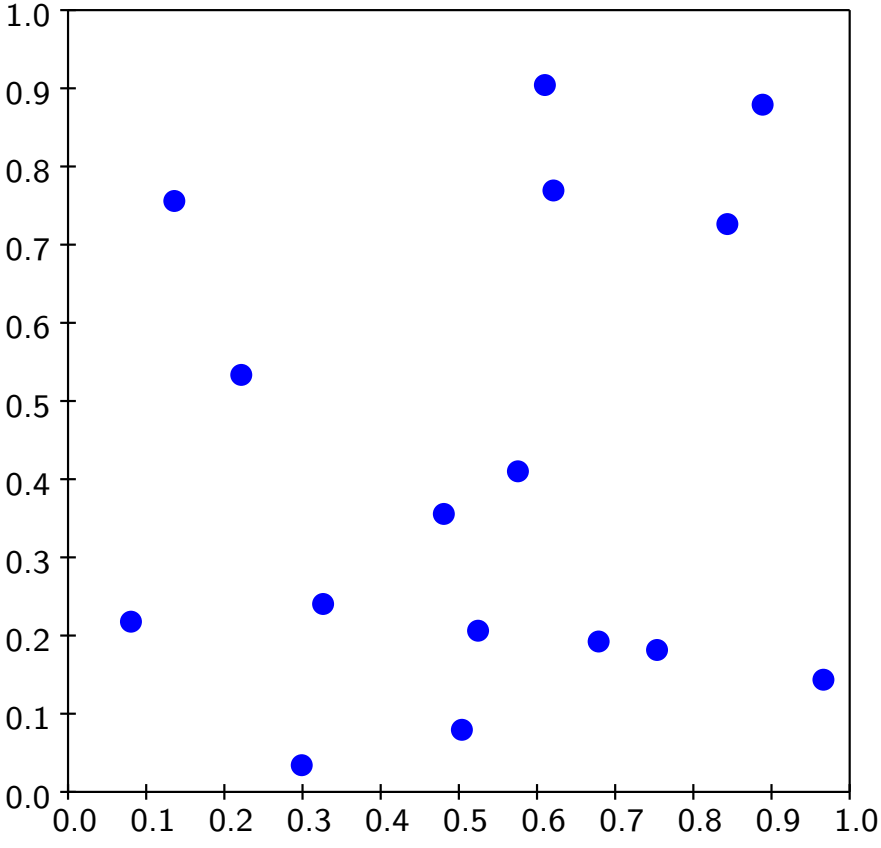
...

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

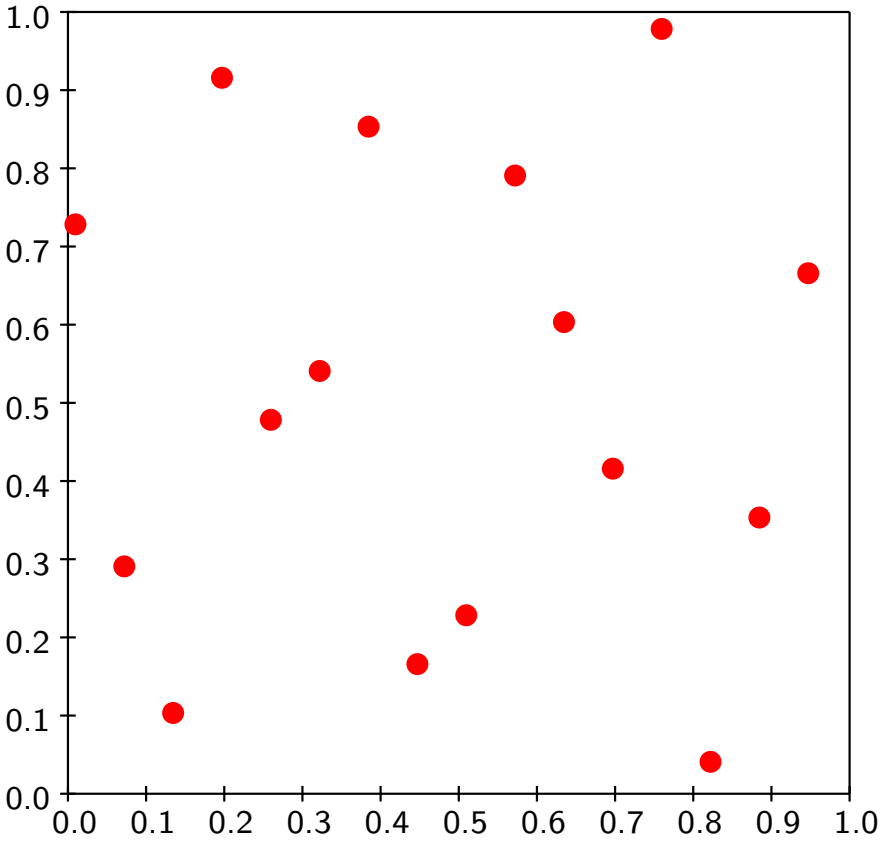
Choice of $n_1, n_2, ..., n_\ell$?

A (4,4) mapping

States of the chains

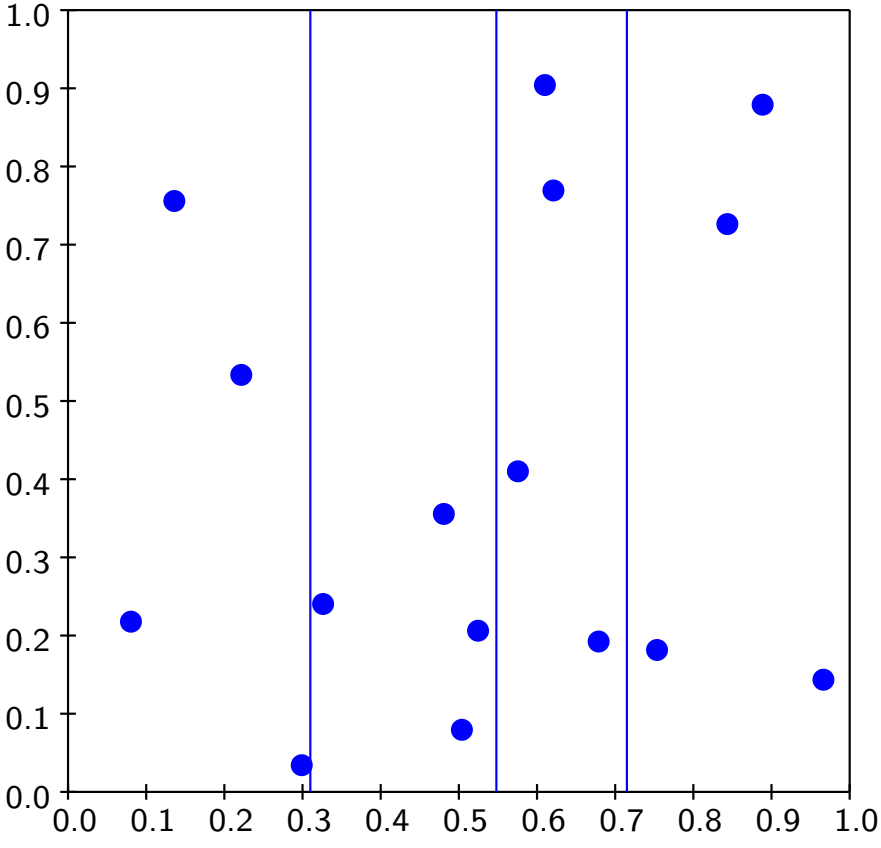


Sobol' net in 2 dimensions after random digital shift

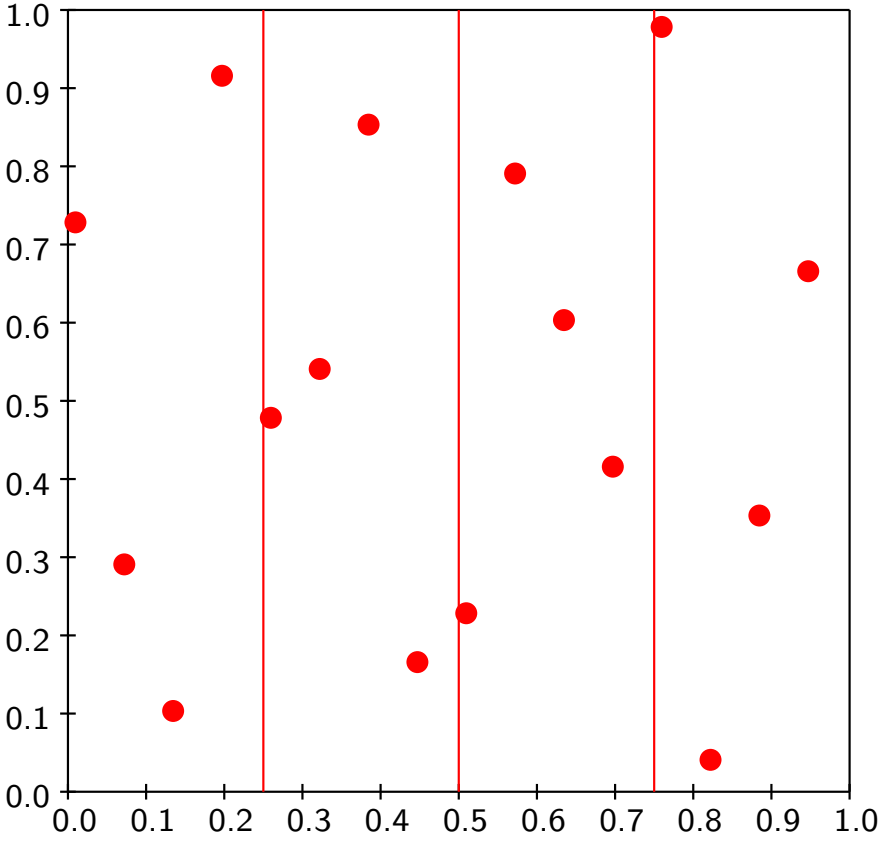


A (4,4) mapping

States of the chains

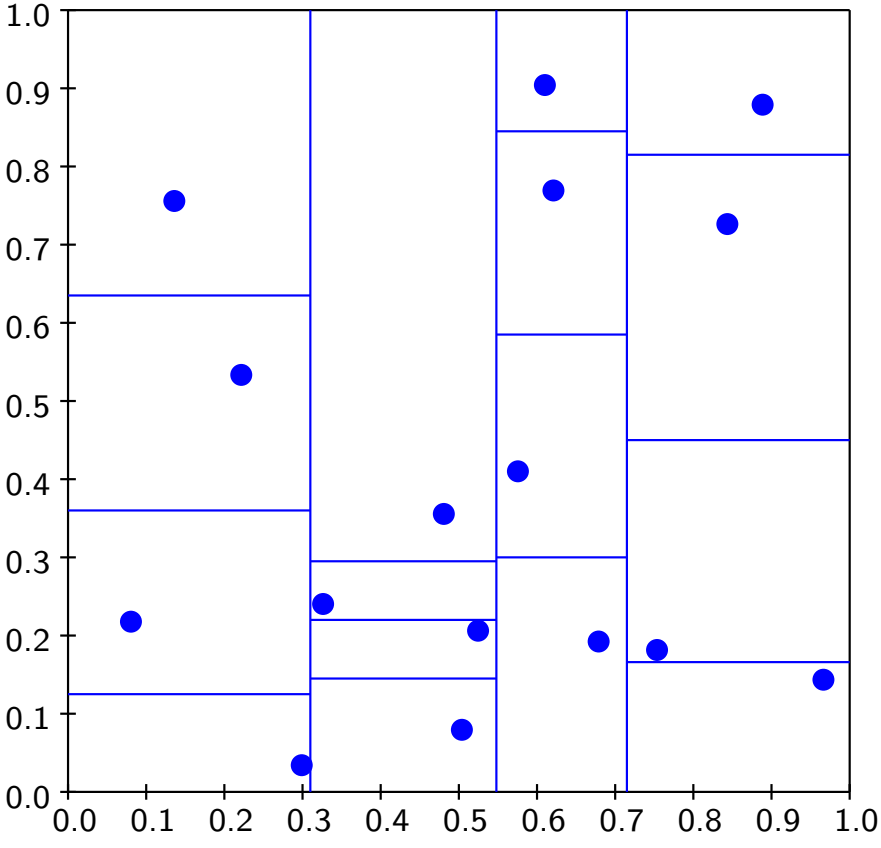


Sobol' net in 2 dimensions after random digital shift

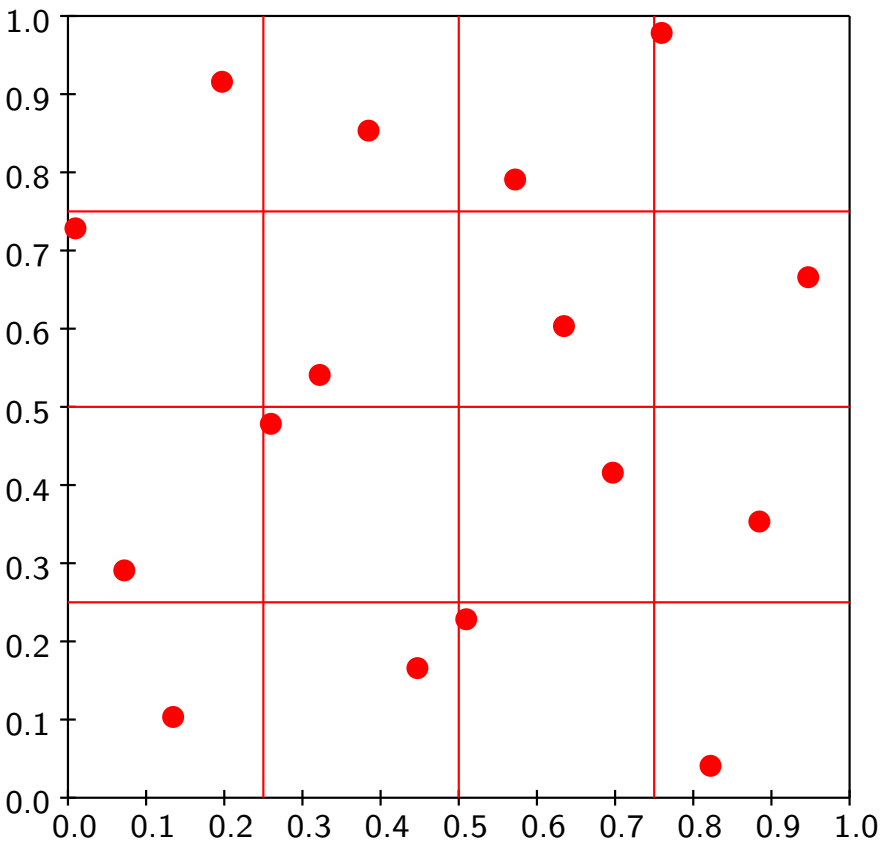


A (4,4) mapping

States of the chains

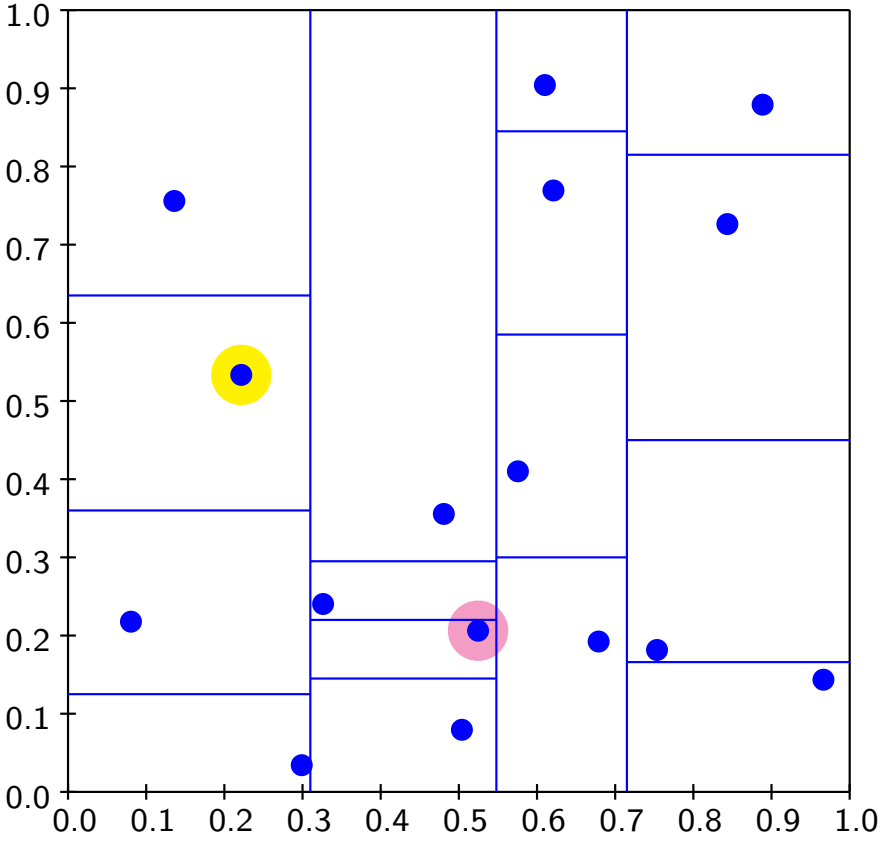


Sobol' net in 2 dimensions after random digital shift

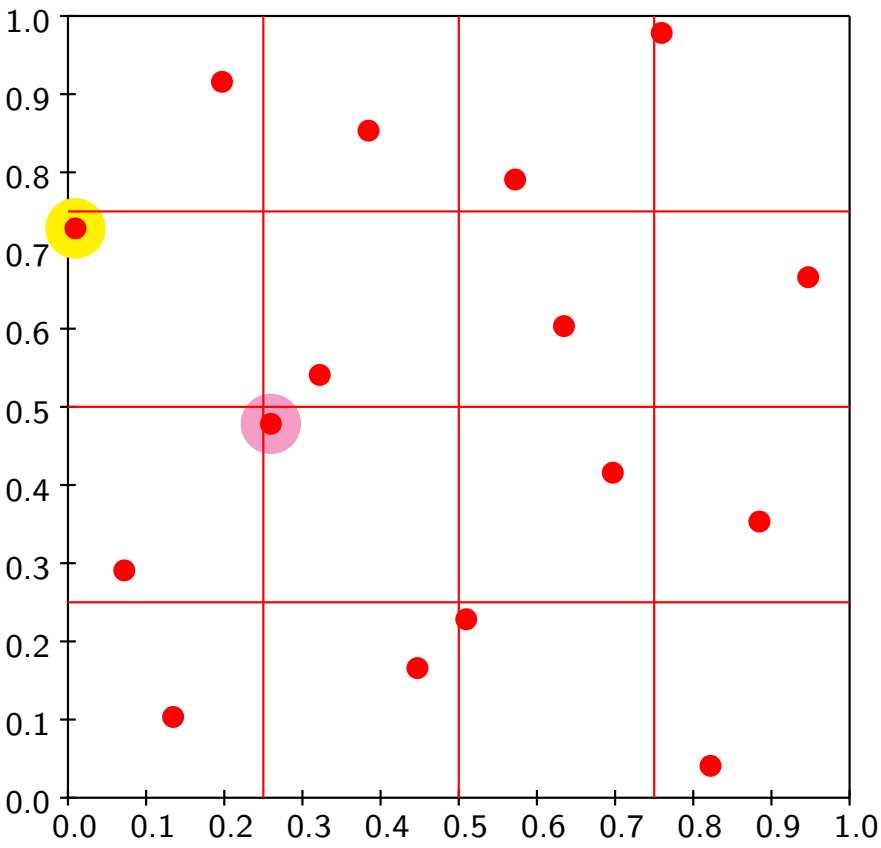


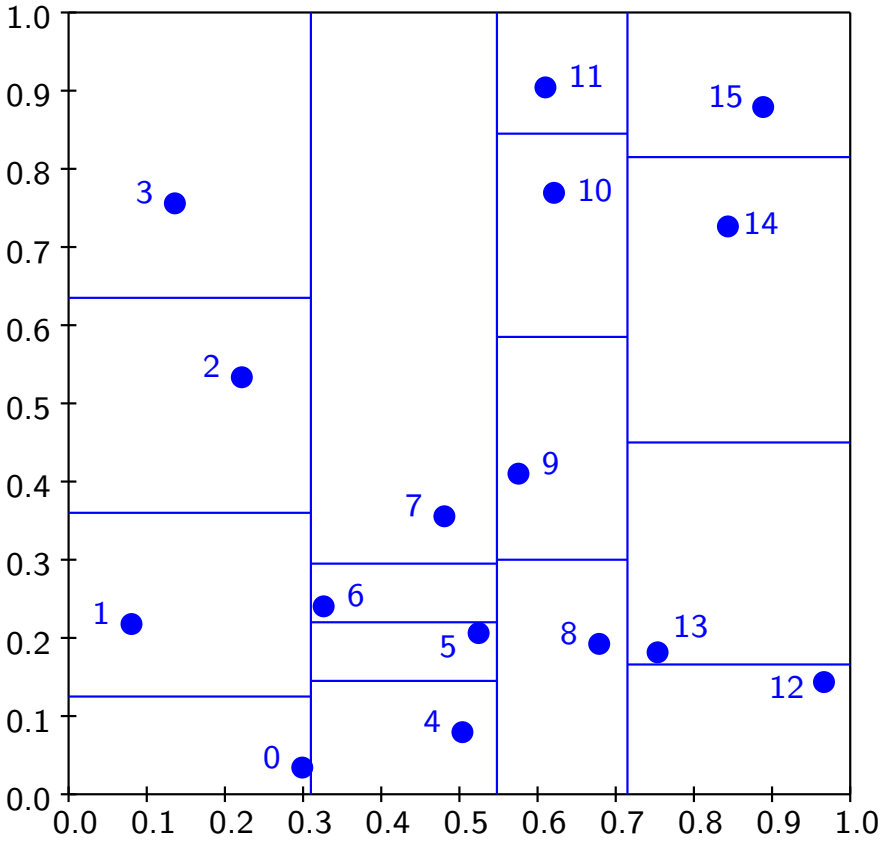
A (4,4) mapping

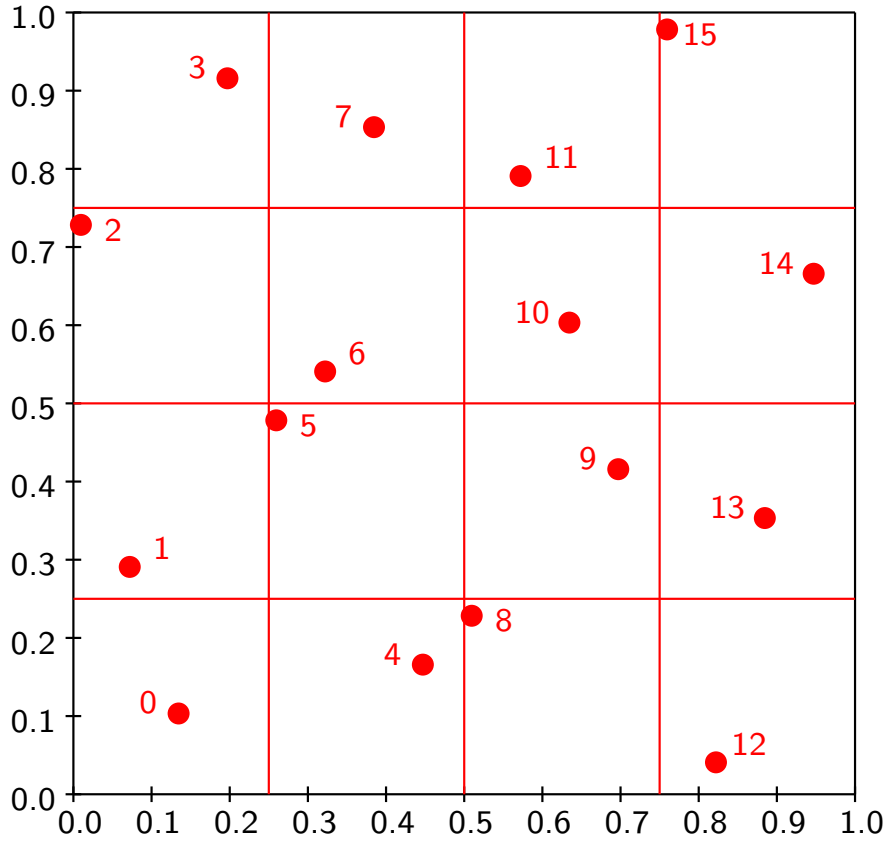
States of the chains



Sobol' net in 2 dimensions after random digital shift




1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
11
15
3
10
14
2
9
7
1
6
8
13
5
12
4
0


1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
15
3
7
11
2
14
10
6
5
9
13
1
8
4
0
12

Mapping chains to points when $\ell > 2$

2. Multivariate split sort:

$n_1 = n_2 = \cdots = 2$.

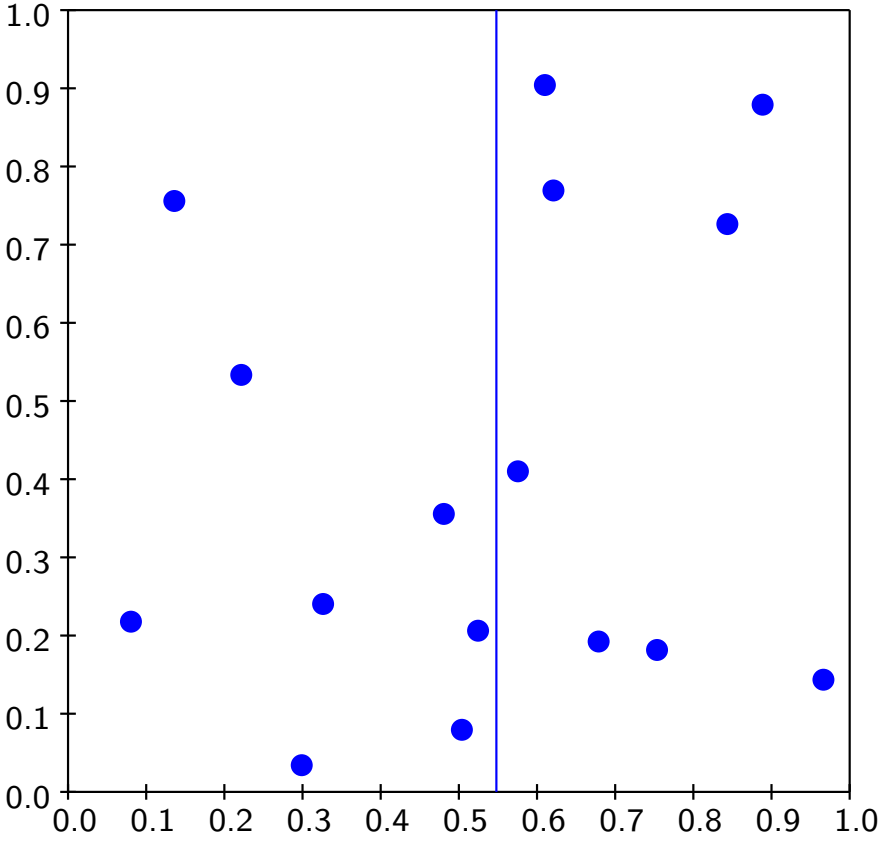
Sort by first coordinate in 2 packets.

Sort each packet by second coordinate in 2 packets.

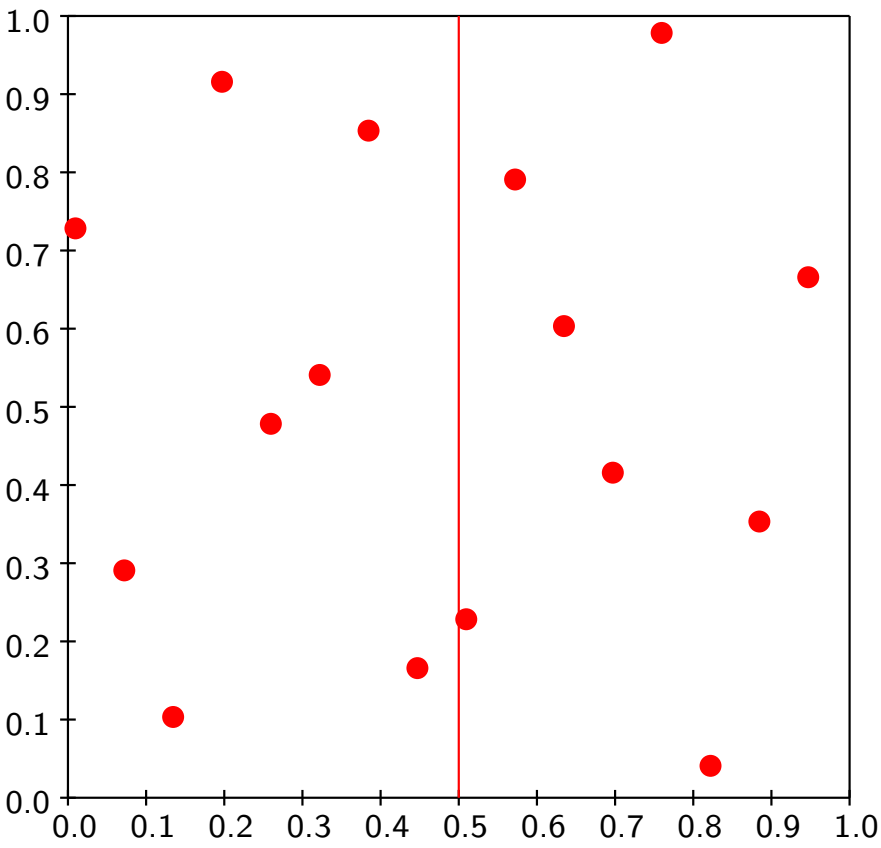
etc.

Mapping by split sort

States of the chains

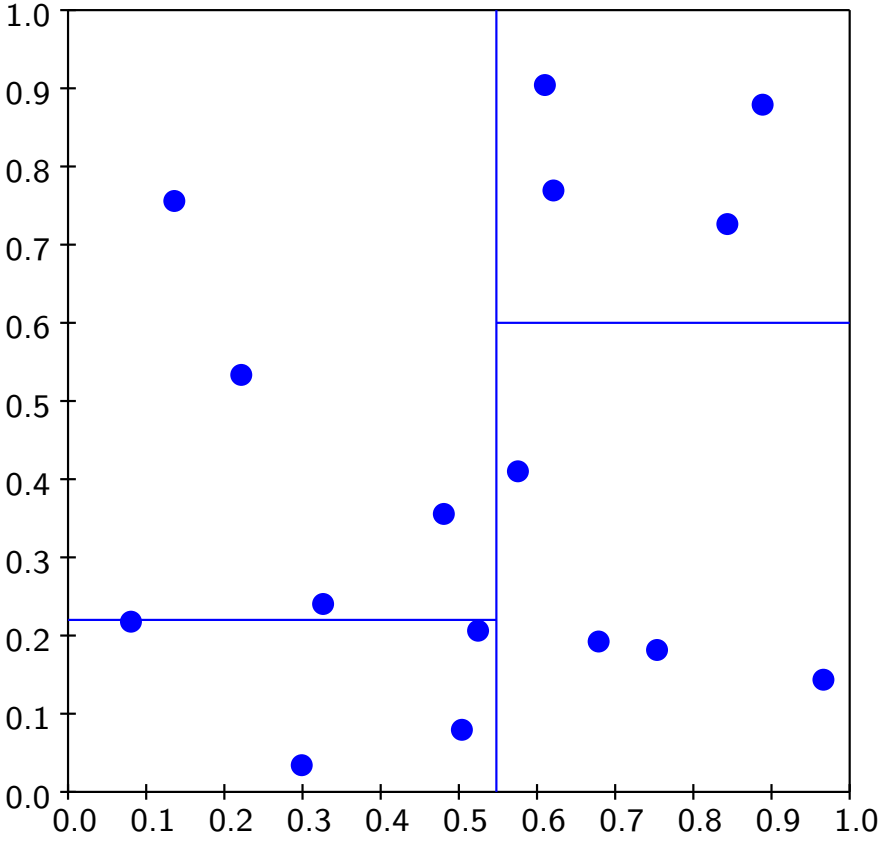


Sobol' net in 2 dimensions after random digital shift

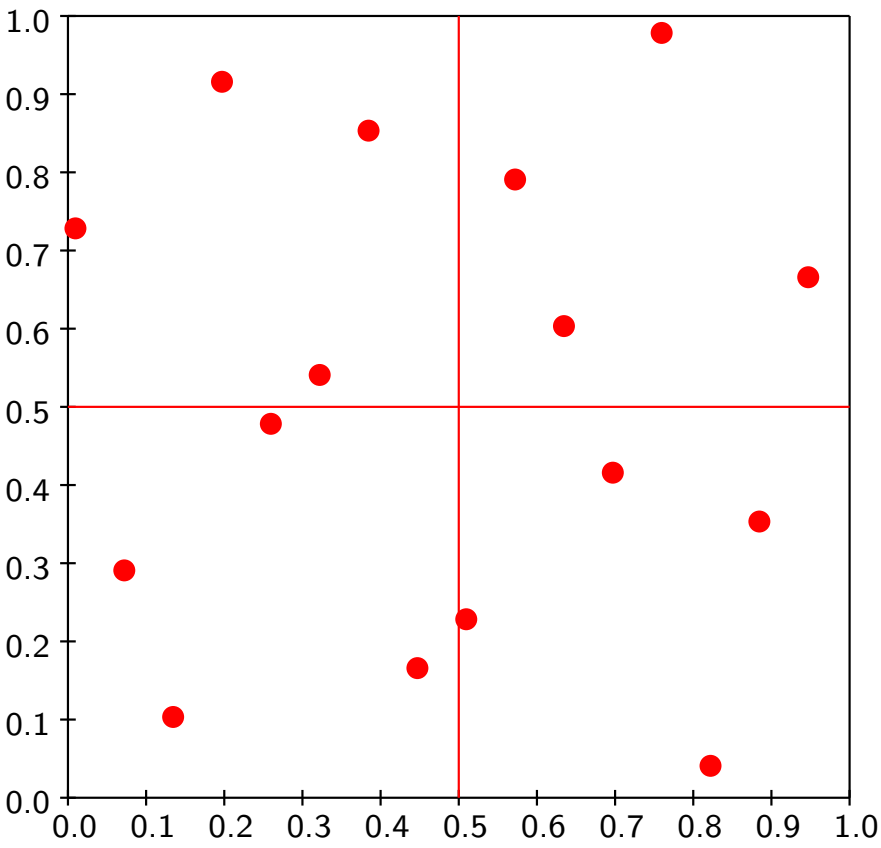


Mapping by split sort

States of the chains

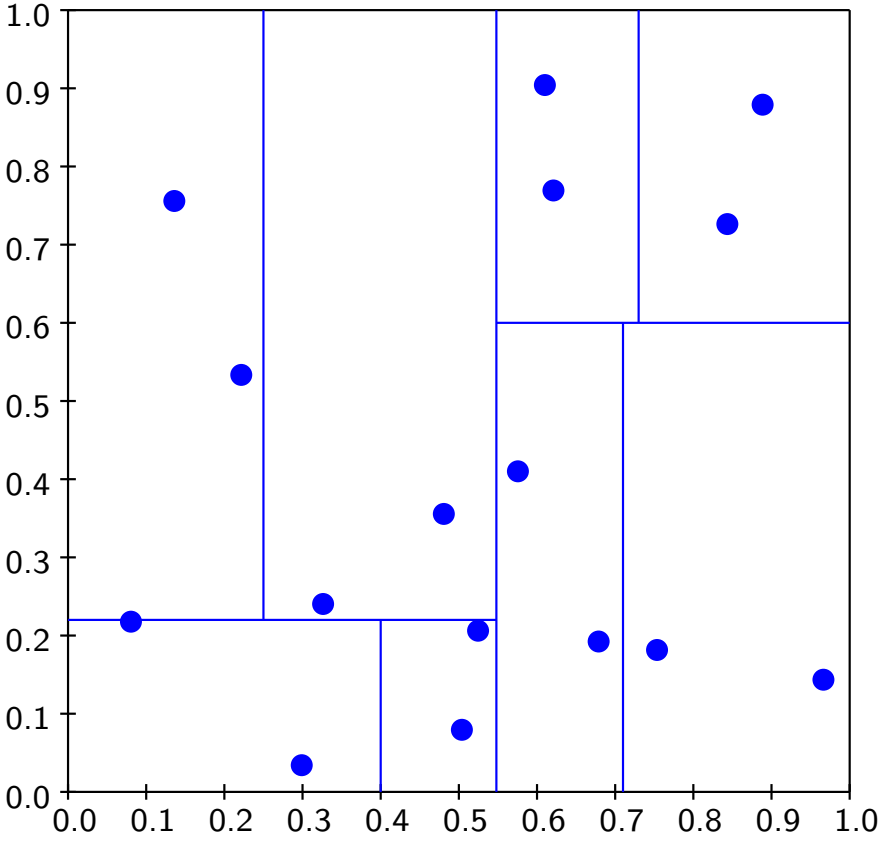


Sobol' net in 2 dimensions after random digital shift

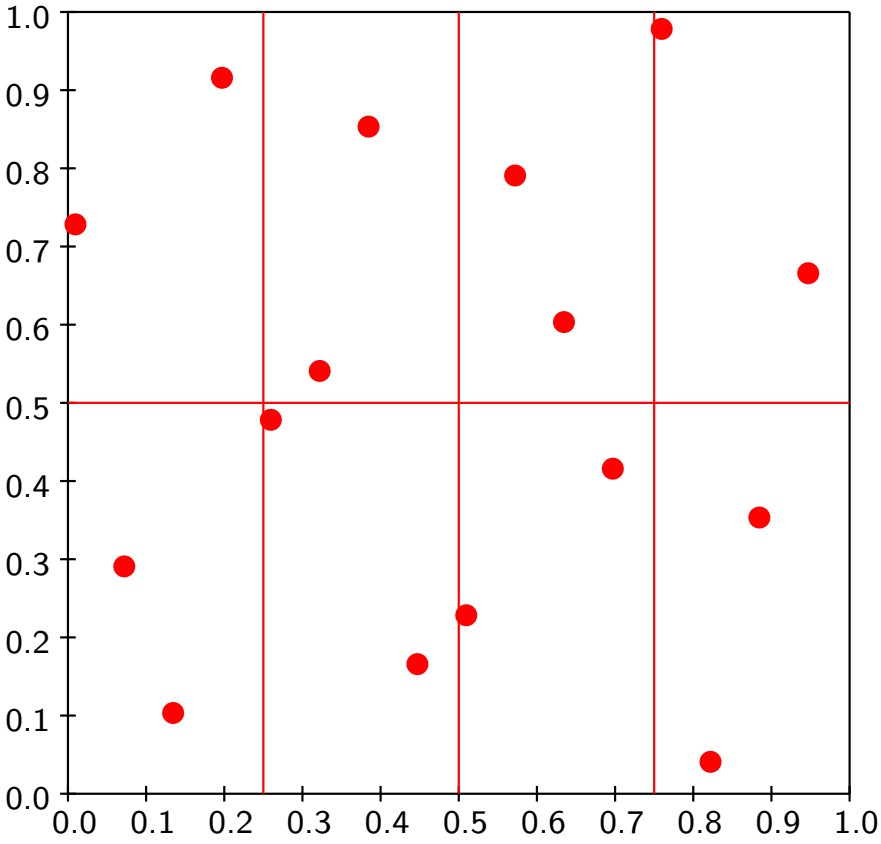


Mapping by split sort

States of the chains

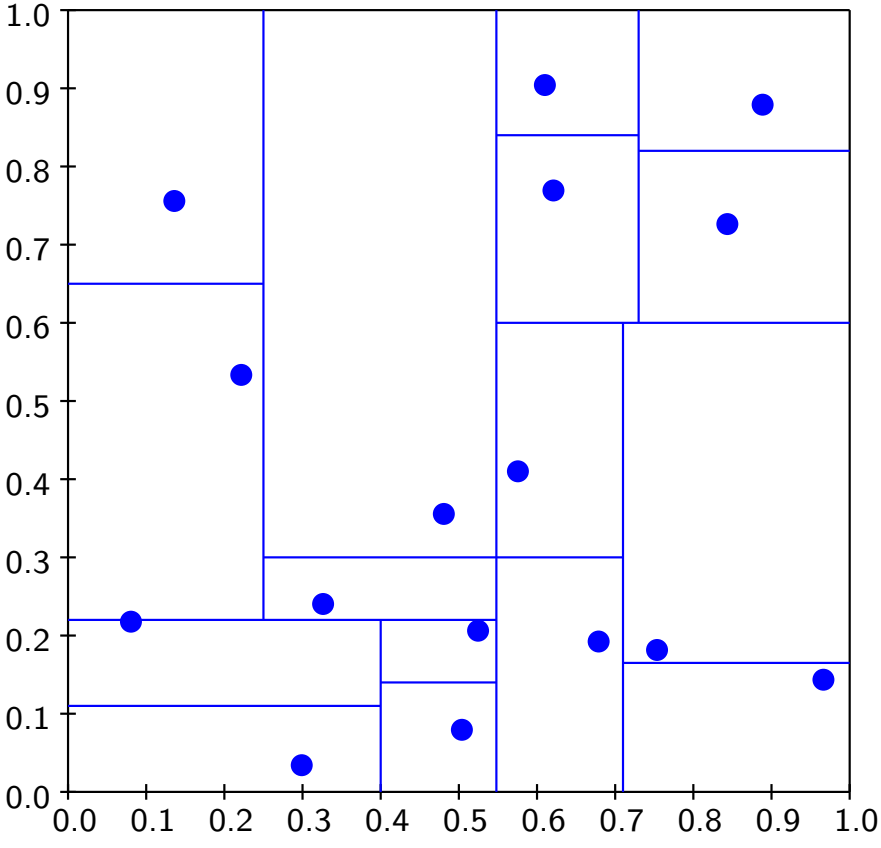


Sobol' net in 2 dimensions after random digital shift

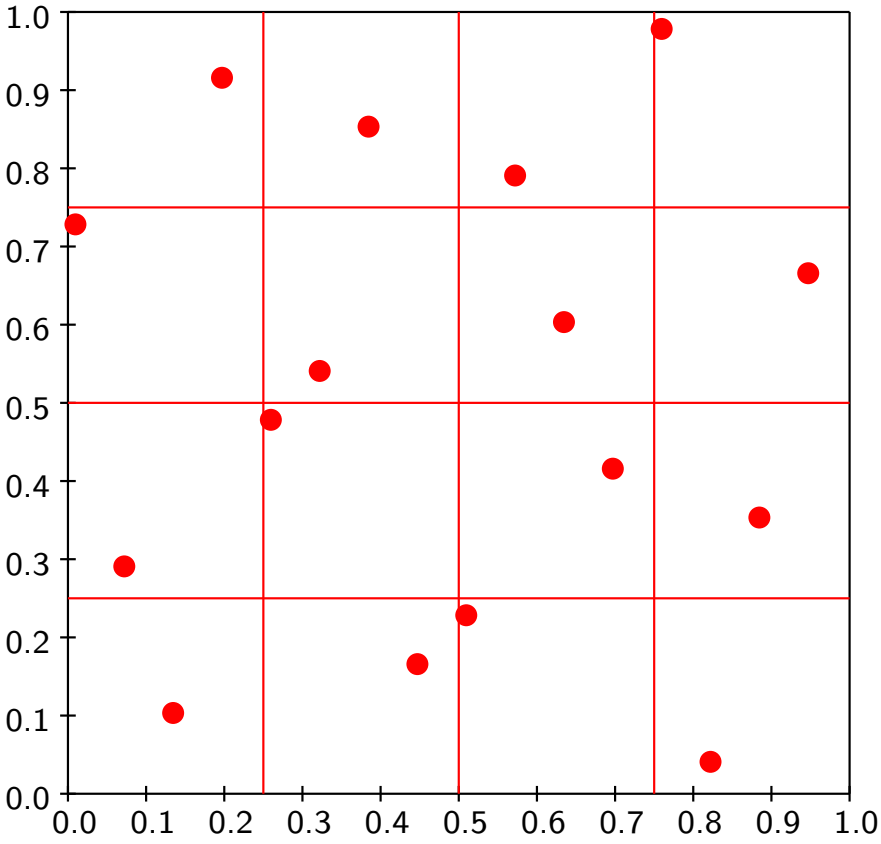


Mapping by split sort

States of the chains

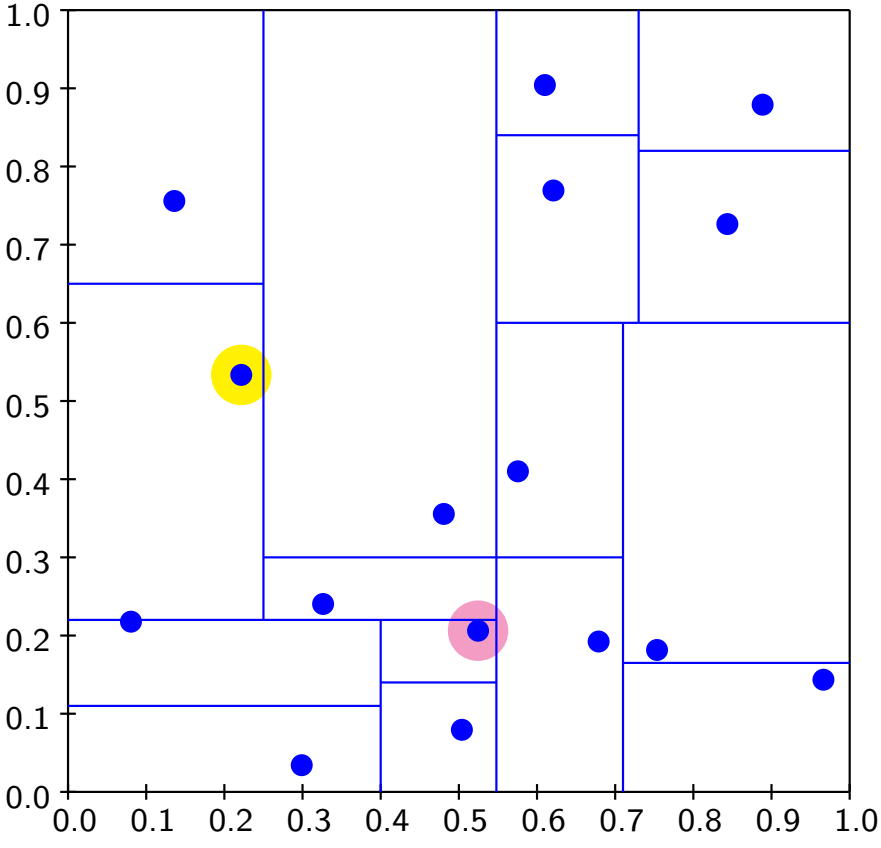


Sobol' net in 2 dimensions after random digital shift

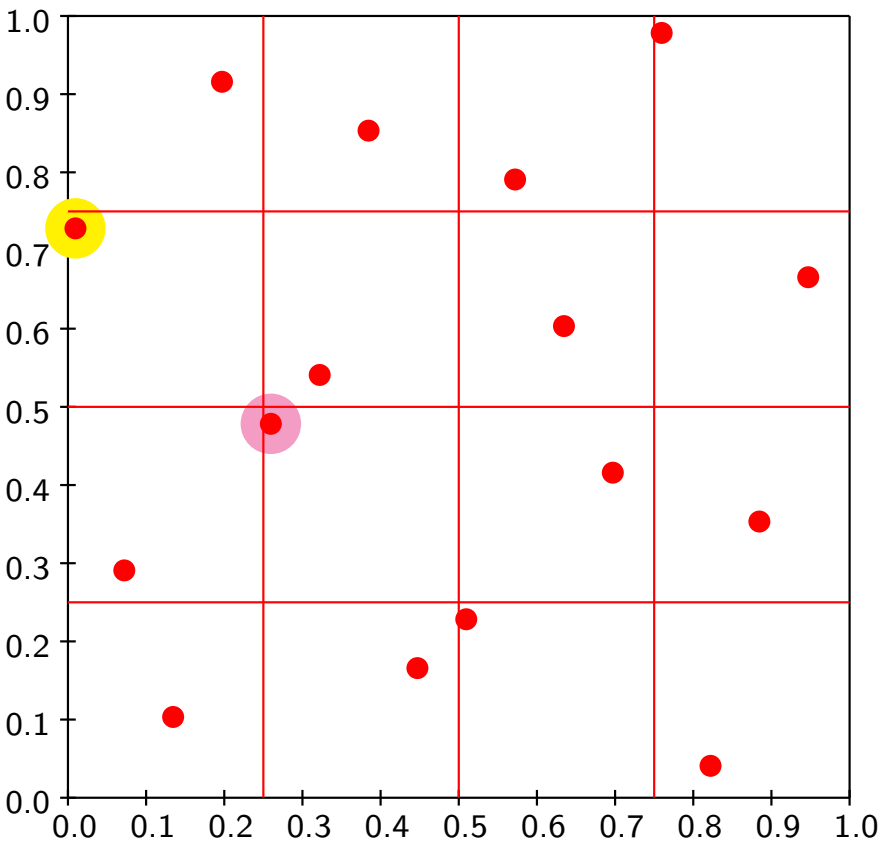


Mapping by split sort

States of the chains



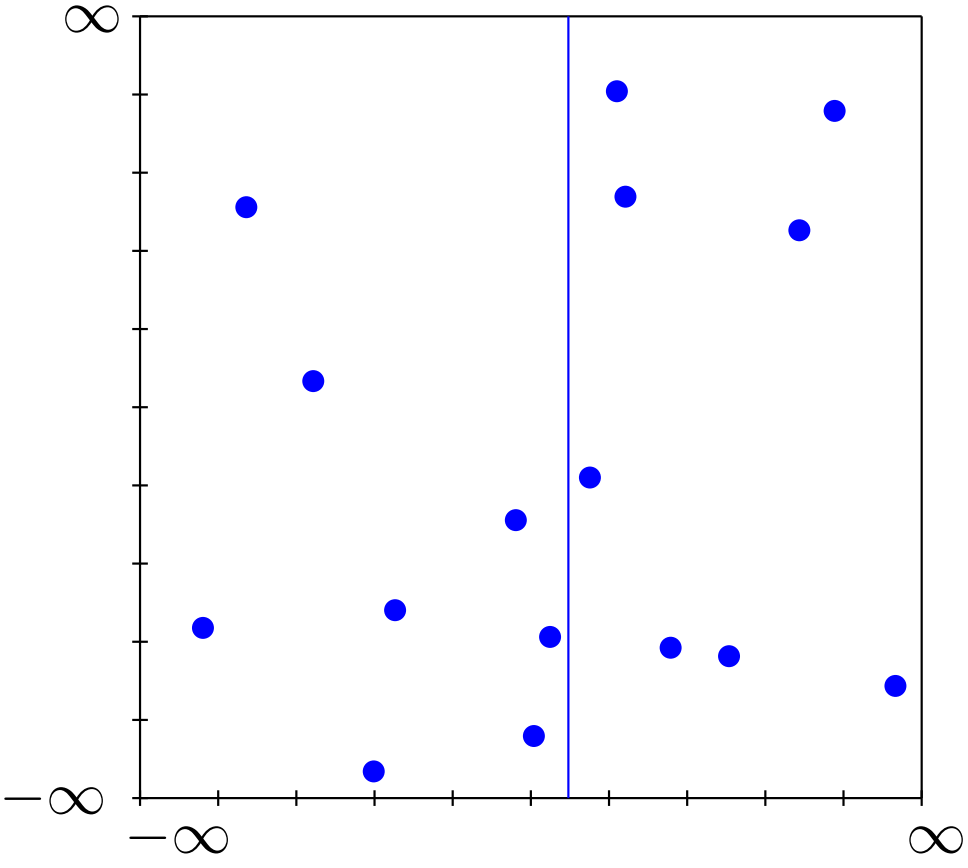
Sobol' net in 2 dimensions after random digital shift



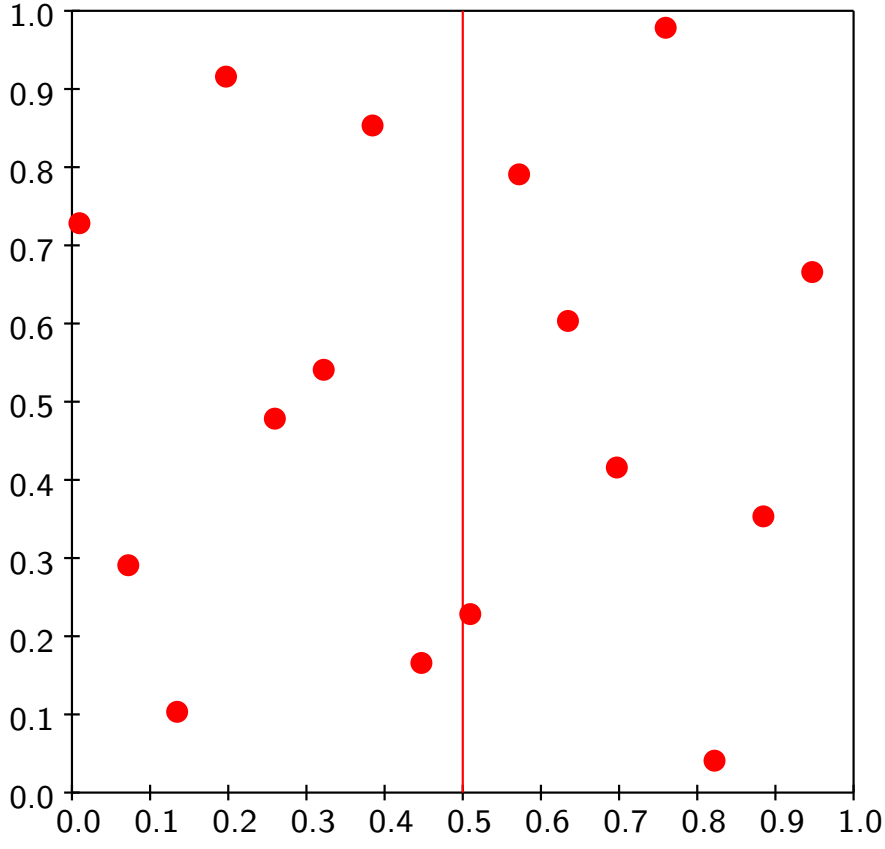
Mapping by batch sort and split sort

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

States of the chains



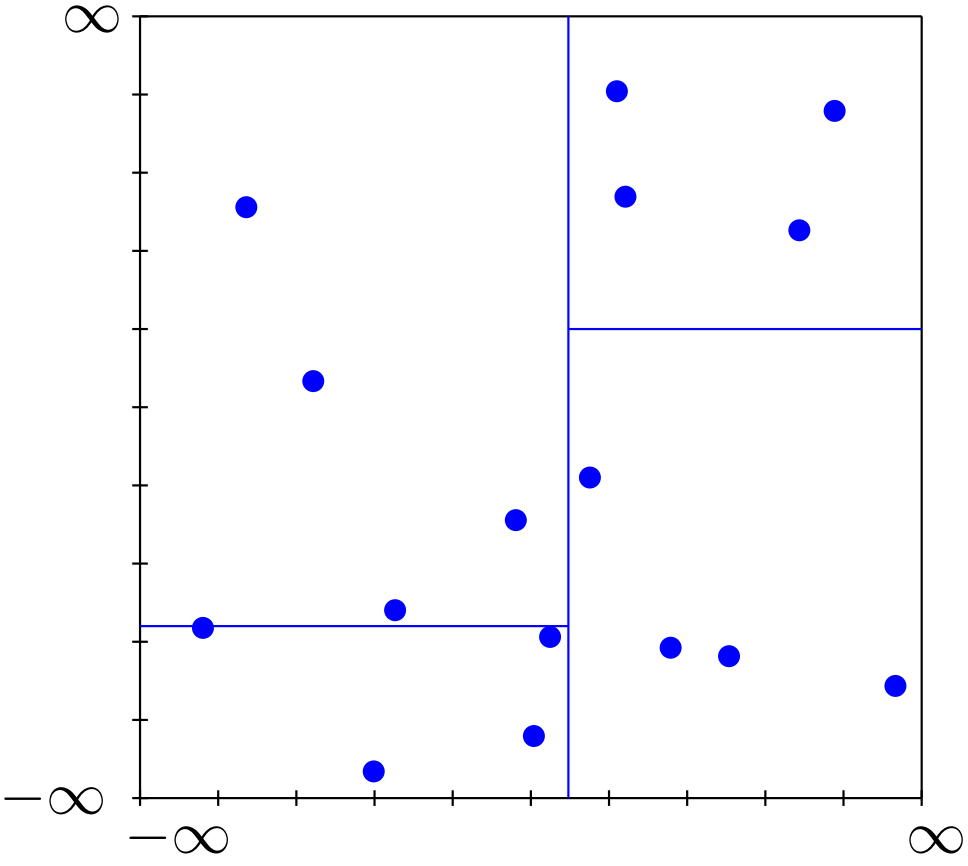
Sobol' net + digital shift



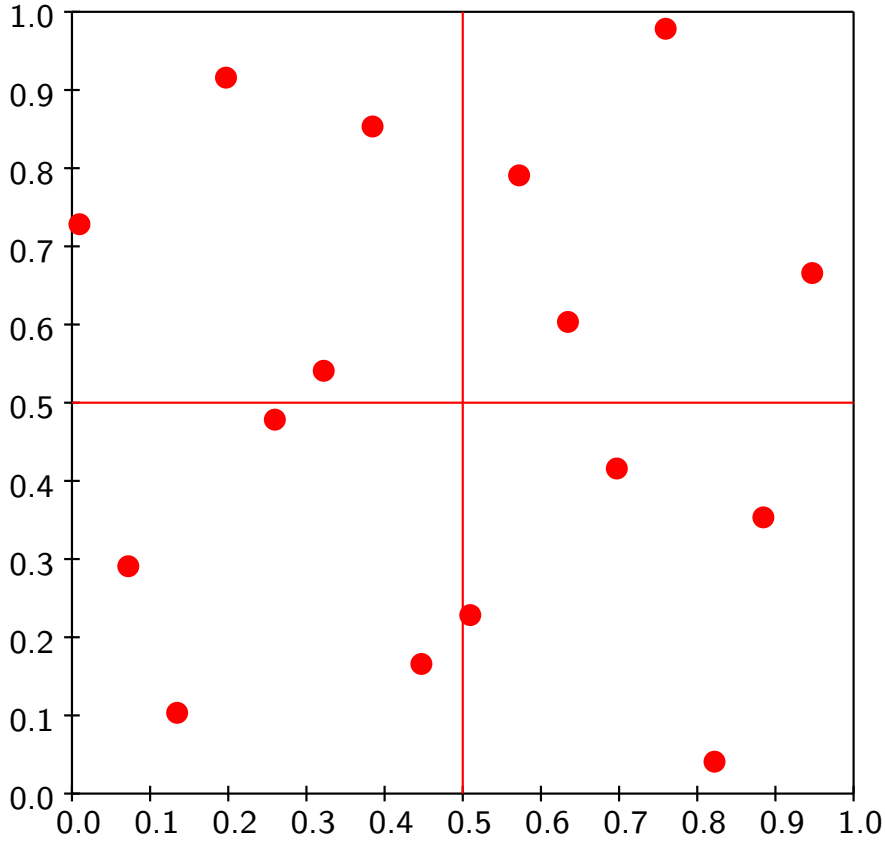
Mapping by batch sort and split sort

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

States of the chains



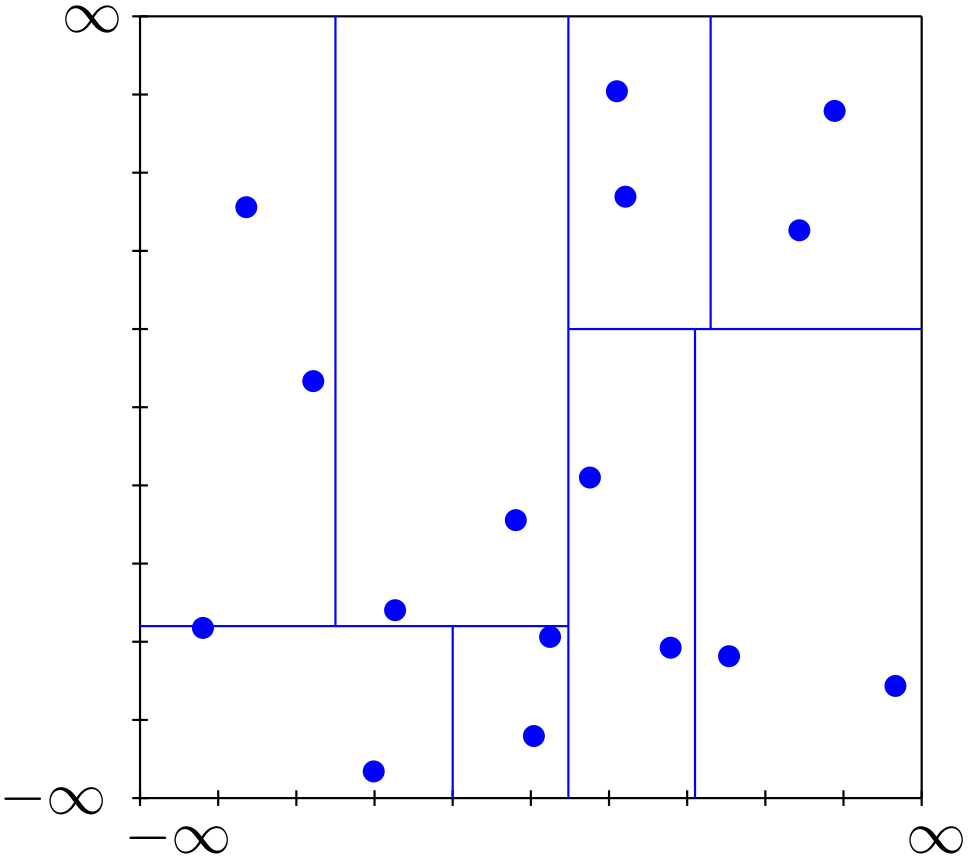
Sobol' net + digital shift



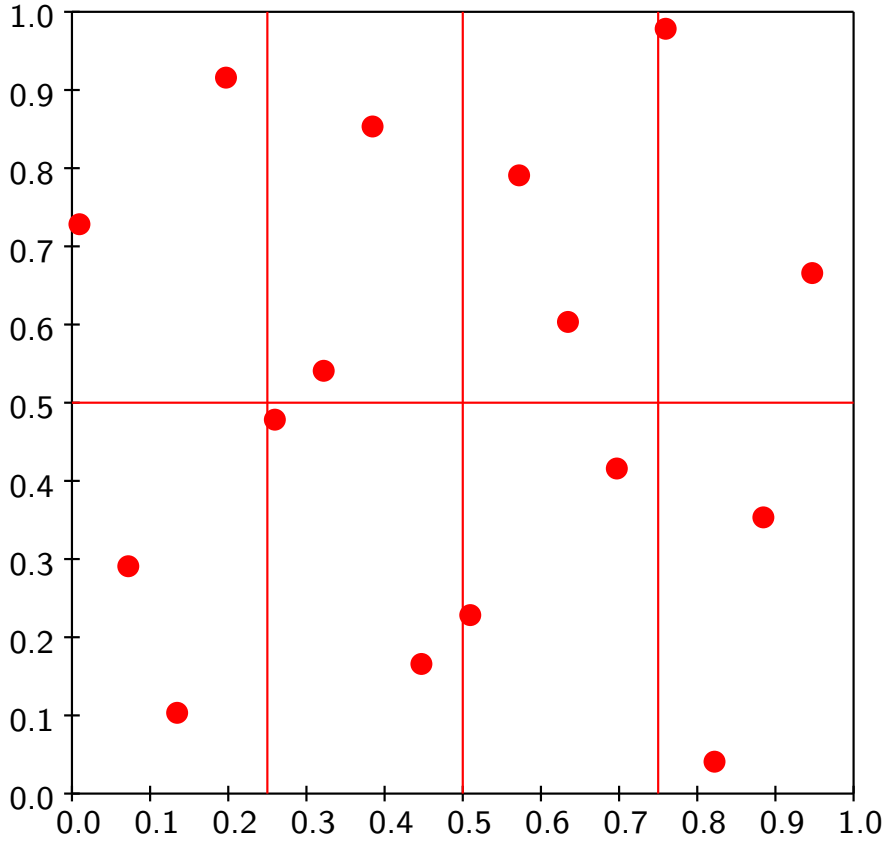
Mapping by batch sort and split sort

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

States of the chains



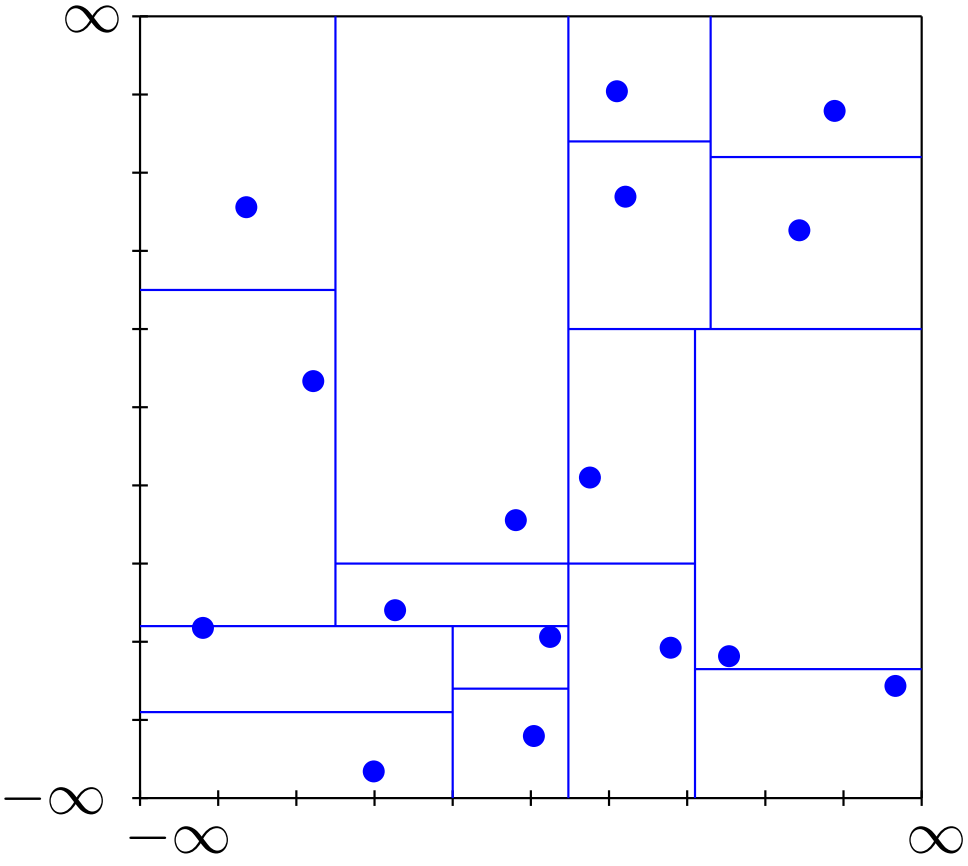
Sobol' net + digital shift



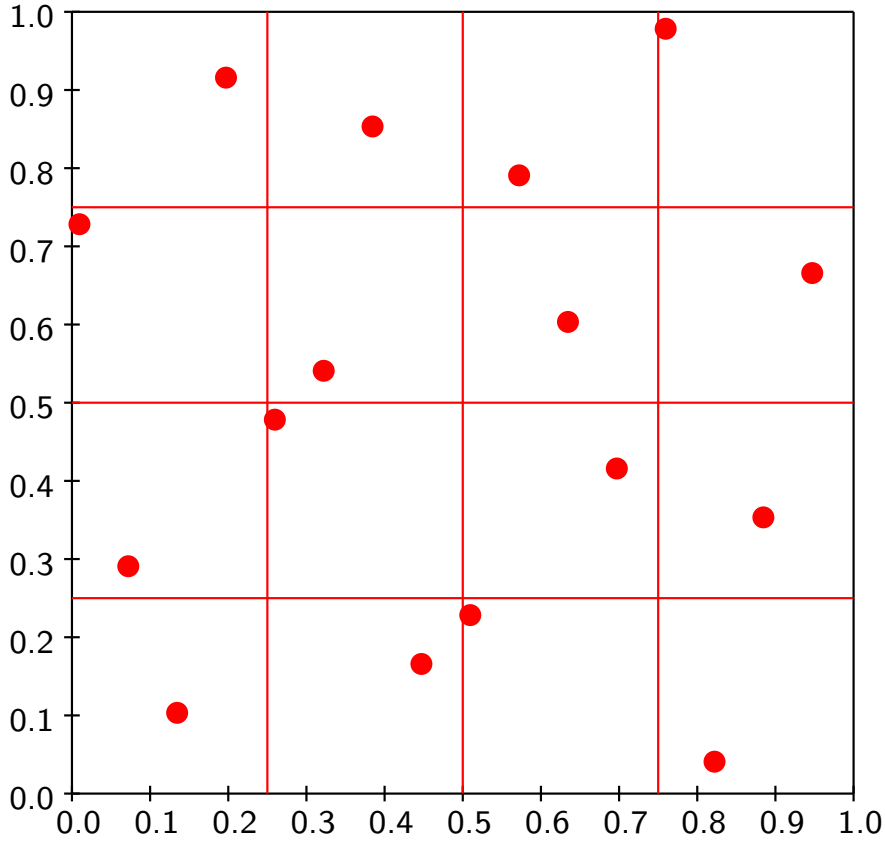
Mapping by batch sort and split sort

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

States of the chains



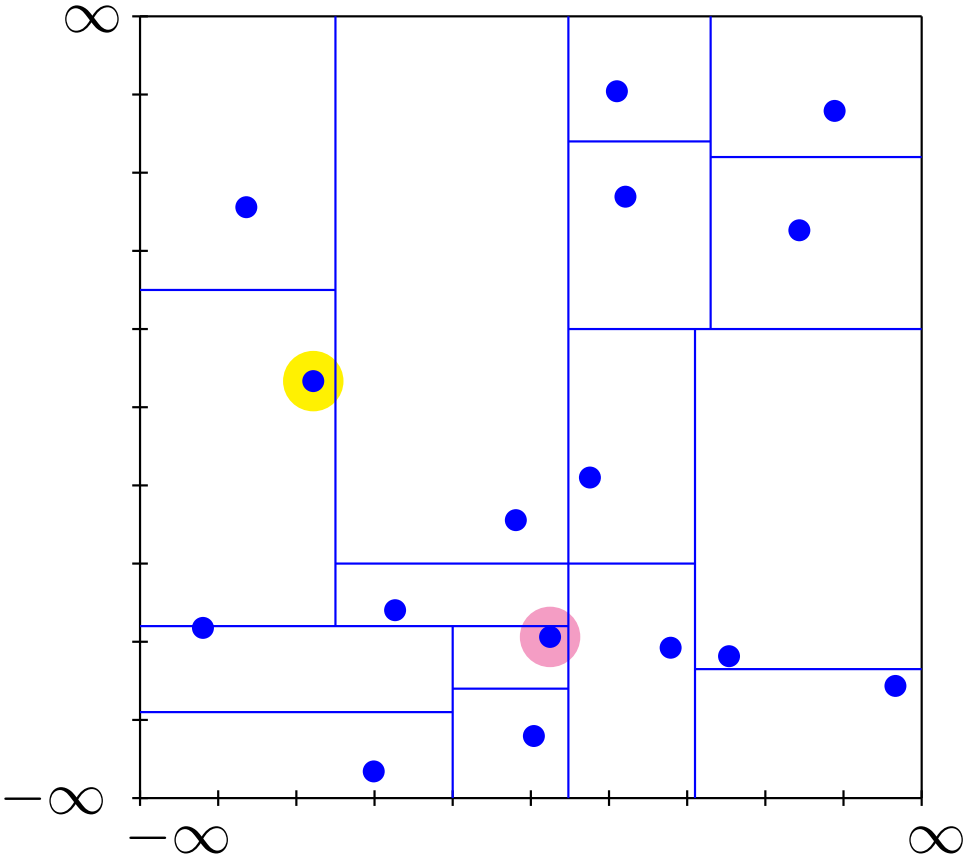
Sobol' net + digital shift



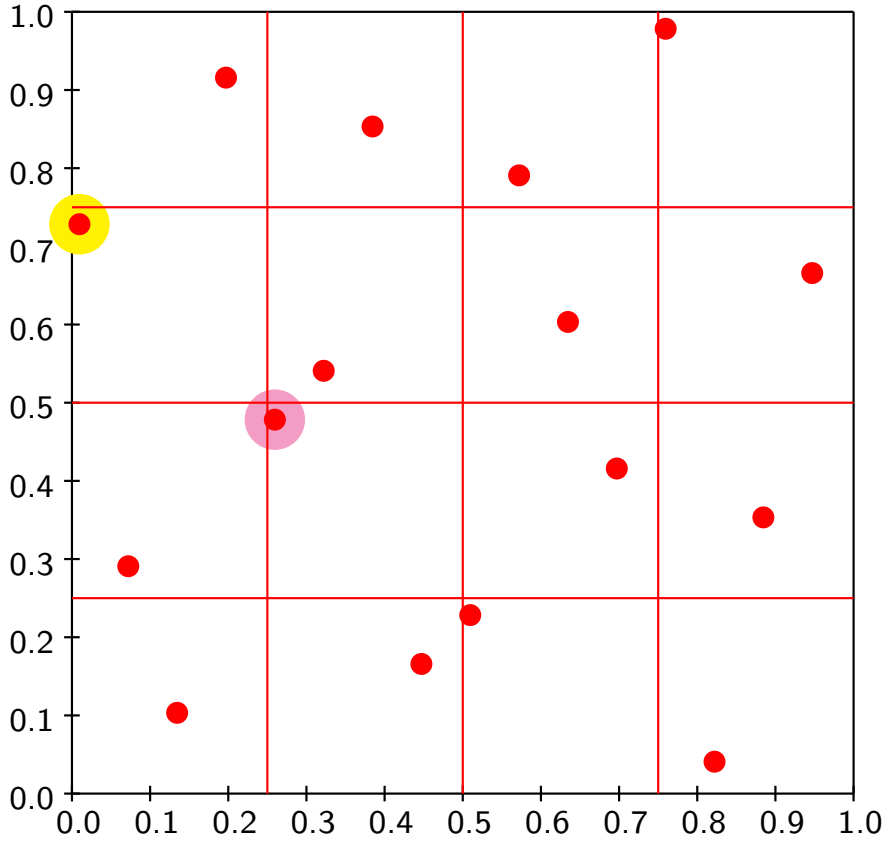
Mapping by batch sort and split sort

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

States of the chains



Sobol' net + digital shift



Lowering the state dimension

For $\ell > 1$: Define a transformation $h : \mathcal{X} \to [0,1)^c$ for $c < \ell$.
Sort the transformed points $h(X_{i,j})$ in c dimensions.
Now we only need $c + d$ dimensions for the RQMC point sets;
c for the mapping and d to advance the chain.

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

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

The mapping h with $c = 1$ can be based on a space-filling curve:
Wächter and Keller [2008] use a Lebesgue Z-curve and mention others;
Gerber and Chopin [2015] use a Hilbert curve and prove $o(n^{-1})$ convergence for the variance when used with digital nets and Owen nested scrambling. A Peano curve would also work in base 3.

Reality check: We only need a good pairing between states and RQMC points. Any good way of doing this is welcome!
Machine learning to the rescue?

Sorting by a Hilbert curve

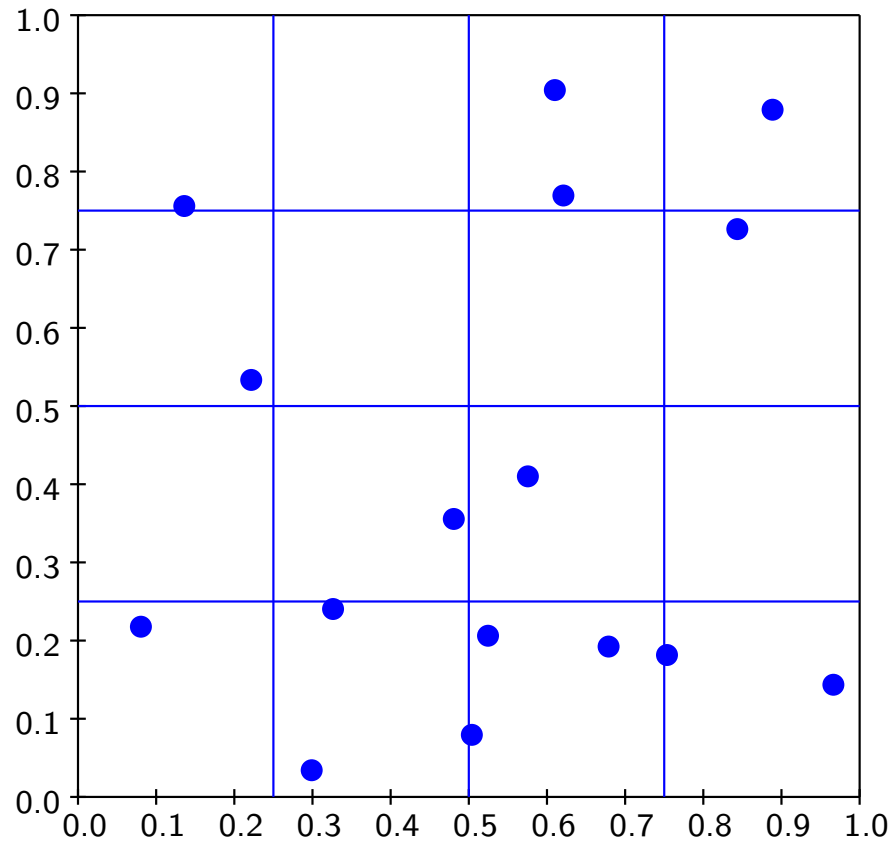
Suppose the state space is $\mathcal{X} = [0,1)^\ell$.
Partition this cube into $2^{m\ell}$ subcubes of equal size.
When a subcube contains more than one point (a collision), we could split it again in 2^ℓ. But in practice, we rather fix m and neglect collisions.

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

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

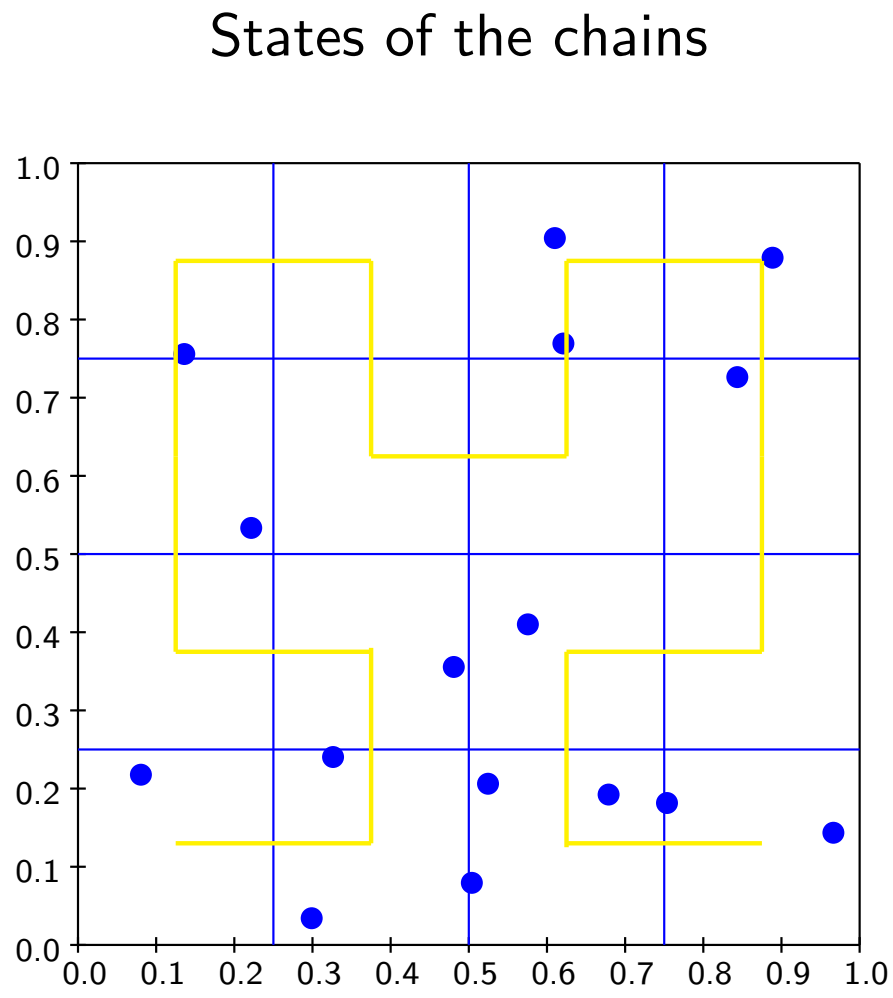
Hilbert curve sort

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



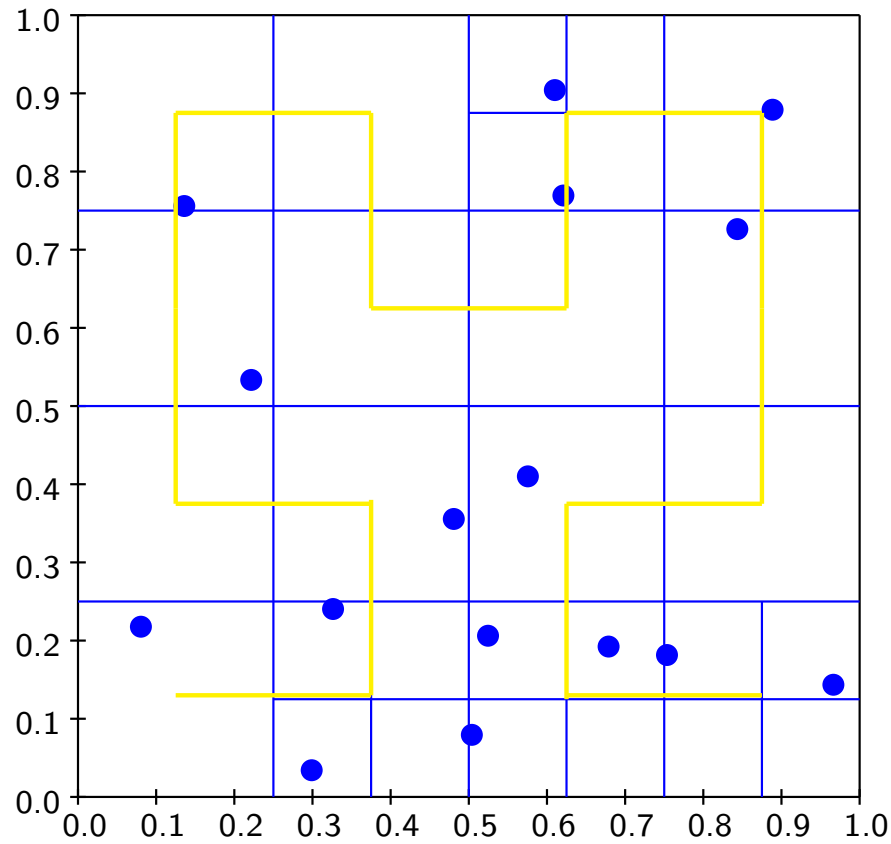
Hilbert curve sort

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



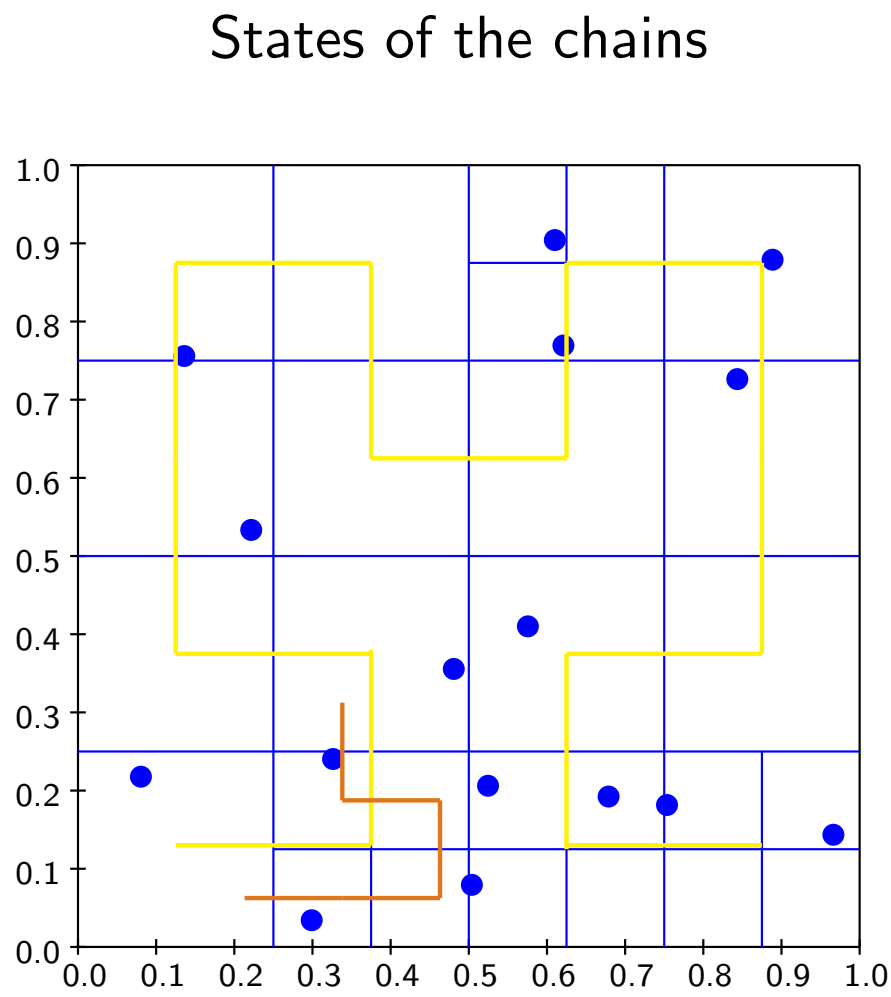
Hilbert curve sort

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



Hilbert curve sort

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



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

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

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

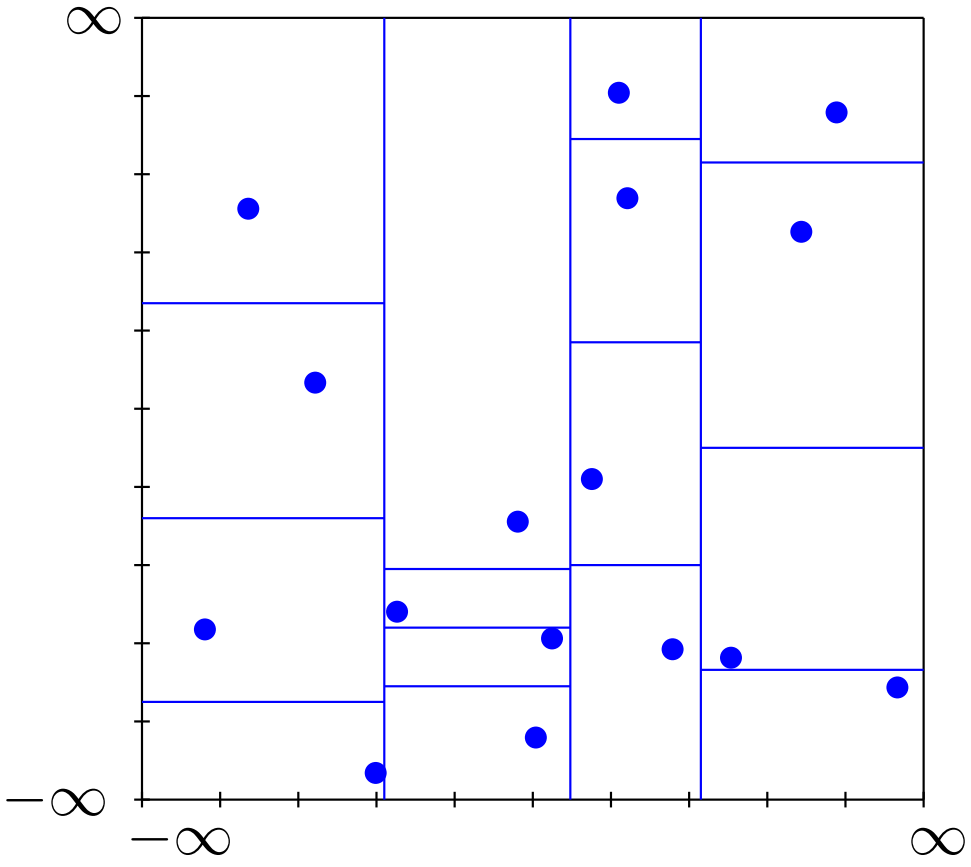
Not easy to find a good ψ in general!
Gerber and Chopin [2015] propose using a logistic transformation for each coordinate, combined with trial and error.

A lousy choice could possibly damage efficiency.

Hilbert curve batch sort

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

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



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.

Proved convergence results

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

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

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

$$V_\infty(g) = \int_0^1 \left| \frac{dg(x)}{dx} \right| dx \qquad \text{(corresponding variation of } g\text{)}$$

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

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

We have

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

Convergence results and proofs, $\ell = 1$

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

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

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

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

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

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

Convergence results and proofs, $\ell = 1$

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

Proposition. (Variance bound.) Under Assumption 2,

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

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

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

These bounds are uniform in j.

Convergence results and proofs, $\ell > 1$

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

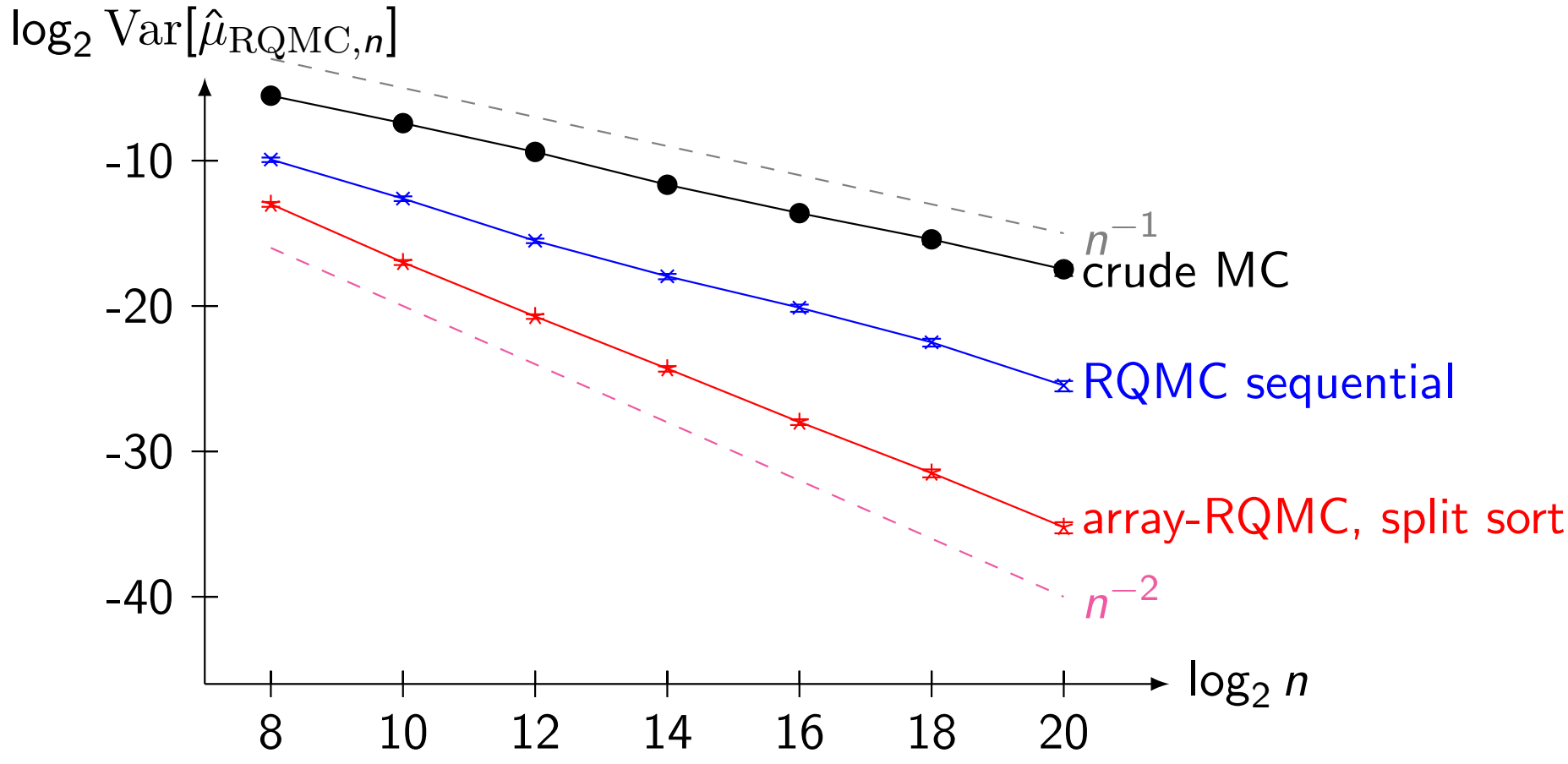
Gerber and Chopin (2015) proved $o(n^{-1})$ for the variance, for Hilbert sort and digital net with nested scrambling.

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

$S(0) = 100$, $K = 100$, $r = \ln(1.09)$, $\sigma = 0.2$,

$t_j = (230 + j)/365$, for $j = 1, \ldots, \tau = 10$. $\quad \mathrm{Var} \approx \mathcal{O}(n^{-\alpha})$.

Sort	RQMC points	$\alpha = \frac{\log_2 \mathrm{Var}[\bar{Y}_{n,j}]}{\log_2 n}$	VRF	CPU (sec)
Split sort	SS	-1.38	2.0×10^2	3093
	Sobol	-2.04	4.0×10^6	1116
	Sobol+NUS	-2.03	2.6×10^6	1402
	Korobov+baker	-2.00	2.2×10^6	903
Batch sort	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.

Example: Asian Call Option

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

Sort	RQMC points	$\frac{\log_2 \mathrm{Var}[\bar{Y}_{n,j}]}{\log_2 n}$	VRF	CPU (sec)
Split sort	SS	-1.38	2.0×10^2	3093
	Sobol	-2.04	4.0×10^6	1116
	Sobol+NUS	-2.03	2.6×10^6	1402
	Korobov+baker	-2.00	2.2×10^6	903
Batch sort	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.

The small Markov chain

RQMC points	$\alpha = \frac{\log_2 \mathrm{MISE}}{\log_2 n}$
Monte Carlo	-1.00
Sobol+DS	-1.88
Lattice	-1.91

A small example with a one-dimensional state

Let $\theta \in [0,1)$ and let G_θ be the cdf of $Y = \theta U + (1-\theta)V$, where U, V are indep. $U(0,1)$. We define a Markov chain by

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

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

We consider various functions g, all with $\mathbb{E}[g(X_j)] = 0$:

$g(x) = x - 1/2$, $g(x) = x^2 - 1/3$,

$g(x) = \sin(2\pi x)$, $g(x) = e^x - e + 1$ (all smooth),

$g(x) = (x - 1/2)^+ - 1/8$ (kink), $\quad g(x) = \mathbb{I}[x \leq 1/3] - 1/3$ (step).

We pretend we do not know $\mathbb{E}[g(X_j)]$, and see how well we can estimate it by simulation.

We also want to see how well we can estimate the exact distribution of X_j (uniform) by the empirical distribution of $X_{0,j}, \ldots, X_{n-1,j}$.

One-dimensional example

We take $\rho = 0.3$ and $j = 5$.
For array-RQMC, we take $X_{i,0} = w_i = (i - 1/2)/n$.

We tried different array-RQMC variants, 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

One-dimensional example

We take $\rho = 0.3$ and $j = 5$.
For array-RQMC, we take $X_{i,0} = w_i = (i - 1/2)/n$.

We tried different array-RQMC variants, 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

We also looked at uniformity measures of the set of n states at step j. For example, the Kolmogorov-Smirnov (KS) and Cramer von Mises (CvM) test statistics, denoted $\mathrm{KS_j}$ and D_j. With ordinary MC, $\mathbb{E}[\mathrm{KS_j}]$ and $\mathbb{E}[D_j]$ converge as $\mathcal{O}(n^{-1})$ for any j.

For stratification, we have a proof that

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

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 $\log_2 n$	$\log_2 \mathrm{Var}[\bar{Y}_{n,j}]$			
	$X_j - \frac{1}{2}$	$X_j^2 - \frac{1}{3}$	$(X_j - \frac{1}{2})^+ - \frac{1}{8}$	$\mathbb{I}[X_j \le \frac{1}{3}] - \frac{1}{3}$
MC	-1.02	-1.01	-1.00	-1.02
LHS	-0.99	-1.00	-1.00	-1.00
SS	-1.98	-2.00	-2.00	-1.49
SSA	-2.65	-2.56	-2.50	-1.50
Sobol	-3.22	-3.14	-2.52	-1.49
Sobol+baker	-3.41	-3.36	-2.54	-1.50
Sobol+NUS	-2.95	-2.95	-2.54	-1.52
Korobov	-2.00	-1.98	-1.98	-1.85
Korobov+baker	-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

Density estimation

slope vs $\log_2 n$	$\log_2 \mathbb{E}[\mathrm{KS}_j^2]$	$\log_2 \mathbb{E}[D_j^2]$	MISE hist. 64
MC	-1.00	-1.00	-1.00
SS	-1.42	-1.50	-1.47
Sobol	-1.46	-1.46	-1.48
Sobol+baker	-1.50	-1.57	-1.58
Korobov	-1.83	-1.93	-1.90
Korobov+baker	-1.55	-1.54	-1.52

Conclusion

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

Many other sorting strategies remain to be explored.

Other examples and applications. Higher dimension.

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

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*, 2017. `http://dx.doi.org/10.1016/j.matcom.2016.07.010`.

- P. L'Ecuyer and C. Sanvido. Coupling from the past with randomized quasi-Monte Carlo. *Mathematics and Computers in Simulation*, 81(3):476–489, 2010.
- C. Wächter and A. Keller. Efficient simultaneous simulation of Markov chains. *Monte Carlo and Quasi-Monte Carlo Methods 2006*, pages 669–684, Berlin, 2008. Springer-Verlag.

Some basic references on QMC and RQMC:

- J. Dick and F. Pillichshammer. *Digital Nets and Sequences: Discrepancy Theory and Quasi-Monte Carlo Integration*. Cambridge University Press, Cambridge, U.K., 2010.
- P. L'Ecuyer. Quasi-Monte Carlo methods with applications in finance. *Finance and Stochastics*, 13(3):307–349, 2009.
- 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.
- *Monte Carlo and Quasi-Monte Carlo Methods 2016, 2014, 2012, 2010, ...* Springer-Verlag, Berlin.

Approx. Zero-Variance Importance Sampling:

L'Ecuyer and B. Tuffin, "Approximate Zero-Variance Simulation," Proceedings of the 2008 Winter Simulation Conference, 170–181.
http://www.informs-sim.org/wsc08papers/019.pdf