Monte Carlo

Quasi Monte-carlo (QMC)

Francesco Banterle, Ph.D. - July 2021

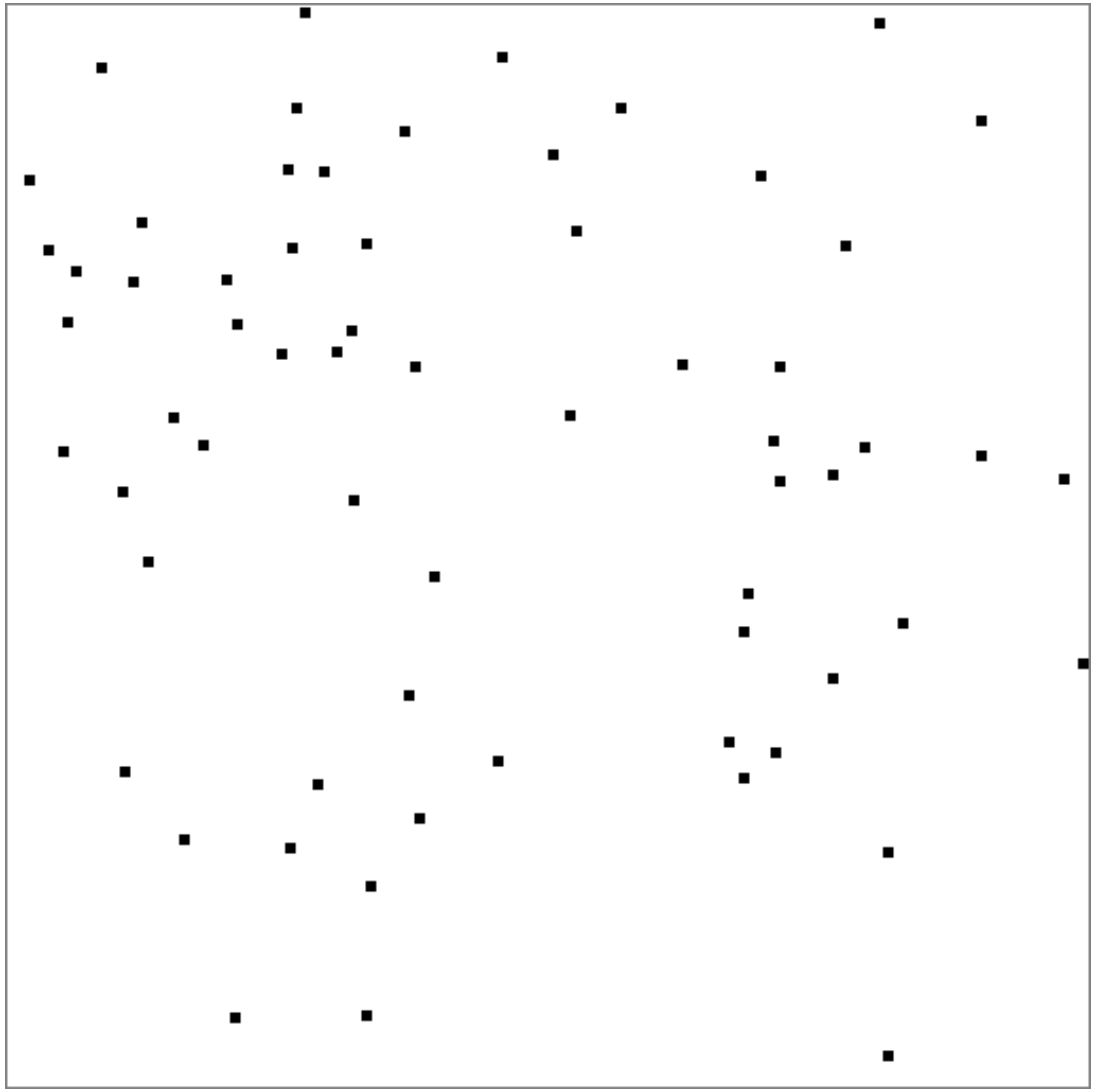
Quasi Monte-Carlo

Introduction

- In Monte-Carlo, we have seen that we use randomness to estimate averages, quantiles, and ratios.
  - The justification why this works is thanks to the **Law of Large Numbers**.
- In Quasi Monte-Carlo or QMC, our goal is to “*bend*” this law using deterministic samples.
  - We may get better results than the ones of classic Monte-Carlo!

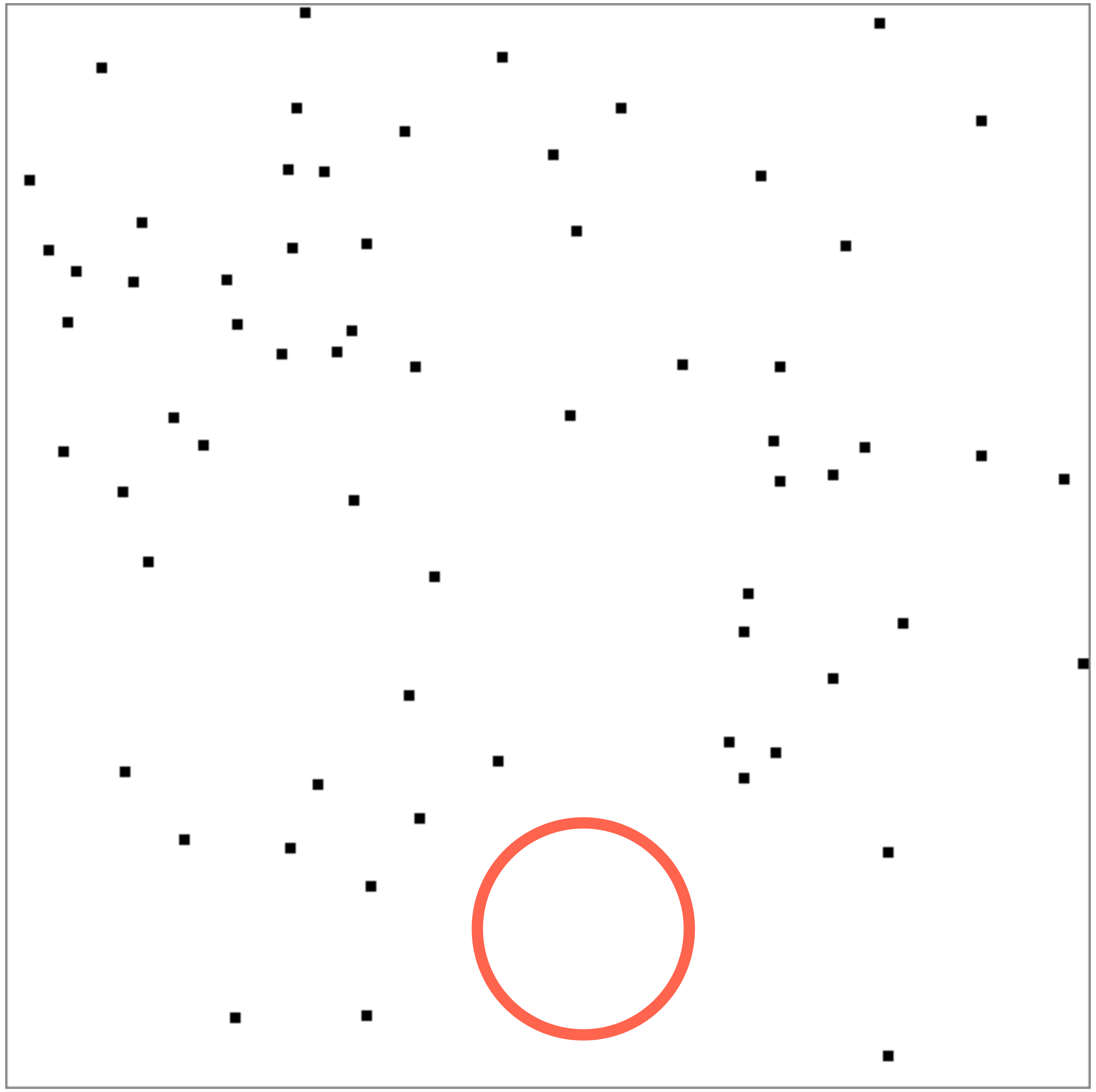
Quasi Monte-Carlo

Motivation



Quasi Monte-Carlo

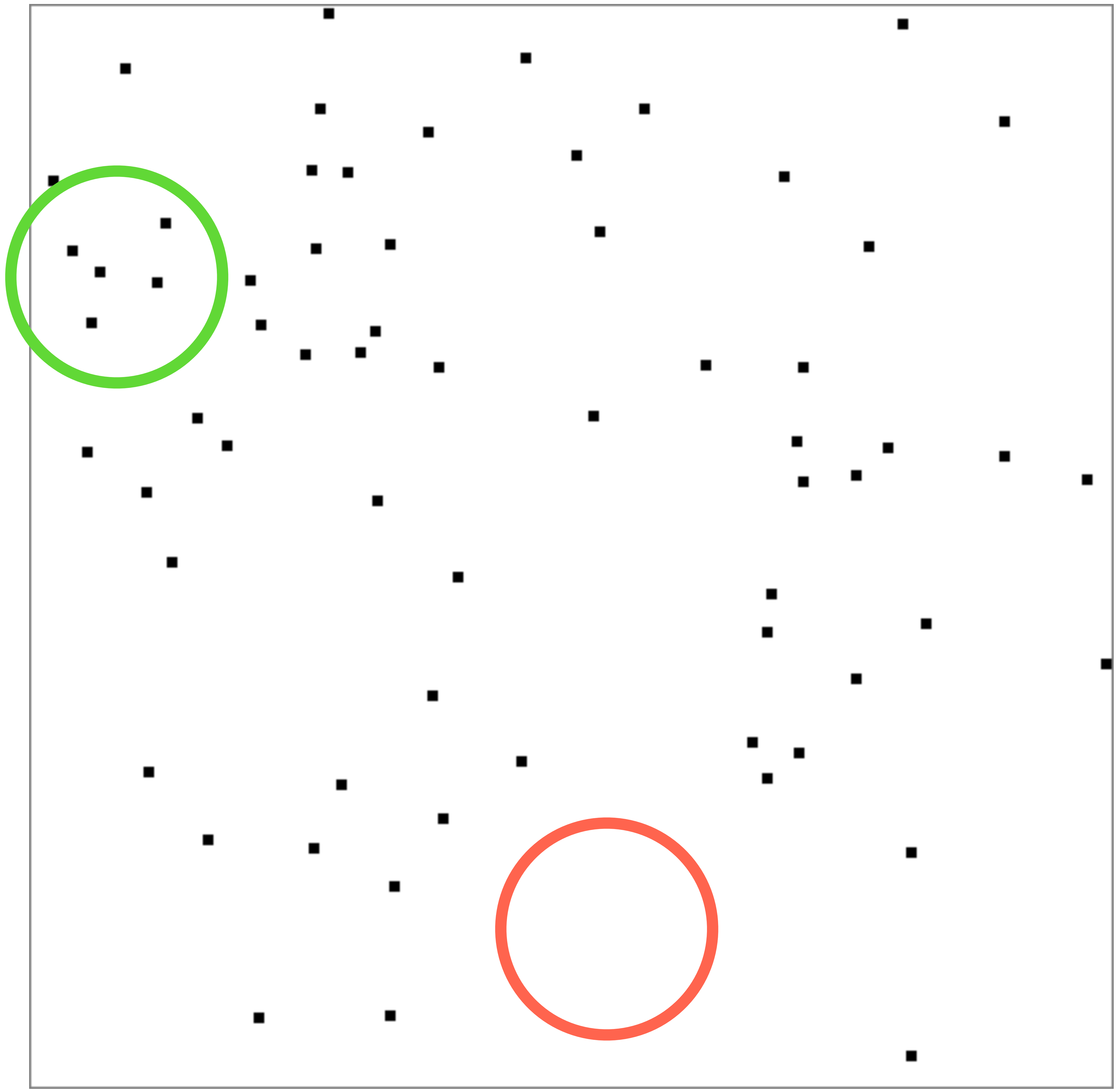
Motivation



Gaps

Quasi Monte-Carlo

Motivation



Cluster

Gaps

Quasi Monte-Carlo

Introduction

- We still estimate:

$$\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x}_i) p(\mathbf{x}_i).$$

- Now, our samples, $\mathbf{x}_i$, are **deterministic points** that fill $[0,1]^d$ in an even way:
  - We are half-way between regular grids and Monte-Carlo.
- In QMC, how to measure the uniformity of our samples is important, and measures are typically called **discrepancies**.

Quasi Monte-Carlo

The Start Discrepancy

- Let's define an interval in d dimension as:

$$\prod_{i=1}^{d} [a_i, b_i) = \left\{ \mathbf{x} \in \mathbb{R}^d \middle| \forall_{j \in [1,d]} \quad x_j \in [a_i, b_i) \right\} \qquad \mathbf{a}, \mathbf{b} \in \mathbb{R}^d \wedge \forall_i a_i \leq b_i.$$

- The **local discrepancy** of n samples $\mathbf{x}_i$ is defined as:

$$\delta(\mathbf{a}) = \frac{1}{n} \sum_{i=1}^{n} 1_{\mathbf{x}_i \in [\mathbf{0}, \mathbf{a})} - \prod_{i=1}^{d} a_j.$$

Quasi Monte-Carlo

The Start Discrepancy

- Let's define an interval in d dimension as:

$$\prod_{i=1}^{d}[a_i, b_i) = \left\{ \mathbf{x} \in \mathbb{R}^d \middle| \forall_{j\in[1,d]} \quad x_j \in [a_i, b_i) \right\} \qquad \mathbf{a}, \mathbf{b} \in \mathbb{R}^d \wedge \forall_i a_i \leq b_i.$$

- The **local discrepancy** of n samples $\mathbf{x}_i$ is defined as:

$$\delta(\mathbf{a}) = \boxed{\frac{1}{n}\sum_{i=1}^{n} 1_{\mathbf{x}_i \in [\mathbf{0},\mathbf{a})}} - \prod_{i=1}^{d} a_j.$$

The number of points in $[\mathbf{0}, \mathbf{a})$

Quasi Monte-Carlo

The Start Discrepancy

- Let's define an interval in d dimension as:

$$\prod_{i=1}^{d}[a_i, b_i) = \left\{ \mathbf{x} \in \mathbb{R}^d \,\middle|\, \forall_{j \in [1,d]} \quad x_j \in [a_i, b_i) \right\} \qquad \mathbf{a}, \mathbf{b} \in \mathbb{R}^d \wedge \forall_i a_i \leq b_i.$$

- The **local discrepancy** of n samples $\mathbf{x}_i$ is defined as:

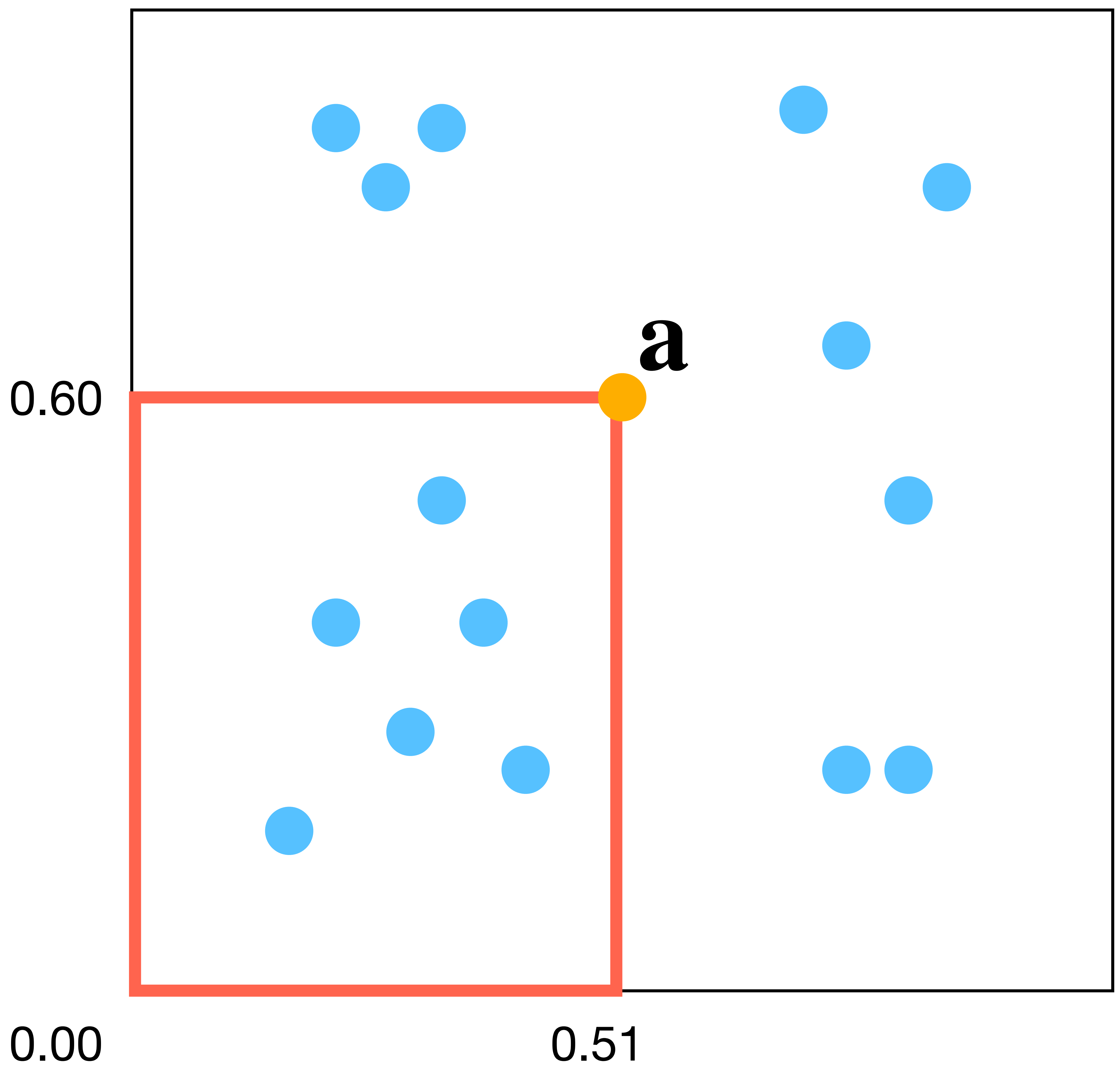
$$\delta(\mathbf{a}) = \frac{1}{n}\sum_{i=1}^{n} 1_{\mathbf{x}_i \in [\mathbf{0}, \mathbf{a})} - \prod_{i=1}^{d} a_j.$$

Volume

The number of points in $[\mathbf{0}, \mathbf{a})$

Quasi Monte-Carlo

The Start Discrepancy: Example



$$\delta(\mathbf{a}) = \frac{6}{14} - 0.6 \cdot 0.50 = 0.42 - 0.30 = 0.12$$

Quasi Monte-Carlo

The Start Discrepancy

- When $\delta(\mathbf{a}) = 0$ there is the perfect balance.

$$D_n^\star = D_n^\star(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sup_{\mathbf{x} \in [0,1)^d} \left| \delta(\mathbf{x}) \right|$$

- A sequence, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, is low discrepancy when:

$$D_n^\star(\mathbf{x}_1, \ldots, \mathbf{x}_n) = O\left(\frac{(\log n)^d}{n} \right), n \to \infty.$$

Quasi Monte-Carlo
Trade-offs

- When we use low discrepancy sequences, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, we cannot use the CLT anymore.
- We have Koksma-Hlawka Theorem:

$$\left| \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x}_i) - \int_{[0,1)^d} f((x) d\mathbf{x} \right| \leq D_n^\star \cdot V_{HK}(f),$$

where V_{HK} is the Hardy and Krause total variation.

- What does this mean?
  - If $V_{HK}(f) < \infty$ and we approximate $D_n^\star = o(n^{-1+\epsilon})$ with $\epsilon > 0$, we have:

$$|\hat{\mu} - \mu| = o(n^{-1+\epsilon}).$$

Low Discrepancy Sequences

Low Discrepancy Sequences

Radical Inverse Function

- The radical inverse function is a simple function defined as:

$$\Phi_b(i) = \sum_{k=0}^{\infty} d_{k,b}(i) b^{-k-1} \qquad b \geq 2 \wedge d_{k,b}(i) \in \{0, \ldots, b-1\}.$$

- This function is based on the fact that we can encode a number i as a sequence of digits:

$$i = \sum_{k=0}^{\infty} d_{k,b}(i) b^{k}.$$

- Φ_b transforms a positive integer into a floating-point in $[0,1)$ by reversing its digits:

$$\Phi(i)_b = 0.d_{i,0} d_{i,1} \ldots d_n.$$

- Van Der Corput's sequence is a simple 1D sequence that is based on the radical inverse function using base 2:

$$x_i = \Phi_2(i).$$

Low Discrepancy Sequences

Radical Inverse Function: Example

$$n = 1 = 1 \times 2^0 + 0 \times 2^2 + \ldots = (\ldots 001)_2$$

$$\Phi(1)_2 = (0.100\ldots)_2 = 1 \times 2^{-1} = 0.5$$

$$n = 2 = 0 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + \ldots = (\ldots 0010)_2$$

$$\Phi(2)_2 = (0.010\ldots)_2 = 0 \times 2^{-1} + 1 \times 2^{-2} = 0.25$$

Low Discrepancy Sequences

Radical Inverse Function: Example

i	Binary	Reversed	$\Phi_2(i)$
1	1	0.1	0.5
2	10	0.01	0.25
3	11	0.11	0.125
4	100	0.001	0.0625

Low Discrepancy Sequences

Halton Sequence

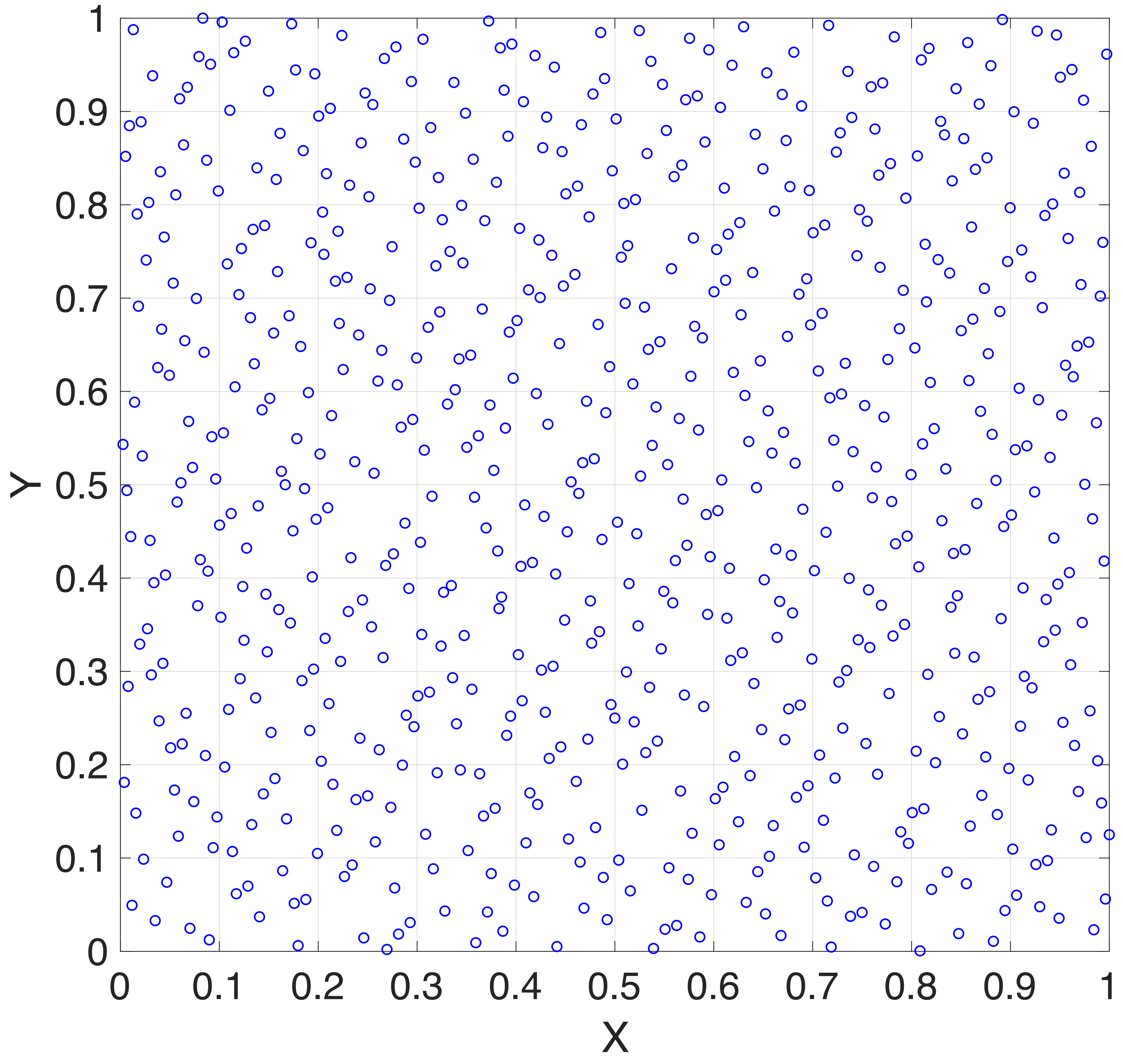
- The Halton sequence employs the radical inverse base.
- In this case, we use a different base for each dimension:
  - Each base needs to be co-prime with the others!
  - A popular choice is to use the first d-prime for generating a d-dimension vector:

$$\mathbf{x}_i = \left(\Phi_2(i), \Phi_3(i), \ldots, \Phi_{p(d)}(i)\right),$$

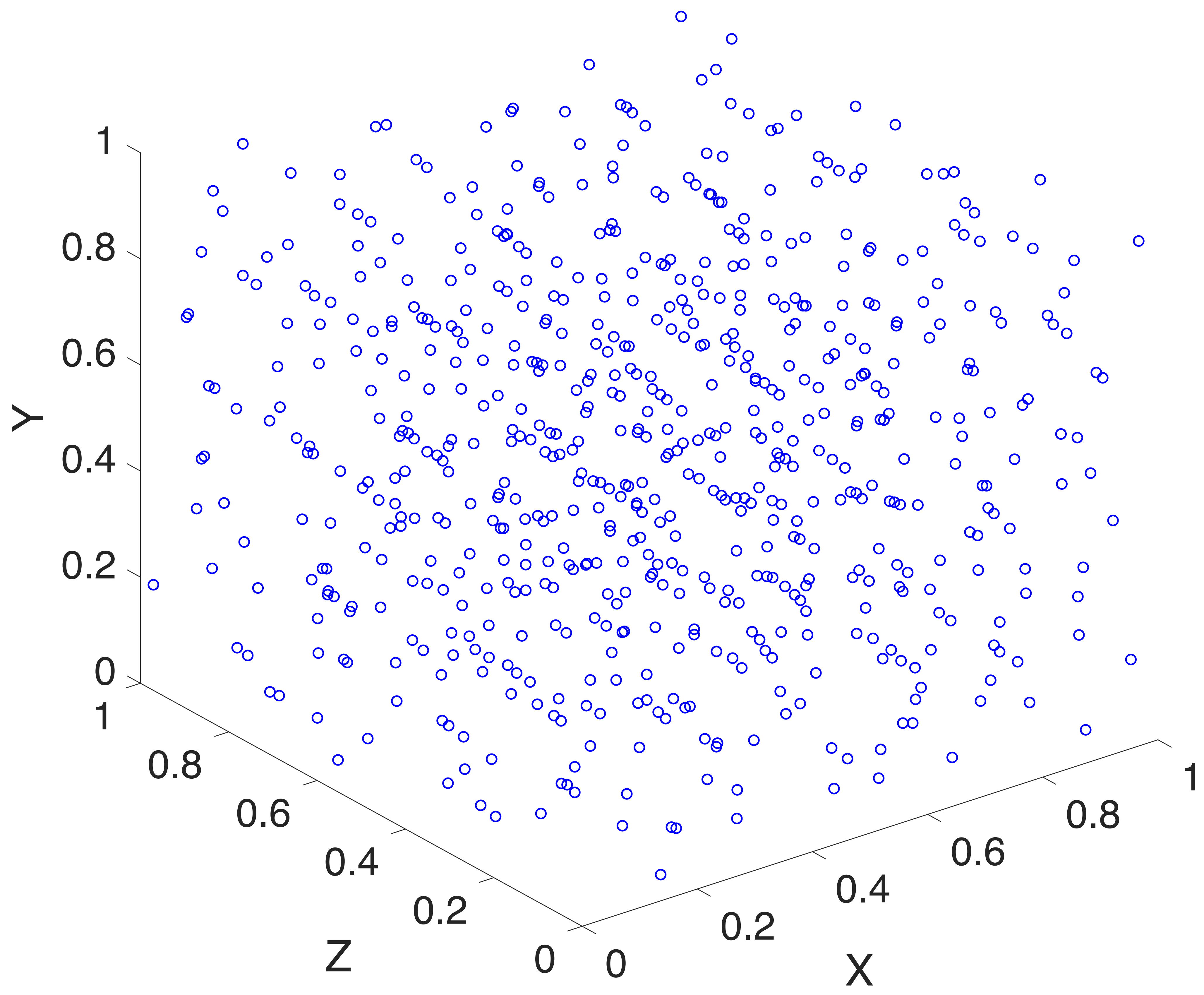
where $p(k)$ is the k-th prime number.

Low Discrepancy Sequences

Halton Sequence: Example



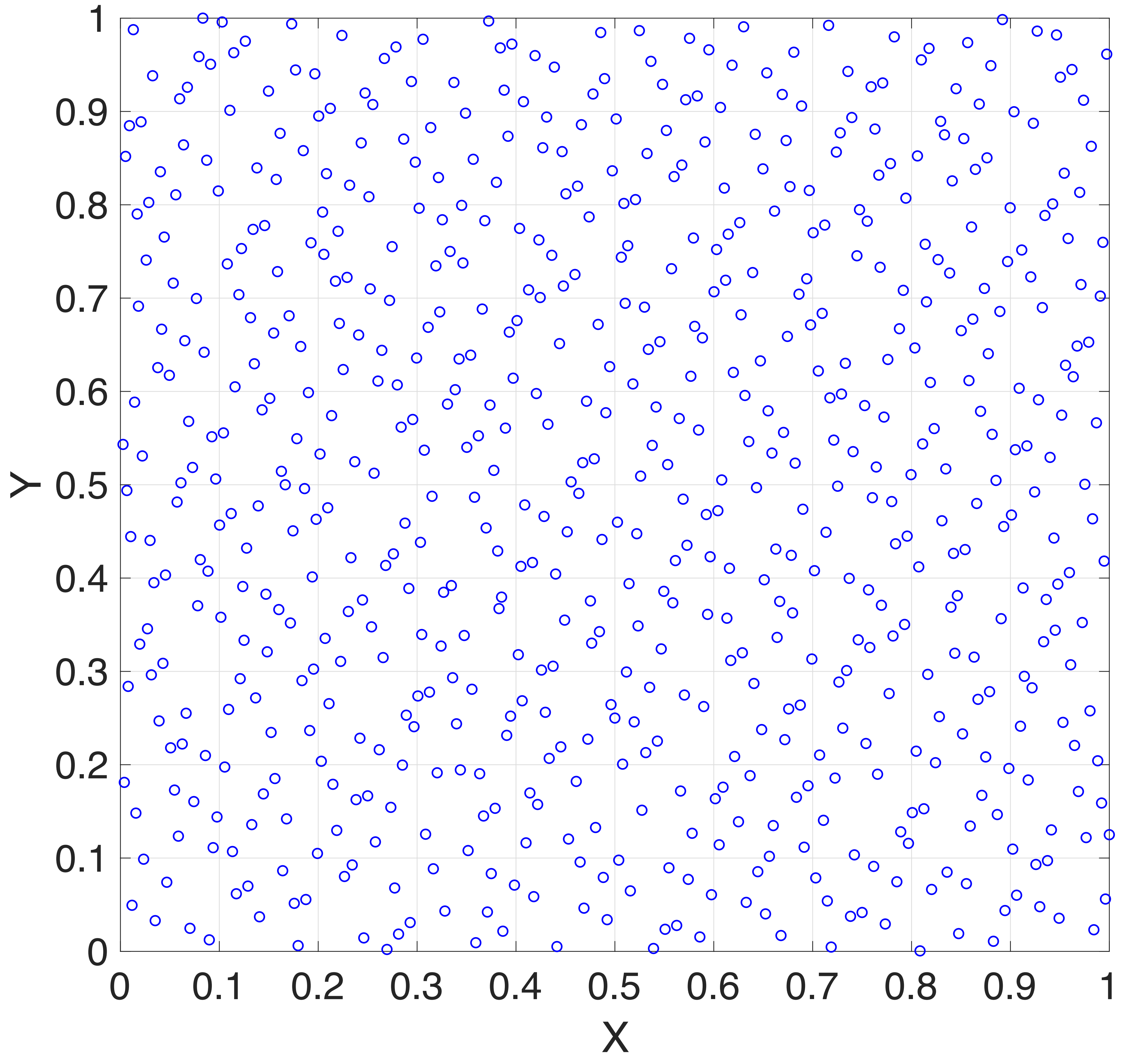
Base X = 2; Base Y = 3



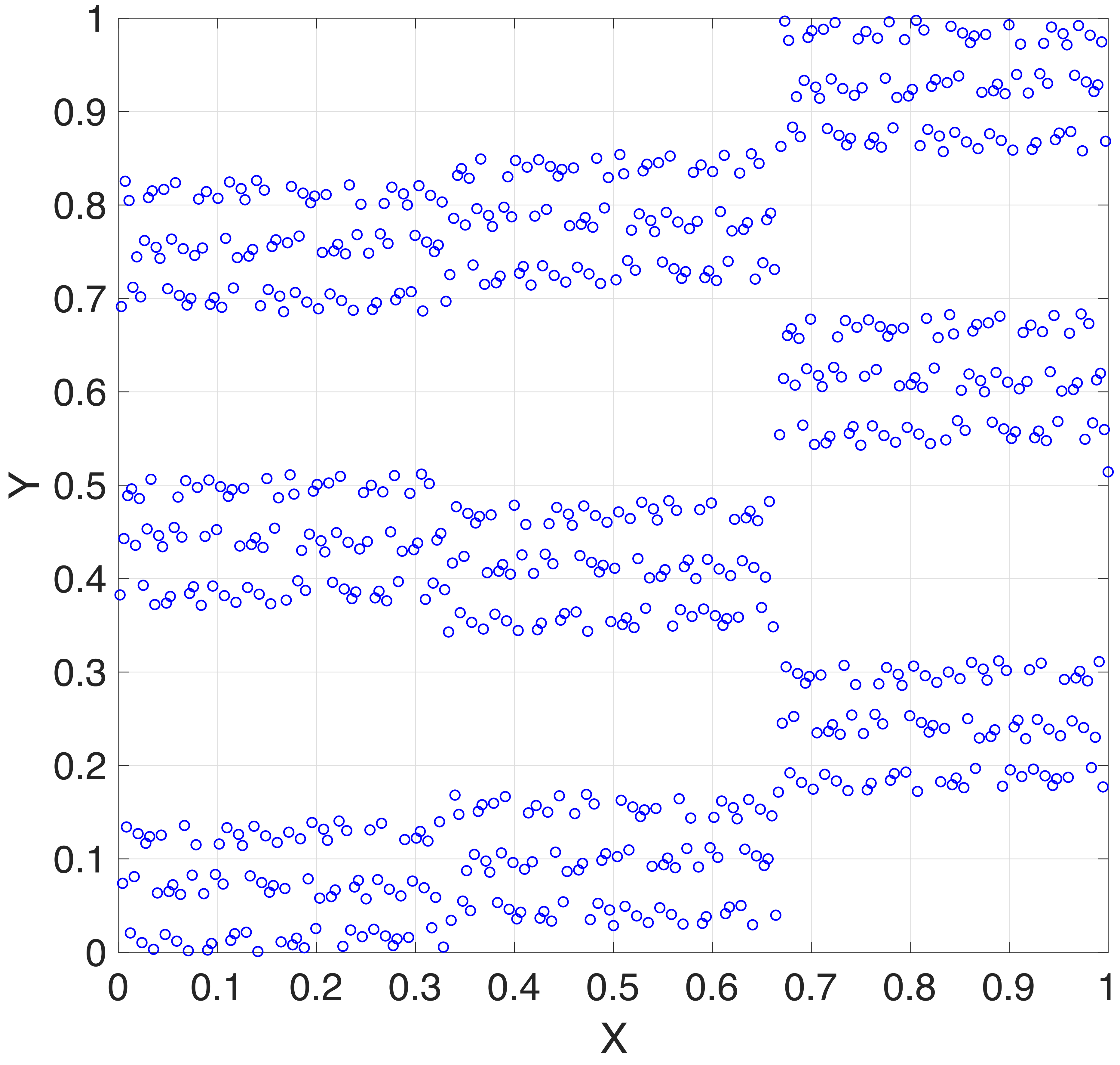
Base X = 2; Base Y = 3; Base Z = 5

Low Discrepancy Sequences

Halton Sequence: Example



Base X = 2; Base Y = 3



Base X = 2; Base Y = 6

Low Discrepancy Sequences

Halton Sequence

- The discrepancy when generating a d-dimensional vector is:

$$O\left(\frac{(\log n)^d}{n}\right),$$

where n is the number of samples.

Low Discrepancy Sequences

Hammersley Sequence

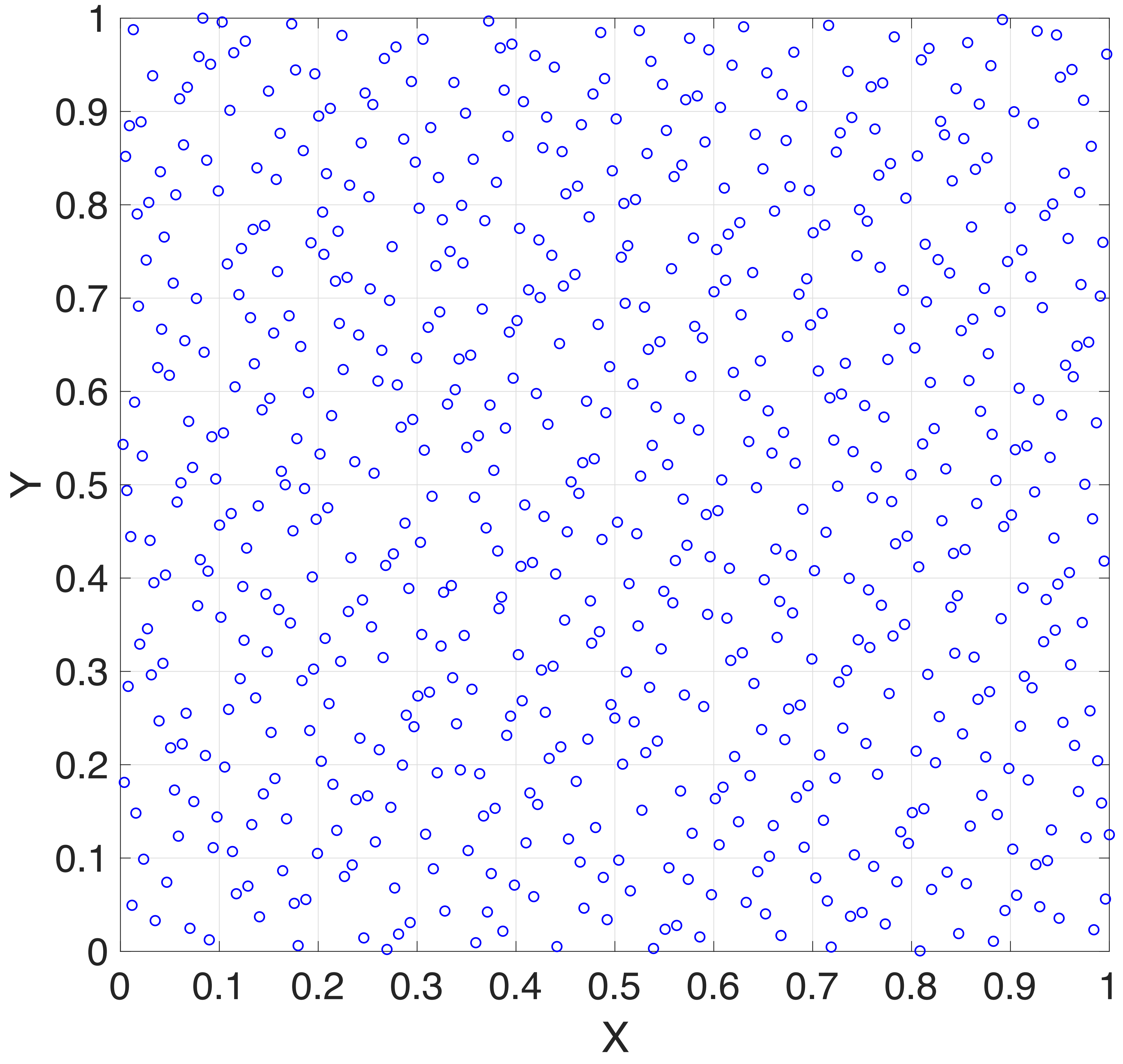
- The Hammersley sequence employs as well the radical inverse base.
- Again, we use a different base for each dimension:
  - Each base needs to be co-prime with the others!
  - As before, we use the first $(d-1)$-prime for generating a d-dimension vector. The vector, compared to Halton's one, has the following change in the generation:

$$\mathbf{x}_i = \left(\Phi_2(i), \Phi_3(i), \ldots, \Phi_{p(d-1)}(i), \frac{i}{n} \right).$$

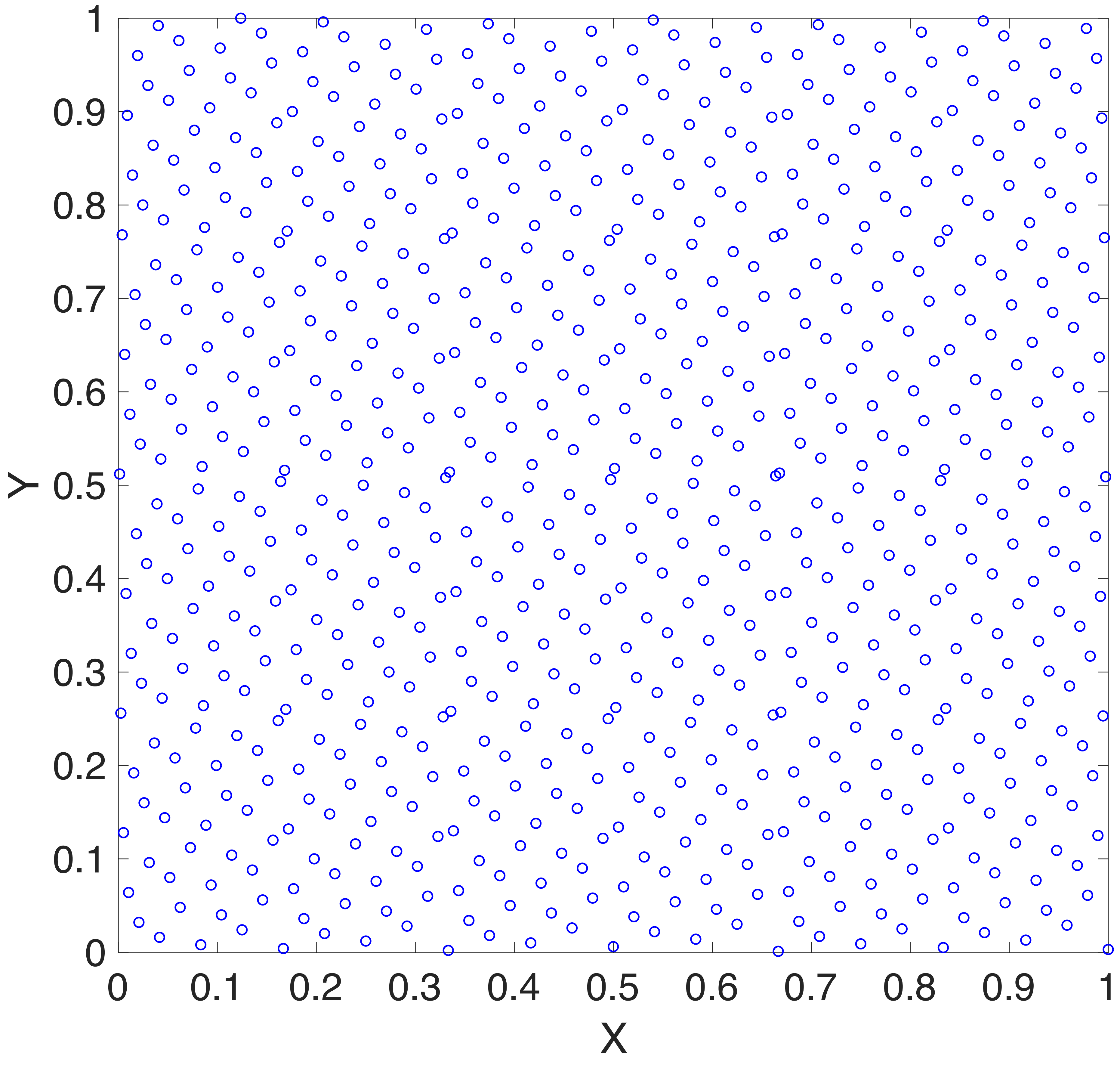
- Note: the number of samples, n, has to be known in advance!

Low Discrepancy Sequences

Hammersley Sequence: Example



Halton Sequence



Hammersley Sequence

Low Discrepancy Sequences

Halton Sequence

- The discrepancy when generating a d-dimensional vector is:

$$O\left(\frac{(\log n)^{d-1}}{n}\right),$$

where n is the number of samples.

Low Discrepancy Sequences
Limitations

- Both Halton sequence and Hammersley sequence have some issues:
  - We may have regular patterns.
  - They are not ideal for parallel applications:
    - All threads will generate the same sequence!
- A possible solution is to randomize such sequences:
  - We apply a random permutation for the digits of a number.

Low Discrepancy Sequences

Other Sequences

- Faure: is based on Van der Corput's sequences, but there is only a base for different dimensions. This is a large prime number:
  - We have permutations with each dimension.

- Sobol: based on algebra of polynomials in $\mathbb{F}_2$:
  - It can be computed using Gray codes.

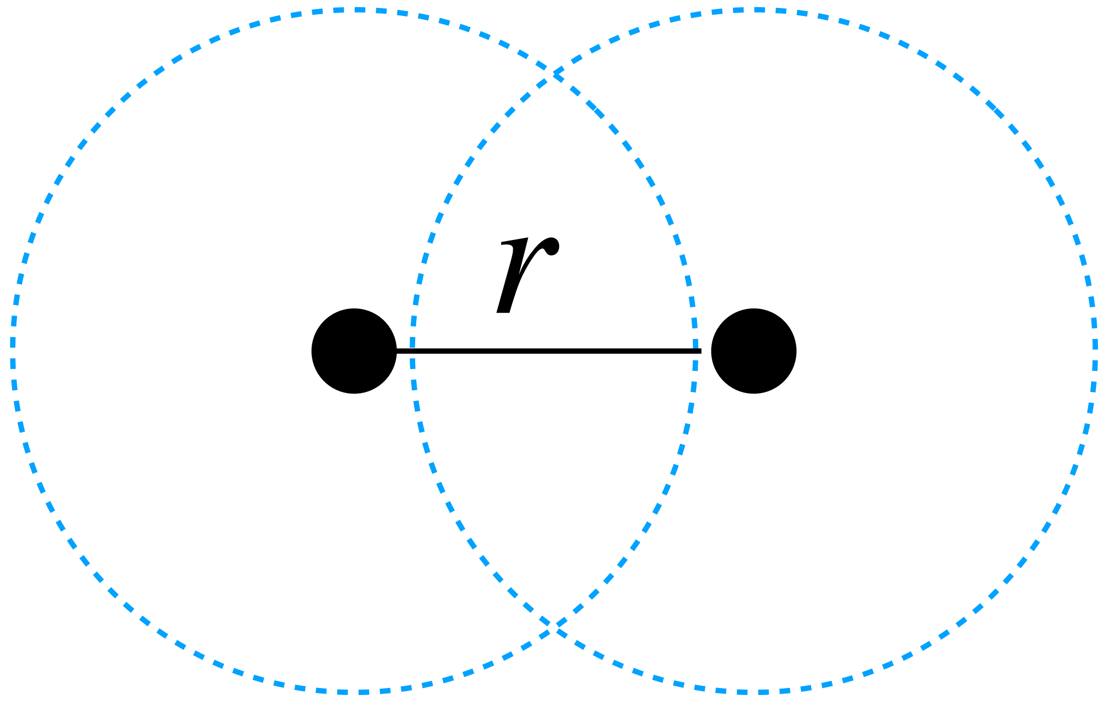
Poisson-Disk Sampling

Poisson-Disk Sampling

Main Idea

- Poisson-disk sampling is a sequential random process for generating samples in a domain.

- Each generated sample/point has to be "**disk-free**" for a minimum distance r:



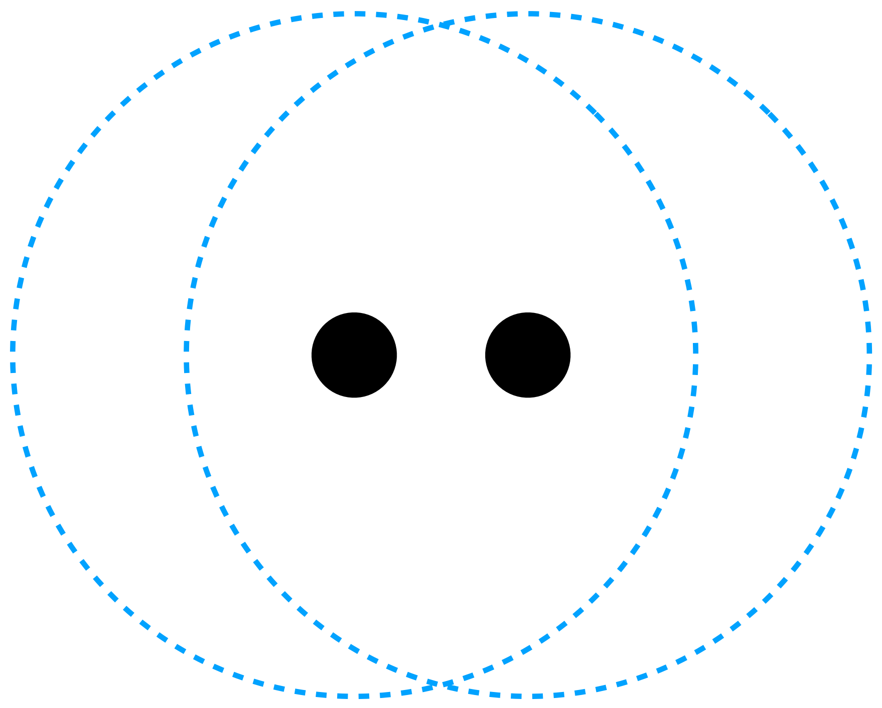
Disk-Free

Poisson-Disk Sampling

Main Idea

- Poisson-disk sampling is a sequential random process for generating samples in a domain.

- Each generated sample/point has to be “**disk-free**” for a minimum distance r:



No Disk-Free

Poisson-Disk Sampling

Main Idea

- This method does not guarantee low-discrepancy, but it creates point-sets without regularity.
- The goal of this sequence is to generate samples with blue noise properties; i.e., the spectrum of a sequence has certain properties:
    - uniformity.
    - isotropic.

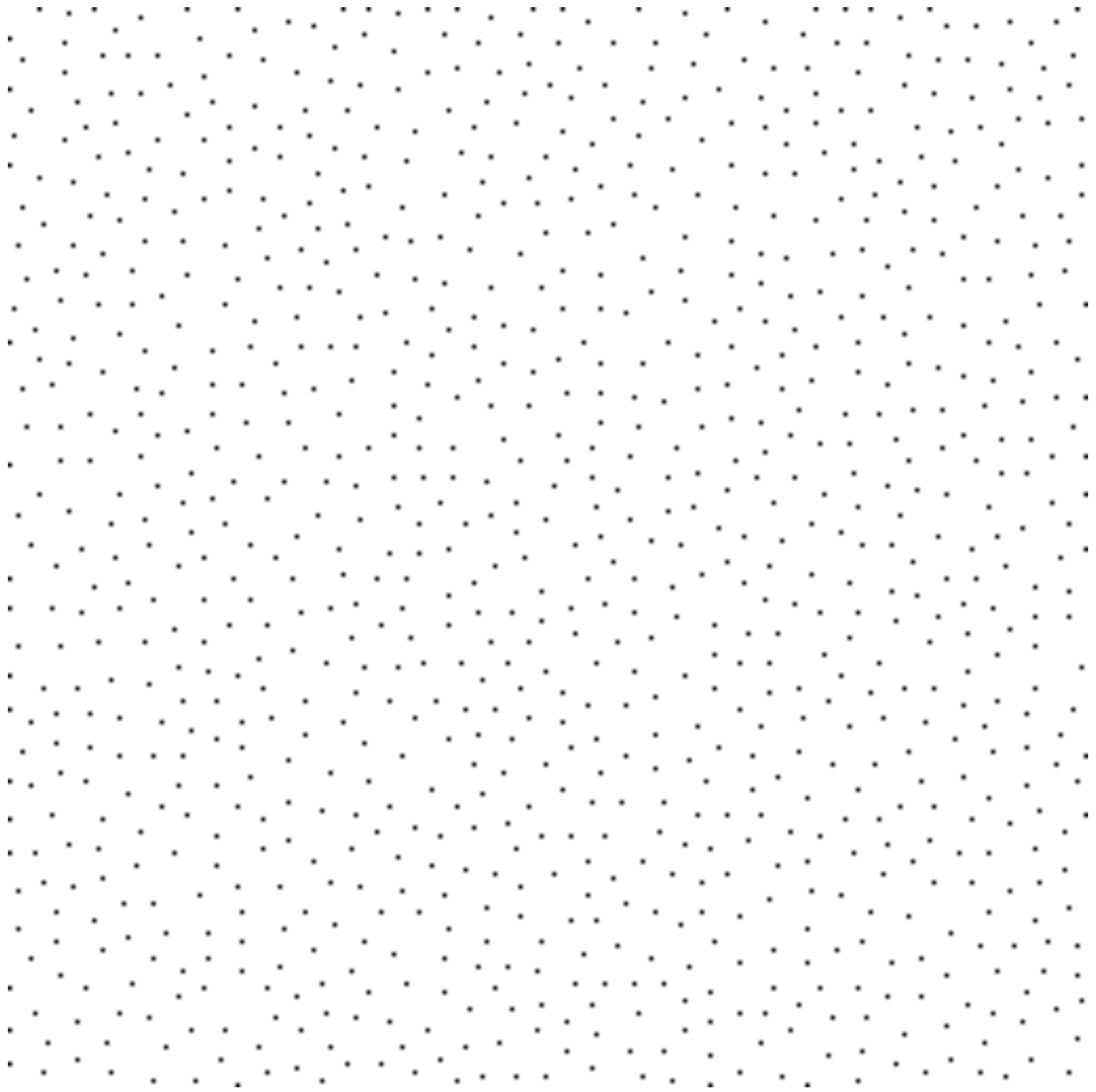
Poisson-Disk Sampling

Main Idea

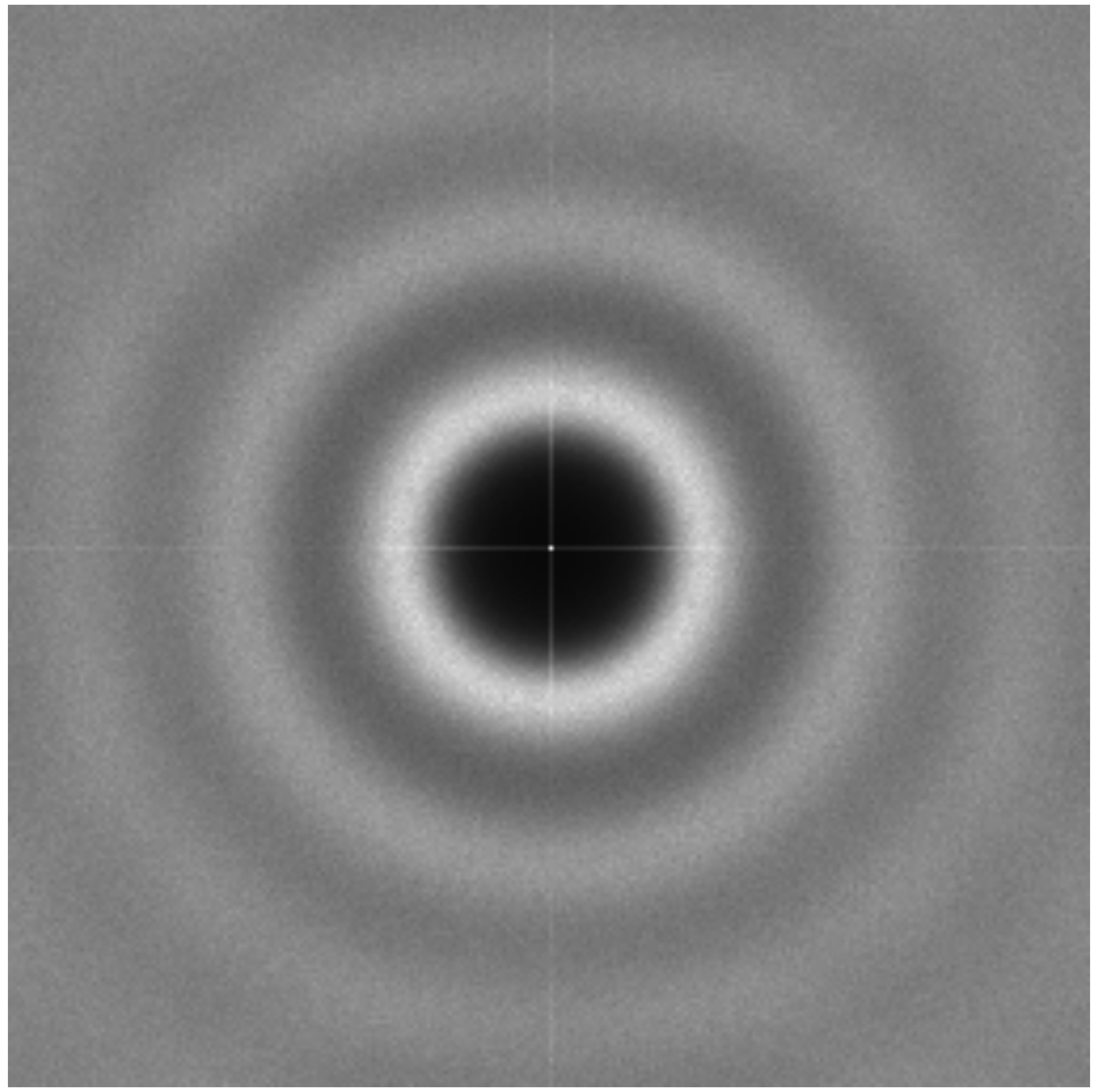
- To achieve Poisson-Disk Sampling, there are a huge literature: 2D, nD, spatially varying radius according a PDF, different distributions, etc.
- The most famous algorithms:
    - Dart Throwing: we draw a sample, $\mathbf{x}_i$, we accept it if its neighbors are at a minimum distance $d \geq r$.
    - Samples removal: we draw a huge number of samples, we remove that samples that close to others; i.e., $d \leq r$.
    - Spatial data structures helps in reducing computational complexity:
        - Bridson 2007 algorithm.

Possin-Disk Sampling

Example



Samples



Periodogram

Randomized QMC

Randomized QMC

Main Idea: Cranley-Patterson Rotation

- One problem of QMC is that if we run it on parallel, all threads will start to generate **exactly the same samples**!

- Another issue is that we cannot have the error estimation that we have in classic Monte-Carlo.

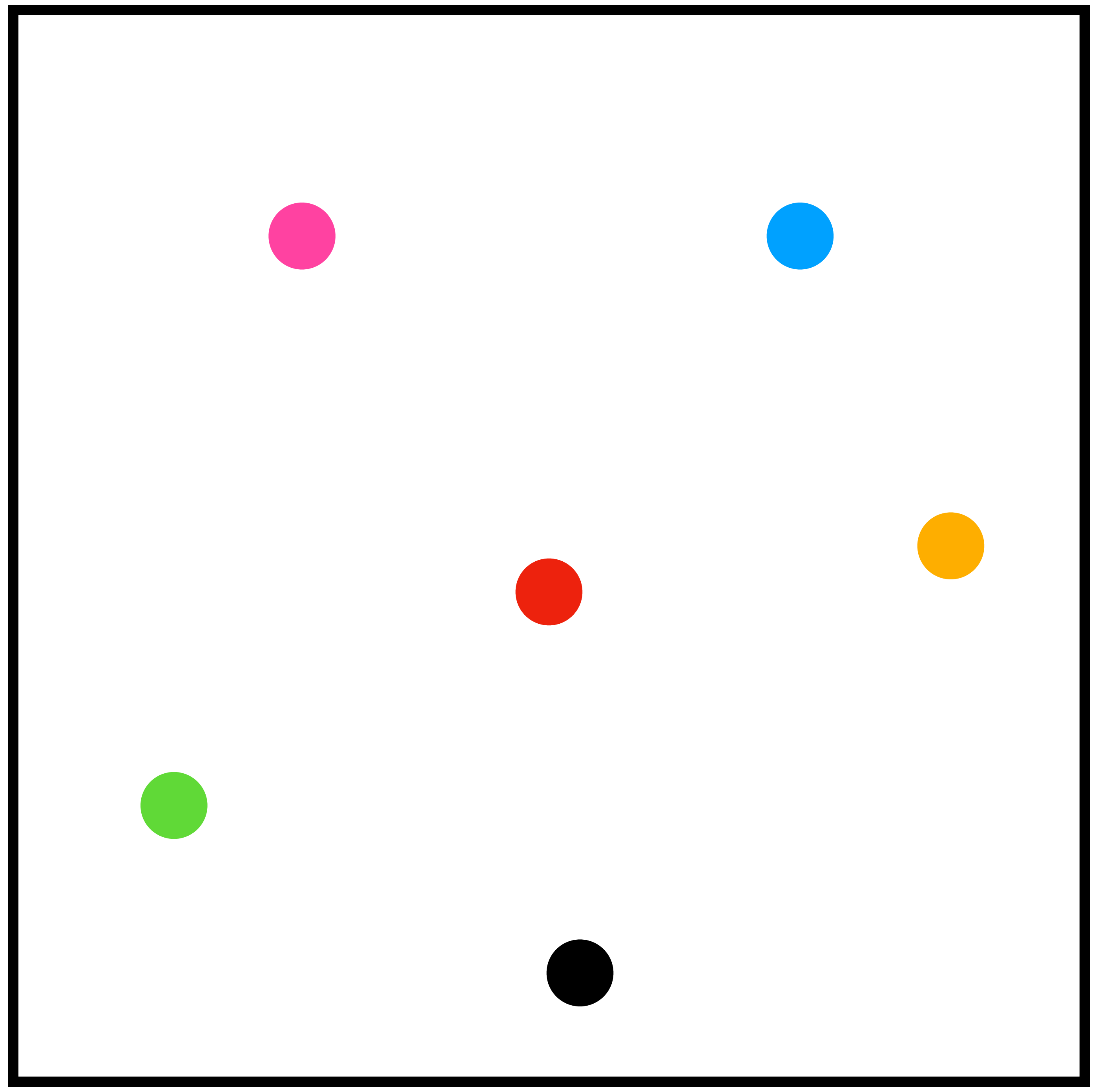
- A solution is to apply a random shift to the sequence:

$$\mathbf{x}_i' = \mathbf{x} + \mathbf{u} \mod 1 \qquad \mathbf{u} \in \mathbf{U}(0,1).$$

- This solution is called Cranley-Patterson rotation.

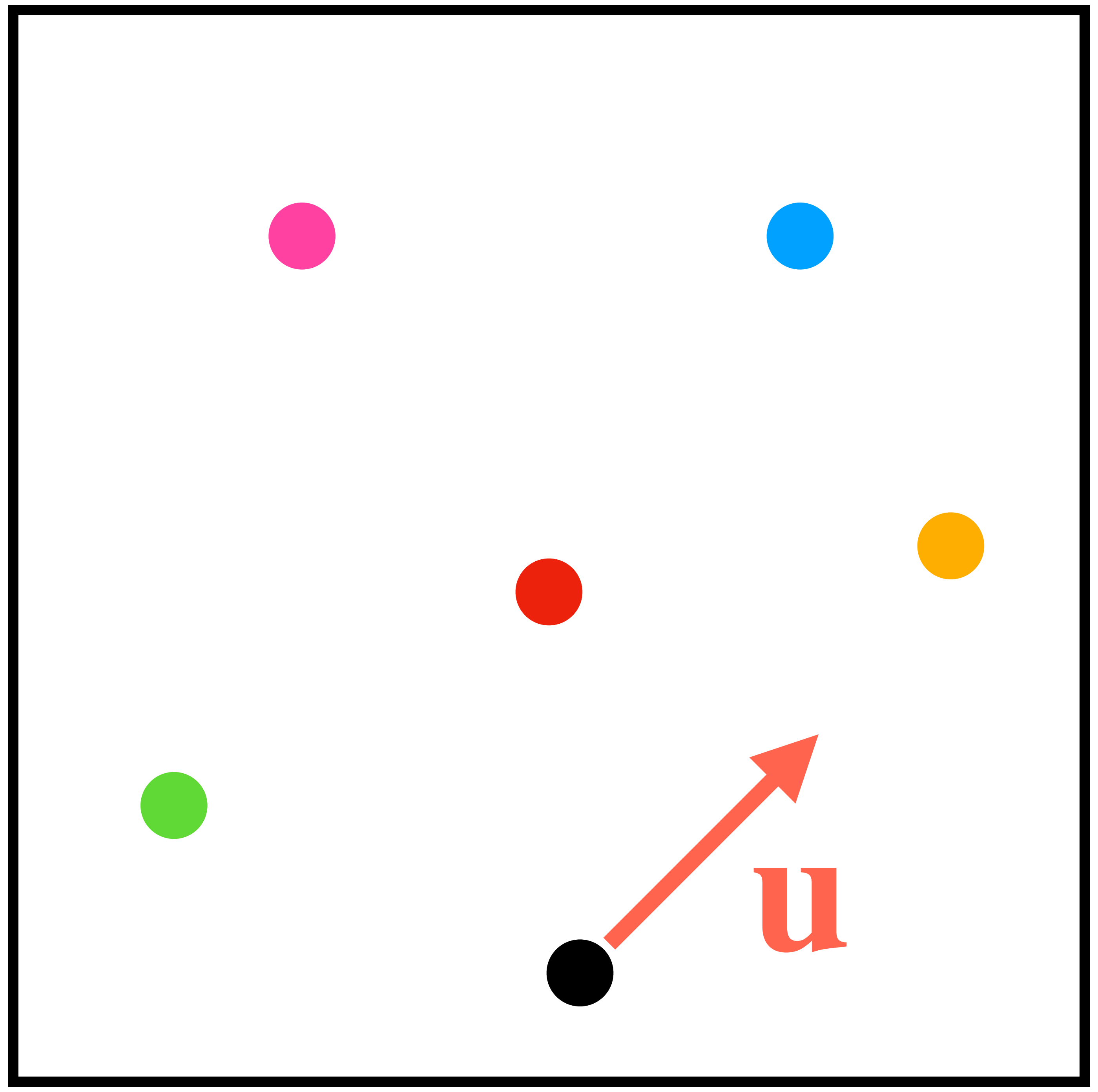
Main Idea: Cranley-Patterson

Example



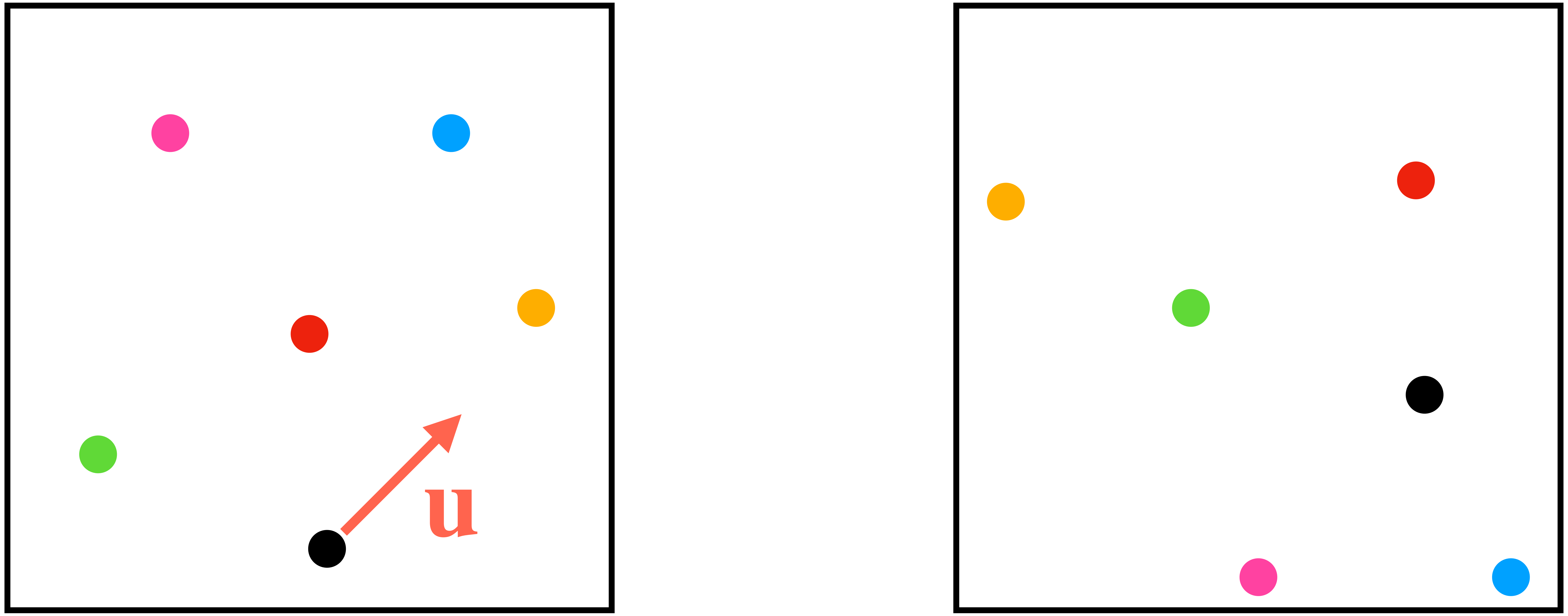
Main Idea: Cranley-Patterson

Example



Main Idea: Cranley-Patterson

Example



Randomized QMC

Main Idea: Scrambling

- Cranley-Patterson rotation works and is low discrepancy. However, it does not preserve stratification properties of a sequence.
- A solution is scrambling the digits of numbers in a sequence. For example in 1D:

$$x = \sum_{i=0}^{\infty} x_i b^{-i-1} \to x' = \sum_{i=0}^{\infty} x_i' b^{-i-1},$$

  - Where we apply random permutations:

$$\begin{aligned} x_0' &= \pi(x_0) \\ x_1' &= \pi_{x_0}(x_1) \\ x_2' &= \pi_{x_0,x_1}(x_2) \\ &\cdots \end{aligned},$$

and π are permutations of $\{0,\ldots,b-1\}$.

Bibliography

- Art Owen. “Chapter 15: The Quasi-Monte Carlo parts” from the book “Monte Carlo theory, methods and examples”. 2019.
- Paolo Brandimarte. “Handbook in Monte-Carlo Simulation”. Wiley. 2014.
- Matt Pharr, Greg Humphreys. “Chapter 7: Sampling and Reconstruction” from the book “Physically Based Rendering - Second Edition”. Morgan Kaufmann. 2010.

Thank you for your attention!