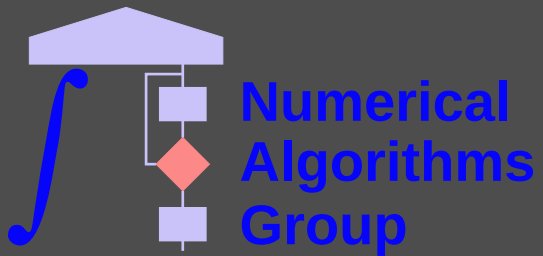
Beyond Monte Carlo

Alexander Keller

keller@informatik.uni-kl.de



Dept. of Computer Science
University of Kaiserslautern

’For every randomized algorithm, there is a clever deterministic one.’

Harald Niederreiter, Claremont, 1998.

’For every randomized algorithm, there is a clever deterministic one.’

Harald Niederreiter, Claremont, 1998.

- no real random on classical deterministic computers
- real random by measuring quantum registers

Beyond Monte Carlo

- **MC**: Monte Carlo
  - random sampling

- **QMC:** Quasi-Monte Carlo integration
  - low-discrepancy sampling by deterministic nets, sequences, and lattices

Beyond Monte Carlo

- **MC**: Monte Carlo
  - random sampling

- **QMC:** Quasi-Monte Carlo integration
  - low-discrepancy sampling by deterministic nets, sequences, and lattices

- **RQMC = MC:** Monte Carlo extensions of quasi-Monte Carlo
  - random field synthesis on good lattice points
  - randomized quasi-Monte Carlo integration

Beyond Monte Carlo

- **MC**: Monte Carlo
  - random sampling

- **QMC:** Quasi-Monte Carlo integration
  - low-discrepancy sampling by deterministic nets, sequences, and lattices

- **RQMC = MC:** Monte Carlo extensions of quasi-Monte Carlo
  - random field synthesis on good lattice points
  - randomized quasi-Monte Carlo integration

- **DRQMC = QMC:** Derandomized randomized quasi-Monte Carlo integration

Beyond Monte Carlo: Applications in Computer Graphics

- **MC**: Industry standard RenderMan by PIXAR
  - stratified random sampling

Beyond Monte Carlo: Applications in Computer Graphics

- **MC**: Industry standard RenderMan by PIXAR
  - stratified random sampling

- **QMC**: Derandomized RenderMan
  - new graphics hardware

Beyond Monte Carlo: Applications in Computer Graphics

- **MC**: Industry standard RenderMan by PIXAR
  - stratified random sampling

- **QMC**: Derandomized RenderMan
  - new graphics hardware

- **RQMC**: Ocean wave synthesis
  - discrete Fourier transform independent of dimension

  RQMC: Error estimation for bidirectional path tracing
  - simpler algorithms

Beyond Monte Carlo: Applications in Computer Graphics

- **MC**: Industry standard RenderMan by PIXAR
  - stratified random sampling

- **QMC**: Derandomized RenderMan
  - new graphics hardware

- **RQMC**: Ocean wave synthesis
  - discrete Fourier transform independent of dimension

  RQMC: Error estimation for bidirectional path tracing
  - simpler algorithms

- **DRQMC**: Industry standard mental ray by mental images
  - deterministic correlated low discrepancy sampling
  - fastest performance

Reworking the Classics of Computer Graphics

- Uncorrelated sampling
  - correlated sampling more efficient

Reworking the Classics of Computer Graphics

- Uncorrelated sampling
  - correlated sampling more efficient
- Uniformity is sufficient
  - low-discrepancy sampling more efficient

Reworking the Classics of Computer Graphics

- Uncorrelated sampling
  - correlated sampling more efficient
- Uniformity is sufficient
  - low-discrepancy sampling more efficient
- Sampling points from classical space filling curves
  - not of low discrepancy

Reworking the Classics of Computer Graphics

- Uncorrelated sampling
  - correlated sampling more efficient
- Uniformity is sufficient
  - low-discrepancy sampling more efficient
- Sampling points from classical space filling curves
  - not of low discrepancy
- Either stratification or Latin hypercube sampling
  - you can have both and even more...

Reworking the Classics of Computer Graphics

- Uncorrelated sampling
  - correlated sampling more efficient
- Uniformity is sufficient
  - low-discrepancy sampling more efficient
- Sampling points from classical space filling curves
  - not of low discrepancy
- Either stratification or Latin hypercube sampling
  - you can have both and even more...
- One dimensional stratified Monte Carlo integration
  - Cranley-Patterson rotations more efficient

Reworking the Classics of Computer Graphics

- Uncorrelated sampling
  - correlated sampling more efficient
- Uniformity is sufficient
  - low-discrepancy sampling more efficient
- Sampling points from classical space filling curves
  - not of low discrepancy
- Either stratification or Latin hypercube sampling
  - you can have both and even more...
- One dimensional stratified Monte Carlo integration
  - Cranley-Patterson rotations more efficient
- Antialiasing only by random sampling
  - deterministic low-discrepancy sampling more efficient

Program

- **Day 1: Monte Carlo**
- Day 2: Quasi-Monte Carlo points
- Day 3: Quasi-Monte Carlo integration
- Day 4: Monte Carlo extensions of quasi-Monte Carlo
- Day 5: Applications to computer graphics

Techniques for basically all high-dimensional integration and transport problems

Day 1: Monte Carlo

- Simulation of random variables and fields
- Monte Carlo integration
- Method of dependent tests
- Multilevel method of dependent tests
- Dependent sampling
- Replication heuristics
- Regularization of the samples

Probability Spaces, Random Variables and Random Fields

- **Definition:** A ***probability space*** is given by a set $\Omega = \{\omega_1, \omega_2, \ldots\}$ of ***elementary events*** ω_i, where each elementary event is assigned a probability with

  $$0 \leq \textbf{Prob}(\omega_i) \leq 1 \qquad \text{and} \qquad \sum_{\omega \in \Omega} \textbf{Prob}(\omega) = 1.$$

  $E \subseteq \Omega$ is called ***event*** with

  $$\textbf{Prob}(E) = \sum_{\omega \in E} \textbf{Prob}(\omega).$$

Probability Spaces, Random Variables and Random Fields

- **Definition:** A ***probability space*** is given by a set $\Omega = \{\omega_1, \omega_2, \ldots\}$ of ***elementary events*** ω_i, where each elementary event is assigned a probability with

  $$0 \leq \textbf{Prob}(\omega_i) \leq 1 \qquad \text{and} \qquad \sum_{\omega \in \Omega} \textbf{Prob}(\omega) = 1.$$

  $E \subseteq \Omega$ is called ***event*** with

  $$\textbf{Prob}(E) = \sum_{\omega \in E} \textbf{Prob}(\omega).$$

- **Definition:** Given a probability space on the set of elementary events Ω, a mapping

  $$\begin{array}{rcl} X : \Omega & \rightarrow & \mathbb{R} \\ \omega & \mapsto & X_\omega \end{array}$$

  is called a ***random variable***. X_ω is called a ***realization***.

Probability Spaces, Random Variables and Random Fields

- **Definition:** A ***probability space*** is given by a set $\Omega = \{\omega_1, \omega_2, \ldots\}$ of ***elementary events*** ω_i, where each elementary event is assigned a probability with

  $$0 \leq \textbf{Prob}(\omega_i) \leq 1 \qquad \text{and} \qquad \sum_{\omega \in \Omega} \textbf{Prob}(\omega) = 1.$$

  $E \subseteq \Omega$ is called ***event*** with

  $$\textbf{Prob}(E) = \sum_{\omega \in E} \textbf{Prob}(\omega).$$

- **Definition:** Given a probability space on the set of elementary events Ω, a mapping

  $$\begin{aligned} X : \Omega &\rightarrow \mathbb{R} \\ \omega &\mapsto X_\omega \end{aligned}$$

  is called a ***random variable***. X_ω is called a ***realization***.

- **Definition:** A ***random field*** (also called ***random function***)

  $$\begin{aligned} X : \Omega &\rightarrow C(s, d) \\ \omega &\mapsto X_\omega \end{aligned}$$

  maps the space of elementary events Ω into the space of continuous functions $C(s, d)$. If $s = 1$ the random fields can be called ***random process***.

Discrete Random Variables

- **Definition:** If the probability space Ω is finite or countable, the random variable X is ***discrete***.

$$\begin{aligned} P_X : \mathbb{R} &\rightarrow [0,1] \\ x &\mapsto \textbf{Prob}(X \leq x) = \sum_{x' \leq x} \textbf{Prob}(X = x') \end{aligned}$$

  is called ***cumulative distribution function (cdf)*** of the random variable X.

Continuous Random Variables

- **Definition:** A ***continuous random variable*** X and its underlying (real) probability space are defined by an integrable density function

  $$p_X : \mathbb{R} \to \mathbb{R}_0^+$$

  with the property $\int_{\mathbb{R}} p_X(x)dx = 1$. A set $A \subseteq \mathbb{R}$ that can be built by the union $A = \cup_k I_k$ of countably many pair-wise disjoint intervals of arbitrary kind (open, closed, half-open, one-sided infinite) is called ***event***. X takes a value from A with

  $$\textbf{Prob}(A) = \int_A p_X(x)dx = \sum_k \int_{I_k} p_X(x)dx.$$

  The ***cumulative distribution function (cdf)*** is

  $$P_X(x) = \textbf{Prob}(X \leq x) = \textbf{Prob}(\{t \in \mathbb{R} | t \leq x\}) = \int_{-\infty}^{x} p_X(t)dt.$$

- Properties of the cumulative distribution function
  - monotonicity and continuity
  - $\lim_{x \to -\infty} P_X(x) = 0$
  - $\lim_{x \to \infty} P_X(x) = 1$

- Properties of the cumulative distribution function
  - monotonicity and continuity
  - $\lim_{x \to -\infty} P_X(x) = 0$
  - $\lim_{x \to \infty} P_X(x) = 1$
- **Corollary:** Any differentiable function P that fulfills the above properties can be assigned a probability density function by

$$p = P'(x).$$

Uniform Distribution $\mathcal{U}$ on $[0,1)^s$

- Probability density function

$$p_{\mathcal{U}}(x) = \begin{cases} 1 & x \in [0,1)^s \\ 0 & \text{else} \end{cases}$$

Uniform Distribution $\mathcal{U}$ *on* $[0,1)^s$

- Probability density function

$$p_{\mathcal{U}}(x) = \begin{cases} 1 & x \in [0,1)^s \\ 0 & \text{else} \end{cases}$$

- Requirements for simulation, i.e. realization
  - fast, deterministic algorithms
  - mimic independence

    $\Rightarrow$ pseudo-random numbers

Uniform Distribution $\mathcal{U}$ on $[0,1)^s$

- Probability density function

$$p_{\mathcal{U}}(x) = \begin{cases} 1 & x \in [0,1)^s \\ 0 & \text{else} \end{cases}$$

- Requirements for simulation, i.e. realization
  - fast, deterministic algorithms
  - mimic independence

  $\Rightarrow$ pseudo-random numbers

- Example: Linear congruential generators (starting value z_0)

$$\begin{aligned} z_{i+1} &= (az_i + c) \bmod m \qquad \in \{0, \ldots, m-1\} \\ \xi_{i+1} &= \frac{z_{i+1}}{m} \end{aligned}$$

Uniform Distribution $\mathcal{U}$ *on* $[0,1)^s$

- Probability density function

$$p_{\mathcal{U}}(x) = \begin{cases} 1 & x \in [0,1)^s \\ 0 & \text{else} \end{cases}$$

- Requirements for simulation, i.e. realization
  - fast, deterministic algorithms
  - mimic independence

    $\Rightarrow$ pseudo-random numbers

- Example: Linear congruential generators (starting value z_0)

$$\begin{aligned} z_{i+1} &= (az_i + c) \bmod m \qquad \in \{0, \ldots, m-1\} \\ \xi_{i+1} &= \frac{z_{i+1}}{m} \end{aligned}$$

  - discrete subset of $[0,1)$
  - finite period
  - choice of a, c, m crucial for good statistical properties
  - parallelization difficult

The Multidimensional Inversion Method

- For $p(x) > 0$ for $x \in I^s$ and $\int_{I^s} p(x)dx < \infty$ realize p-distributed samples

$$P^{-1}(x) := (y^{(1)}, \ldots, y^{(s)}) = y$$

from $x \sim \mathcal{U}$ by successively determining

$$\begin{aligned} y^{(1)} \quad &\text{using} \quad x^{(1)} = F_1(y^{(1)}) \,, \\ y^{(2)} \quad &\text{using} \quad x^{(2)} = F_2(y^{(1)}, y^{(2)}) \\ &\vdots \end{aligned}$$

using the bijections

$$F_j(t_1, \ldots, t_j) := \frac{\int_0^{t_j} \int_0^1 \cdots \int_0^1 p(t_1, \ldots, t_{j-1}, \tau_j, \ldots, \tau_s) d\tau_j \cdots d\tau_s}{\int_0^1 \int_0^1 \cdots \int_0^1 p(t_1, \ldots, t_{j-1}, \tau_j, \ldots, \tau_s) d\tau_j \cdots d\tau_s}$$

The Multidimensional Inversion Method

- For $p(x) > 0$ for $x \in I^s$ and $\int_{I^s} p(x)dx < \infty$ realize p-distributed samples

$$P^{-1}(x) := (y^{(1)}, \ldots, y^{(s)}) = y$$

from $x \sim \mathcal{U}$ by successively determining

$$\begin{aligned} y^{(1)} \quad &\text{using} \quad x^{(1)} = F_1(y^{(1)}) \,, \\ y^{(2)} \quad &\text{using} \quad x^{(2)} = F_2(y^{(1)}, y^{(2)}) \\ &\vdots \end{aligned}$$

using the bijections

$$F_j(t_1, \ldots, t_j) := \frac{\int_0^{t_j} \int_0^1 \cdots \int_0^1 p(t_1, \ldots, t_{j-1}, \tau_j, \ldots, \tau_s) d\tau_j \cdots d\tau_s}{\int_0^1 \int_0^1 \cdots \int_0^1 p(t_1, \ldots, t_{j-1}, \tau_j, \ldots, \tau_s) d\tau_j \cdots d\tau_s}$$

- If $p(x) = \prod_{j=1}^s p^{(j)}(x^{(j)})$

$$F_j(t_j) = \frac{\int_0^{t_j} p^{(j)}(\tau) d\tau}{\int_0^1 p^{(j)}(\tau) d\tau}$$

The Multidimensional Inversion Method

- For $p(x) > 0$ for $x \in I^s$ and $\int_{I^s} p(x)dx < \infty$ realize p-distributed samples

  $$P^{-1}(x) := (y^{(1)}, \ldots, y^{(s)}) = y$$

  from $x \sim \mathcal{U}$ by successively determining

  $$\begin{aligned} y^{(1)} \quad &\text{using} \quad x^{(1)} = F_1(y^{(1)}) \,, \\ y^{(2)} \quad &\text{using} \quad x^{(2)} = F_2(y^{(1)}, y^{(2)}) \\ &\vdots \end{aligned}$$

  using the bijections

  $$F_j(t_1, \ldots, t_j) := \frac{\int_0^{t_j} \int_0^1 \cdots \int_0^1 p(t_1, \ldots, t_{j-1}, \tau_j, \ldots, \tau_s) d\tau_j \cdots d\tau_s}{\int_0^1 \int_0^1 \cdots \int_0^1 p(t_1, \ldots, t_{j-1}, \tau_j, \ldots, \tau_s) d\tau_j \cdots d\tau_s}$$

- If $p(x) = \prod_{j=1}^s p^{(j)}(x^{(j)})$

  $$F_j(t_j) = \frac{\int_0^{t_j} p^{(j)}(\tau) d\tau}{\int_0^1 p^{(j)}(\tau) d\tau}$$

- **Note:** P^{-1} not unique, since there exist many mappings of the unit cube onto itself

Composition Method

- Simulation of composite probability density functions

$$p(x) = \sum_{i=1}^{K} w_i p_i(x) \qquad w_i \in \mathbb{R}^+, \sum_{i=1}^{K} w_i = 1$$

1. fix index i using $\xi \sim \mathcal{U}$

$$\sum_{j=1}^{i-1} w_j \leq \xi < \sum_{j=1}^{i} w_j,$$

   i.e. simulate a discrete random variable with **Prob**$(\omega_i) = w_i$

2. efficiently simulate p_i by

$$\frac{\xi - \sum_{j=1}^{i-1} w_j}{w_i} \in I$$

   using only one random number

Composition Method

- Simulation of composite probability density functions

$$p(x) = \sum_{i=1}^{K} w_i p_i(x) \qquad w_i \in \mathbb{R}^+, \sum_{i=1}^{K} w_i = 1$$

1. fix index i using $\xi \sim \mathcal{U}$

$$\sum_{j=1}^{i-1} w_j \leq \xi < \sum_{j=1}^{i} w_j,$$

   i.e. simulate a discrete random variable with **Prob**$(\omega_i) = w_i$

2. efficiently simulate p_i by

$$\frac{\xi - \sum_{j=1}^{i-1} w_j}{w_i} \in I$$

   using only one random number

- **Note:** The composition method can raise variance.
- Applications: Russian Roulette, stochastic evaluation of sums

Selection Methods

- Neumann rejection method, if $\|p\|_\infty < b < \infty$
  - Choose two independent realizations of uniform random numbers $\xi, \zeta \sim \mathcal{U}$
  - If $p(\xi) > b\zeta$ take ξ as a sample
  - else reject ξ and try again
- Efficiency depends on graph of p

Selection Methods

- Neumann rejection method, if $\|p\|_\infty < b < \infty$
  - Choose two independent realizations of uniform random numbers $\xi, \zeta \sim \mathcal{U}$
  - If $p(\xi) > b\zeta$ take ξ as a sample
  - else reject ξ and try again
- Efficiency depends on graph of p
- Generalized Neumann rejection method
  - density separable, i.e. $p(x) = p_1(x^{(1)}) \cdot p_2(x^{(2)})$
  - multidimensional inversion method on invertible part p_2
  - Neumann rejection method on p_1

Selection Methods

- Neumann rejection method, if $\|p\|_\infty < b < \infty$
  - Choose two independent realizations of uniform random numbers $\xi, \zeta \sim \mathcal{U}$
  - If $p(\xi) > b\zeta$ take ξ as a sample
  - else reject ξ and try again
- Efficiency depends on graph of p
- Generalized Neumann rejection method
  - density separable, i.e. $p(x) = p_1(x^{(1)}) \cdot p_2(x^{(2)})$
  - multidimensional inversion method on invertible part p_2
  - Neumann rejection method on p_1
- Metropolis sampling algorithm
  - construct Markov chain with desired density p as stationary density

Selection Methods

- Neumann rejection method, if $\|p\|_\infty < b < \infty$
  - Choose two independent realizations of uniform random numbers $\xi, \zeta \sim \mathcal{U}$
  - If $p(\xi) > b\zeta$ take ξ as a sample
  - else reject ξ and try again
- Efficiency depends on graph of p
- Generalized Neumann rejection method
  - density separable, i.e. $p(x) = p_1(x^{(1)}) \cdot p_2(x^{(2)})$
  - multidimensional inversion method on invertible part p_2
  - Neumann rejection method on p_1
- Metropolis sampling algorithm
  - construct Markov chain with desired density p as stationary density
- **Construction dimension**, i.e. random numbers required for one realization
  - now only finite expectation

Special Methods: Normal Distribution $\mathcal{N}(\mu, \sigma)$

- Probability density function

$$f_{\mathcal{N}(\mu,\sigma)}(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

  - expectation μ
  - variance σ^2

- Trick: Simulate a pair $(X, Y) \sim \mathcal{N}(0, 1) \times \mathcal{N}(0, 1)$

$$f_{\mathcal{N}(0,1)}(x) \cdot f_{\mathcal{N}(0,1)}(y) dx dy = \frac{1}{2\pi} \cdot e^{-\frac{x^2+y^2}{2}} dx dy$$

Special Methods: Normal Distribution $\mathcal{N}(\mu, \sigma)$

- Probability density function

$$f_{\mathcal{N}(\mu,\sigma)}(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

  - expectation μ
  - variance σ^2

- Trick: Simulate a pair $(X, Y) \sim \mathcal{N}(0, 1) \times \mathcal{N}(0, 1)$

$$f_{\mathcal{N}(0,1)}(x) \cdot f_{\mathcal{N}(0,1)}(y)dxdy = \frac{1}{2\pi} \cdot e^{-\frac{x^2+y^2}{2}} dxdy = \frac{1}{2\pi} \cdot e^{-\frac{r^2}{2}} rdrd\phi$$

Special Methods: Normal Distribution $\mathcal{N}(\mu, \sigma)$

- Probability density function

$$f_{\mathcal{N}(\mu,\sigma)}(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

  - expectation μ
  - variance σ^2

- Trick: Simulate a pair $(X, Y) \sim \mathcal{N}(0, 1) \times \mathcal{N}(0, 1)$

$$f_{\mathcal{N}(0,1)}(x) \cdot f_{\mathcal{N}(0,1)}(y)dxdy = \frac{1}{2\pi} \cdot e^{-\frac{x^2+y^2}{2}} dxdy = \frac{1}{2\pi} \cdot e^{-\frac{r^2}{2}} rdrd\phi$$

- Polar method (Box-Müller)

$$(X, Y) = \sqrt{-2\ln(1-\xi)} \cdot (\cos 2\pi\nu, \sin 2\pi\nu)$$

  where $\xi, \nu \sim \mathcal{U}$ on $[0, 1)$

Simulation of Periodic Random Fields

- Typical realization procedure of $X : \Omega \to C(s, d)$

1. Realize Gaussian noise on s-dimensional regular grid K

$$\vec{N}_\omega(\vec{k}) \sim (\mathcal{N}(0,1) \times i\mathcal{N}(0,1))^d, \qquad \vec{k} \in K$$

2. Shape noise by spectrum S of phenomenon

$$\vec{\hat{X}}_\omega(\vec{k}) = S(\vec{k})\vec{N}_\omega(\vec{k})$$

3. Band limited evaluation by fast Fourier transform for each dimension

$$\vec{X}_\omega(\vec{x}) = \sum_{\vec{k} \in K} \vec{\hat{X}}_\omega(\vec{k}) e^{2\pi i \vec{k}^T \cdot \vec{x}} \qquad \in C(s, d)$$

Simulation of Periodic Random Fields

- Typical realization procedure of $X : \Omega \to C(s, d)$

  1. Realize Gaussian noise on s-dimensional regular grid K

     $$\vec{N}_\omega(\vec{k}) \sim (\mathcal{N}(0,1) \times i\mathcal{N}(0,1))^d, \qquad \vec{k} \in K$$

  2. Shape noise by spectrum S of phenomenon

     $$\vec{\hat{X}}_\omega(\vec{k}) = S(\vec{k})\vec{N}_\omega(\vec{k})$$

  3. Band limited evaluation by fast Fourier transform for each dimension

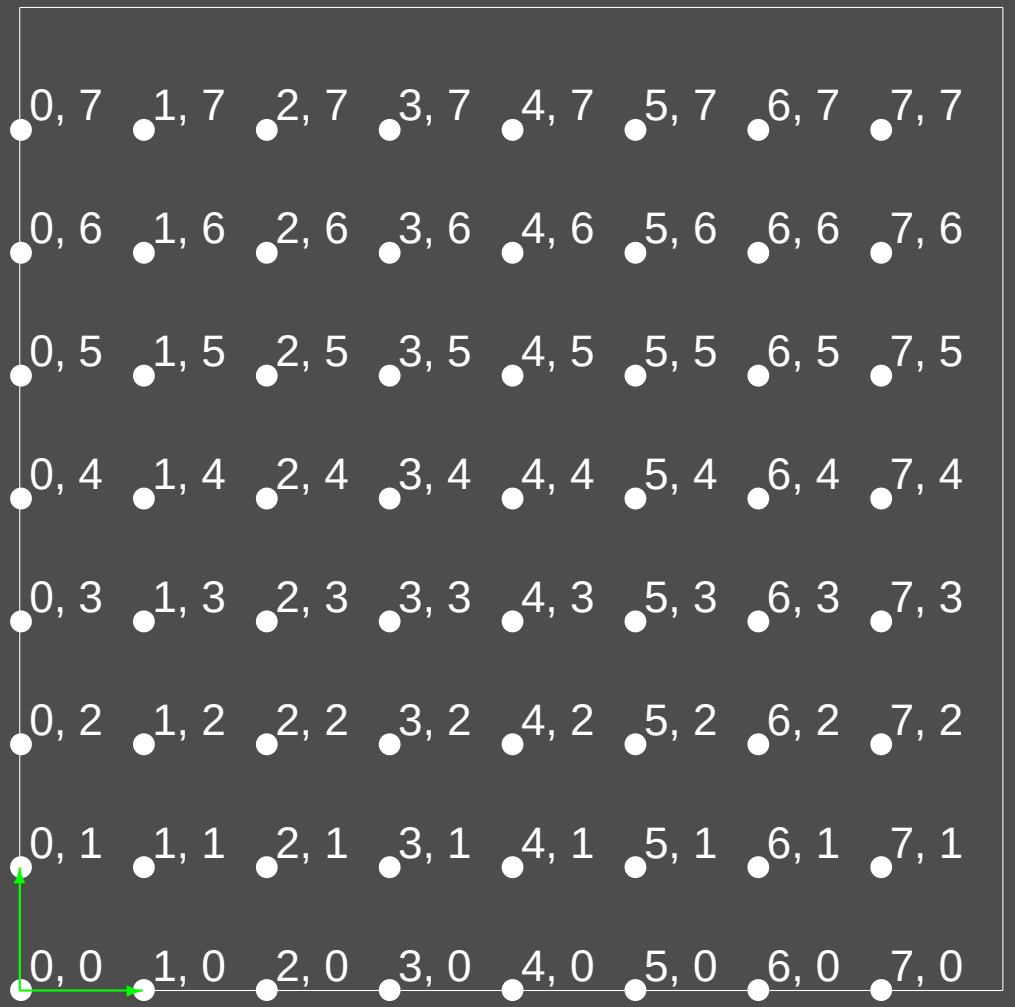
     $$\vec{X}_\omega(\vec{x}) = \sum_{\vec{k} \in K} \vec{\hat{X}}_\omega(\vec{k}) e^{2\pi i \vec{k}^T \cdot \vec{x}} \qquad \in C(s, d)$$

- Standard tensor product approach is exponential in $s = \dim \vec{x} = \dim \vec{k}$

  $\Rightarrow$ **Curse of dimension**

Curse of Dimension from Regular Grids

- Lattices of rank s with $N = n^s$ points from tensor product approach



- $\mathcal{O}\left(n^s \log n\right)$ for s fast Fourier transforms

Curse of Dimension

- **Theorem (Bakhvalov):** *Let C_M^r denote the set of functions on $[0,1)^s$ with r continuous, bounded derivates, i.e.*

$$\left| \frac{\partial^r f(x)}{\partial x_1^{\alpha_1} \cdots \partial x_s^{\alpha_s}} \right| \leq M \text{ for } f \in C_M^r$$

for all $\alpha_1, \ldots, \alpha_s$, such that $\sum_{i=1}^s \alpha_i = r$. Then there exists a function $f \in C_M^r$ such that the error of approximating the integral of f using any N point quadrature rule with weights w_i and function values $f(x_i)$ is

$$\left| \int_{[0,1)^s} f(x)dx - \sum_{i=0}^{N-1} w_i f(x_i) \right| > k \cdot N^{-\frac{r}{s}}$$

where the constant $k > 0$ depends on M and r.

Curse of Discontinuities

- Consider

$$f(x) = \begin{cases} 1 & \text{if } x < X^* \\ 0 & \text{if } x \geq X^* \end{cases}$$

  with $x_i = \frac{i}{n}$ and $x_i \neq X^*$. Then

$$\left| \int_0^1 f(x)dx - \frac{1}{n} \sum_{i=0}^{n-1} f(x_i) \right| \sim \frac{1}{n}$$

- $\mathcal{O}\left(N^{-\frac{1}{s}}\right)$ error for s dimensions

Information Based Complexity Theory

- **Goal:** Find ϵ-approximations to numerical problems
  - minimal cost algorithm for maximum error ϵ

Information Based Complexity Theory

- **Goal:** Find ϵ-approximations to numerical problems
  - minimal cost algorithm for maximum error ϵ
- **Problem statement**
  - **Global information**
    - function classes
  - **Local, partial information**
    - point sampling (standard information)
  - **Model of computation**
    - real number model
    - scalar products as class of algorithms

Information Based Complexity Theory

- **Goal:** Find ϵ-approximations to numerical problems
  - minimal cost algorithm for maximum error ϵ
- **Problem statement**
  - **Global information**
    - function classes
  - **Local, partial information**
    - point sampling (standard information)
  - **Model of computation**
    - real number model
    - scalar products as class of algorithms
- **Analysis of ϵ-complexity**
  - lower bound by abstract structures
  - upper bound by algorithm

  $\Rightarrow$ match bounds

Information Based Complexity Theory

- **Goal:** Find ϵ-approximations to numerical problems
  - minimal cost algorithm for maximum error ϵ
- **Problem statement:** Deterministic numerical integration
  - **Global information**
    - function class: $f \in C_M^r([0,1]^s)$
  - **Local, partial information**
    - point sampling (standard information): $f(x)$
  - **Model of computation**
    - real number model
    - scalar products as class of algorithms: $\sum_{i=1}^{N(f)} w_i f(x_i)$
- **Analysis of ϵ-complexity:** $\mathcal{O}(N^{-\frac{r}{s}})$
  - lower bound by abstract structures: Bakhvalov's theorem
  - upper bound by algorithm: Newton-Cotes quadrature formulas

  $\Rightarrow$ matching bounds

Information Based Complexity Theory

- **Goal:** Find ϵ-approximations to numerical problems
  - minimal cost algorithm for maximum error ϵ
- **Problem statement:** Stochastic numerical integration
  - **Global information**
    - function class: $f \in L^2([0,1]^s)$
  - **Local, partial information**
    - point sampling (standard information): $f(x)$
  - **Model of computation**
    - real number model
    - scalar products as class of algorithms: $\sum_{i=1}^{N(f)} w_i f(x_i)$
- **Analysis of ϵ-complexity:** $\mathcal{O}(N^{-\frac{1}{2}})$
  - lower bound by abstract structures
  - upper bound by algorithm: Monte Carlo integration

  $\Rightarrow$ matching bounds

Information Based Complexity Theory

- **Goal:** Find ϵ-approximations to numerical problems
  - minimal cost algorithm for maximum error ϵ
- **Problem statement:** Stochastic numerical integration
  - **Global information**
    - function class: $f \in C^r_M([0,1]^s)$
  - **Local, partial information**
    - point sampling (standard information): $f(x)$
  - **Model of computation**
    - real number model
    - scalar products as class of algorithms: $\sum_{i=1}^{N(f)} w_i f(x_i)$
- **Analysis of ϵ-complexity:** $\mathcal{O}(N^{-\frac{r}{s}-\frac{1}{2}})$
  - lower bound by abstract structures
  - upper bound by algorithm: Monte Carlo with separation of the main part

  $\Rightarrow$ matching bounds

Monte Carlo Integration

- Principle: Construct random variable with desired functional as expectation
- Numerical integration by **random sampling**

$$\textbf{Prob}\left(\left\{\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| < \frac{3\sigma(f)}{\sqrt{N}}\right\}\right) \approx 0.997 \qquad x_i \sim \mathcal{U}$$

Monte Carlo Integration

- Principle: Construct random variable with desired functional as expectation
- Numerical integration by **random sampling**

$$\textbf{Prob}\left(\left\{\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| < \frac{3\sigma(f)}{\sqrt{N}}\right\}\right) \approx 0.997 \qquad x_i \sim \mathcal{U}$$

- Simple, independent of dimension and smoothness, only $f \in L^2$
- Problems
  - Noise, slow convergence, difficult parallelization and reproducability
  - No real random numbers

Monte Carlo Integration

- Principle: Construct random variable with desired functional as expectation
- Numerical integration by **random sampling**

$$\textbf{Prob}\left(\left\{\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| < \frac{3\sigma(f)}{\sqrt{N}}\right\}\right) \approx 0.997 \qquad x_i \sim \mathcal{U}$$

- Simple, independent of dimension and smoothness, only $f \in L^2$
- Problems
  - Noise, slow convergence, difficult parallelization and reproducability
  - No real random numbers
- Computational complexity

$$N \cdot t_S \cdot \sigma^2(f) = N \cdot t_S \cdot \mathbf{E}\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right|^2$$

Monte Carlo Integration

- Principle: Construct random variable with desired functional as expectation
- Numerical integration by **random sampling**

$$\textbf{Prob}\left(\left\{\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| < \frac{3\sigma(f)}{\sqrt{N}}\right\}\right) \approx 0.997 \qquad x_i \sim \mathcal{U}$$

- Simple, independent of dimension and smoothness, only $f \in L^2$
- Problems
  - Noise, slow convergence, difficult parallelization and reproducability
  - No real random numbers
- Computational complexity

$$N \cdot t_S \cdot \sigma^2(f) = N \cdot t_S \cdot \mathbf{E}\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right|^2$$

- Increase efficiency, not only variance reduction !!!

$$\frac{1}{t_S \cdot \sigma^2(f)}$$

Error Control

- Unbiased estimator Y

$$\mathbf{E}Y = \int_{I^s} f(x)dx$$

Error Control

- Unbiased estimator Y

$$\mathbf{E}Y = \int_{I^s} f(x)dx$$

- Bias of estimator Y

$$\beta_Y := \mathbf{E}Y - \int_{I^s} f(x)dx$$

Error Control

- Unbiased estimator Y

$$\mathbf{E}Y = \int_{I^s} f(x)dx$$

- Bias of estimator Y

$$\beta_Y := \mathbf{E}Y - \int_{I^s} f(x)dx$$

- Consistent estimator Y

$$\mathbf{Prob}\left(\lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N-1} y_i = \int_{I^s} f(x)dx\right) = 1$$

Error Control

- Unbiased estimator Y

$$\mathbf{E}Y = \int_{I^s} f(x)dx$$

- Bias of estimator Y

$$\beta_Y := \mathbf{E}Y - \int_{I^s} f(x)dx$$

- Consistent estimator Y

$$\mathbf{Prob}\left(\lim_{N\to\infty}\frac{1}{N}\sum_{i=0}^{N-1} y_i = \int_{I^s} f(x)dx\right) = 1$$

- Error estimate of the estimate

$$\sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right) \approx \frac{1}{N-1}\left[\sum_{i=0}^{N-1}(f(x_i))^2 - \frac{1}{N}\left(\sum_{i=0}^{N-1} f(x_i)\right)^2\right]$$

  - adaptive sampling

Correlated Sampling: Separation of the Main Part

- Variance reduction by approximation, method of control variables
- Search g with

$$\|f - g\|_\infty < \tau \in \mathbb{R}^+$$

- Then

$$\int_{I^s} f(x)dx = \underbrace{\int_{I^s} g(x)dx}_{\text{analytical}} + \underbrace{\int_{I^s} f(x) - g(x)dx}_{\text{Monte Carlo}}$$

Correlated Sampling: Separation of the Main Part

- Variance reduction by approximation, method of control variables
- Search g with

$$\|f - g\|_\infty < \tau \in \mathbb{R}^+$$

- Then

$$\int_{I^s} f(x)dx = \underbrace{\int_{I^s} g(x)dx}_{\text{analytical}} + \underbrace{\int_{I^s} f(x) - g(x)dx}_{\text{Monte Carlo}}$$

$$\approx \int_{I^s} g(x)dx + \frac{1}{N}\sum_{i=0}^{N-1} (f(x_i) - g(x_i))$$

Note: The independent evaluation would destroy the advantages of the method.

- Variance of Monte Carlo part

$$\sigma^2(f - g) \leq \int_{I^s} |f(x) - g(x)|^2 dx \leq \tau^2$$

Correlated Sampling: Separation of the Main Part

- Variance reduction by approximation, method of control variables
- Search g with

$$\|f - g\|_\infty < \tau \in \mathbb{R}^+$$

- Then

$$\int_{I^s} f(x)dx = \underbrace{\int_{I^s} g(x)dx}_{\text{analytical}} + \underbrace{\int_{I^s} f(x) - g(x)dx}_{\text{Monte Carlo}}$$

$$\approx \int_{I^s} g(x)dx + \frac{1}{N}\sum_{i=0}^{N-1} (f(x_i) - g(x_i))$$

Note: The independent evaluation would destroy the advantages of the method.

- Variance of Monte Carlo part

$$\sigma^2(f - g) \leq \int_{I^s} |f(x) - g(x)|^2 dx \leq \tau^2$$

- Lower bound $\mathcal{O}\left(N^{-\frac{r}{s}-\frac{1}{2}}\right)$ for $f \in C_M^r\left([0,1)^s\right)$ obtained by Newton-Cotes methods

The Method of Dependent Tests

- Principle: Construct random field with desired function as expectation
- Method of dependent tests (parametric Monte Carlo integration)

$$\begin{aligned} g(y) &:= \int_{I^s} f(x,y)dx \\ &\approx \frac{1}{N}\sum_{i=0}^{N-1} f(x_i,y) \end{aligned}$$

for integro-approximation problems

The Method of Dependent Tests

- Principle: Construct random field with desired function as expectation
- Method of dependent tests (parametric Monte Carlo integration)

$$\begin{aligned} g(y) &:= \int_{I^s} f(x,y)dx \\ &\approx \frac{1}{N}\sum_{i=0}^{N-1} f(x_i,y) \end{aligned}$$

  for integro-approximation problems

- Computational complexity

$$N \cdot t_S \cdot \mathbf{E}\left\| \int_{I^s} f(x,y)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i,y) \right\|_{L^2}^2$$

The Method of Dependent Tests

- Principle: Construct random field with desired function as expectation
- Method of dependent tests (parametric Monte Carlo integration)

$$\begin{aligned} g(y) &:= \int_{I^s} f(x,y)dx \\ &\approx \frac{1}{N}\sum_{i=0}^{N-1} f(x_i,y) \end{aligned}$$

  for integro-approximation problems

- Computational complexity

$$N \cdot t_S \cdot \mathbf{E}\left\| \int_{I^s} f(x,y)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i,y) \right\|_{L^2}^2$$

- **Note: One single set** $(x_i)_{i=0}^{N-1} \subset I^s$ of i.i.d. random samples

  $\Rightarrow$ exploit induced grid structure

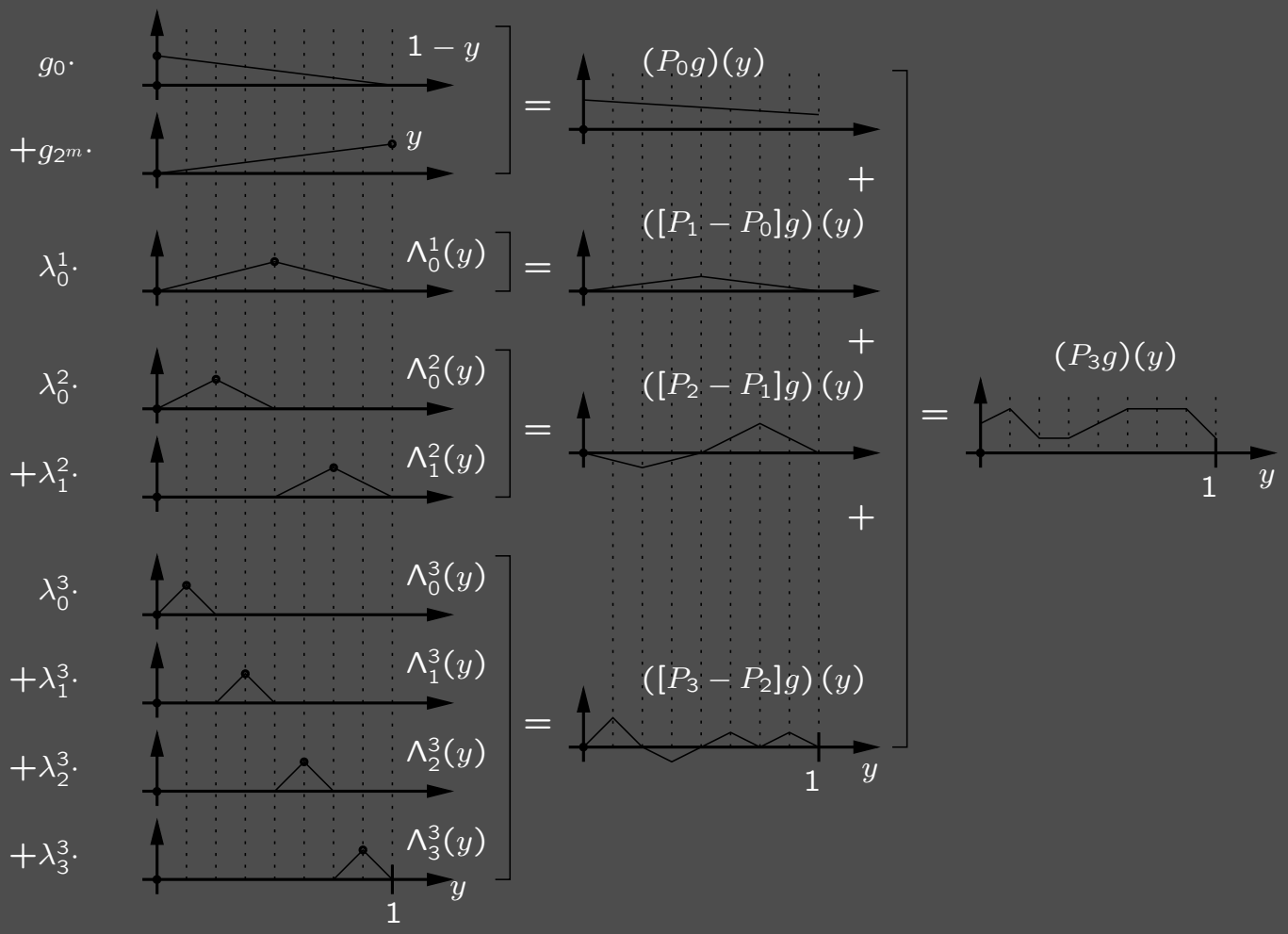
- Examples
  - accumulation buffer
  - multilevel method of dependent tests

Hierarchical Function Representation

- Use multilevel function representation **[Heinrich 1998]**

$$P_m g = P_0 g + \sum_{l=1}^{m} [P_l - P_{l-1}] g$$

for an arbitrary sequence $(P_l)_{l=0}^{m}$ of interpolation operators



Multilevel Method of Dependent Tests

- Linear Lagrange interpolation of $g_k := g(y_k) = (P_m g)(y_k)$ in $y_k = \frac{k}{2^m}$

Multilevel Method of Dependent Tests

- Linear Lagrange interpolation of $g_k := g(y_k) = (P_m g)(y_k)$ in $y_k = \frac{k}{2^m}$
- Method of dependent tests

$$G_k^l := \frac{1}{N_l} \sum_{i=0}^{N_l - 1} f(x_i, y_k) \text{ with } N_l := N \cdot \frac{2^m + 1}{2^l + 1} \cdot 2^{\alpha l} \cdot \frac{2^\alpha - 1}{2^{\alpha(m+1)} - 1}$$

Multilevel Method of Dependent Tests

- Linear Lagrange interpolation of $g_k := g(y_k) = (P_m g)(y_k)$ in $y_k = \frac{k}{2^m}$
- Method of dependent tests

$$G_k^l := \frac{1}{N_l} \sum_{i=0}^{N_l - 1} f(x_i, y_k) \text{ with } N_l := N \cdot \frac{2^m + 1}{2^l + 1} \cdot 2^{\alpha l} \cdot \frac{2^\alpha - 1}{2^{\alpha(m+1)} - 1}$$

- Compute approximation $g_i \approx \hat{g}_i$
  - boundary $g_0 \approx \hat{g}_0 := G_0^0$ and $g_{2^m} \approx \hat{g}_{2^m} := G_{2^m}^0$
  - refinement

$$g_{(2k+1)2^{m-l}} = \underbrace{\frac{g_{k2^{m-(l-1)}} + g_{(k+1)2^{m-(l-1)}}}{2}}_{\text{Predictor}} + \underbrace{\lambda_k^l}_{\text{Update}}$$

Multilevel Method of Dependent Tests

- Linear Lagrange interpolation of $g_k := g(y_k) = (P_m g)(y_k)$ in $y_k = \frac{k}{2^m}$
- Method of dependent tests

$$G_k^l := \frac{1}{N_l} \sum_{i=0}^{N_l - 1} f(x_i, y_k) \text{ with } N_l := N \cdot \frac{2^m + 1}{2^l + 1} \cdot 2^{\alpha l} \cdot \frac{2^\alpha - 1}{2^{\alpha(m+1)} - 1}$$

- Compute approximation $g_i \approx \hat{g}_i$
  - boundary $g_0 \approx \hat{g}_0 := G_0^0$ and $g_{2^m} \approx \hat{g}_{2^m} := G_{2^m}^0$
  - refinement

$$g_{(2k+1)2^{m-l}} = \underbrace{\frac{g_{k2^{m-(l-1)}} + g_{(k+1)2^{m-(l-1)}}}{2}}_{\text{Predictor}} + \underbrace{\lambda_k^l}_{\text{Update}}$$

$$\approx \hat{g}_{(2k+1)2^{m-l}}$$

Multilevel Method of Dependent Tests

- Linear Lagrange interpolation of $g_k := g(y_k) = (P_m g)(y_k)$ in $y_k = \frac{k}{2^m}$
- Method of dependent tests

$$G_k^l := \frac{1}{N_l} \sum_{i=0}^{N_l - 1} f(x_i, y_k) \text{ with } N_l := N \cdot \frac{2^m + 1}{2^l + 1} \cdot 2^{\alpha l} \cdot \frac{2^\alpha - 1}{2^{\alpha(m+1)} - 1}$$

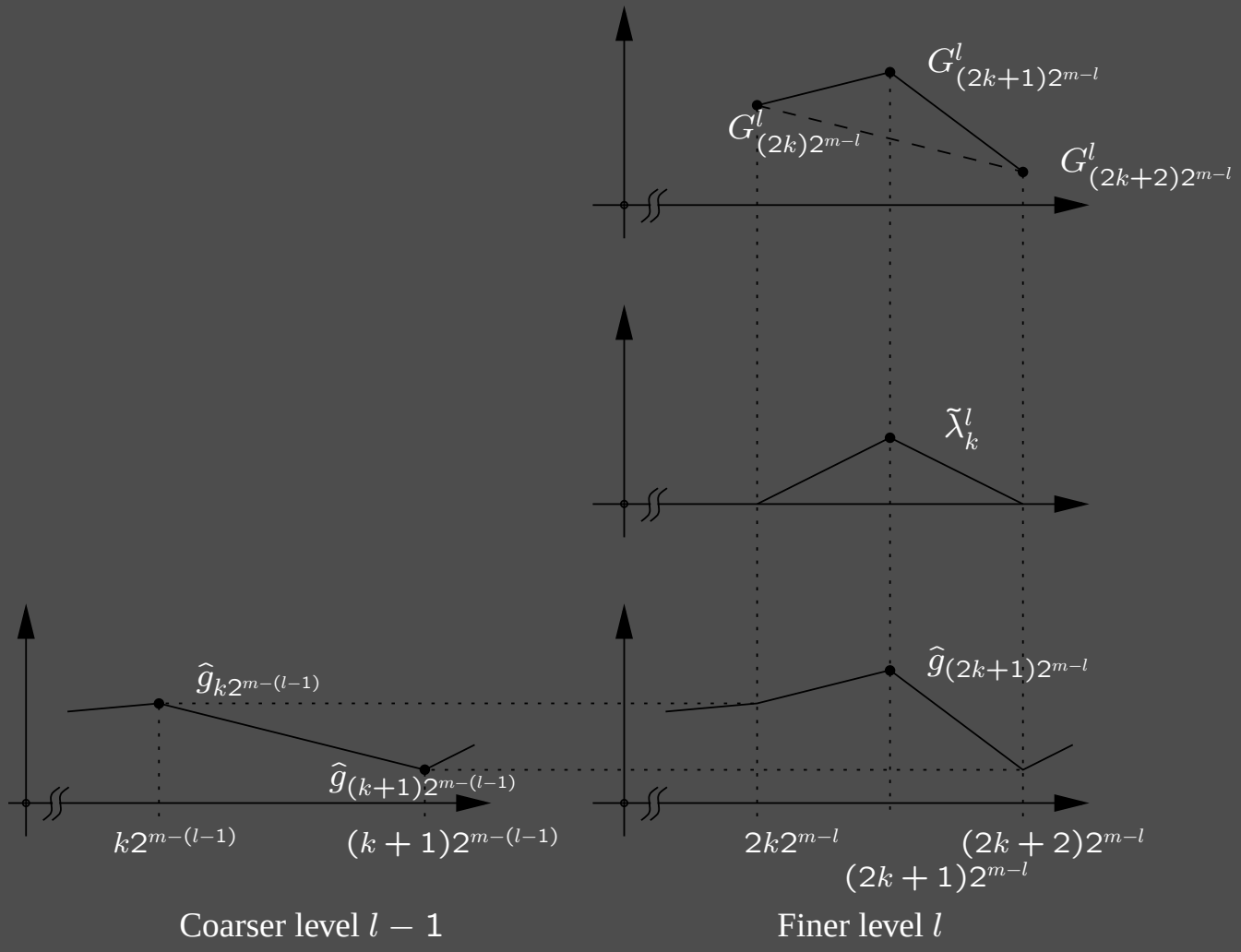
- Compute approximation $g_i \approx \hat{g}_i$
  - boundary $g_0 \approx \hat{g}_0 := G_0^0$ and $g_{2^m} \approx \hat{g}_{2^m} := G_{2^m}^0$
  - refinement

$$g_{(2k+1)2^{m-l}} = \underbrace{\frac{g_{k2^{m-(l-1)}} + g_{(k+1)2^{m-(l-1)}}}{2}}_{\text{Predictor}} + \underbrace{\lambda_k^l}_{\text{Update}}$$

$$\approx \hat{g}_{(2k+1)2^{m-l}} := \frac{\hat{g}_{k2^{m-(l-1)}} + \hat{g}_{(k+1)2^{m-(l-1)}}}{2} + \underbrace{G_{(2k+1)2^{m-l}}^l - \frac{G_{(2k)2^{m-l}}^l + G_{(2k+2)2^{m-l}}^l}{2}}_{=:\tilde{\lambda}_k^l}$$

Implementation

- In-place reconstruction



Coarser level $l-1$ Finer level l

Efficiency Issues

- Individual functionals
  - same high variance
  - same sampling rate, even if correlated
  - converged samples

Efficiency Issues

- Individual functionals
  - same high variance
  - same sampling rate, even if correlated
  - converged samples
- One function
  - small detail contribution if correlated

$$\tilde{\lambda}_k^l = \frac{1}{N_l} \sum_{i=0}^{N_l-1} \left(f(x_i, y_{(2k+1)2^{m-l}}) - \frac{f(x_i, y_{(2k)2^{m-l}}) + f(x_i, y_{(2k+2)2^{m-l}})}{2} \right)$$

  - adapt sampling rate N_l to support size

  $\Rightarrow$ reduced computational cost by exploiting correlation

Efficiency Issues

- Individual functionals
  - same high variance
  - same sampling rate, even if correlated
  - converged samples
- One function
  - small detail contribution if correlated

$$\tilde{\lambda}_k^l = \frac{1}{N_l} \sum_{i=0}^{N_l-1} \left(f(x_i, y_{(2k+1)2^{m-l}}) - \frac{f(x_i, y_{(2k)2^{m-l}}) + f(x_i, y_{(2k+2)2^{m-l}})}{2} \right)$$

  - adapt sampling rate N_l to support size

  $\Rightarrow$ reduced computational cost by exploiting correlation
- Localization heuristics
  - range check
  - predictor-corrector difference
  - relative error

Efficiency Issues

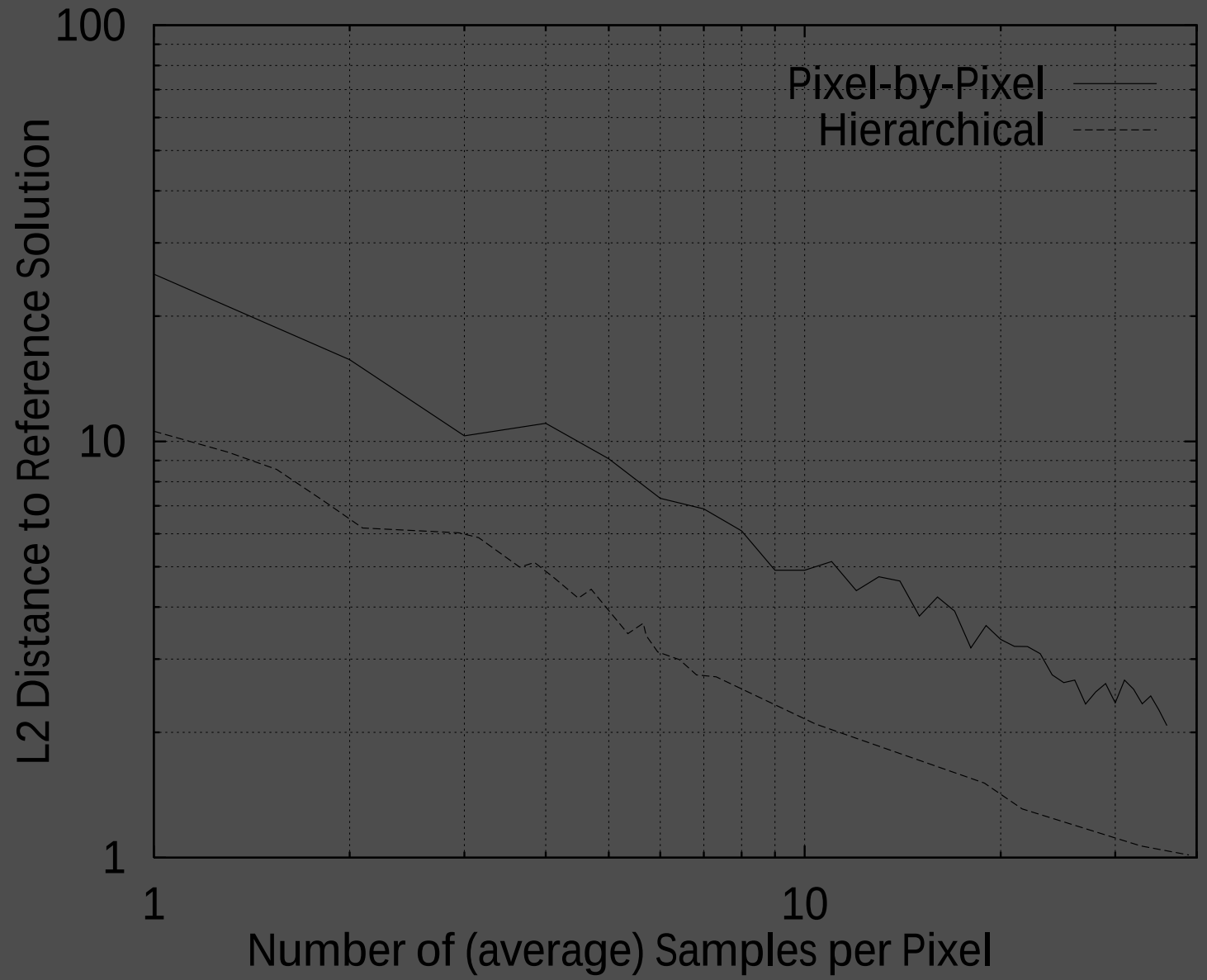
- Individual functionals
  - same high variance
  - same sampling rate, even if correlated
  - converged samples
- One function
  - small detail contribution if correlated

$$\tilde{\lambda}_k^l = \frac{1}{N_l} \sum_{i=0}^{N_l-1} \left(f(x_i, y_{(2k+1)2^{m-l}}) - \frac{f(x_i, y_{(2k)2^{m-l}}) + f(x_i, y_{(2k+2)2^{m-l}})}{2} \right)$$

  - adapt sampling rate N_l to support size

  $\Rightarrow$ reduced computational cost by exploiting correlation
- Localization heuristics
  - range check
  - predictor-corrector difference
  - relative error
- With lifting scheme on arbitrary topology and boundaries

Numerical Results



Importance Sampling

- Integral transformation by introducing a probability density p

$$\int_{I^s} f(x)dx = \int_{I^s} f(x)\frac{p(x)}{p(x)}dx = \int_{I^s} \frac{f(y)}{p(y)}dP(y) \approx \frac{1}{N}\sum_{i=0}^{N-1} \frac{f(y_i)}{p(y_i)} \qquad y_i \sim p$$

Importance Sampling

- Integral transformation by introducing a probability density p

$$\int_{I^s} f(x)dx = \int_{I^s} f(x)\frac{p(x)}{p(x)}dx = \int_{I^s} \frac{f(y)}{p(y)}dP(y) \approx \frac{1}{N}\sum_{i=0}^{N-1} \frac{f(y_i)}{p(y_i)} \qquad y_i \sim p$$

- Variance

$$\sigma^2\left(\frac{f}{p}\right) = \int_{I^s} \frac{f^2(x)}{p(x)}dx - \left(\int_{I^s} f(x)dx\right)^2$$

Importance Sampling

- Integral transformation by introducing a probability density p

$$\int_{I^s} f(x)dx = \int_{I^s} f(x)\frac{p(x)}{p(x)}dx = \int_{I^s} \frac{f(y)}{p(y)}dP(y) \approx \frac{1}{N}\sum_{i=0}^{N-1} \frac{f(y_i)}{p(y_i)} \qquad y_i \sim p$$

- Variance

$$\sigma^2\left(\frac{f}{p}\right) = \int_{I^s} \frac{f^2(x)}{p(x)}dx - \left(\int_{I^s} f(x)dx\right)^2$$

- Often $f(x) = g(x)p(x)$

$$\int_{I^s} f(x)dx = \int_{I^s} g(x)p(x)dx = \int_{I^s} g(y)dP(y) \approx \frac{1}{N}\sum_{i=0}^{N-1} g(y_i) \qquad y_i \sim p$$

Importance Sampling

- Integral transformation by introducing a probability density p

$$\int_{I^s} f(x)dx = \int_{I^s} f(x)\frac{p(x)}{p(x)}dx = \int_{I^s} \frac{f(y)}{p(y)}dP(y) \approx \frac{1}{N}\sum_{i=0}^{N-1} \frac{f(y_i)}{p(y_i)} \qquad y_i \sim p$$

- Variance

$$\sigma^2\left(\frac{f}{p}\right) = \int_{I^s} \frac{f^2(x)}{p(x)}dx - \left(\int_{I^s} f(x)dx\right)^2$$

- Often $f(x) = g(x)p(x)$

$$\int_{I^s} f(x)dx = \int_{I^s} g(x)p(x)dx = \int_{I^s} g(y)dP(y) \approx \frac{1}{N}\sum_{i=0}^{N-1} g(y_i) \qquad y_i \sim p$$

- Often separating the main part is more efficient than importance sampling

Replication: Independent and Dependent Sampling

- Replication heuristic

$$\left(w_j, R_j\right)_{j=0}^{M-1}$$

  - weight functions $w_j(x) : I^s \rightarrow \mathbb{R}$, and
  - mappings $R_j(x) : I^s \rightarrow I^s$ so that

$$\int_{I^s} f(x)dx = \int_{I^s} \sum_{j=0}^{M-1} w_j(x) f(R_j(x))dx = \sum_{j=0}^{M-1} \int_{I^s} w_j(x) f(R_j(x))dx$$

Replication: Independent and Dependent Sampling

- Replication heuristic

$$\left(w_j, R_j\right)_{j=0}^{M-1}$$

  - weight functions $w_j(x) : I^s \rightarrow \mathbb{R}$, and
  - mappings $R_j(x) : I^s \rightarrow I^s$ so that

$$\int_{I^s} f(x)dx = \int_{I^s} \sum_{j=0}^{M-1} w_j(x) f(R_j(x))dx = \sum_{j=0}^{M-1} \int_{I^s} w_j(x) f(R_j(x))dx$$

- Either independent integral estimation

$$\sum_{j=0}^{M-1} \int_{I^s} w_j(x) f(R_j(x))dx \approx \sum_{j=0}^{M-1} \frac{1}{N_j} \sum_{i=0}^{N_j-1} w_j(x_{i,j}) f(R_j(x_{i,j})),$$

Replication: Independent and Dependent Sampling

- Replication heuristic

  $$\left(w_j, R_j\right)_{j=0}^{M-1}$$

  - weight functions $w_j(x) : I^s \rightarrow \mathbb{R}$, and
  - mappings $R_j(x) : I^s \rightarrow I^s$ so that

$$\int_{I^s} f(x)dx = \int_{I^s} \sum_{j=0}^{M-1} w_j(x) f(R_j(x))dx = \sum_{j=0}^{M-1} \int_{I^s} w_j(x) f(R_j(x))dx$$

- Either independent integral estimation

$$\sum_{j=0}^{M-1} \int_{I^s} w_j(x) f(R_j(x))dx \approx \sum_{j=0}^{M-1} \frac{1}{N_j} \sum_{i=0}^{N_j-1} w_j(x_{i,j}) f(R_j(x_{i,j})),$$

  or dependent, i.e. correlated sampling

$$\int_{I^s} \sum_{j=0}^{M-1} w_j(x) f(R_j(x))dx \approx \frac{1}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} w_j(x_i) f(R_j(x_i)),$$

Replication Heuristics: Multiple importance sampling

- Simple importance sampling can cause infinite variance
- For a set of techniques p_j, i.e. $R_j := P_j^{-1}$, the weights are

Heuristic	independent sampling	dependent sampling
Power ($\beta \in \mathbb{R}^+$)	$w_j(x) := \frac{N_j^\beta p_j^\beta(x)}{\sum_{k=0}^{M-1} N_k^\beta p_k^\beta(x)} \cdot \frac{1}{p_j(x)}$	$w_j(x) = \frac{p_j^\beta(x)}{\sum_{k=0}^{M-1} p_k^\beta(x)} \cdot \frac{1}{p_j(x)}$
Balance ($\beta = 1$)	$w_j(x) := \frac{N_j}{\sum_{k=0}^{M-1} N_k p_k(x)}$	$w_j(x) = \frac{1}{\sum_{k=0}^{M-1} p_k(x)}$
Uniform ($\beta = 0$)	$w_j(x) := \frac{N_j}{p_j(x) \sum_{k=0}^{M-1} N_k}$	$w_j(x) = \frac{1}{M p_j(x)}$

Replication Heuristics: Multiple importance sampling

- Simple importance sampling can cause infinite variance
- For a set of techniques p_j, i.e. $R_j := P_j^{-1}$, the weights are

Heuristic	independent sampling	dependent sampling
Power ($\beta \in \mathbb{R}^+$)	$w_j(x) := \frac{N_j^\beta p_j^\beta(x)}{\sum_{k=0}^{M-1} N_k^\beta p_k^\beta(x)} \cdot \frac{1}{p_j(x)}$	$w_j(x) = \frac{p_j^\beta(x)}{\sum_{k=0}^{M-1} p_k^\beta(x)} \cdot \frac{1}{p_j(x)}$
Balance ($\beta = 1$)	$w_j(x) := \frac{N_j}{\sum_{k=0}^{M-1} N_k p_k(x)}$	$w_j(x) = \frac{1}{\sum_{k=0}^{M-1} p_k(x)}$
Uniform ($\beta = 0$)	$w_j(x) := \frac{N_j}{p_j(x) \sum_{k=0}^{M-1} N_k}$	$w_j(x) = \frac{1}{M p_j(x)}$

- Problem of insufficient techniques

Stratification

- Partition of integration domain $I^s = \cup_{k=1}^{K} A_k$
- Monte Carlo integration on each of the disjoint strata A_k

$$\int_{I^s} f(x)dx = \sum_{k=1}^{K} \int_{A_k} f(x)dx$$

Stratification

- Partition of integration domain $I^s = \cup_{k=1}^{K} A_k$
- Monte Carlo integration on each of the disjoint strata A_k

$$\int_{I^s} f(x)dx = \sum_{k=1}^{K} \int_{A_k} f(x)dx \approx \sum_{k=1}^{K} \frac{\lambda_s(A_k)}{N_k} \sum_{i=0}^{N_k-1} f(x_{k,i})$$

Stratification

- Partition of integration domain $I^s = \cup_{k=1}^{K} A_k$
- Monte Carlo integration on each of the disjoint strata A_k

$$\int_{I^s} f(x)dx = \sum_{k=1}^{K} \int_{A_k} f(x)dx \approx \sum_{k=1}^{K} \frac{\lambda_s(A_k)}{N_k} \sum_{i=0}^{N_k-1} f(x_{k,i})$$

- Variance reduction for standard choice $N_k = \lambda_s(A_k)N$

$$\sum_{k=1}^{K} \frac{\lambda_s(A_k)}{N_k} \int_{A_k} \left(f(y) - \frac{1}{\lambda_s(A_k)} \int_{A_k} f(x)dx \right)^2 dy \leq \frac{\sigma^2(f)}{N}$$

$\Rightarrow$ at least as good as uniform random sampling

Stratification

- Partition of integration domain $I^s = \cup_{k=1}^{K} A_k$
- Monte Carlo integration on each of the disjoint strata A_k

$$\int_{I^s} f(x)dx = \sum_{k=1}^{K} \int_{A_k} f(x)dx \approx \sum_{k=1}^{K} \frac{\lambda_s(A_k)}{N_k} \sum_{i=0}^{N_k-1} f(x_{k,i})$$

- Variance reduction for standard choice $N_k = \lambda_s(A_k)N$

$$\sum_{k=1}^{K} \frac{\lambda_s(A_k)}{N_k} \int_{A_k} \left(f(y) - \frac{1}{\lambda_s(A_k)} \int_{A_k} f(x)dx \right)^2 dy \leq \frac{\sigma^2(f)}{N}$$

  $\Rightarrow$ at least as good as uniform random sampling

- $\lambda_s(A_k) = \frac{1}{N}$ yields

$$\int_{I^s} f(x)dx \approx \frac{1}{N} \sum_{k=0}^{N-1} f(x_k|_{A_k})$$

  - Lloyd-relaxation
  - jittered sampling

Stratification by Lloyd-Relaxation

- Algorithm (similar to vector quantization)
  - Take N random initial points
  - Loop: Move each point into the center of gravity of its Voronoi-cell
- Periodic boundary conditions

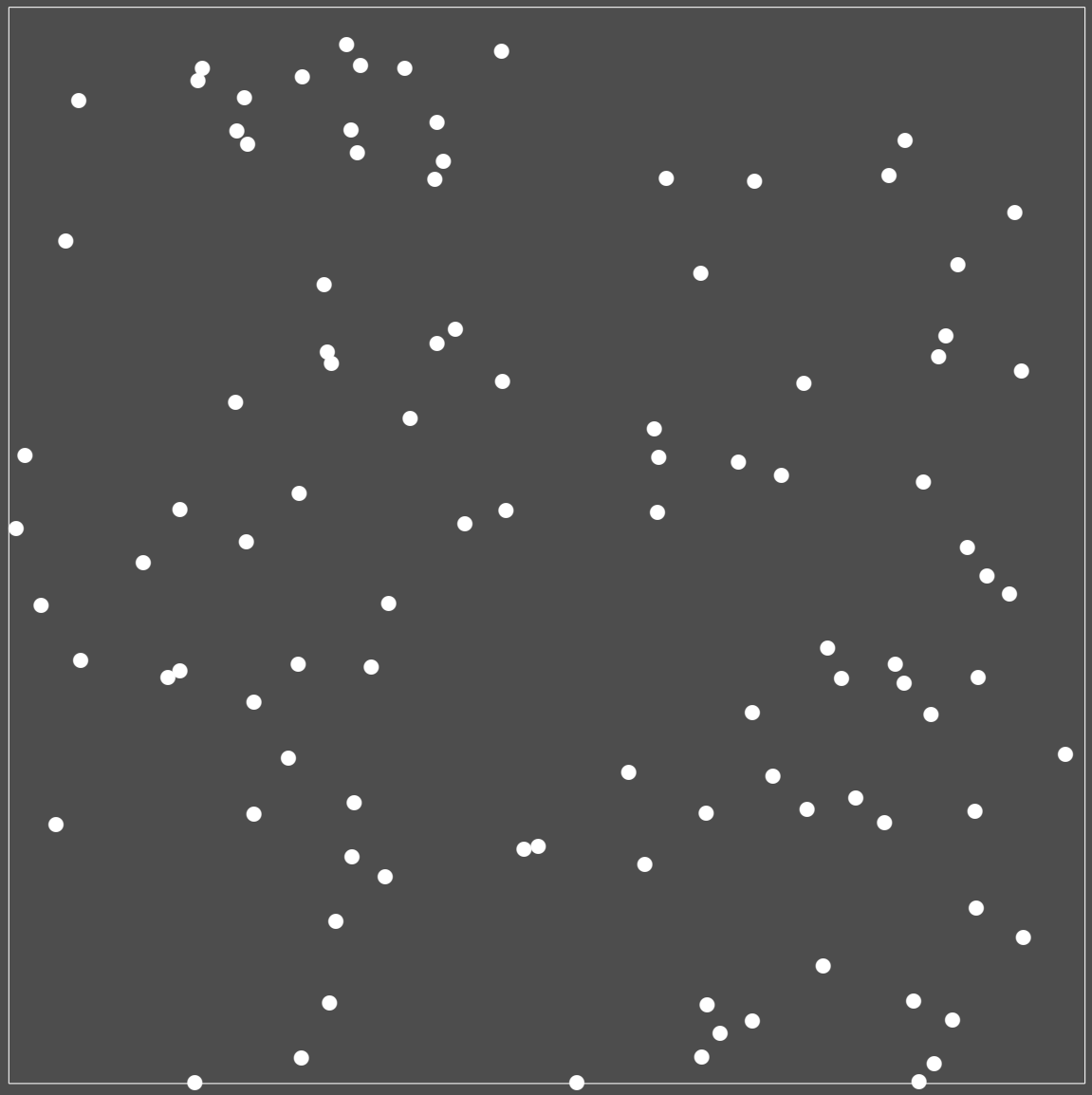
\+ Fast convergence to regular patterns

$\Rightarrow$ *Small* number of relaxation steps yields blue-noise-samples

\- Expensive iteration step

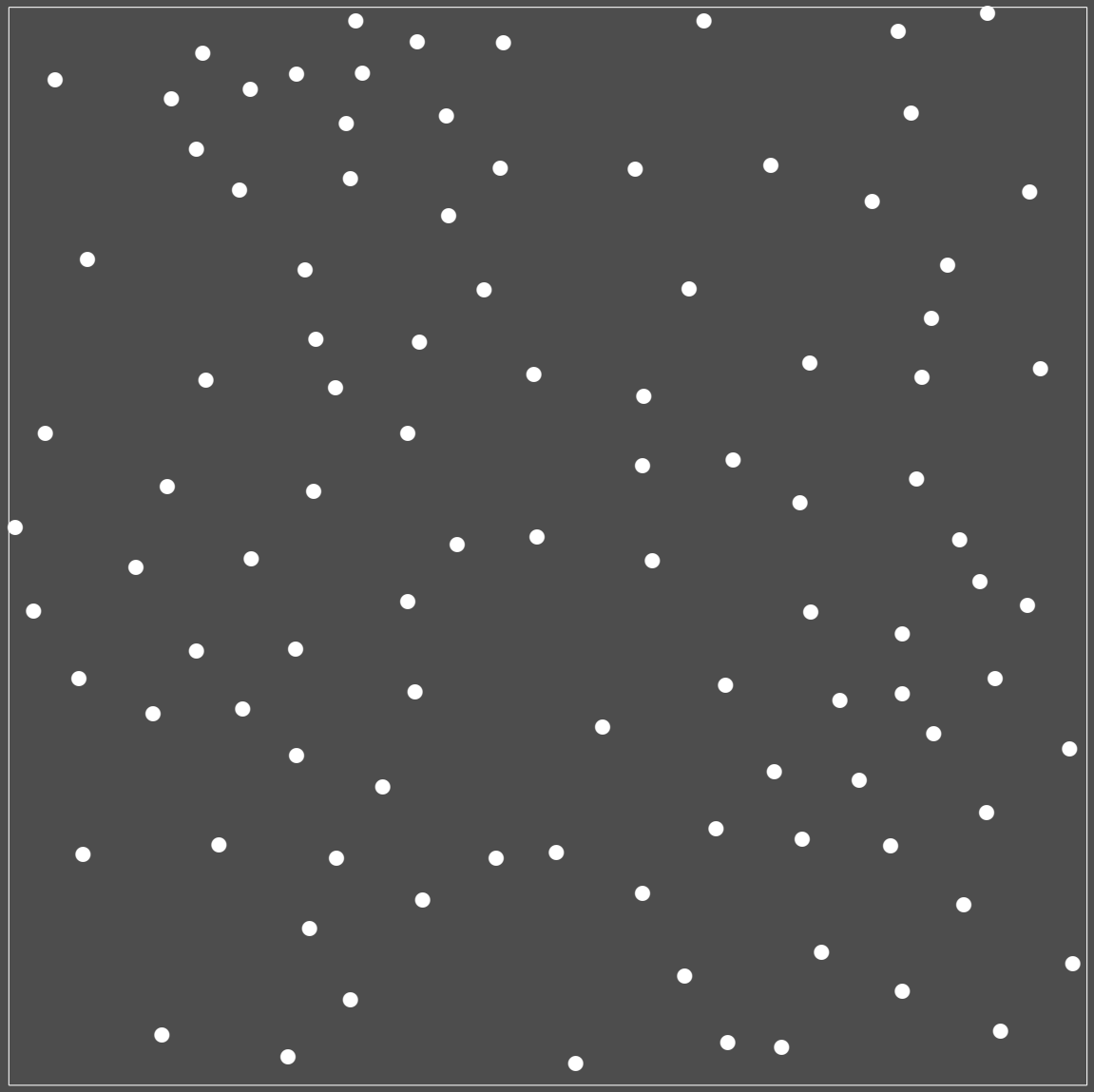
\- No incremental sampling

Stratification by Lloyd-Relaxation



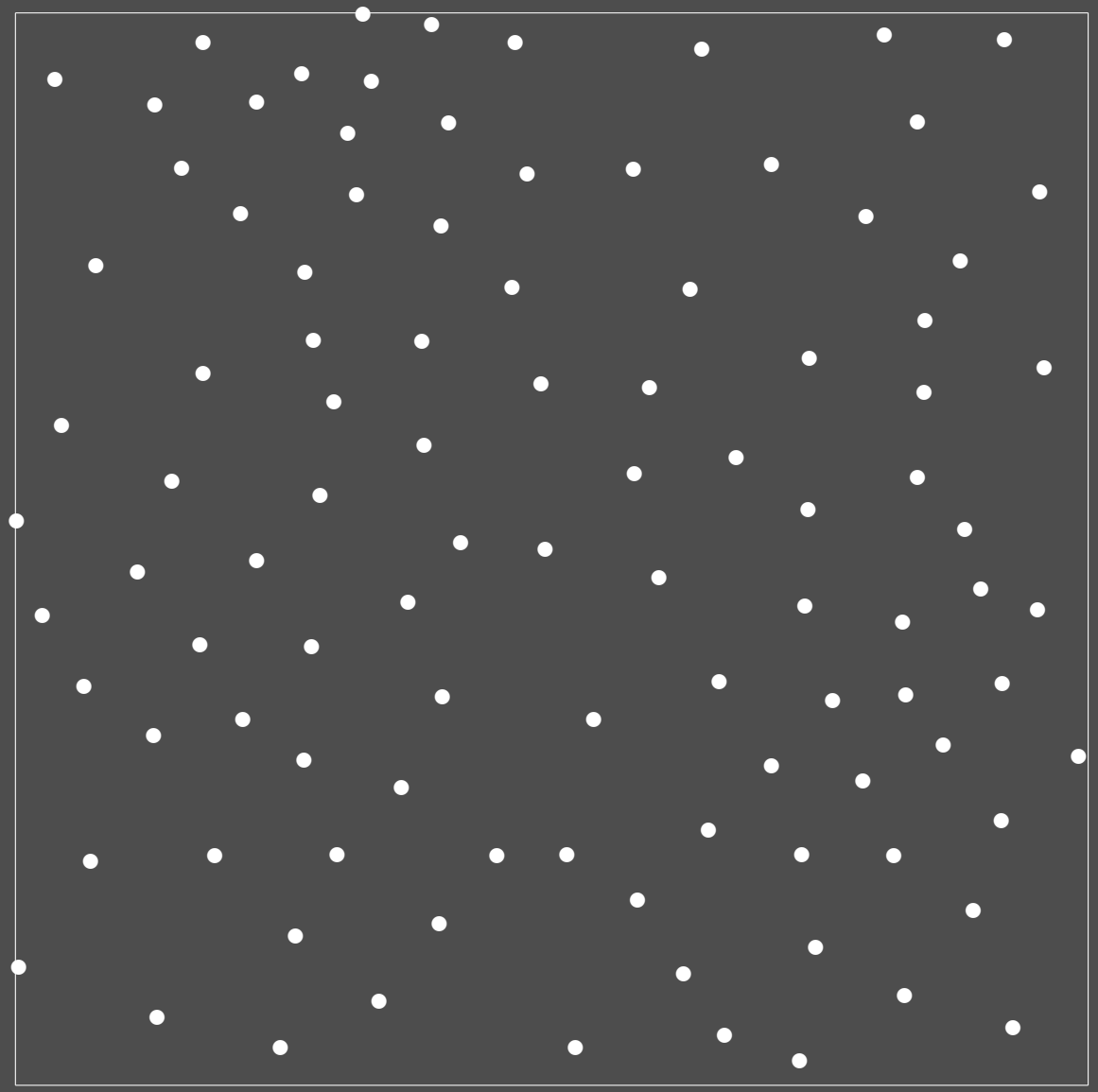
- Iteration 0

Stratification by Lloyd-Relaxation



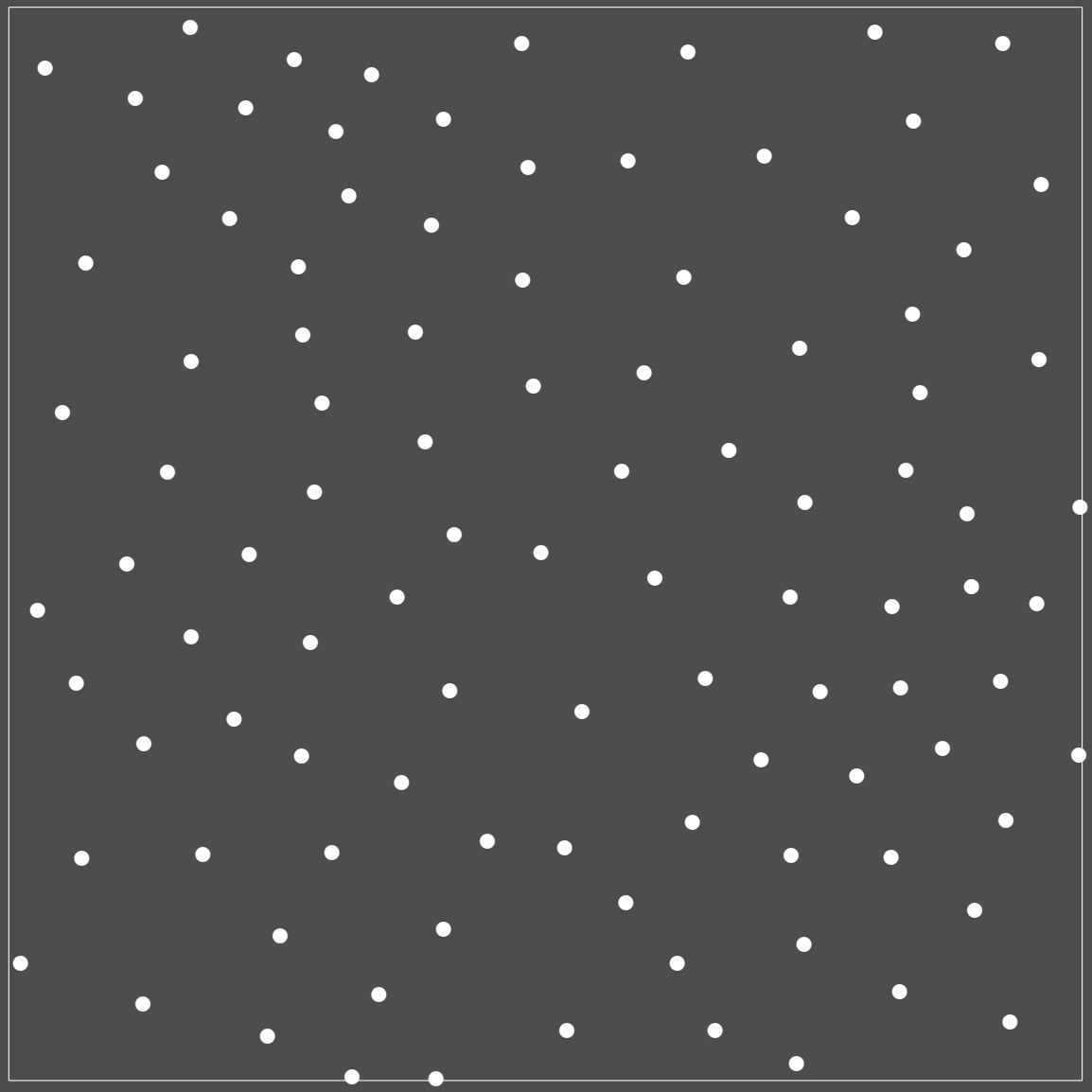
- Iteration 1

Stratification by Lloyd-Relaxation



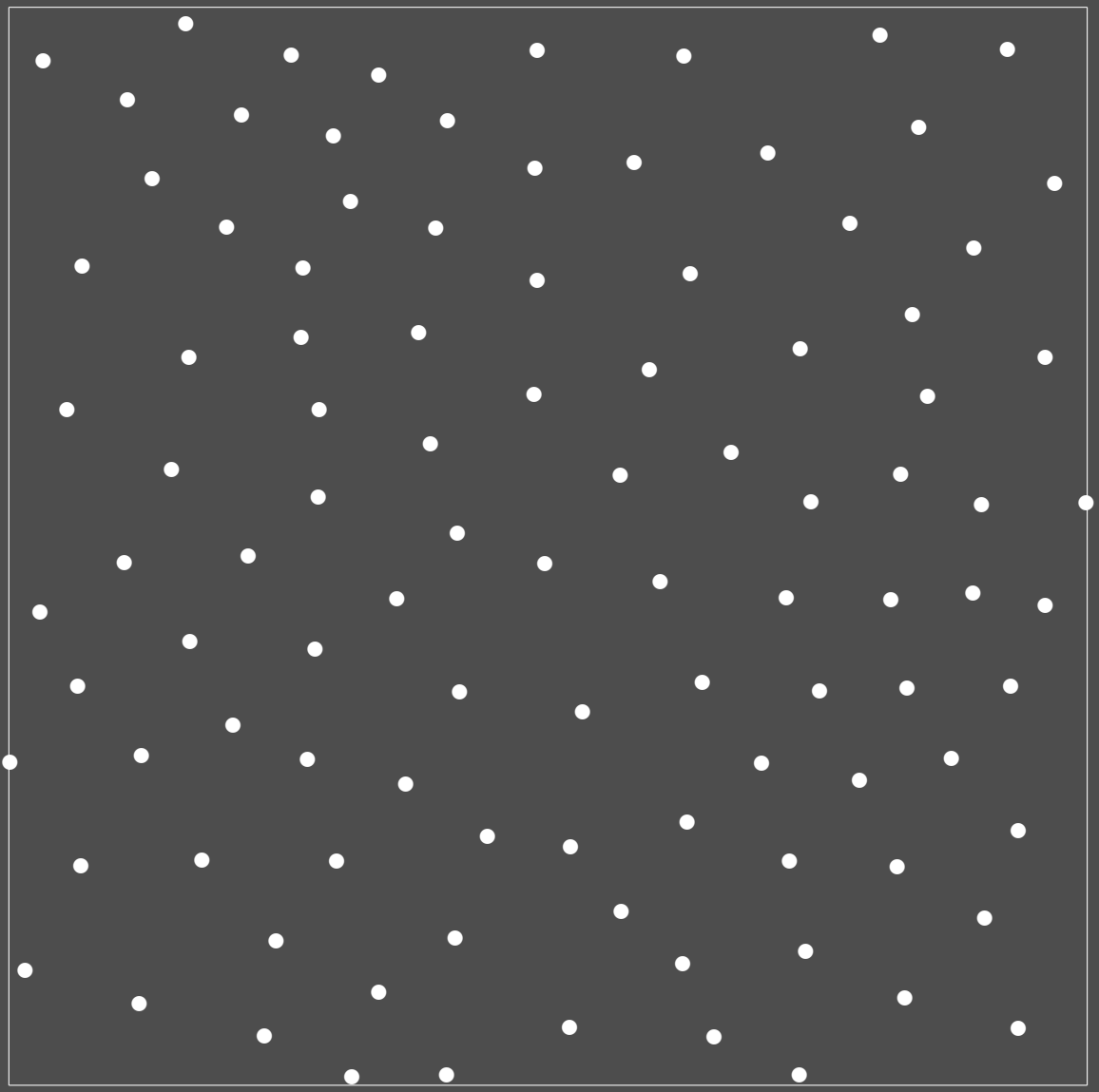
- Iteration 2

Stratification by Lloyd-Relaxation



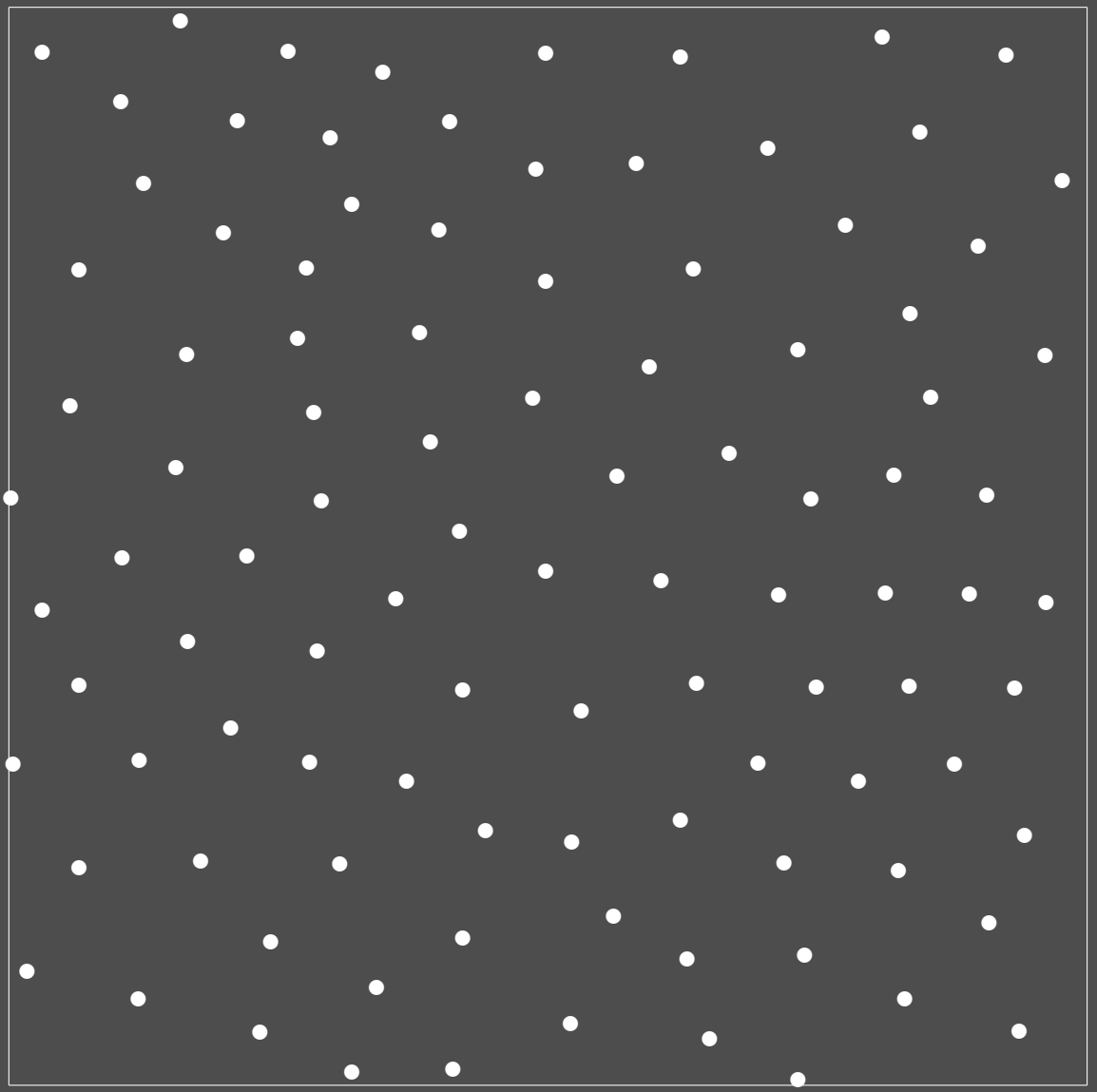
- Iteration 3

Stratification by Lloyd-Relaxation



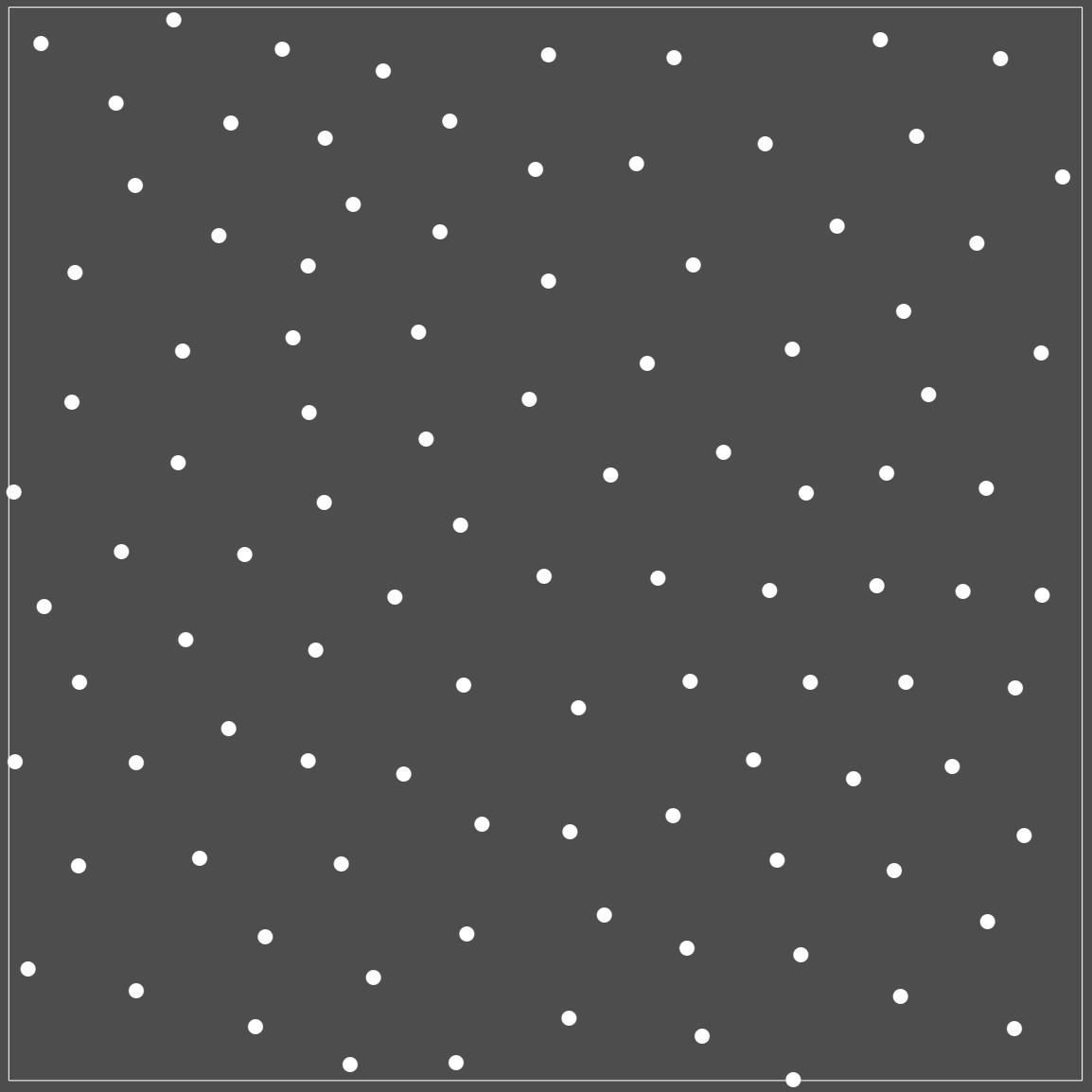
- Iteration 4

Stratification by Lloyd-Relaxation



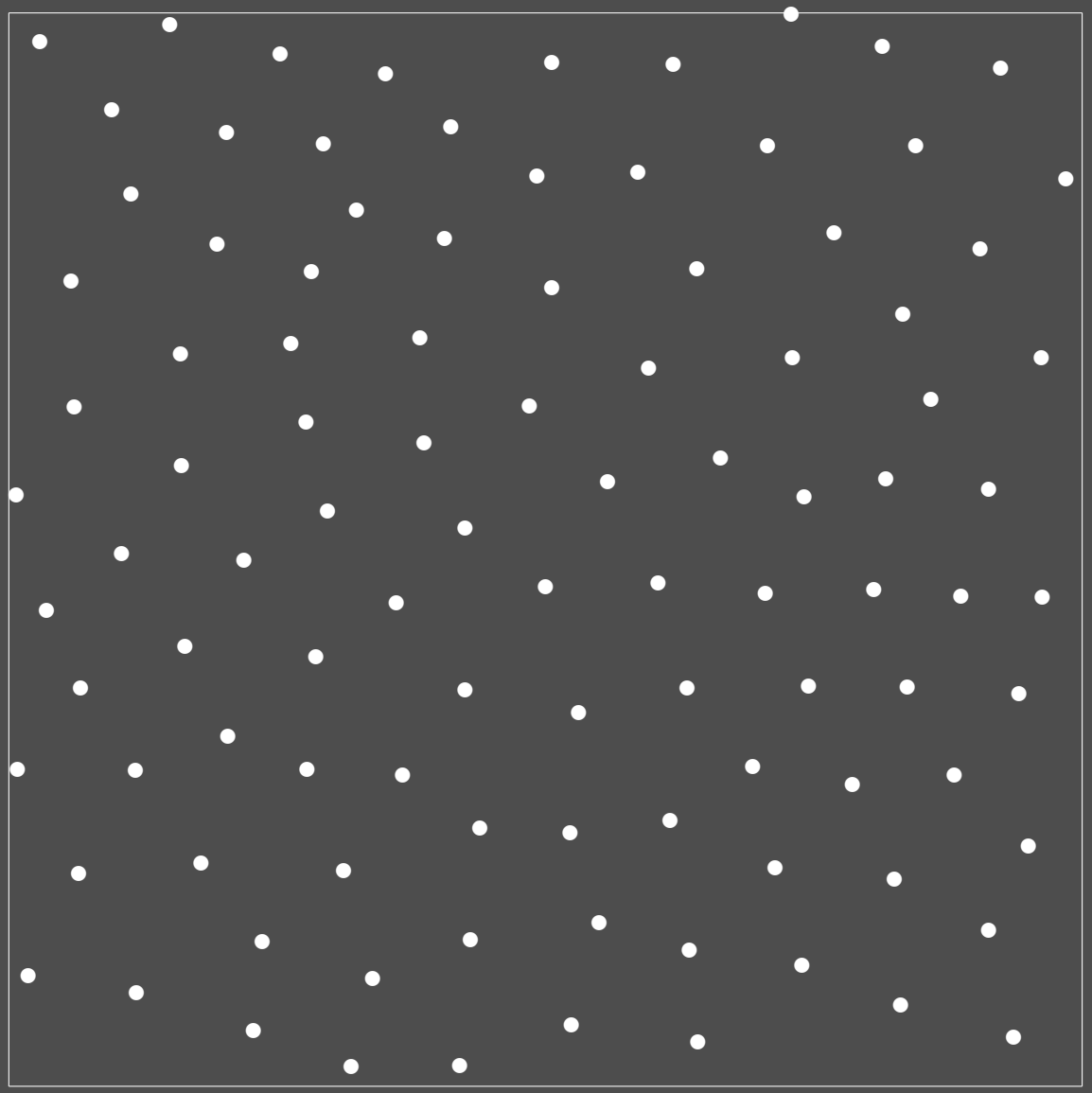
- Iteration 5

Stratification by Lloyd-Relaxation



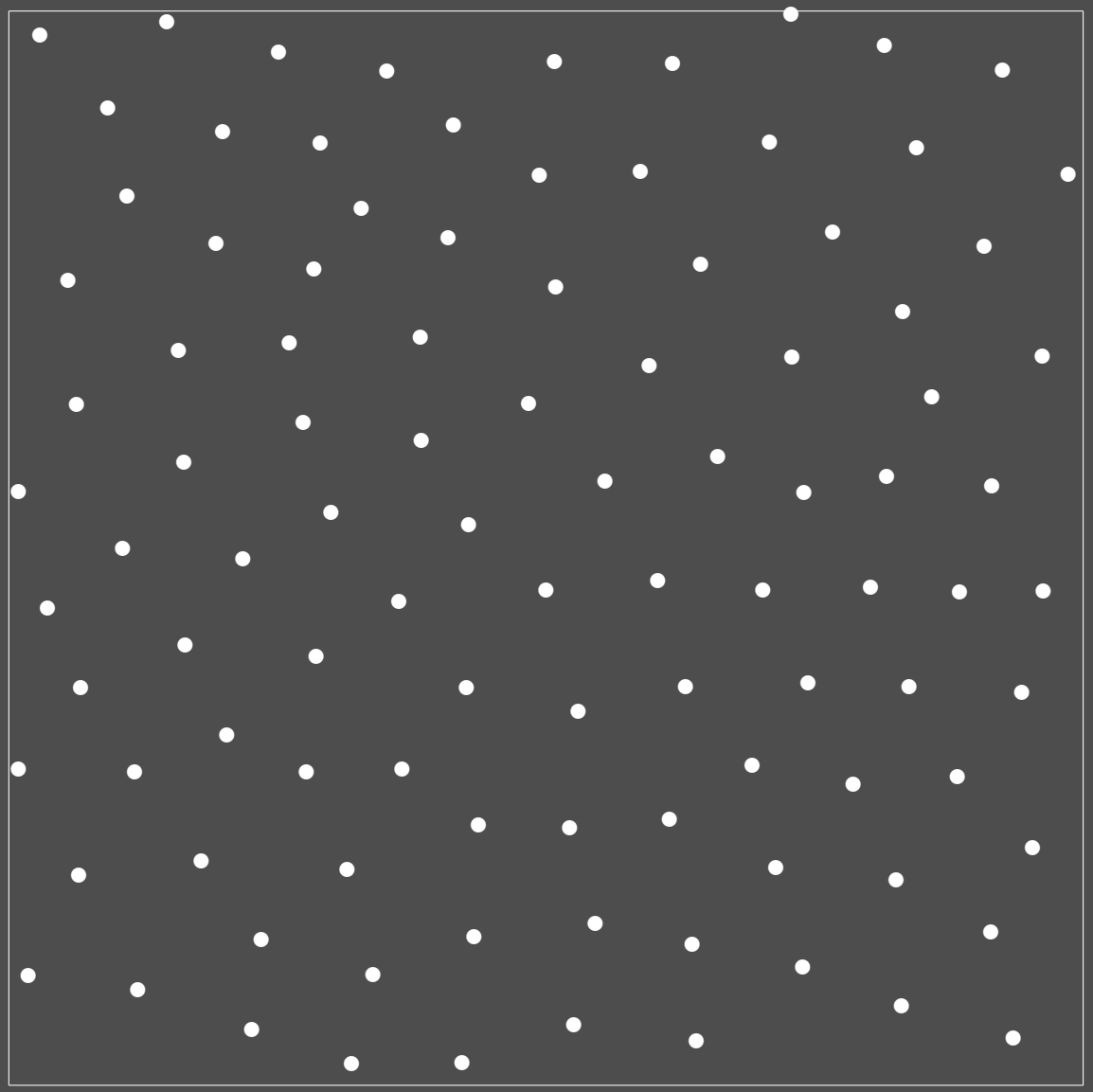
- Iteration 6

Stratification by Lloyd-Relaxation



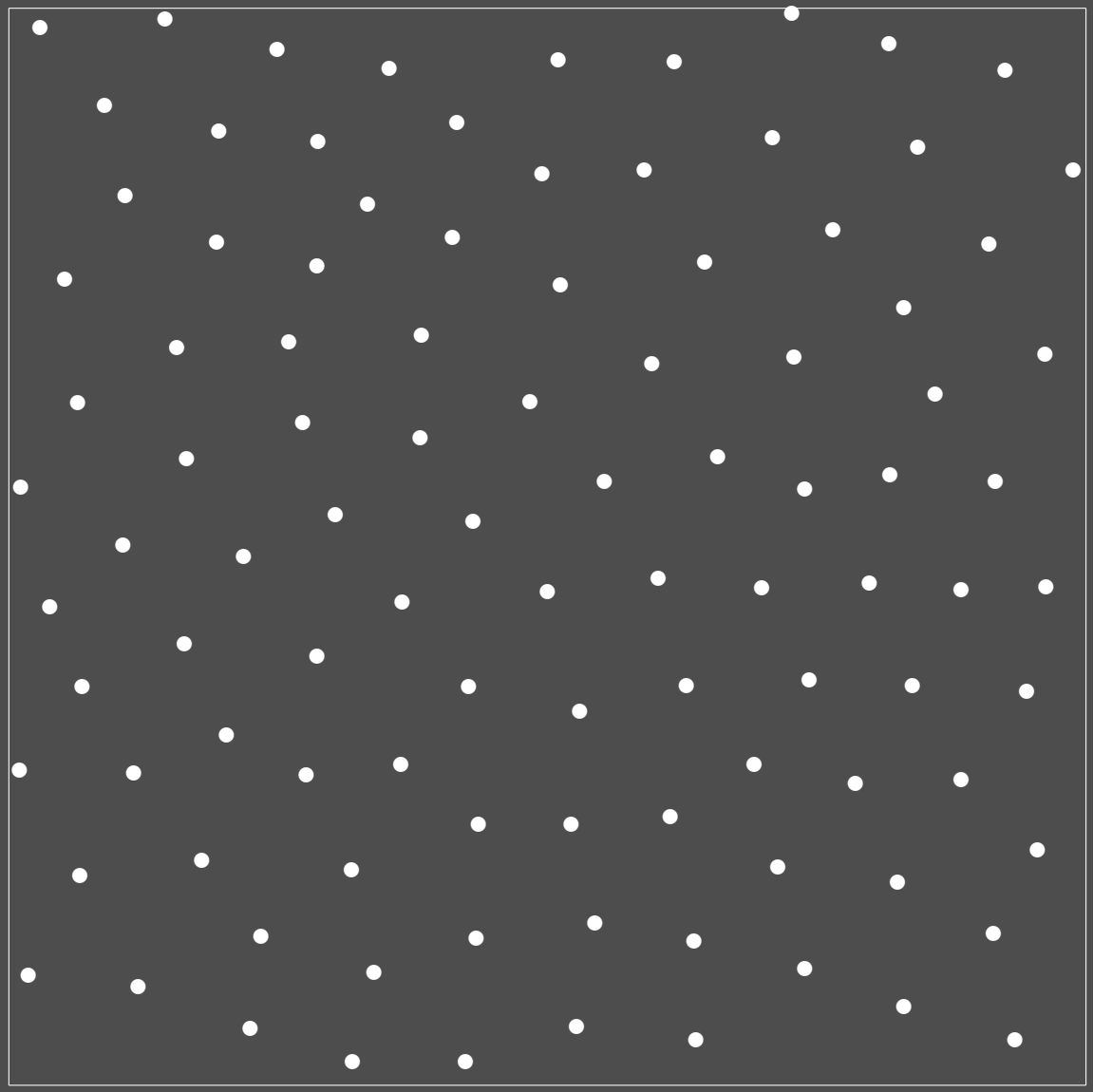
- Iteration 7

Stratification by Lloyd-Relaxation



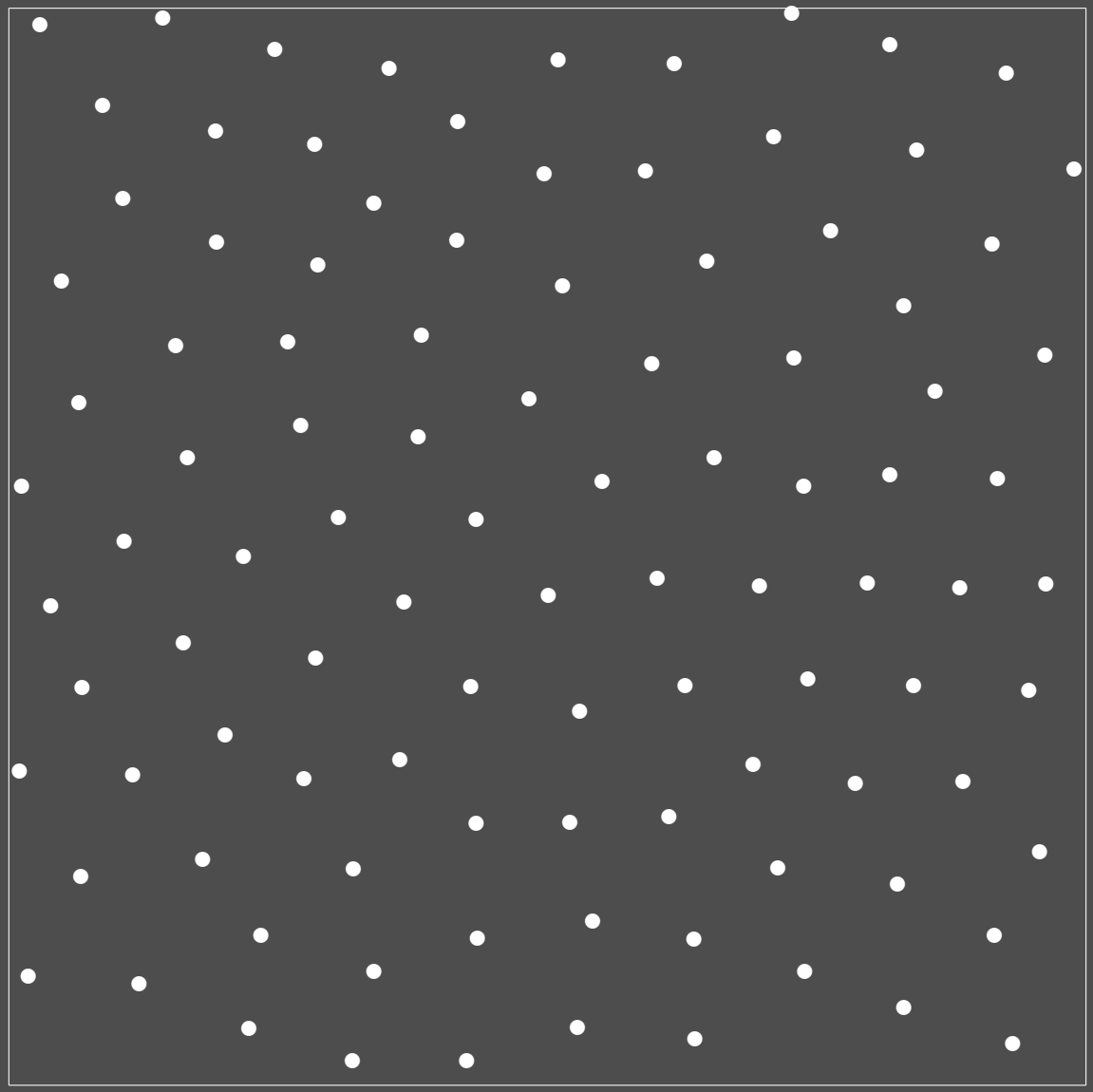
- Iteration 8

Stratification by Lloyd-Relaxation



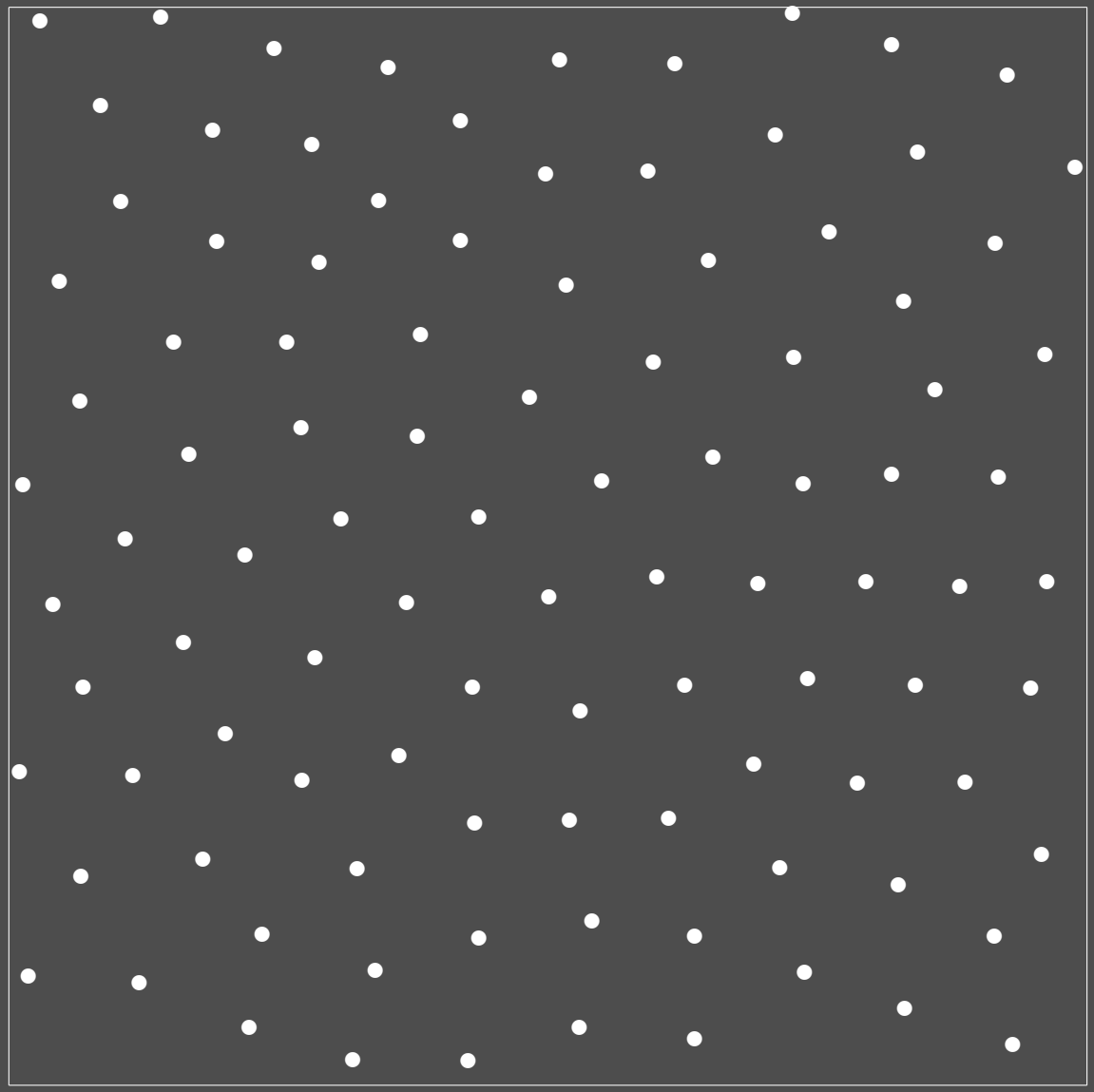
- Iteration 9

Stratification by Lloyd-Relaxation



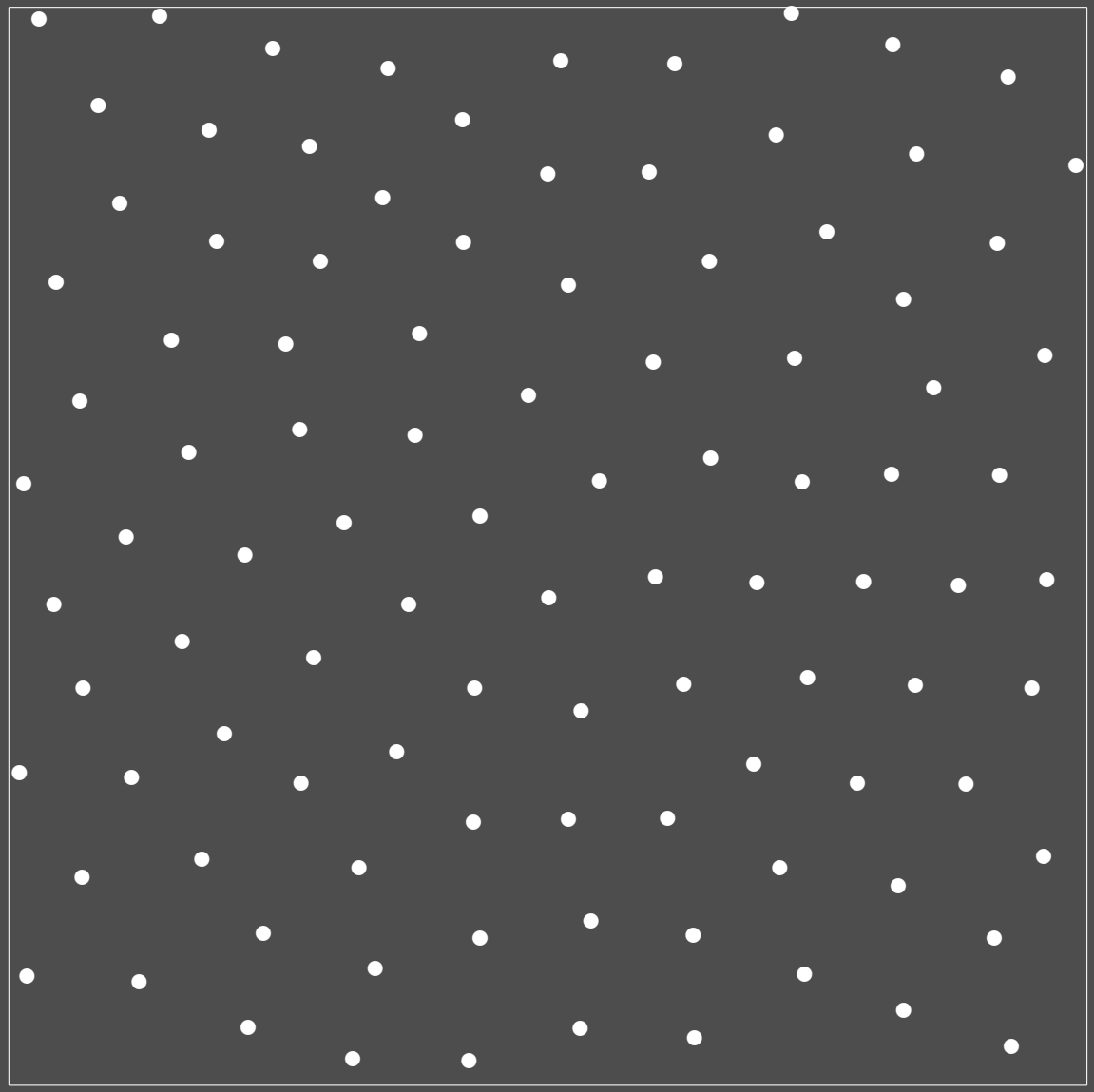
- Iteration 10

Stratification by Lloyd-Relaxation



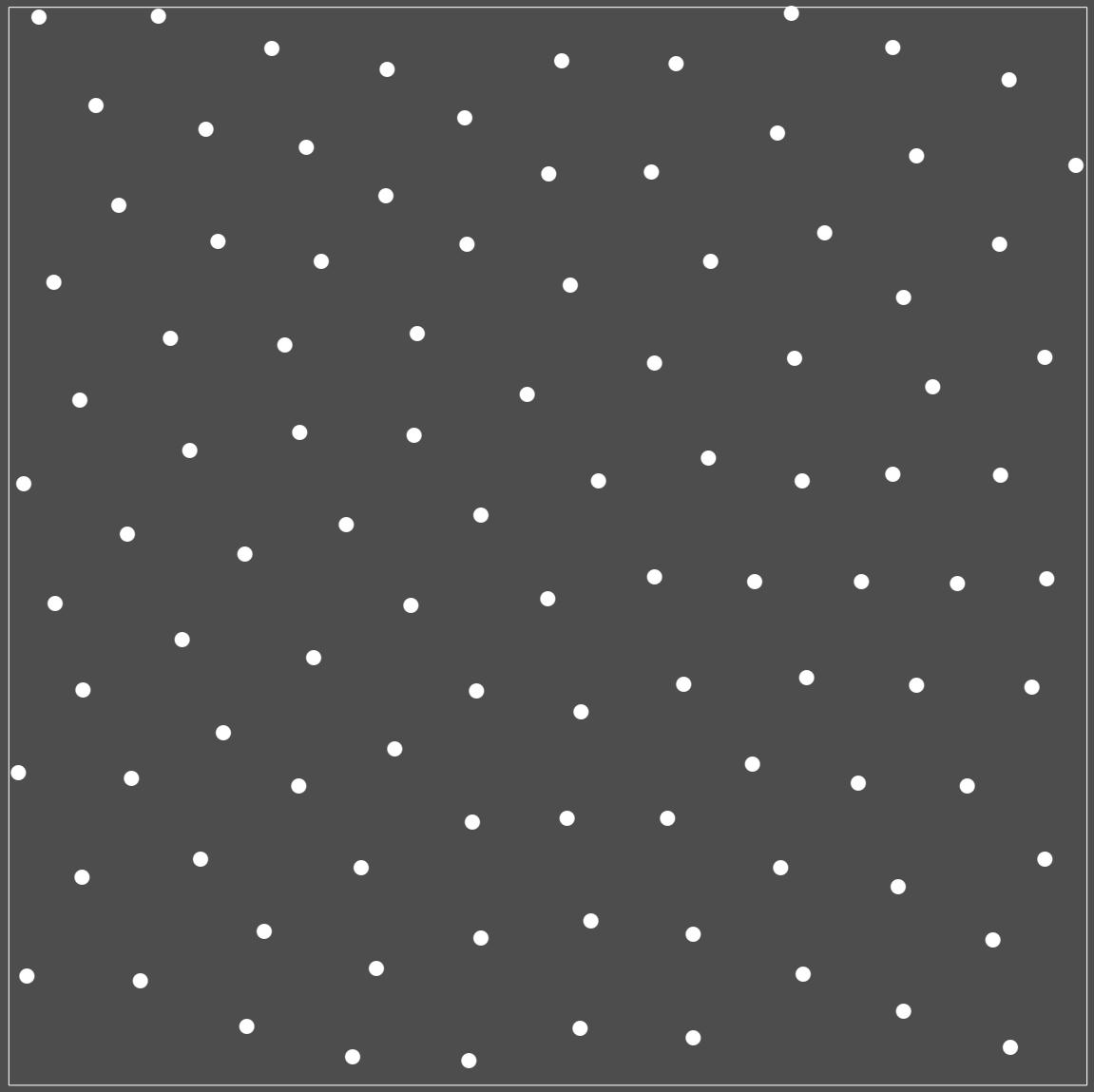
- Iteration 11

Stratification by Lloyd-Relaxation



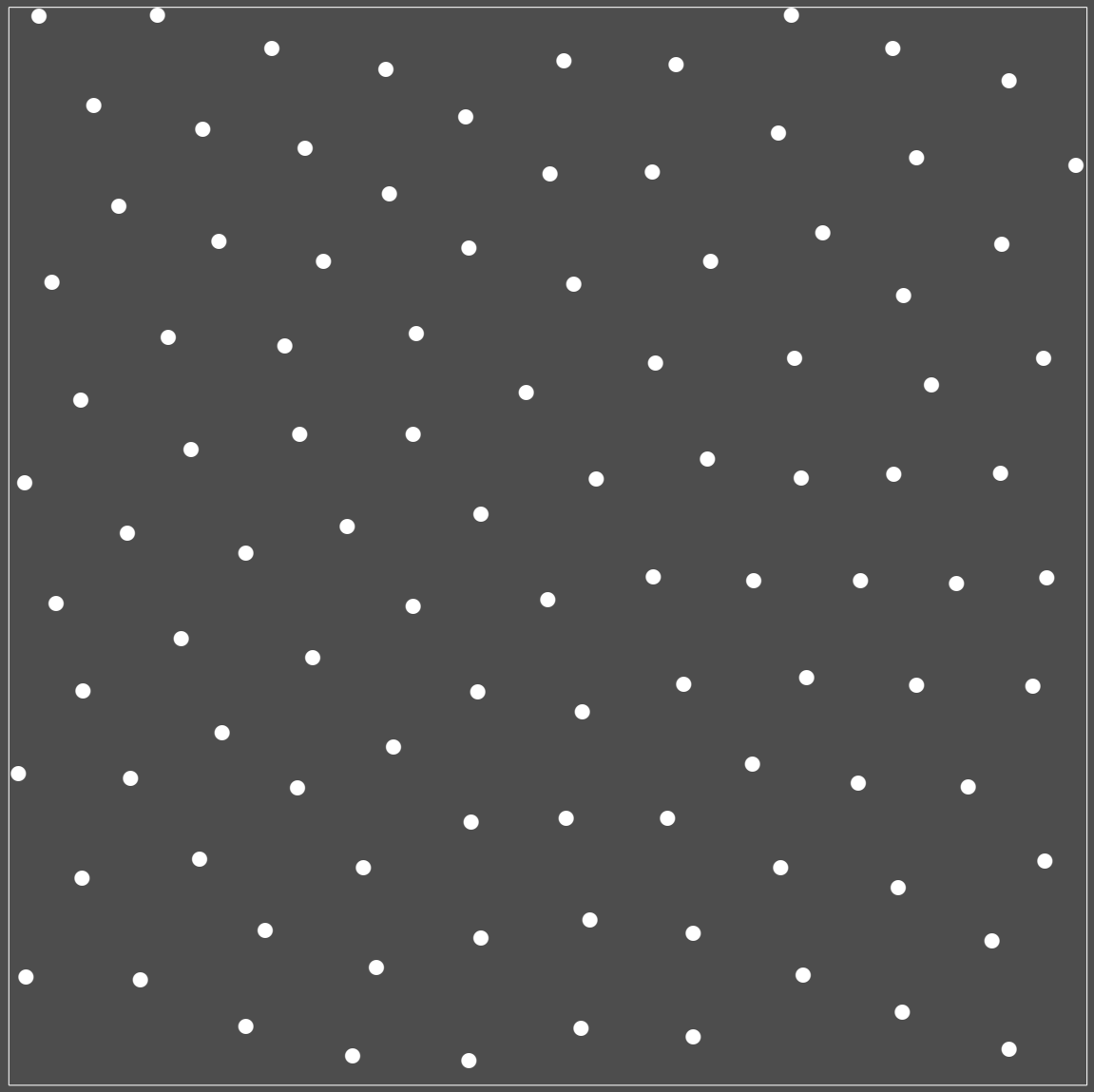
- Iteration 12

Stratification by Lloyd-Relaxation



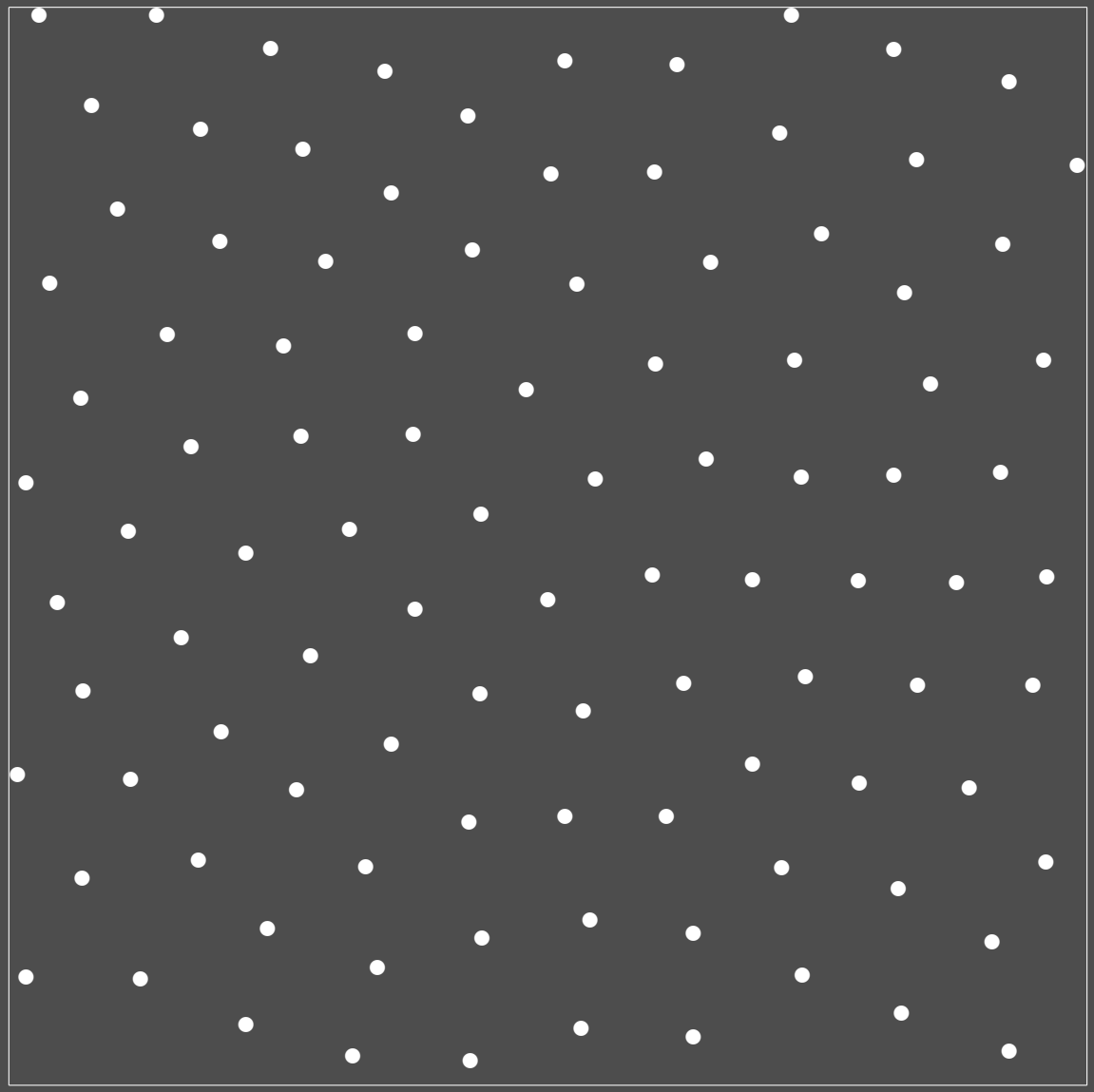
- Iteration 13

Stratification by Lloyd-Relaxation



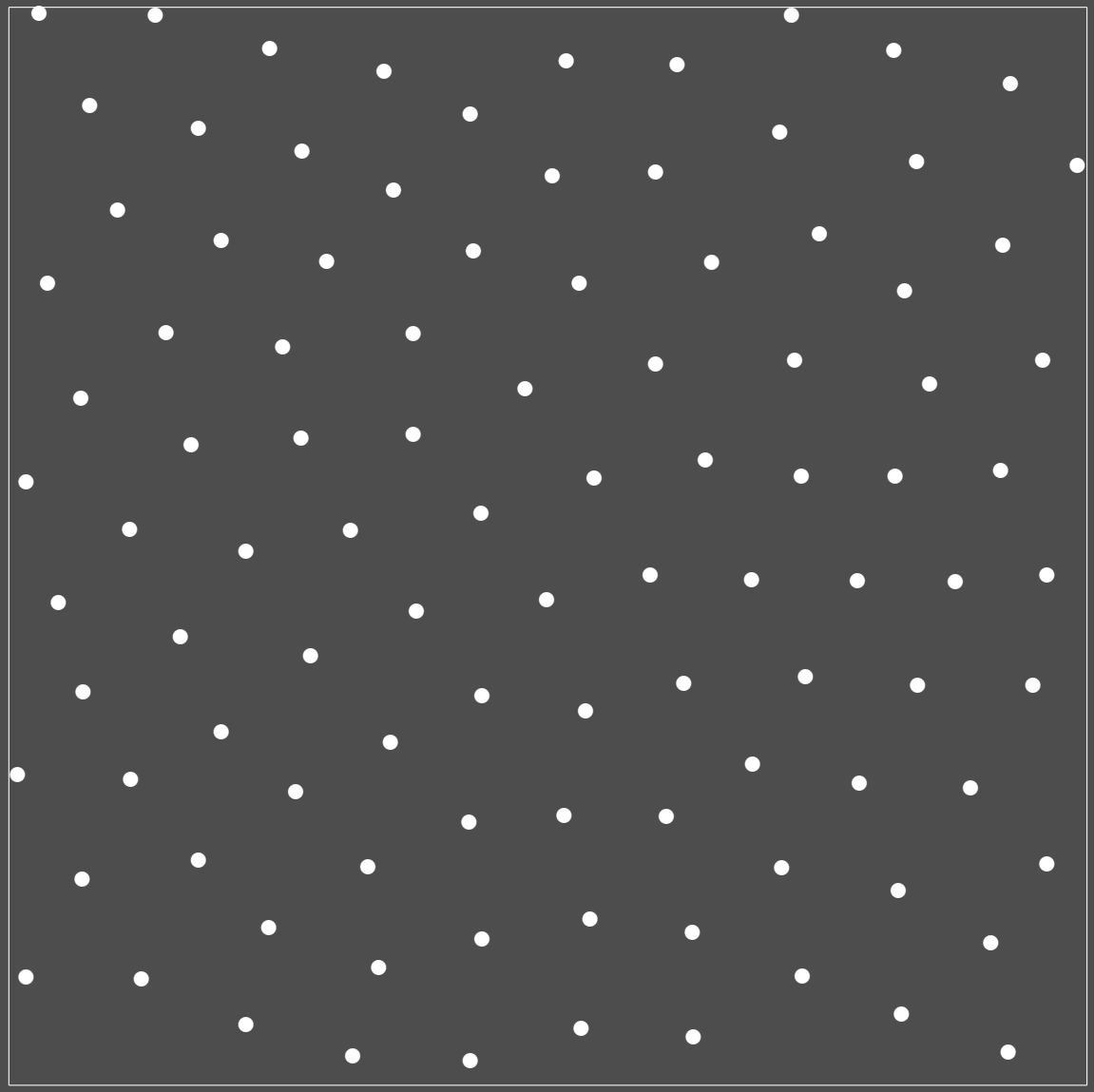
- Iteration 14

Stratification by Lloyd-Relaxation



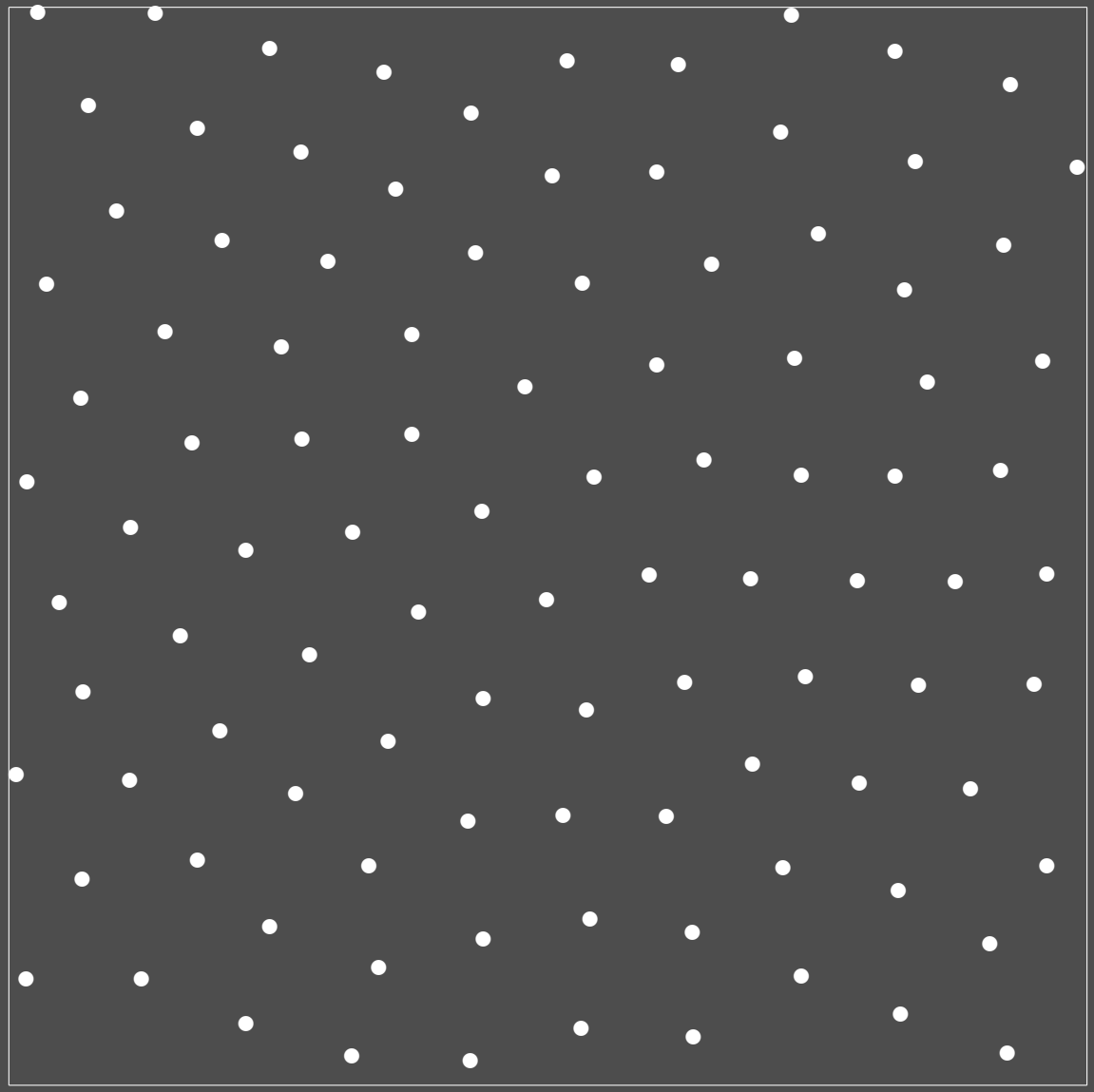
- Iteration 15

Stratification by Lloyd-Relaxation



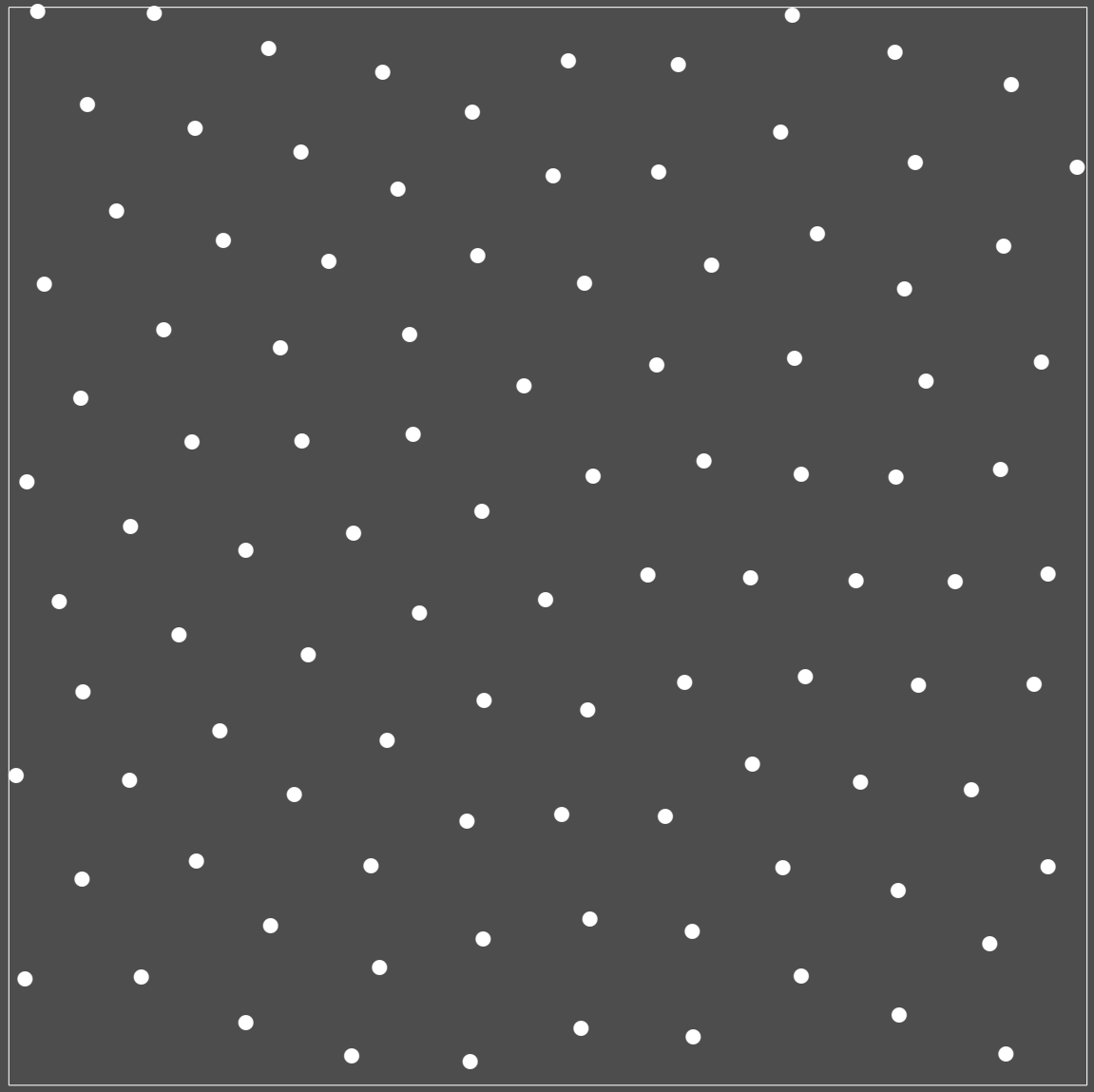
- Iteration 16

Stratification by Lloyd-Relaxation



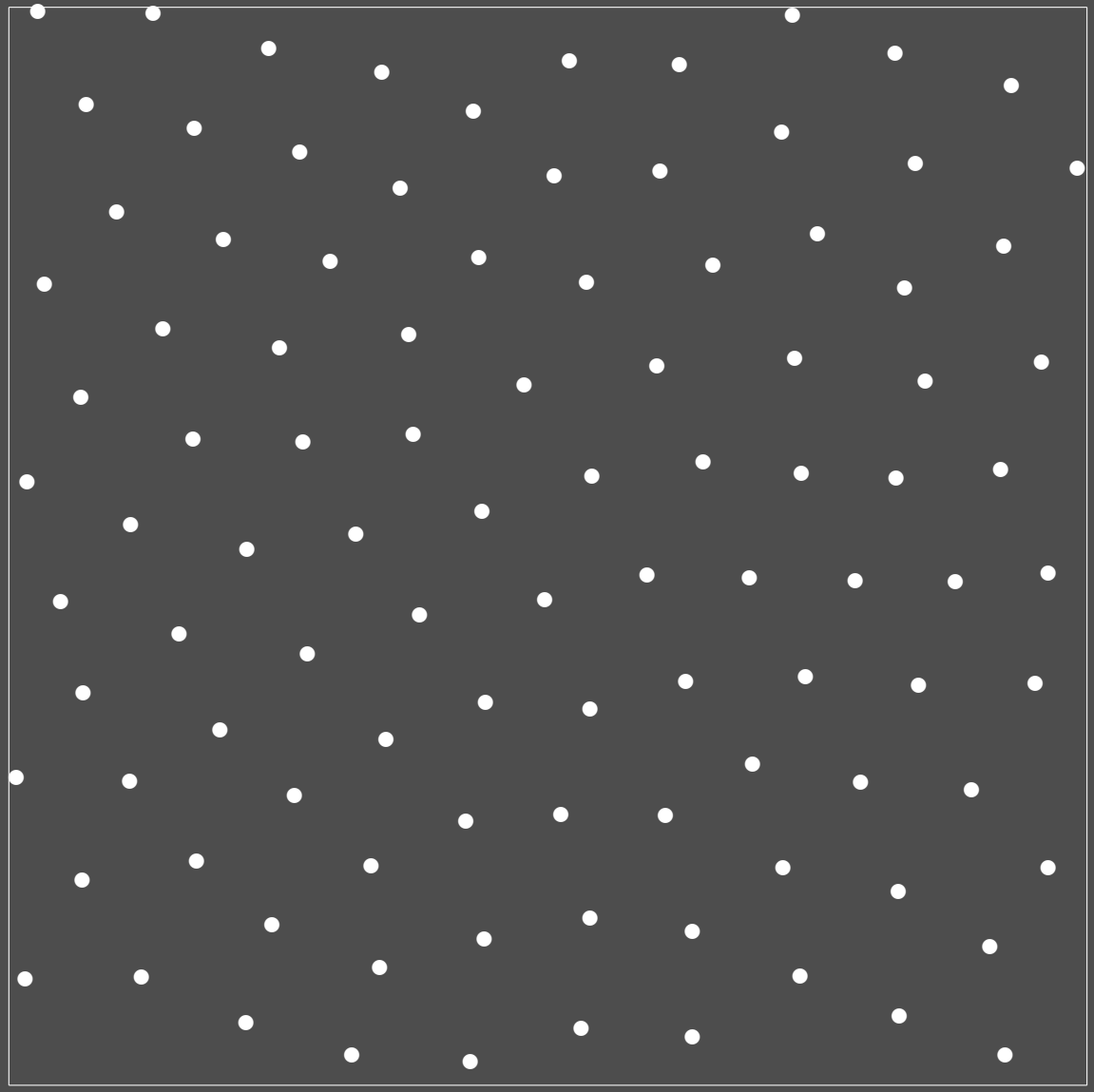
- Iteration 17

Stratification by Lloyd-Relaxation



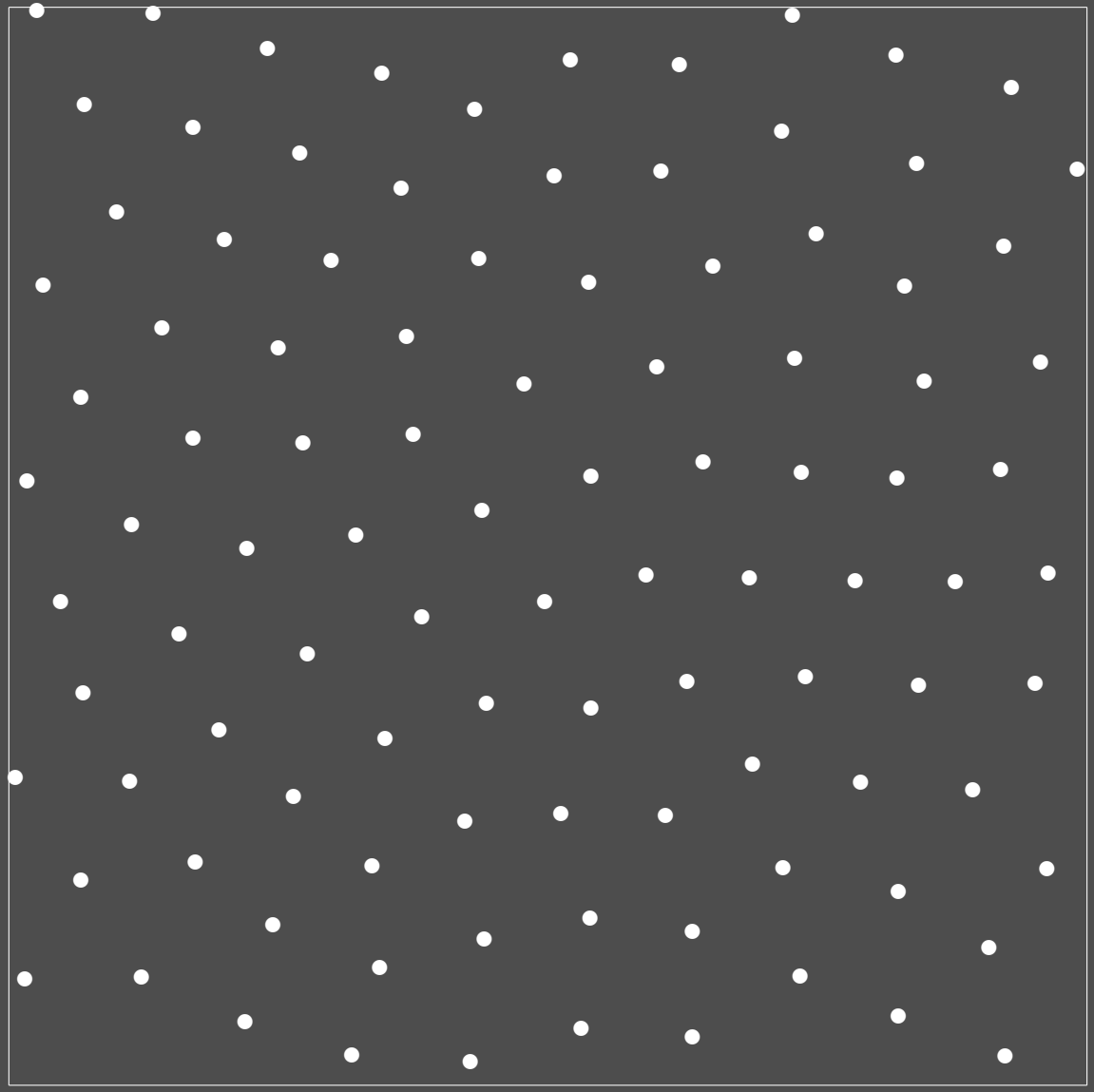
- Iteration 18

Stratification by Lloyd-Relaxation



- Iteration 19

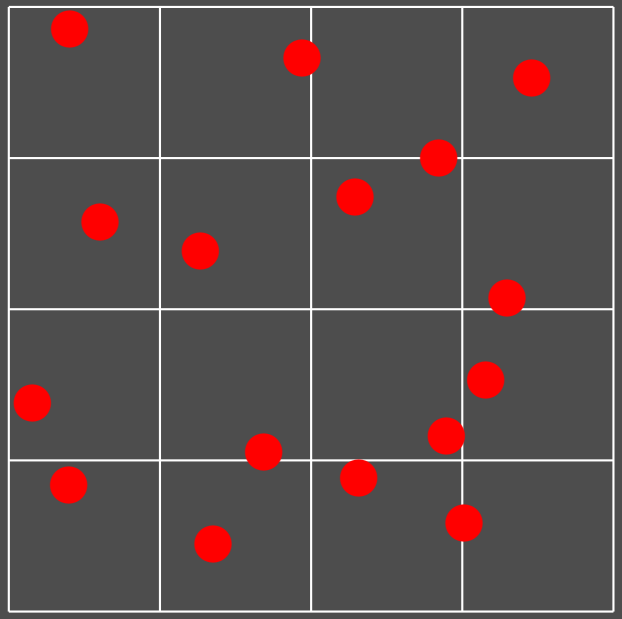
Stratification by Lloyd-Relaxation



- Iteration 20

Stratification: Jittered Sampling

- Division of each axis into N_j intervals for $N = \prod_{j=1}^{s} N_j$



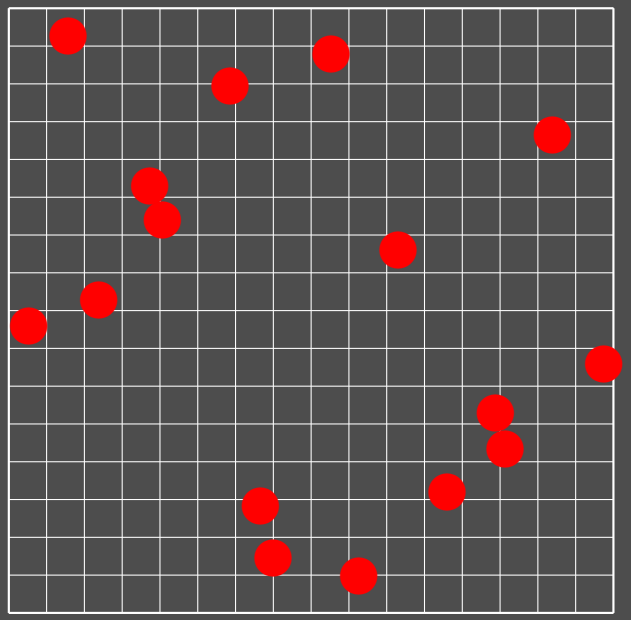
- Increased efficiency by increased uniformity of distribution
- Problem: N must be factorized

Latin Hypercube Sampling (N-Rooks Sampling)

- Using s uniform random permutations $\sigma_N^{(j)}$ of size N yields

$$x_i = \left(\frac{\sigma_N^{(1)}(i) + \xi}{N}, \ldots, \frac{\sigma_N^{(s)}(i) + \xi}{N} \right)$$

where $\sigma_N^{(1)}$ can be chosen as identity

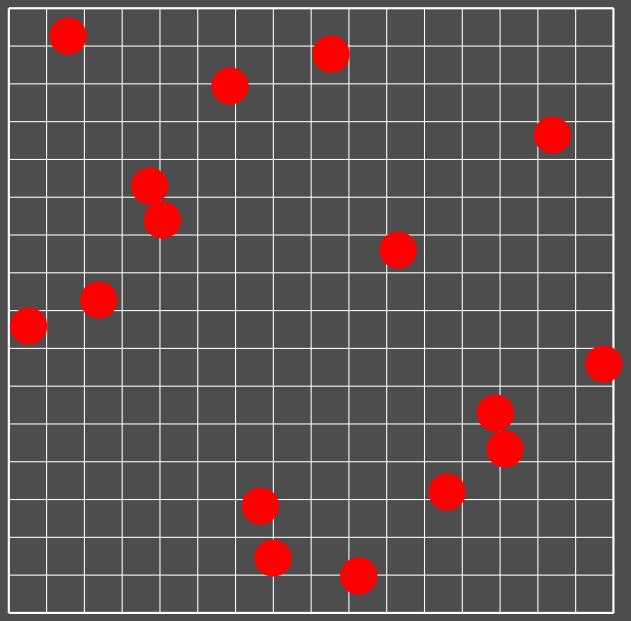


Latin Hypercube Sampling (N-Rooks Sampling)

- Using s uniform random permutations $\sigma_N^{(j)}$ of size N yields

$$x_i = \left(\frac{\sigma_N^{(1)}(i) + \xi}{N}, \ldots, \frac{\sigma_N^{(s)}(i) + \xi}{N} \right)$$

where $\sigma_N^{(1)}$ can be chosen as identity



- Cannot be much worse than uniform random sampling

$$\sigma^2(f_{\text{LHS}}) \leq \frac{N}{N-1}\sigma^2(f_{\text{MC}})$$

Replication Heuristics: Stratification

- Heuristic with
  - weights $w_j = \lambda_s(A_j)$, and
  - mappings $R_j : I^s \rightarrow A_j$
- Independent sampling for $N_j = \lambda_s(A_j)N$

$$\int_{I^s} f(x)dx \approx \sum_{j=0}^{M-1} \frac{1}{N_j} \sum_{i=0}^{N_j-1} \lambda_s(A_j) f(R_j(x_{i,j})) = \frac{1}{N} \sum_{j=0}^{M-1} \sum_{i=0}^{N_j-1} f(R_j(x_{i,j}))$$

- Dependent sampling

$$\int_{I^s} f(x)dx \approx \frac{1}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \lambda_s(A_j) f(R_j(x_i))$$

Replication Heuristics: Regularization

- Antithetic variables

$$\int_I f(x)dx = \int_I \frac{1}{2}f(x) + \frac{1}{2}f(1-x)dx \approx \frac{1}{2N}\sum_{i=0}^{N-1}\left(f(x_i) + f(1-x_i)\right)$$

  - sample points doubled and symmetrized
  - more efficient if variance reduced to less than half of original variance
  - good for monotonic problems
  - effect killed by independent sampling !

Replication Heuristics: Regularization

- Antithetic variables

$$\int_I f(x)dx = \int_I \frac{1}{2}f(x) + \frac{1}{2}f(1-x)dx \approx \frac{1}{2N}\sum_{i=0}^{N-1}\left(f(x_i) + f(1-x_i)\right)$$

  - sample points doubled and symmetrized
  - more efficient if variance reduced to less than half of original variance
  - good for monotonic problems
  - effect killed by independent sampling !

- Combining stratification

$$f_{\text{strat}}(x) = \frac{1}{2}\left(f\left(\frac{x}{2}\right) + f\left(1-\frac{x}{2}\right)\right)$$

  and antithetic variables

$$\int_I f_{\text{strat, anti}}(x)dx \approx \frac{1}{4N}\sum_{i=0}^{N-1}\left(f\left(\frac{x_i}{2}\right) + f\left(1-\frac{x_i}{2}\right) + f\left(\frac{1}{2}+\frac{x_i}{2}\right) + f\left(\frac{1}{2}-\frac{x_i}{2}\right)\right)$$

Splitting

- Instead of

$$\int_{I^{s1}} \int_{I^{s2}} f(x,y)dydx \approx \frac{1}{N} \sum_{i=0}^{N-1} f(x_i, y_i)$$

  computational complexity can be improved by

$$\int_{I^{s1}} \int_{I^{s2}} f(x,y)dydx \approx \frac{1}{NM} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} f(x_i, y_{i,j})$$

- Low pass filtering of problematic dimensions of the integrand
  - e.g. splitting for shadow rays

Replication Heuristics: Dependent Splitting

- Splitting considered as a replication heuristic restricted to selected dimensions

$$\int_{I^{s1}} \int_{I^{s2}} f(x,y)dydx = \int_{I^{s1}} \int_{I^{s2}} \sum_{j=0}^{M-1} w_j(x,y) f(x, R_j(x,y))dydx$$

$$\approx \frac{1}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} w_j(x_i,y_i) f(x_i, R_j(x_i,y_i))dydx$$

- Realize splitting much more efficiently by e.g.
  - stratification heuristic (independent sampling)
  - randomized quadratures (dependent sampling)

Summary

- Simulation of random variables and fields
- Monte Carlo integration
- Method of dependent tests
- Efficiency and time complexity
- Dependent sampling
- Replication

Summary

- Simulation of random variables and fields
- Monte Carlo integration
- Method of dependent tests
- Efficiency and time complexity
- Dependent sampling
- Replication

$\Rightarrow$ **Use as few random numbers as possible**

Beyond Monte Carlo

- Day 1: Monte Carlo
- **Day 2: Quasi-Monte Carlo points**
- Day 3: Quasi-Monte Carlo integration
- Day 4: Monte Carlo extensions of quasi-Monte Carlo
- Day 5: Applications to computer graphics

Day 2: Quasi-Monte Carlo Points

- Discrepancy
- Deterministic low discrepancy
  - Halton and Hammersley points
  - Scrambling
  - (t, m, s)-nets and (t, s)-sequences
  - Digital constructions
  - Good lattice points

Discrepancy

- **Definition:** The ***discrepancy***

$$D(P_N, \mathcal{A}) := \sup_{A \in \mathcal{A}} \left| \lambda_s(A) - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i) \right|$$

  is a measure of the uniform distribution of a given point set $P_N = \{x_0, \ldots, x_{N-1}\}$ with respect to non-empty families $\mathcal{A}$ of Lebesgue-measurable subsets of I^s. χ_A is the characteristic function of the set A.

Discrepancy

- **Definition:** The ***discrepancy***

$$D(P_N, \mathcal{A}) := \sup_{A \in \mathcal{A}} \left| \lambda_s(A) - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i) \right|$$

  is a measure of the uniform distribution of a given point set $P_N = \{x_0, \ldots, x_{N-1}\}$ with respect to non-empty families $\mathcal{A}$ of Lebesgue-measurable subsets of I^s. χ_A is the characteristic function of the set A.

- $D(P_N, \mathcal{A}) \sim$ worst case integration error

Discrepancy

- **Definition:** The ***discrepancy***

$$D(P_N, \mathcal{A}) := \sup_{A \in \mathcal{A}} \left| \lambda_s(A) - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i) \right|$$

  is a measure of the uniform distribution of a given point set $P_N = \{x_0, \ldots, x_{N-1}\}$ with respect to non-empty families $\mathcal{A}$ of Lebesgue-measurable subsets of I^s. χ_A is the characteristic function of the set A.

- $D(P_N, \mathcal{A}) \sim$ worst case integration error

- (Star-) discrepancy

$$D^*(P_N) := D\left(P_N, \left\{A | A = \prod_{j=1}^{s} [0, a_j) \subset I^s\right\}\right)$$

- Extreme discrepancy

$$D(P_N) := D\left(P_N, \left\{A | A = \prod_{j=1}^{s} [a_j, b_j) \subset I^s\right\}\right)$$

Discrepancy

- **Definition:** The ***discrepancy***

$$D(P_N, \mathcal{A}) := \sup_{A \in \mathcal{A}} \left| \lambda_s(A) - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i) \right|$$

  is a measure of the uniform distribution of a given point set $P_N = \{x_0, \ldots, x_{N-1}\}$ with respect to non-empty families $\mathcal{A}$ of Lebesgue-measurable subsets of I^s. χ_A is the characteristic function of the set A.

- $D(P_N, \mathcal{A}) \sim$ worst case integration error

- (Star-) discrepancy

$$D^*(P_N) := D\left(P_N, \left\{A | A = \prod_{j=1}^{s} [0, a_j) \subset I^s\right\}\right)$$

- Extreme discrepancy

$$D(P_N) := D\left(P_N, \left\{A | A = \prod_{j=1}^{s} [a_j, b_j) \subset I^s\right\}\right)$$

- The (Star-) discrepancy and extreme discrepancy are anisotropic measures

Discrepancy Bounds

- Case $s = 1$: Discrepancy is size of largest gap

$$D^*(P_N) \geq \frac{1}{2N}$$

$$D(P_N) \geq \frac{1}{N}$$

- General case

$$D^*(P_N) \geq B_s \frac{\log^{\frac{s-1}{2}} N}{N}$$

Discrepancy Bounds

- Case $s = 1$: Discrepancy is size of largest gap

$$D^*(P_N) \geq \frac{1}{2N}$$

$$D(P_N) \geq \frac{1}{N}$$

- General case

$$D^*(P_N) \geq B_s \frac{\log^{\frac{s-1}{2}} N}{N}$$

- Discrepancy of random points

$$D^*(P_N^{\text{random}}) \in \mathcal{O}\left(\sqrt{\frac{\log \log N}{N}}\right)$$

- Discrepancy of regular grids

$$D^*(P_N) \in \mathcal{O}\left(\frac{1}{\sqrt[s]{N}}\right)$$

Discrepancy Bounds

- Case $s = 1$: Discrepancy is size of largest gap

$$D^*(P_N) \geq \frac{1}{2N}$$

$$D(P_N) \geq \frac{1}{N}$$

- General case

$$D^*(P_N) \geq B_s \frac{\log^{\frac{s-1}{2}} N}{N}$$

- Discrepancy of random points

$$D^*(P_N^{\text{random}}) \in \mathcal{O}\left(\sqrt{\frac{\log \log N}{N}}\right)$$

- Discrepancy of regular grids

$$D^*(P_N) \in \mathcal{O}\left(\frac{1}{\sqrt[s]{N}}\right)$$

  - includes points taken from space filling curves like e.g. the Hilbert curve

Uniform and Completely Uniform Distribution

- By the theory of uniform distribution

  (x_i) is uniformly distributed in I^s

  $\Leftrightarrow \lim_{N\to\infty} D(P_N) = 0$

  $\Leftrightarrow \lim_{N\to\infty} D^*(P_N) = 0$

Uniform and Completely Uniform Distribution

- By the theory of uniform distribution

  (x_i) is uniformly distributed in I^s

  $\Leftrightarrow \lim_{N\to\infty} D(P_N) = 0$

  $\Leftrightarrow \lim_{N\to\infty} D^*(P_N) = 0$

- **Definition:** A sequence (x_i) of numbers in I is **completely uniformly distributed** if for every $s \in \mathbb{N}$ the sequence of points $(x_n, x_{n+1}, \ldots, x_{n+s-1})$ is uniformly distributed in I^s for $n \in \mathbb{N}_0$.
- Formalization of independence

Quasi-Monte Carlo Point Sets

- Low discrepancy means

$$D^*(P_N) \in \mathcal{O}\left(\frac{\log^s N}{N}\right)$$

Quasi-Monte Carlo Point Sets

- Low discrepancy means

$$D^*(P_N) \in \mathcal{O}\left(\frac{\log^s N}{N}\right)$$

- Low discrepancy sequences cannot be completely uniformly distributed

Quasi-Monte Carlo Point Sets

- Low discrepancy means

$$D^*(P_N) \in \mathcal{O}\left(\frac{\log^s N}{N}\right)$$

- Low discrepancy sequences cannot be completely uniformly distributed

- Quasi-Monte Carlo points means
  - low discrepancy and
  - deterministic points

$\Rightarrow$ Discrete density approximation of uniform distribution $\mathcal{U}$

Halton Sequence and Hammersley Points

- Radical inverse (van der Corput sequence) in base b

$$i = \sum_{j=0}^{\infty} a_j(i) b^j \mapsto \Phi_b(i) := \sum_{j=0}^{\infty} a_j(i) b^{-j-1}$$

Halton Sequence and Hammersley Points

- Radical inverse (van der Corput sequence) in base b

$$i = \sum_{j=0}^{\infty} a_j(i) b^j \mapsto \Phi_b(i) := \sum_{j=0}^{\infty} a_j(i) b^{-j-1}$$

Note: The radical inverses are not completely uniform distributed !!!

Halton Sequence and Hammersley Points

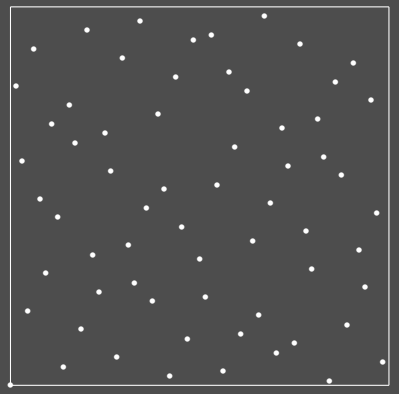
- Radical inverse (van der Corput sequence) in base b

$$i = \sum_{j=0}^{\infty} a_j(i) b^j \mapsto \Phi_b(i) := \sum_{j=0}^{\infty} a_j(i) b^{-j-1}$$

Note: The radical inverses are not completely uniform distributed !!!

- Halton sequence $x_i := \left(\Phi_{b_1}(i), \ldots, \Phi_{b_s}(i)\right)$ where b_i is the i-th prime number

$$D^*(P_N^{\text{Halton}}) < \frac{s}{N} + \frac{1}{N} \prod_{j=1}^{s} \left(\frac{b_j - 1}{2 \log b_j} \log N + \frac{b_j + 1}{2} \right)$$



Halton Sequence and Hammersley Points

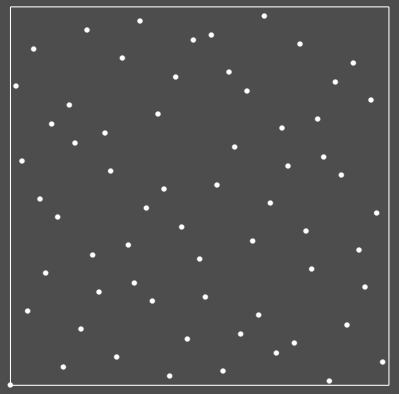
- Radical inverse (van der Corput sequence) in base b

$$i = \sum_{j=0}^{\infty} a_j(i) b^j \mapsto \Phi_b(i) := \sum_{j=0}^{\infty} a_j(i) b^{-j-1}$$

Note: The radical inverses are not completely uniform distributed !!!

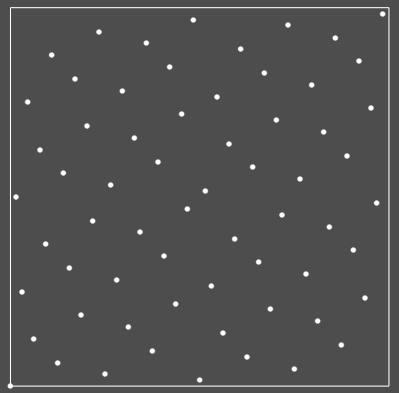
- Halton sequence $x_i := \left(\Phi_{b_1}(i), \ldots, \Phi_{b_s}(i)\right)$ where b_i is the i-th prime number

$$D^*(P_N^{\text{Halton}}) < \frac{s}{N} + \frac{1}{N} \prod_{j=1}^{s} \left(\frac{b_j - 1}{2 \log b_j} \log N + \frac{b_j + 1}{2} \right)$$



- Hammersley point set $x_i := \left(\frac{i}{N}, \Phi_{b_1}(i), \ldots, \Phi_{b_{s-1}}(i)\right)$

$$D^*(P_N^{\text{Hammersley}}) < \frac{s}{N} + \frac{1}{N} \prod_{j=1}^{s-1} \left(\frac{b_j - 1}{2 \log b_j} \log N + \frac{b_j + 1}{2} \right)$$



Algorithm: Radical Inversion

```
double RadicalInverse(const int Base, int i)
{
  double Digit, Radical, Inverse;

  Digit = Radical = 1.0 / (double) Base;
  Inverse = 0.0;

  while(i)
  {
    Inverse += Digit * (double) (i % Base);
    Digit *= Radical;
    i /= Base;
  }

  return Inverse;
}
```

Algorithm: Incremental Radical Inversion

```
double NextRadicalInverse(const double Radical, double Inverse)
// Radical = 1.0 / Base
{
   const double AlmostOne = 1.0 - 1e-10;
   double NextInverse, Digit1, Digit2;

   NextInverse = Inverse + Radical;

   if(NextInverse < AlmostOne)
      return NextInverse;
   else
   {
      Digit1 = Radical;
      Digit2 = Radical * Radical;

      while(Inverse + Digit2 >= AlmostOne)
      {
         Digit1 = Digit2;
         Digit2 *= Radical;
      }

      return Inverse + (Digit1 - 1.0) + Digit2;
   }
}
```

Other Discrepancies

- Isotropic discrepancy $J(P_N)$
  - $\mathcal{A}$ is family of all convex subsets of I^s

Other Discrepancies

- Isotropic discrepancy $J(P_N)$
  - $\mathcal{A}$ is family of all convex subsets of I^s
  - by

$$D^*(P_N) \leq D(P_N) \leq 2^s D^*(P_N)$$
$$D(P_N) \leq J(P_N) \leq 4sD(P_N)^{1/s}$$

    * upper bound

$$J(P_N) \leq 4sD(P_N)^{1/s} \leq 4s(2^s D^*(P_N))^{1/s} = 8sD^*(P_N)^{1/s}$$

    * lower bound

$$J(P_N) \geq D(P_N) \geq D^*(P_N)$$

Other Discrepancies

- Isotropic discrepancy $J(P_N)$
  - $\mathcal{A}$ is family of all convex subsets of I^s
  - by

$$D^*(P_N) \leq D(P_N) \leq 2^s D^*(P_N)$$
$$D(P_N) \leq J(P_N) \leq 4sD(P_N)^{1/s}$$

    - upper bound

$$J(P_N) \leq 4sD(P_N)^{1/s} \leq 4s(2^s D^*(P_N))^{1/s} = 8sD^*(P_N)^{1/s}$$

    - lower bound

$$J(P_N) \geq D(P_N) \geq D^*(P_N)$$

- Triangle discrepancy
- Edge discrepancy

Computing Discrepancies

- L_2-norm based discrepancy

$$D_2^*(P_N) := \sqrt{\int_{I^s} \left(\lambda_s(A(x)) - \frac{1}{N} \sum_{i=0}^{N-1} \chi_{A(x)}(x_i) \right)^2 dx}$$

  where $A(x) = \prod_{j=1}^{s} [0, x^{(j)})$

- Can be efficiently computed in contrast to L_∞-norm based discrepancies

Computing Discrepancies

- L_2-norm based discrepancy

$$D_2^*(P_N) := \sqrt{\int_{I^s} \left(\lambda_s(A(x)) - \frac{1}{N} \sum_{i=0}^{N-1} \chi_{A(x)}(x_i) \right)^2 dx}$$

  where $A(x) = \prod_{j=1}^{s} [0, x^{(j)})$

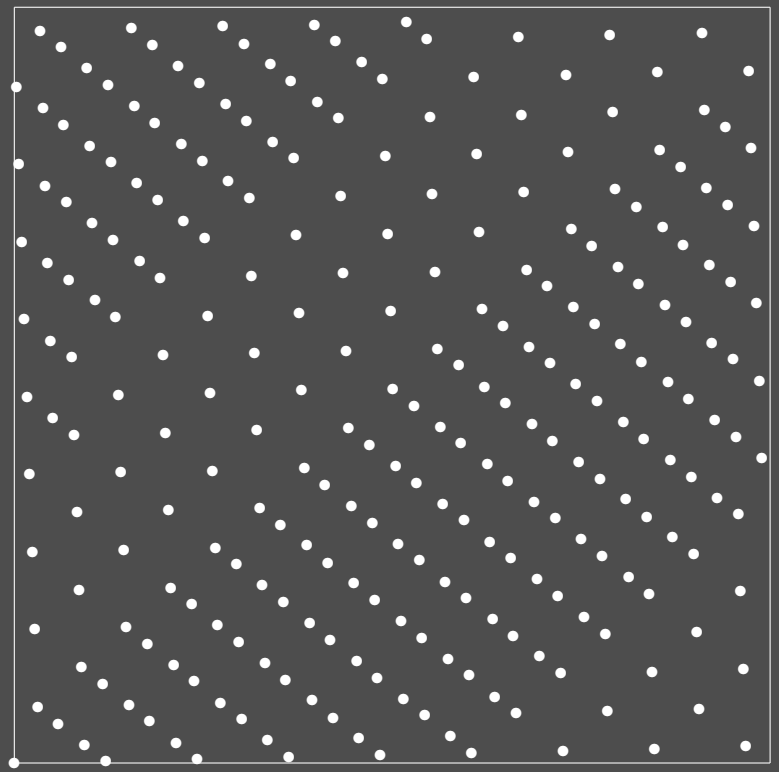
- Can be efficiently computed in contrast to L_∞-norm based discrepancies
- Numerical example: Triangular discrepancy

$$D(P_N, \mathcal{T}) \leq J(P_N) \leq 16\sqrt{D^*(P_N)}$$

N	10000 random triangles	100000 random triangles	theoretical bound
4	0.539712	0.591708	16.971
16	0.18326	0.230355	9.381
64	0.0660696	0.0777368	5.099
256	0.032454	0.0364673	2.739
1024	0.0118695	0.0178952	1.458
4096	0.00521621	0.00715305	0.771

Correlation Problems of Projections

- Dimensions 7 and 8 of the Halton sequence



Scrambling Permutations by Faure

- Scrambled radical inverse

$$i = \sum_{j=0}^{\infty} a_j(i) b^j \mapsto \sum_{j=0}^{\infty} \sigma_b(a_j(i)) b^{-j-1},$$

  using permutations σ_b by Faure

$$\begin{aligned}
\sigma_2 &= (0,1) \\
\sigma_3 &= (0,1,2) \\
\sigma_4 &= (0,2,1,3) \\
\sigma_5 &= (0,3,2,1,4) \\
\sigma_6 &= (0,2,4,1,3,5) \\
\sigma_7 &= (0,2,5,3,1,4,6) \\
\sigma_8 &= (0,4,2,6,1,5,3,7) \\
&\vdots
\end{aligned}$$

- Construction rule
  - b is even: Take $2\sigma_{\frac{b}{2}}$ and append $2\sigma_{\frac{b}{2}} + 1$
  - b is odd: Take σ_{b-1}, increment each value $\geq \frac{b-1}{2}$ and insert $\frac{b-1}{2}$ in the middle

Scrambled Halton Sequence and Hammersley Points

- Scrambled Halton sequence

$$x_i := \left(\Phi_{b_1}(i, \sigma_{b_1}), \ldots, \Phi_{b_s}(i, \sigma_{b_s})\right)$$

- Scrambled Hammersley point set

$$x_i := \left(\frac{i}{N}, \Phi_{b_1}(i, \sigma_{b_1}), \ldots, \Phi_{b_{s-1}}(i, \sigma_{b_{s-1}})\right)$$

Scrambled Halton Sequence and Hammersley Points

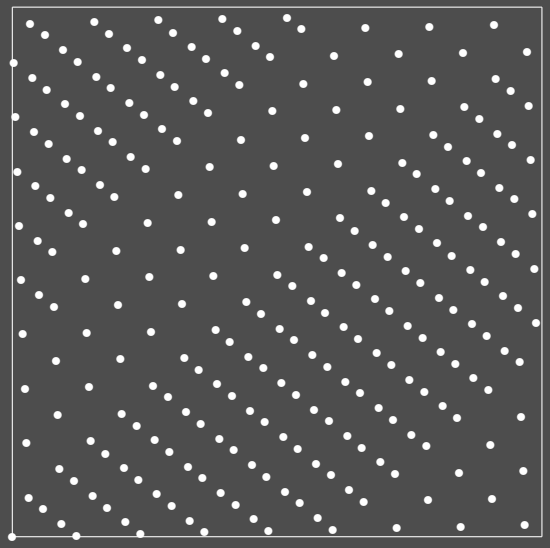
- Scrambled Halton sequence

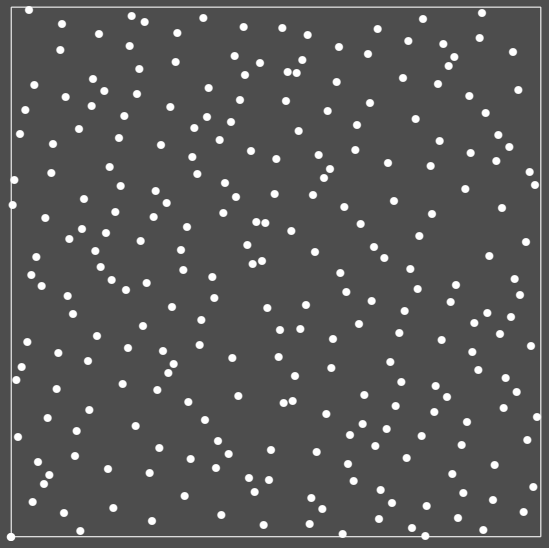
$$x_i := \left(\Phi_{b_1}(i, \sigma_{b_1}), \ldots, \Phi_{b_s}(i, \sigma_{b_s})\right)$$

- Scrambled Hammersley point set

$$x_i := \left(\frac{i}{N}, \Phi_{b_1}(i, \sigma_{b_1}), \ldots, \Phi_{b_{s-1}}(i, \sigma_{b_{s-1}})\right)$$

- Improvement by scrambling (scrambled Halton sequence dimensions 7 and 8)





(t, m, s)-*Nets in Base* b

- Elementary interval

$$E := \prod_{j=1}^{s} \left[\frac{a_j}{b^{l_j}}, \frac{a_j + 1}{b^{l_j}} \right) \subseteq I^s \text{ for integers } l_j \geq 0 \text{ and } 0 \leq a_j < b^{l_j}$$

- Consequently its volume is

$$\lambda_s(E) = \prod_{j=1}^{s} \frac{1}{b^{l_j}} = \frac{1}{b^{\sum_{j=1}^{s} l_j}}$$

(t, m, s)-*Nets in Base* b

- Elementary interval

$$E := \prod_{j=1}^{s} \left[\frac{a_j}{b^{l_j}}, \frac{a_j + 1}{b^{l_j}} \right) \subseteq I^s \text{ for integers } l_j \geq 0 \text{ and } 0 \leq a_j < b^{l_j}$$

- Consequently its volume is

$$\lambda_s(E) = \prod_{j=1}^{s} \frac{1}{b^{l_j}} = \frac{1}{b^{\sum_{j=1}^{s} l_j}}$$

- **Definition:** For two integers $0 \leq t \leq m$, a finite point set of b^m points in s dimensions is called a (t, m, s)-*net in base* b, if every elementary interval of volume $\lambda_s(E) = b^{t-m}$ contains exactly b^t points.

(t, m, s)-*Nets in Base* b

- Elementary interval

$$E := \prod_{j=1}^{s} \left[\frac{a_j}{b^{l_j}}, \frac{a_j + 1}{b^{l_j}} \right) \subseteq I^s \text{ for integers } l_j \geq 0 \text{ and } 0 \leq a_j < b^{l_j}$$

- Consequently its volume is

$$\lambda_s(E) = \prod_{j=1}^{s} \frac{1}{b^{l_j}} = \frac{1}{b^{\sum_{j=1}^{s} l_j}}$$

- **Definition:** For two integers $0 \leq t \leq m$, a finite point set of b^m points in s dimensions is called a (t, m, s)-*net in base* b, if every elementary interval of volume $\lambda_s(E) = b^{t-m}$ contains exactly b^t points.

- For (t, m, s)-nets in base b we have

$$D^*(P_N) \leq B(s, b) b^t \frac{\log^{s-1} N}{N} + \mathcal{O}\left(b^t \frac{\log^{s-2} N}{N} \right)$$

  - t is the quality parameter

(t, m, s)-*Nets in Base* b

- Elementary interval

$$E := \prod_{j=1}^{s} \left[\frac{a_j}{b^{l_j}}, \frac{a_j + 1}{b^{l_j}} \right) \subseteq I^s \text{ for integers } l_j \geq 0 \text{ and } 0 \leq a_j < b^{l_j}$$

- Consequently its volume is

$$\lambda_s(E) = \prod_{j=1}^{s} \frac{1}{b^{l_j}} = \frac{1}{b^{\sum_{j=1}^{s} l_j}}$$

- **Definition:** For two integers $0 \leq t \leq m$, a finite point set of b^m points in s dimensions is called a (t, m, s)-*net in base* b, if every elementary interval of volume $\lambda_s(E) = b^{t-m}$ contains exactly b^t points.

- For (t, m, s)-nets in base b we have

$$D^*(P_N) \leq B(s, b) b^t \frac{\log^{s-1} N}{N} + \mathcal{O}\left(b^t \frac{\log^{s-2} N}{N} \right)$$

  - t is the quality parameter

- **Note:** So far the concept applies to random and deterministic points

Structure of $(0, m, 2)$*-Nets in Base* $b = 2$

- (t, m, s)-net in base b:
  - Set P_N of $N = b^m$ s-dimensional points of low discrepancy
  - Every *elementary interval* of volume b^{t-m} contains exactly b^t points

Structure of $(0, m, 2)$*-Nets in Base* $b = 2$

- (t, m, s)-net in base b:
  - Set P_N of $N = b^m$ s-dimensional points of low discrepancy
  - Every *elementary interval* of volume b^{t-m} contains exactly b^t points

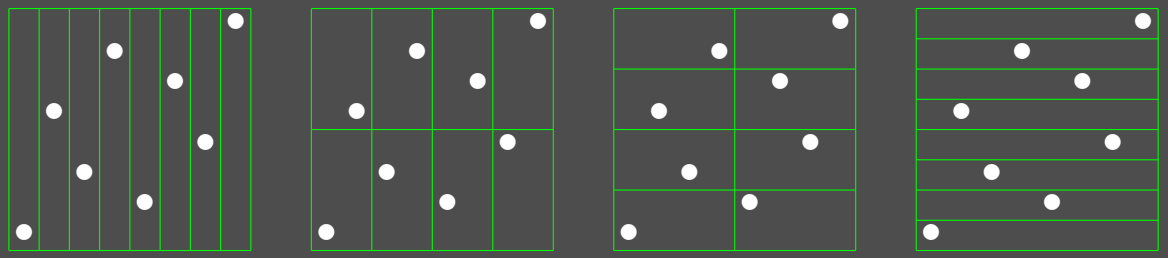
- $(0, m, 2)$-net in base $b = 2$
  - Set P_N of $N = 2^m$ 2-dimensional points of low discrepancy
  - Every *elementary interval* of volume $2^{-m} = \frac{1}{N}$ contains exactly 1 point

Structure of $(0, m, 2)$*-Nets in Base* $b = 2$

- (t, m, s)-net in base b:
  - Set P_N of $N = b^m$ s-dimensional points of low discrepancy
  - Every *elementary interval* of volume b^{t-m} contains exactly b^t points

- $(0, m, 2)$-net in base $b = 2$
  - Set P_N of $N = 2^m$ 2-dimensional points of low discrepancy
  - Every *elementary interval* of volume $2^{-m} = \frac{1}{N}$ contains exactly 1 point

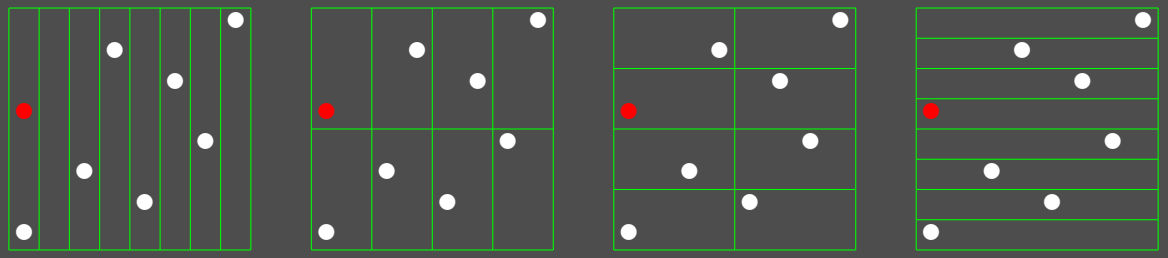
- Example: All elementary volumes of a $(0, 3, 2)$-net in base $b = 2$:



  - more general than stratification and Latin hypercube sampling

Example of a $(1, 3, 2)$*-Net in Base* $b = 2$

- All elementary volumes of a $(0, 3, 2)$-net in base $b = 2$:

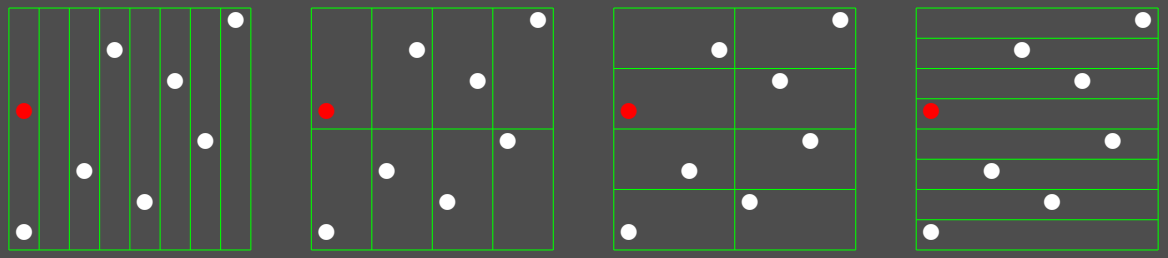


$\lambda_s(E) = b^{t-m} = 2^{0-3} = \frac{1}{8}$ with exactly $b^t = 2^0 = 1$ point

$\Rightarrow$ it cannot be a $(0, 3, 2)$-net !

Example of a $(1, 3, 2)$*-Net in Base* $b = 2$

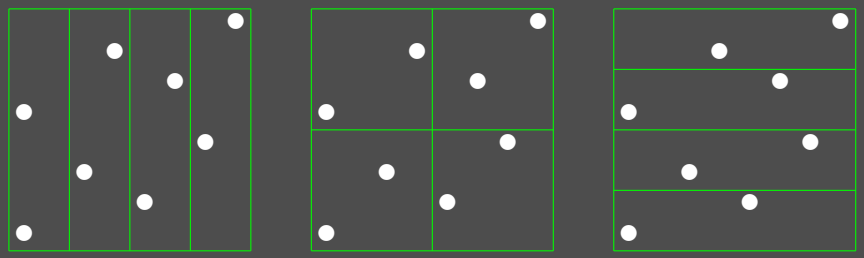
- All elementary volumes of a $(0, 3, 2)$-net in base $b = 2$:



$\lambda_s(E) = b^{t-m} = 2^{0-3} = \frac{1}{8}$ with exactly $b^t = 2^0 = 1$ point

$\Rightarrow$ it cannot be a $(0, 3, 2)$-net !

- All elementary volumes of a $(1, 3, 2)$-net in base $b = 2$:



$\lambda_s(E) = b^{t-m} = 2^{1-3} = \frac{1}{4}$ with exactly $b^t = 2^1 = 2$ points

$\Rightarrow$ it is only a $(1, 3, 2)$-net...

Structure of $(0, 2n, 2)$*-Nets in Base* $b = 2$

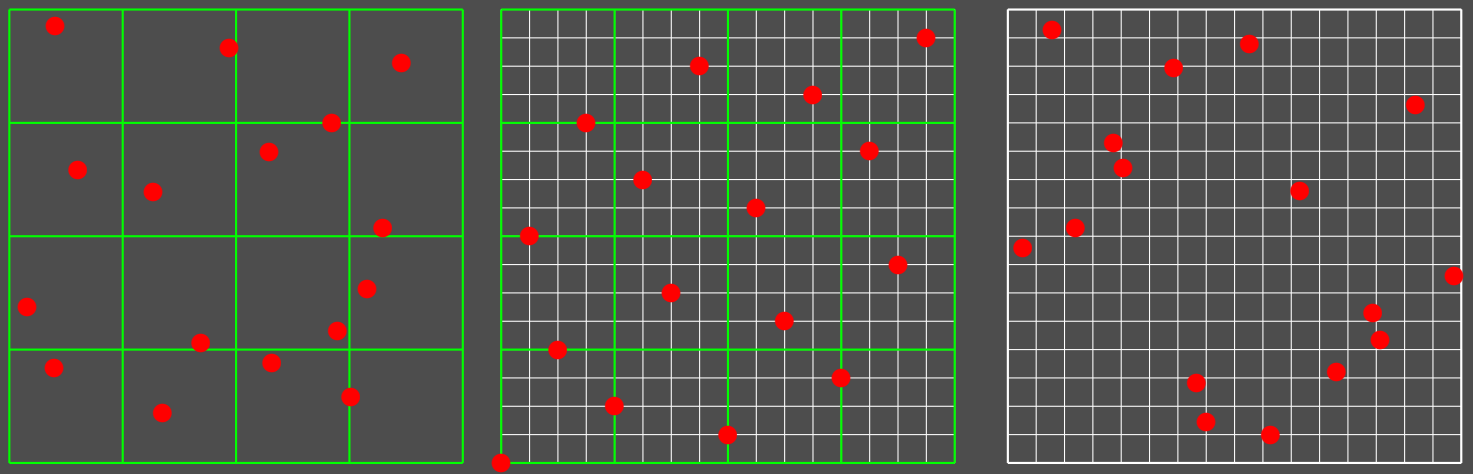
- (t, m, s)-net in base b:
  - Set P_N of $N = b^m$ s-dimensional points of low discrepancy
  - Every *elementary interval* of volume b^{t-m} contains exactly b^t points

Structure of $(0, 2n, 2)$*-Nets in Base* $b = 2$

- (t, m, s)-net in base b:
  - Set P_N of $N = b^m$ s-dimensional points of low discrepancy
  - Every *elementary interval* of volume b^{t-m} contains exactly b^t points
- $(0, 2n, 2)$-net in base $b = 2$
  - Set P_N of $N = (2^n)^2$ 2-dimensional points of low discrepancy
  - Every *elementary interval* of volume $2^{-2n} = \frac{1}{N}$ contains exactly 1 point

Structure of $(0, 2n, 2)$*-Nets in Base* $b = 2$

- (t, m, s)-net in base b:
  - Set P_N of $N = b^m$ s-dimensional points of low discrepancy
  - Every *elementary interval* of volume b^{t-m} contains exactly b^t points
- $(0, 2n, 2)$-net in base $b = 2$
  - Set P_N of $N = (2^n)^2$ 2-dimensional points of low discrepancy
  - Every *elementary interval* of volume $2^{-2n} = \frac{1}{N}$ contains exactly 1 point



jittered **and** LHS (N -rooks)

- (t, m, s)-nets: Much more general concept of stratification

(t, s)-*Sequences in Base* $b = 2$

- **Definition:** For $t \geq 0$, an infinite point sequence is called a (t, s)-*sequence* in base b, if for all $k \geq 0$ and $m \geq t$, the vectors $x_{kb^m+1}, \ldots, x_{(k+1)b^m} \in I^s$ form a (t, m, s)-net.

(t, s)-*Sequences in Base* $b = 2$

- **Definition:** For $t \geq 0$, an infinite point sequence is called a (t, s)-*sequence* in base b, if for all $k \geq 0$ and $m \geq t$, the vectors $x_{kb^m+1}, \ldots, x_{(k+1)b^m} \in I^s$ form a (t, m, s)-net.

- For (t, s)-sequence in base b we have

$$D^*(P_N) \leq C(s, b) b^t \frac{\log^s N}{N} + \mathcal{O}\left(b^t \frac{\log^{s-1} N}{N}\right)$$

(t, s)-*Sequences in Base* $b = 2$

- **Definition:** For $t \geq 0$, an infinite point sequence is called a (t, s)-*sequence* in base b, if for all $k \geq 0$ and $m \geq t$, the vectors $x_{kb^m+1}, \ldots, x_{(k+1)b^m} \in I^s$ form a (t, m, s)-net.

- For (t, s)-sequence in base b we have

$$D^*(P_N) \leq C(s, b)b^t \frac{\log^s N}{N} + \mathcal{O}\left(b^t \frac{\log^{s-1} N}{N}\right)$$

- Adding the component $\frac{i}{N} = \frac{i}{b^m}$ to a (t, s)-sequence yields a $(t, m, s + 1)$-net
- $(0, s)$-sequences can only exist for $b \geq s$

(t, s)-*Sequences in Base* $b = 2$

- **Definition:** For $t \geq 0$, an infinite point sequence is called a (t, s)-*sequence* in base b, if for all $k \geq 0$ and $m \geq t$, the vectors $x_{kb^m+1}, \ldots, x_{(k+1)b^m} \in I^s$ form a (t, m, s)-net.

- For (t, s)-sequence in base b we have

$$D^*(P_N) \leq C(s, b)b^t \frac{\log^s N}{N} + \mathcal{O}\left(b^t \frac{\log^{s-1} N}{N}\right)$$

- Adding the component $\frac{i}{N} = \frac{i}{b^m}$ to a (t, s)-sequence yields a $(t, m, s + 1)$-net
- $(0, s)$-sequences can only exist for $b \geq s$

- Examples
  - Van der Corput sequences are $(0, 1)$-sequences in base b
  - adding the component $\frac{i}{N}$ with $N = b^m$ yields a $(0, m, 2)$-net
    - e.g. Hammersley point set for $s = 2$ and $N = 2^m$ points
    - many applications in finance and particle transport problems

Digital (t, m, s)-Nets and (t, s)-Sequences

- Fixed-point numbers with M digits in base b

$$[0, 1)_{b,M} := \left\{ kb^{-M} \mid k = 0, \ldots, b^M - 1 \right\} \subset [0, 1)$$

Digital (t, m, s)-Nets and (t, s)-Sequences

- Fixed-point numbers with M digits in base b

$$[0,1)_{b,M} := \left\{ kb^{-M} \mid k = 0, \ldots, b^M - 1 \right\} \subset [0,1)$$

- Components $A_i^{(j)}$ of a point set $A = \{A_0, \ldots, A_{N-1}\}$

$$A_i^{(j)} = \sum_{k=1}^{M} a_{i,k}^{(j)} \cdot b^{-k}$$

Digital (t, m, s)-Nets and (t, s)-Sequences

- Fixed-point numbers with M digits in base b

$$[0,1)_{b,M} := \left\{ kb^{-M} \mid k = 0, \ldots, b^M - 1 \right\} \subset [0,1)$$

- Components $A_i^{(j)}$ of a point set $A = \{A_0, \ldots, A_{N-1}\}$

$$A_i^{(j)} \quad = \quad \sum_{k=1}^{M} a_{i,k}^{(j)} \cdot b^{-k} =_b 0.a_{i,1}^{(j)} a_{i,2}^{(j)} \ldots a_{i,M}^{(j)} \in [0,1)_{b,M}$$

Digital (t, m, s)*-Nets and* (t, s)*-Sequences*

- Fixed-point numbers with M digits in base b

$$[0,1)_{b,M} := \left\{ kb^{-M} \mid k = 0, \ldots, b^M - 1 \right\} \subset [0,1)$$

- Components $A_i^{(j)}$ of a point set $A = \{A_0, \ldots, A_{N-1}\}$

$$A_i^{(j)} = \sum_{k=1}^{M} a_{i,k}^{(j)} \cdot b^{-k} =_b 0.a_{i,1}^{(j)} a_{i,2}^{(j)} \ldots a_{i,M}^{(j)} \in [0,1)_{b,M} \text{ where}$$

$$a_{i,k}^{(j)} := \eta_k^{(j)} \left(\sum_{l=0}^{M-1} c_{k,l}^{(j)} \cdot \psi_l\left(d_{i,l}\right) \right)$$

for $1 \leq j \leq s$ and

$$i =: \sum_{l=0}^{M-1} d_{i,l} \cdot b^l \qquad d_{i,l} \in \mathbb{Z}_b := \{0, \ldots, b-1\}$$

Digital (t, m, s)*-Nets and* (t, s)*-Sequences*

- Fixed-point numbers with M digits in base b

$$[0,1)_{b,M} := \left\{ kb^{-M} \mid k = 0, \ldots, b^M - 1 \right\} \subset [0,1)$$

- Components $A_i^{(j)}$ of a point set $A = \{A_0, \ldots, A_{N-1}\}$

$$A_i^{(j)} = \sum_{k=1}^{M} a_{i,k}^{(j)} \cdot b^{-k} =_b 0.a_{i,1}^{(j)} a_{i,2}^{(j)} \ldots a_{i,M}^{(j)} \in [0,1)_{b,M} \text{ where}$$

$$a_{i,k}^{(j)} := \eta_k^{(j)} \left(\sum_{l=0}^{M-1} c_{k,l}^{(j)} \cdot \psi_l\left(d_{i,l}\right) \right)$$

for $1 \leq j \leq s$ and

$$i =: \sum_{l=0}^{M-1} d_{i,l} \cdot b^l \qquad d_{i,l} \in \mathbb{Z}_b := \{0, \ldots, b-1\}$$

- Arithmetic in commutative ring $(R, +, \cdot)$ with $|R| = b$ elements
- Bijections $\eta_k^{(j)} : R \to \mathbb{Z}_b$ and $\psi_l : \mathbb{Z}_b \to R$

Digital (t, m, s)*-Nets and* (t, s)*-Sequences*

- Fixed-point numbers with M digits in base b

$$[0,1)_{b,M} := \left\{ kb^{-M} \mid k = 0, \ldots, b^M - 1 \right\} \subset [0,1)$$

- Components $A_i^{(j)}$ of a point set $A = \{A_0, \ldots, A_{N-1}\}$

$$A_i^{(j)} = \sum_{k=1}^{M} a_{i,k}^{(j)} \cdot b^{-k} =_b 0.a_{i,1}^{(j)} a_{i,2}^{(j)} \ldots a_{i,M}^{(j)} \in [0,1)_{b,M} \text{ where}$$

$$a_{i,k}^{(j)} := \eta_k^{(j)} \left(\sum_{l=0}^{M-1} c_{k,l}^{(j)} \cdot \psi_l\left(d_{i,l}\right) \right)$$

for $1 \leq j \leq s$ and

$$i =: \sum_{l=0}^{M-1} d_{i,l} \cdot b^l \qquad d_{i,l} \in \mathbb{Z}_b := \{0, \ldots, b-1\}$$

- Arithmetic in commutative ring $(R, +, \cdot)$ with $|R| = b$ elements
- Bijections $\eta_k^{(j)} : R \to \mathbb{Z}_b$ and $\psi_l : \mathbb{Z}_b \to R$

$\Rightarrow$ If now A is a (t, m, s)-net, it is called a ***digital*** (t, m, s)***-net***

$\Rightarrow$ If now A is a (t, s)-sequence, it is called a ***digital*** (t, s)***-sequence***

Deterministic Constructions of Digital Point Sets

- Generator matrix

$$C^{(j)} := \left(c_{k,l}^{(j)} \right)_{k=1,l=0}^{M,M-1} \in R^{M \times M}$$

- van der Corput, Sobol', Faure, Niederreiter, and Niederreiter-Xing
  - increased quality by decreased parameter t
  - difficult computation of the generator matrices

Deterministic Constructions of Digital Point Sets

- Generator matrix

$$C^{(j)} := \left(c_{k,l}^{(j)} \right)_{k=1,l=0}^{M,M-1} \in R^{M \times M}$$

- van der Corput, Sobol', Faure, Niederreiter, and Niederreiter-Xing
  - increased quality by decreased parameter t
  - difficult computation of the generator matrices
- Fast evaluation by
  - Gray codes
  - vectorization
  - buffering of invariants
  - rings implemented as lookup tables
- Very often

$$\vec{a}_i^{(j)} = C^{(j)} \vec{d}_i$$

Vectorization Example for Base $b = 2$

- Ring $R = (\{0, 1\}, +, \cdot) = \mathbb{Z}_2$ by bit vector operations
- One component at M bits precision

$$x_i = \left(\frac{1}{2} \cdots \frac{1}{2^M}\right) \cdot C \cdot \begin{pmatrix} d_0(i) \\ \vdots \\ d_{M-1}(i) \end{pmatrix} \qquad \text{where } i = \sum_{k=0}^{m-1} d_k(i) 2^k$$

Vectorization Example for Base $b = 2$

- Ring $R = (\{0, 1\}, +, \cdot) = \mathbb{Z}_2$ by bit vector operations
- One component at M bits precision

$$x_i = \left(\frac{1}{2} \cdots \frac{1}{2^M}\right) \cdot C \cdot \begin{pmatrix} d_0(i) \\ \vdots \\ d_{M-1}(i) \end{pmatrix} \qquad \text{where } i = \sum_{k=0}^{m-1} d_k(i) 2^k$$

- Basic vectorized algorithm

```
double x(int i)
{
  for(int y = 0, int k = 0; i; i /= 2, k++)
    if(i & 1)
      y ^= C[k];

  return (double) y / (double) (1 << (M + 1));
}
```

Examples Matrices for Base $b = 2$

- $(0, m, 1)$-nets at $N = 2^m$

$$C_1 = \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 0 \\ & & \ddots & & \\ 0 & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

implements $x = \frac{i}{N}$

Examples Matrices for Base $b = 2$

- $(0, 1)$-sequences: Bit reversal, or $\phi_2(i)$ by van der Corput

$$C_2 = I$$

Examples Matrices for Base $b = 2$

- $(0, 1)$-sequences: Bit reversal, or $\phi_2(i)$ by van der Corput

$$C_2 = I$$

- Algorithm

```
double RadicalInverse(unsigned int bits) // M=32 bits version
{
  bits = ( bits               <<  16) | ( bits               >>  16);
  bits = ((bits & 0x00ff00ff) <<   8) | ((bits & 0xff00ff00) >>   8);
  bits = ((bits & 0x0f0f0f0f) <<   4) | ((bits & 0xf0f0f0f0) >>   4);
  bits = ((bits & 0x33333333) <<   2) | ((bits & 0xcccccccc) >>   2);
  bits = ((bits & 0x55555555) <<   1) | ((bits & 0xaaaaaaaa) >>   1);

  return (double) bits / (double) 0x100000000L;
}
```

Examples Matrices for Base $b = 2$

- $(0, 1)$-sequences: Sobol' scrambled radical inverse

$$C_3 = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 1 & \cdots & 0 & 0 \\ 1 & 1 & 1 & \ddots & 0 & 0 \end{pmatrix} = \binom{k-1}{l-1} \bmod 2$$

Examples Matrices for Base $b = 2$

- $(0, 1)$-sequences: Sobol' scrambled radical inverse

$$C_3 = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 1 & \cdots & 0 & 0 \\ 1 & 1 & 1 & \ddots & 0 & 0 \end{pmatrix} = \binom{k-1}{l-1} \bmod 2$$

- Algorithm

```
double SobolRadicalInverse(int i)
{
  int r, v;

  v = 1 << M;

  for(r = 0; i; i >>= 1)
  {
    if(i & 1)
      r ^= v;

    v ^= v >> 1;
  }

  return (double) r / (double) (1 << (M + 1));
}
```

Examples Matrices for Base $b = 2$

- $(0, 1)$-sequences: Larcher-Pillichshammer scrambled radical inverse

$$C_4 = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & \cdots & 0 & 0 \\ & & \ddots & & \\ 1 & 1 & \cdots & 1 & 0 \\ 1 & 1 & \cdots & 1 & 1 \end{pmatrix}$$

Examples Matrices for Base $b = 2$

- $(0, 1)$-sequences: Larcher-Pillichshammer scrambled radical inverse

$$C_4 = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & \cdots & 0 & 0 \\ & & \ddots & & \\ 1 & 1 & \cdots & 1 & 0 \\ 1 & 1 & \cdots & 1 & 1 \end{pmatrix}$$

- Algorithm

```
double LarcherPillichshammerRadicalInverse(int i)
{
  int r, v;

  v = 1 << M;

  for(r = 0; i; i >>= 1)
  {
    if(i & 1)
      r ^= v;

    v |= v >> 1;
  }

  return (double) r / (double) (1 << (M + 1));
}
```

Digital $(0, m, s)$***-Nets and*** $(0, s)$***-Sequences in Base*** $b = 2$

- $(0, m, 2)$-nets at $N = 2^m$
  - Hammersley points (worst constant)

    (C_1, C_2)
  - Larcher-Pillichshammer points (best constant)

    (C_1, C_4)

Digital $(0, m, s)$*-Nets and* $(0, s)$*-Sequences in Base* $b = 2$

- $(0, m, 2)$-nets at $N = 2^m$
  - Hammersley points (worst constant)

    (C_1, C_2)
  - Larcher-Pillichshammer points (best constant)

    (C_1, C_4)
- $(0, 2)$-sequence: Sobol' LP_0-sequence

  (C_2, C_3)

Digital $(0, m, s)$*-Nets and* $(0, s)$*-Sequences in Base* $b = 2$

- $(0, m, 2)$-nets at $N = 2^m$
  - Hammersley points (worst constant)

    (C_1, C_2)
  - Larcher-Pillichshammer points (best constant)

    (C_1, C_4)
- $(0, 2)$-sequence: Sobol' LP_0-sequence

  (C_2, C_3)
- $(0, m, 3)$-net at $N = 2^m$: Sobol' LP_0-net

  (C_1, C_2, C_3)
- Very useful in particle transport, especially computer graphics

Software

- **Numerical Recipes**
  - Sobol’ sequence
- **http://www.mcqmc.org/Software.html**
  - Sobol’ sequence
  - Faure sequence
  - Niederreiter sequence
- **http://www.multires.caltech.edu/software/libseq/index.html**
  - general package
  - several sequences (Halton, Niederreiter, ...)
- **http://www.dismat.oeaw.ac.at/pirs/niedxing.html**
  - generator matrices for the Niederreiter-Xing sequence

Good Lattice Points: Rank-1 Lattices

- **Definition:** A discrete subset

$$L := P_N + \mathbb{Z}^s \subset \mathbb{R}^s$$

that is closed under addition and subtraction is called a **lattice**.

Good Lattice Points: Rank-1 Lattices

- **Definition:** A discrete subset

  $$L := P_N + \mathbb{Z}^s \subset \mathbb{R}^s$$

  that is closed under addition and subtraction is called a **lattice**.

- Rank-1 lattice

  $$\vec{x}_i := \frac{i}{N}\vec{g}$$

  by suitable generating vector $\vec{g} \in \mathbb{N}^s$

- Low discrepancy constructions
  - Fibonacci lattices for $s = 2$
  - lattices with generator vector of Korobov-form $\vec{g} = (1, l, l^2, \ldots)$

Good Lattice Points: Rank-1 Lattices

- **Definition:** A discrete subset

  $$L := P_N + \mathbb{Z}^s \subset \mathbb{R}^s$$

  that is closed under addition and subtraction is called a **lattice**.

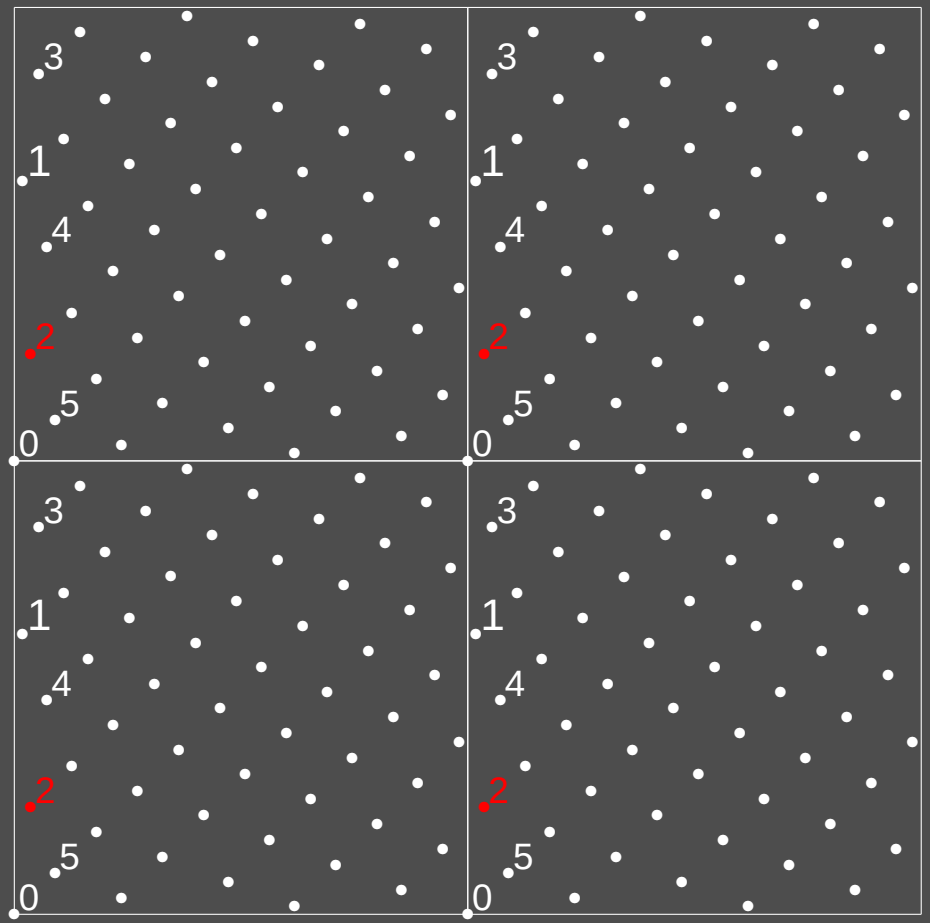
- Rank-1 lattice

  $$\vec{x}_i := \frac{i}{N}\vec{g}$$

  by suitable generating vector $\vec{g} \in \mathbb{N}^s$

- Low discrepancy constructions
  - Fibonacci lattices for $s = 2$
  - lattices with generator vector of Korobov-form $\vec{g} = (1, l, l^2, \ldots)$
- No explicit construction - only tables

- One-periodic pattern $L \cap [0,1)^s$



- Low discrepancy
- Much better discrepancy than regular grids

Example: Fibonacci Rank-1 *Lattice*

- Fibonacci numbers: $F_1 = F_2 = 1$, $F_k = F_{k-1} + F_{k-2}$ for $k > 2$
- **Fibonacci lattice** by generator vector $\vec{g} = (1, F_{k-1})$ at $N = F_k$ points

$$\vec{x}_i := \frac{i}{F_k}(1, F_{k-1})$$

  - Low discrepancy

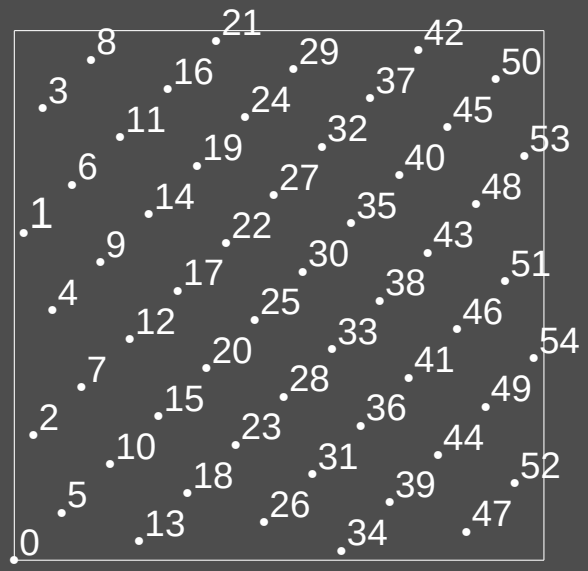
Example: Fibonacci Rank-1 *Lattice*

- Fibonacci numbers: $F_1 = F_2 = 1$, $F_k = F_{k-1} + F_{k-2}$ for $k > 2$
- **Fibonacci lattice** by generator vector $\vec{g} = (1, F_{k-1})$ at $N = F_k$ points

$$\vec{x}_i := \frac{i}{F_k}(1, F_{k-1})$$

  - Low discrepancy

- Example: $N = F_{10} = 55$, $\vec{x}_i := \frac{i}{55}(1, 34)$



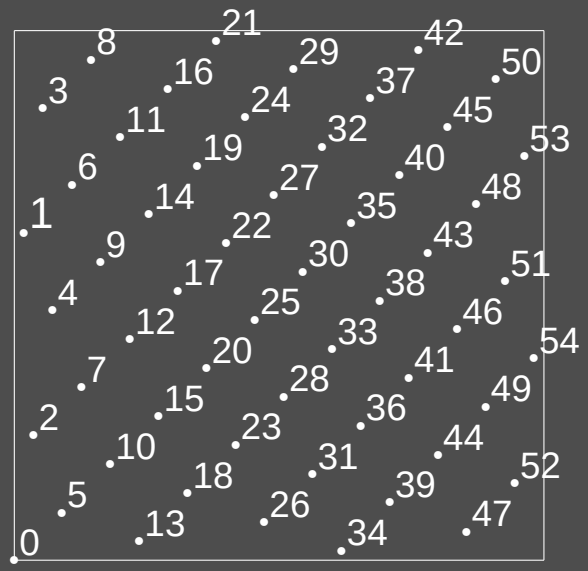
Example: Fibonacci Rank-1 *Lattice*

- Fibonacci numbers: $F_1 = F_2 = 1$, $F_k = F_{k-1} + F_{k-2}$ for $k > 2$
- **Fibonacci lattice** by generator vector $\vec{g} = (1, F_{k-1})$ at $N = F_k$ points

$$\vec{x}_i := \frac{i}{F_k}(1, F_{k-1})$$

  - Low discrepancy

- Example: $N = F_{10} = 55$, $\vec{x}_i := \frac{i}{55}(1, 34)$



- **Note:** N grows exponentially for Fibonacci lattices

Lattice Sequences

- Rank-1 lattice

$$\vec{x}_i = \frac{i}{N} \cdot \vec{g}$$

- Hide N by choosing $N = b^m$ and

$$\vec{x}_i = \phi_b(i) \cdot \vec{g}$$

Lattice Sequences

- Rank-1 lattice

$$\vec{x}_i = \frac{i}{N} \cdot \vec{g}$$

- Hide N by choosing $N = b^m$ and

$$\vec{x}_i = \phi_b(i) \cdot \vec{g}$$

- Similar to (t, s)-sequences: $\vec{x}_{kb^m}, \ldots, \vec{x}_{(k+1)b^m-1}$ form a shifted lattice

Lattice Sequences

- Rank-1 lattice

$$\vec{x}_i = \frac{i}{N} \cdot \vec{g}$$

- Hide N by choosing $N = b^m$ and

$$\vec{x}_i = \phi_b(i) \cdot \vec{g}$$

- Similar to (t, s)-sequences: $\vec{x}_{kb^m}, \ldots, \vec{x}_{(k+1)b^m-1}$ form a shifted lattice

- Shift Δ in the $k+1$st run for $N = b^m$

$$\begin{aligned} \phi_b(i + kb^m) \cdot \vec{g} &= \phi_b(i) + \phi_b(kb^m) \\ &= \phi_b(i) \cdot \vec{g} + \underbrace{\phi_b(k) b^{-m-1} \vec{g}}_{=:\Delta} \end{aligned}$$

Summary

- Quasi-Monte Carlo Points
  - low discrepancy
  - deterministic
  - intrinsic stratification (Latin hypercube, symmetrized, regularized, antithetic)
    - no extra programming

Summary

- Quasi-Monte Carlo Points
  - low discrepancy
  - deterministic
  - intrinsic stratification (Latin hypercube, symmetrized, regularized, antithetic)
    - no extra programming
  - no completely uniform distribution due to correlation

Beyond Monte Carlo

- Day 1: Monte Carlo
- Day 2: Quasi-Monte Carlo points
- **Day 3: Quasi-Monte Carlo integration**
- Day 4: Monte Carlo extensions of quasi-Monte Carlo
- Day 5: Applications to computer graphics

Day 3: Quasi-Monte Carlo Integration

- Koksma-Hlawka inequality and variation in the sense of Hardy and Krause
- Discrete density approximation
- Error control
- Transferring Monte Carlo techniques to quasi-Monte Carlo
- Integrands of infinite variation
- Discrete Fourier transform on good lattice points

Quasi-Monte Carlo Integration

- Numerical integration by **Quasi-Monte Carlo points**

$$\left| \int_{I^s} f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} f(x_i) \right| < V(f) D^*(P_N)$$

with variation $V(f)$ in the sense of Hardy and Krause and star-discrepancy

$$D^*(P_N) := \sup_{A=\prod_{j=1}^{s}[0,a_j)\subseteq I^s} \left| \underbrace{\int_{I^s} \chi_A(x)dx}_{=\lambda_s(A)} - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i) \right|$$

Quasi-Monte Carlo Integration

- Numerical integration by **Quasi-Monte Carlo points**

$$\left| \int_{I^s} f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} f(x_i) \right| < V(f)D^*(P_N)$$

with variation $V(f)$ in the sense of Hardy and Krause and star-discrepancy

$$D^*(P_N) := \sup_{A=\prod_{j=1}^{s}[0,a_j)\subseteq I^s} \left| \underbrace{\int_{I^s} \chi_A(x)dx}_{=\lambda_s(A)} - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i) \right|$$

- Deterministic error bound by the Koksma-Hlawka inequality

- Independent of dimension by using quasi-Monte Carlo points
  - roughly quadratically faster as compared to random sampling

Theorem: The Koksma-Hlawka Inequality

$$\left| \int_I f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} f(x_i) \right| \leq V(f) D^*(P_N)$$

- Proof for $s = 1$: Decompose

  $$f(x) = f(1) - \int_x^1 f'(u)du = f(1) - \int_I \chi_{[0,u]}(x) f'(u)du$$

  and define

  $$V(f) := \int_I \left| \frac{\partial f(u)}{\partial u} \right| du$$

- **Note:**

  $$\chi_{[0,u]}(x) = \begin{cases} 1 & x \in [0,u) \\ 0 & \text{else} \end{cases} = \begin{cases} 1 & x < u \\ 0 & \text{else} \end{cases}$$

Theorem: The Koksma-Hlawka Inequality

$$\left| \int_I f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} f(x_i) \right| \leq V(f) D^*(P_N)$$

- Proof for $s = 1$: Decompose

  $$f(x) = f(1) - \int_x^1 f'(u)du = f(1) - \int_I \chi_{[0,u]}(x) f'(u)du$$

  and define

  $$V(f) := \int_I \left| \frac{\partial f(u)}{\partial u} \right| du$$

- **Note:**

  $$\chi_{[0,u]}(x) = \begin{cases} 1 & x \in [0,u) \\ 0 & \text{else} \end{cases} = \begin{cases} 1 & x < u \\ 0 & \text{else} \end{cases} = \begin{cases} 1 & u > x \\ 0 & \text{else} \end{cases}$$

$$\left|\int_I f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right|$$

$$= \left|\int_I f(1) - \int_I \chi_{[0,u]}(x) f'(u)dudx - \frac{1}{N}\sum_{i=0}^{N-1}\left(f(1) - \int_I \chi_{[0,u]}(x_i) f'(u)du\right)\right|$$

$$
\begin{aligned}
&\left| \int_I f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} f(x_i) \right| \\
&= \left| \int_I f(1) - \int_I \chi_{[0,u]}(x) f'(u) du dx - \frac{1}{N} \sum_{i=0}^{N-1} \left(f(1) - \int_I \chi_{[0,u]}(x_i) f'(u) du \right) \right| \\
&= \left| f(1) - \int_I \int_I \chi_{[0,u]}(x) f'(u) du dx - f(1) + \frac{1}{N} \sum_{i=0}^{N-1} \int_I \chi_{[0,u]}(x_i) f'(u) du \right|
\end{aligned}
$$

$$
\begin{aligned}
&\left| \int_I f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i) \right| \\
&= \left| \int_I f(1) - \int_I \chi_{[0,u]}(x) f'(u)dudx - \frac{1}{N}\sum_{i=0}^{N-1} \left(f(1) - \int_I \chi_{[0,u]}(x_i) f'(u)du \right) \right| \\
&= \left| f(1) - \int_I \int_I \chi_{[0,u]}(x) f'(u)dudx - f(1) + \frac{1}{N}\sum_{i=0}^{N-1} \int_I \chi_{[0,u]}(x_i) f'(u)du \right| \\
&= \left| \frac{1}{N}\sum_{i=0}^{N-1} \int_I \chi_{[0,u]}(x_i) f'(u)du - \int_I \int_I \chi_{[0,u]}(x)dx f'(u)du \right|
\end{aligned}
$$

$$
\begin{aligned}
&\left| \int_I f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i) \right| \\
&= \left| \int_I f(1) - \int_I \chi_{[0,u]}(x) f'(u)dudx - \frac{1}{N}\sum_{i=0}^{N-1} \left(f(1) - \int_I \chi_{[0,u]}(x_i) f'(u)du \right) \right| \\
&= \left| f(1) - \int_I \int_I \chi_{[0,u]}(x) f'(u)dudx - f(1) + \frac{1}{N}\sum_{i=0}^{N-1} \int_I \chi_{[0,u]}(x_i) f'(u)du \right| \\
&= \left| \frac{1}{N}\sum_{i=0}^{N-1} \int_I \chi_{[0,u]}(x_i) f'(u)du - \int_I \int_I \chi_{[0,u]}(x)dx f'(u)du \right| \\
&= \left| \int_I f'(u) \left[\frac{1}{N}\sum_{i=0}^{N-1} \chi_{[0,u]}(x_i) - \int_I \chi_{[0,u]}(x)dx \right] du \right|
\end{aligned}
$$

$$
\begin{aligned}
&\left|\int_I f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| \\
&= \left|\int_I f(1) - \int_I \chi_{[0,u]}(x) f'(u)dudx - \frac{1}{N}\sum_{i=0}^{N-1}\left(f(1) - \int_I \chi_{[0,u]}(x_i) f'(u)du\right)\right| \\
&= \left|f(1) - \int_I\int_I \chi_{[0,u]}(x) f'(u)dudx - f(1) + \frac{1}{N}\sum_{i=0}^{N-1}\int_I \chi_{[0,u]}(x_i) f'(u)du\right| \\
&= \left|\frac{1}{N}\sum_{i=0}^{N-1}\int_I \chi_{[0,u]}(x_i) f'(u)du - \int_I\int_I \chi_{[0,u]}(x)dx f'(u)du\right| \\
&= \left|\int_I f'(u)\left[\frac{1}{N}\sum_{i=0}^{N-1}\chi_{[0,u]}(x_i) - \int_I \chi_{[0,u]}(x)dx\right]du\right| \\
&\leq \int_I \left|f'(u)\right|\left|\frac{1}{N}\sum_{i=0}^{N-1}\chi_{[0,u]}(x_i) - \int_I \chi_{[0,u]}(x)dx\right|du
\end{aligned}
$$

$$
\begin{aligned}
&\left|\int_I f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| \\
&= \left|\int_I f(1) - \int_I \chi_{[0,u]}(x) f'(u)dudx - \frac{1}{N}\sum_{i=0}^{N-1}\left(f(1) - \int_I \chi_{[0,u]}(x_i) f'(u)du\right)\right| \\
&= \left|f(1) - \int_I\int_I \chi_{[0,u]}(x) f'(u)dudx - f(1) + \frac{1}{N}\sum_{i=0}^{N-1}\int_I \chi_{[0,u]}(x_i) f'(u)du\right| \\
&= \left|\frac{1}{N}\sum_{i=0}^{N-1}\int_I \chi_{[0,u]}(x_i) f'(u)du - \int_I\int_I \chi_{[0,u]}(x)dx f'(u)du\right| \\
&= \left|\int_I f'(u)\left[\frac{1}{N}\sum_{i=0}^{N-1}\chi_{[0,u]}(x_i) - \int_I \chi_{[0,u]}(x)dx\right]du\right| \\
&\leq \int_I \left|f'(u)\right|\left|\frac{1}{N}\sum_{i=0}^{N-1}\chi_{[0,u]}(x_i) - \int_I \chi_{[0,u]}(x)dx\right|du \\
&\leq \int_I \left|f'(u)\right|du \cdot \sup_{u\in I}\left|\frac{1}{N}\sum_{i=0}^{N-1}\chi_{[0,u]}(x_i) - \int_I \chi_{[0,u]}(x)dx\right|
\end{aligned}
$$

$$
\begin{aligned}
&\left|\int_I f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| \\
&= \left|\int_I f(1) - \int_I \chi_{[0,u]}(x) f'(u)dudx - \frac{1}{N}\sum_{i=0}^{N-1}\left(f(1) - \int_I \chi_{[0,u]}(x_i) f'(u)du\right)\right| \\
&= \left|f(1) - \int_I\int_I \chi_{[0,u]}(x) f'(u)dudx - f(1) + \frac{1}{N}\sum_{i=0}^{N-1}\int_I \chi_{[0,u]}(x_i) f'(u)du\right| \\
&= \left|\frac{1}{N}\sum_{i=0}^{N-1}\int_I \chi_{[0,u]}(x_i) f'(u)du - \int_I\int_I \chi_{[0,u]}(x)dx f'(u)du\right| \\
&= \left|\int_I f'(u)\left[\frac{1}{N}\sum_{i=0}^{N-1}\chi_{[0,u]}(x_i) - \int_I \chi_{[0,u]}(x)dx\right]du\right| \\
&\leq \int_I \left|f'(u)\right|\left|\frac{1}{N}\sum_{i=0}^{N-1}\chi_{[0,u]}(x_i) - \int_I \chi_{[0,u]}(x)dx\right|du \\
&\leq \int_I \left|f'(u)\right|du \cdot \sup_{u\in I}\left|\frac{1}{N}\sum_{i=0}^{N-1}\chi_{[0,u]}(x_i) - \int_I \chi_{[0,u]}(x)dx\right| \\
&= V(f)D^*(P_N) \qquad \textit{q.e.d.}
\end{aligned}
$$

Variation in the Sense of Vitali

- Difference operator for intervals of the form $A = \prod_{i=1}^{s} [a_i, b_i) \subseteq I^s$

$$\Delta(f, A) := \sum_{j_1=0}^{1} \cdots \sum_{j_s=0}^{1} (-1)^{\sum_{k=1}^{s} j_k} f(j_1 a_1 + (1 - j_1) b_1, \ldots, j_s a_s + (1 - j_s) b_s)$$

Variation in the Sense of Vitali

- Difference operator for intervals of the form $A = \prod_{i=1}^{s} [a_i, b_i) \subseteq I^s$

$$\Delta(f, A) := \sum_{j_1=0}^{1} \cdots \sum_{j_s=0}^{1} (-1)^{\sum_{k=1}^{s} j_k} f(j_1 a_1 + (1 - j_1) b_1, \ldots, j_s a_s + (1 - j_s) b_s)$$

- Variation in the sense of Vitali

$$V^{(s)}(f) := \sup_{\mathcal{P}} \sum_{A \in \mathcal{P}} |\Delta(f, A)|$$

where $\mathcal{P}$ is the set of partitions of I^s into subintervals A as above

Variation in the Sense of Vitali

- Difference operator for intervals of the form $A = \prod_{i=1}^{s} [a_i, b_i) \subseteq I^s$

$$\Delta(f, A) := \sum_{j_1=0}^{1} \cdots \sum_{j_s=0}^{1} (-1)^{\sum_{k=1}^{s} j_k} f(j_1 a_1 + (1-j_1) b_1, \ldots, j_s a_s + (1-j_s) b_s)$$

- Variation in the sense of Vitali

$$V^{(s)}(f) := \sup_{\mathcal{P}} \sum_{A \in \mathcal{P}} |\Delta(f, A)|$$

  where $\mathcal{P}$ is the set of partitions of I^s into subintervals A as above

- If f has a continuous derivative

$$V^{(s)}(f) = \int_{I^s} \left| \frac{\partial^s f(u_1, \ldots, u_s)}{\partial u_1 \cdots \partial u_s} \right| du$$

Variation in the Sense of Vitali

- Difference operator for intervals of the form $A = \prod_{i=1}^{s} [a_i, b_i) \subseteq I^s$

$$\Delta(f, A) := \sum_{j_1=0}^{1} \cdots \sum_{j_s=0}^{1} (-1)^{\sum_{k=1}^{s} j_k} f(j_1 a_1 + (1-j_1) b_1, \ldots, j_s a_s + (1-j_s) b_s)$$

- Variation in the sense of Vitali

$$V^{(s)}(f) := \sup_{\mathcal{P}} \sum_{A \in \mathcal{P}} |\Delta(f, A)|$$

  where $\mathcal{P}$ is the set of partitions of I^s into subintervals A as above

- If f has a continuous derivative

$$V^{(s)}(f) = \int_{I^s} \left| \frac{\partial^s f(u_1, \ldots, u_s)}{\partial u_1 \cdots \partial u_s} \right| du$$

- Problem if f constant in only some of the variables $u_1, \ldots, u_s$

$$\Rightarrow \Delta(f, A) = 0 \qquad \Rightarrow V^{(s)}(f) = 0$$

Variation in the Sense of Hardy and Krause

- Restrict variation in the sense of Vitali

  $$V^{(k)}(f; i_1, \ldots, i_k)$$

  to the k-dimensional face $\{(u_1, \ldots, u_s) \in [0,1]^s | u_j = 1 \text{ for } j \neq i_1, \ldots, i_k\}$

Variation in the Sense of Hardy and Krause

- Restrict variation in the sense of Vitali

  $$V^{(k)}(f; i_1, \ldots, i_k)$$

  to the k-dimensional face $\{(u_1, \ldots, u_s) \in [0,1]^s | u_j = 1 \text{ for } j \neq i_1, \ldots, i_k\}$

- Variation in the sense of Hardy and Krause

  $$V(f) := \sum_{k=1}^{s} \sum_{1 \leq i_1 < \cdots < i_k \leq s} V^{(k)}(f; i_1, \ldots, i_k)$$

Variation in the Sense of Hardy and Krause

- Restrict variation in the sense of Vitali

  $$V^{(k)}(f; i_1, \ldots, i_k)$$

  to the k-dimensional face $\{(u_1, \ldots, u_s) \in [0,1]^s | u_j = 1 \text{ for } j \neq i_1, \ldots, i_k\}$

- Variation in the sense of Hardy and Krause

  $$V(f) := \sum_{k=1}^{s} \sum_{1 \leq i_1 < \cdots < i_k \leq s} V^{(k)}(f; i_1, \ldots, i_k)$$

- **Definition:**

  f is of bounded variation in the sense of Hardy and Krause, if $V(f)$ is finite.

Variation in the Sense of Hardy and Krause

- Restrict variation in the sense of Vitali

  $$V^{(k)}(f; i_1, \ldots, i_k)$$

  to the k-dimensional face $\{(u_1, \ldots, u_s) \in [0,1]^s | u_j = 1 \text{ for } j \neq i_1, \ldots, i_k\}$

- Variation in the sense of Hardy and Krause

  $$V(f) := \sum_{k=1}^{s} \sum_{1 \leq i_1 < \cdots < i_k \leq s} V^{(k)}(f; i_1, \ldots, i_k)$$

- **Definition:**

  f is of bounded variation in the sense of Hardy and Krause, if $V(f)$ is finite.

- Estimating the variation in the sense of Hardy and Krause
  - use regular grid at $N = n^s$ samples
  - compute difference operator $\triangle$ on the grid
  - sum up the approximations of the single Vitali variations
  - $n \to \infty$

Variation Reduction

- Transfer Monte Carlo variance reduction techniques to quasi-Monte Carlo
  - separation of the main part
  - multilevel method of dependent tests
  - importance sampling
  - replication heuristics (presmoothing the integrand)

Variation Reduction

- Transfer Monte Carlo variance reduction techniques to quasi-Monte Carlo
  - separation of the main part
  - multilevel method of dependent tests
  - importance sampling
  - replication heuristics (presmoothing the integrand)

- Quasi-Monte Carlo importance sampling

$$\left| \int_{I^s} f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} \frac{f(y_i)}{p(y_i)} \right| \leq V\left(\frac{f}{p}\right) D^*(P_N)$$

  where $y_i \sim p$ by the multidimensional inversion method

  - Similar to the Monte Carlo case, the variation is not changed
  - For low discrepancy points P_N quadratically faster than random sampling

Approximating Continuous by Discrete Measures

- Often integrands of the form $f = gp$
  - p can be modeled using the multidimensional inversion method
  - g is hard to handle (e.g. discontinuous, expensive)

Approximating Continuous by Discrete Measures

- Often integrands of the form $f = gp$
  - p can be modeled using the multidimensional inversion method
  - g is hard to handle (e.g. discontinuous, expensive)
- Avoid weighting by small probabilities

$$\int_{I^s} f(x)dx = \int_{I^s} g(x)p(x)dx = \int_{I^s} g(y)dP(y)$$

- Approximate measure P by discrete measure

$$P_N := \frac{1}{N}\sum_{i=0}^{N-1} \delta_{y_i}$$

  modeled by $y_i = P^{-1}(x_i)$ from $x_i \sim \mathcal{U}$

- Then

$$\int_{I^s} g(y)dP(y) \approx \int_{I^s} g(y)dP_N(y) := \frac{1}{N}\sum_{i=0}^{N-1} g(y_i)$$

Discrepancy Bounds for Transformed Points

- **Definition:** The **discrepancy with respect to the density** p is

$$D^*(p, C_N) := \sup_{A \in \mathcal{J}^*} \left| \int_{I^s} \chi_A(x) p(x) dx - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(y_i) \right|$$

where $C_N = \{y_0, \ldots, y_{N-1}\}$

Discrepancy Bounds for Transformed Points

- **Definition:** The **discrepancy with respect to the density** p is

$$D^*(p, C_N) := \sup_{A \in \mathcal{J}^*} \left| \int_{I^s} \chi_A(x) p(x) dx - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(y_i) \right|$$

  where $C_N = \{y_0, \ldots, y_{N-1}\}$

- Multidimensional inversion method: If p is separable, i.e. $p(x) = \prod_{j=1}^{s} p^{(j)}(x^{(j)})$

$$D^*(p, C_N) = D^*(P_N)$$

  otherwise

$$D^*(p, C_N) \leq c \left(D^*(P_N)\right)^{\frac{1}{s}} \qquad c \in \mathbb{R}^+$$

Discrepancy Bounds for Transformed Points

- **Definition:** The **discrepancy with respect to the density** p is

$$D^*(p, C_N) := \sup_{A \in \mathcal{J}^*} \left| \int_{I^s} \chi_A(x) p(x) dx - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(y_i) \right|$$

  where $C_N = \{y_0, \ldots, y_{N-1}\}$

- Multidimensional inversion method: If p is separable, i.e. $p(x) = \prod_{j=1}^{s} p^{(j)}(x^{(j)})$

$$D^*(p, C_N) = D^*(P_N)$$

  otherwise

$$D^*(p, C_N) \leq c \, (D^*(P_N))^{\frac{1}{s}} \qquad c \in \mathbb{R}^+$$

  Discrete density approximation by elements of low discrepancy outperforms random sampling !!!

Discrepancy Bounds for Transformed Points

- **Definition:** The **discrepancy with respect to the density** p is

$$D^*(p, C_N) := \sup_{A \in \mathcal{J}^*} \left| \int_{I^s} \chi_A(x) p(x) dx - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(y_i) \right|$$

  where $C_N = \{y_0, \ldots, y_{N-1}\}$

- Multidimensional inversion method: If p is separable, i.e. $p(x) = \prod_{j=1}^{s} p^{(j)}(x^{(j)})$

$$D^*(p, C_N) = D^*(P_N)$$

  otherwise

$$D^*(p, C_N) \leq c \left(D^*(P_N)\right)^{\frac{1}{s}} \qquad c \in \mathbb{R}^+$$

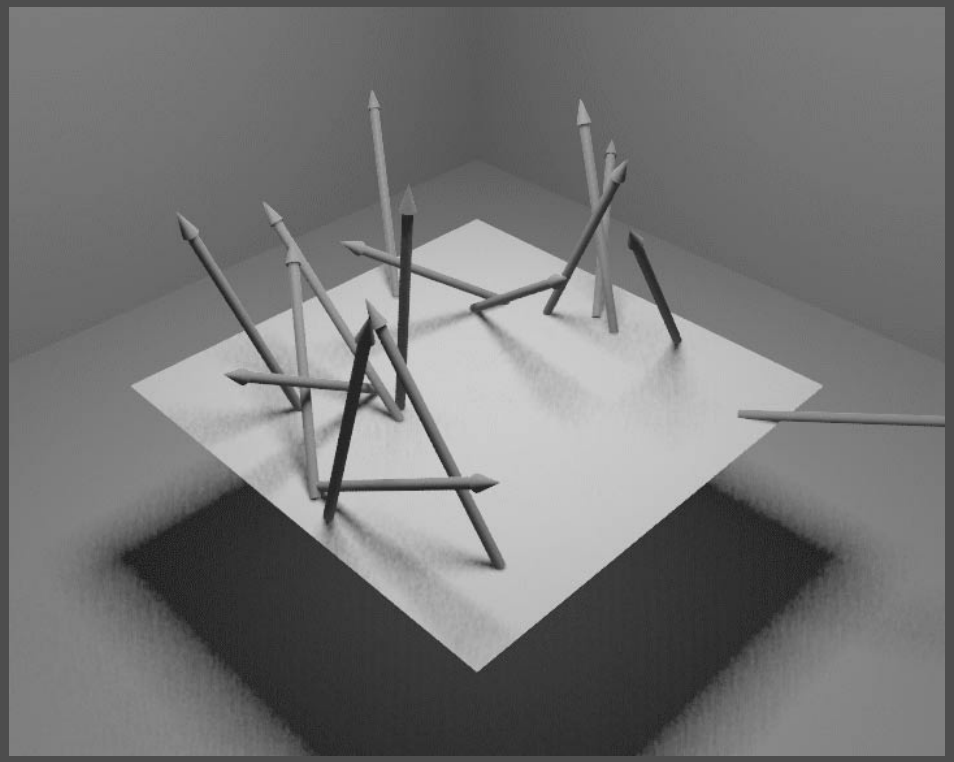
  Discrete density approximation by elements of low discrepancy outperforms random sampling !!!

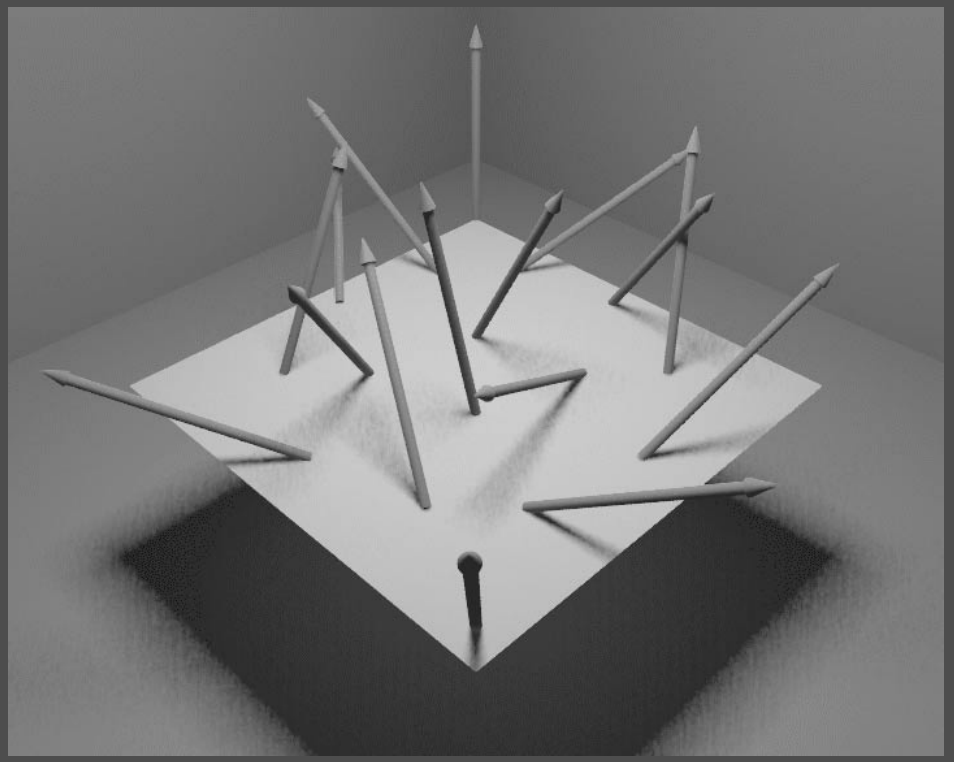
- Generalized Koksma-Hlawka inequality

$$\left| \int_{I^s} g(x) p(x) dx - \frac{1}{N} \sum_{i=0}^{N-1} g(y_i) \right| \leq V(g) D^*(p, C_N)$$

Discrete Density Approximation

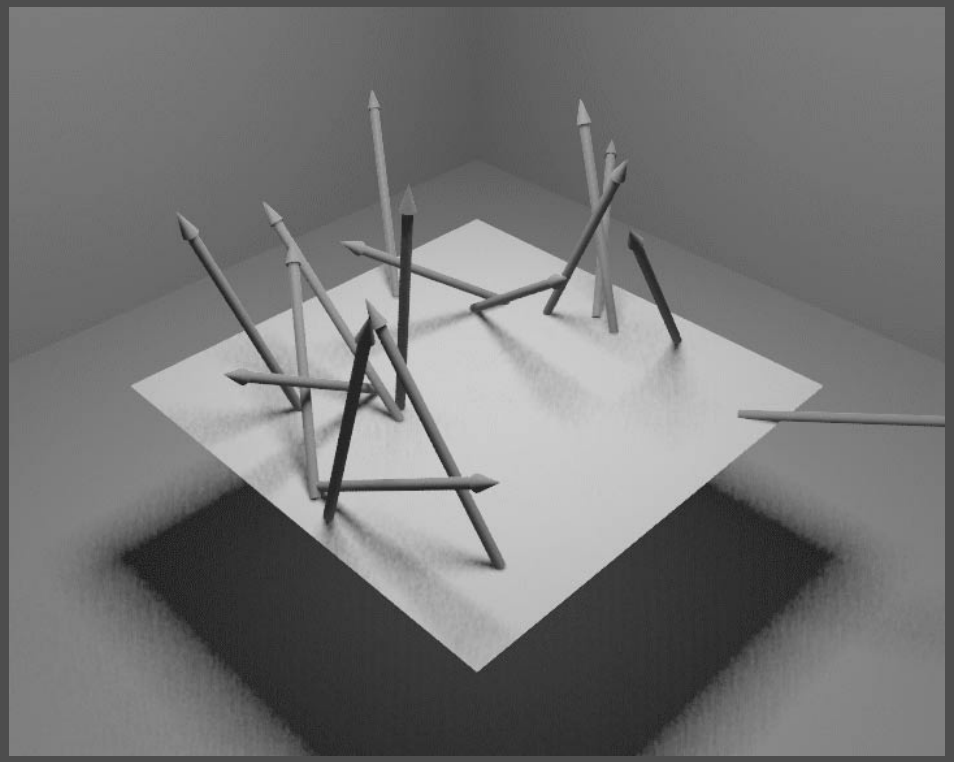
- Example: Particle emission (jittered sampling and Hammersley points at $N = 16$)

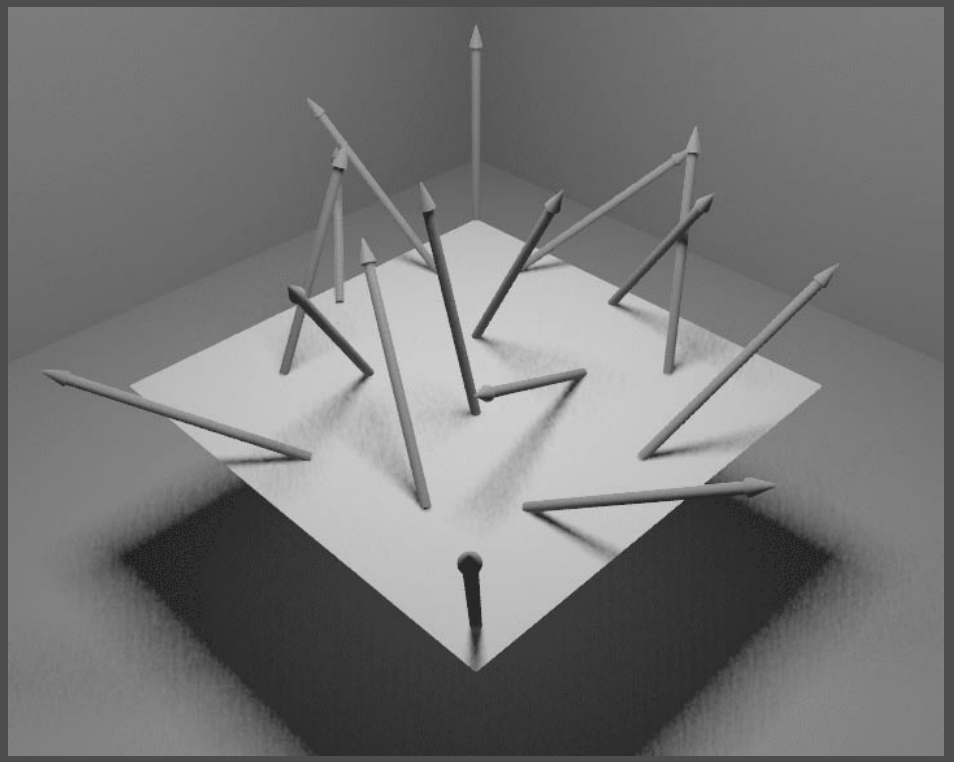




Discrete Density Approximation

- Example: Particle emission (jittered sampling and Hammersley points at $N = 16$)

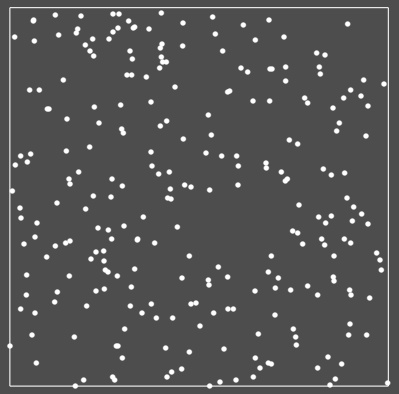


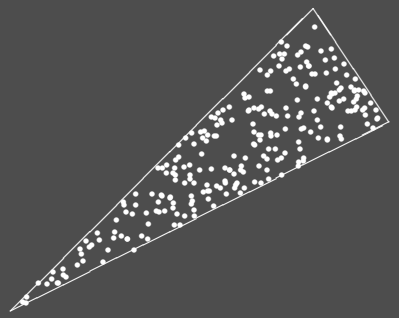


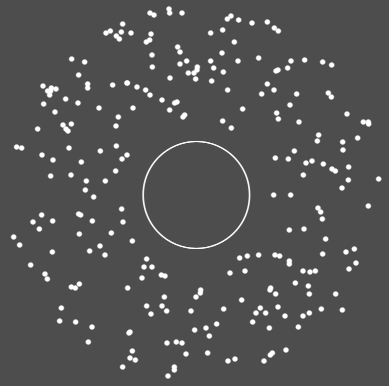
- **Note:** Assigning dimensions is crucial

Discrete Density Approximation

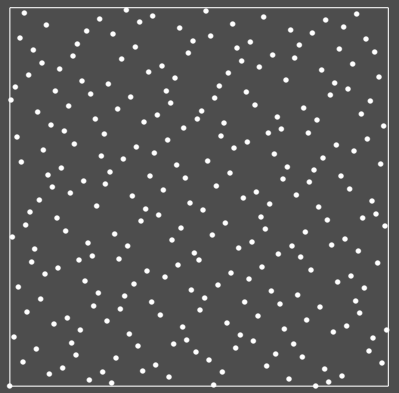
Random

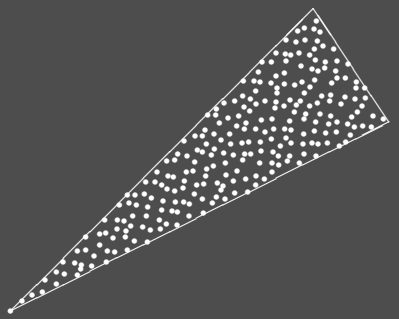


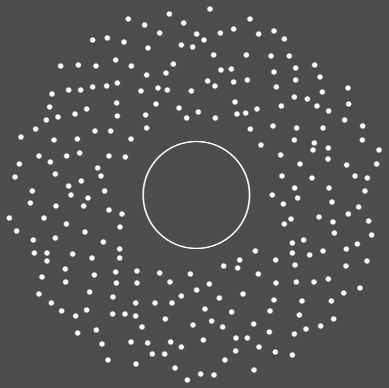




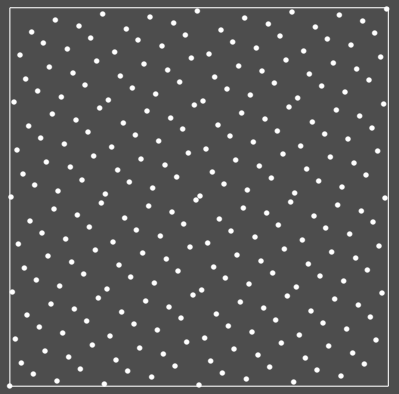
Halton

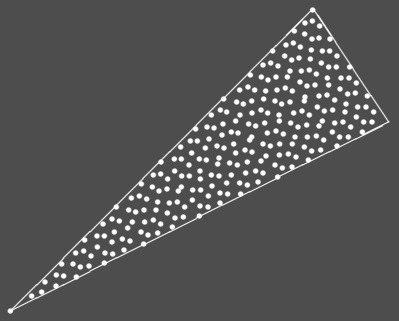


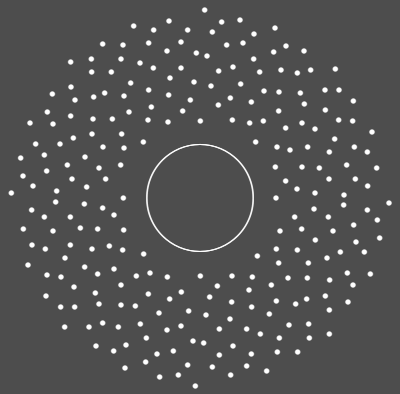




Hammersley





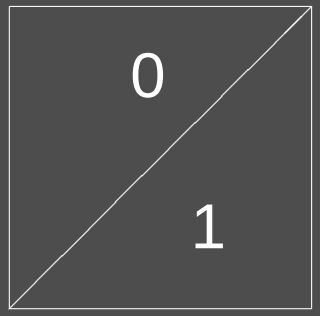


Infinite Variation

- Quasi-Monte Carlo is roughly quadratically faster than random sampling
- Case $s = 1$: $V(f) < \infty$ for piecewise continuous functions
- General case: Usually infinite variation for piecewise continuous functions

Infinite Variation

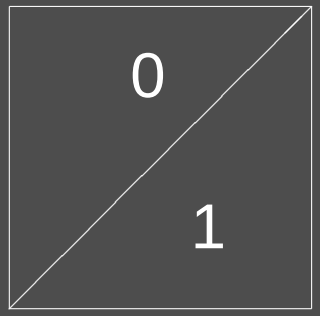
- Quasi-Monte Carlo is roughly quadratically faster than random sampling
- Case $s = 1$: $V(f) < \infty$ for piecewise continuous functions
- General case: Usually infinite variation for piecewise continuous functions
- In computer graphics: Triangles and edges



$$V(f) = \infty \qquad \sigma^2(f) = \frac{1}{4}$$

Infinite Variation

- Quasi-Monte Carlo is roughly quadratically faster than random sampling
- Case $s = 1$: $V(f) < \infty$ for piecewise continuous functions
- General case: Usually infinite variation for piecewise continuous functions
- In computer graphics: Triangles and edges



$$V(f) = \infty \qquad \sigma^2(f) = \frac{1}{4}$$

- Proof for the Hammersley points at $N = 2^l$

$$\left| \int_{I^2} f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} f(x_i) \right| = \begin{cases} \frac{1}{2\sqrt{N}} & l \text{ even} \\ \frac{1}{\sqrt{2N}} & \text{else} \end{cases}$$

Far Too Pessimistic Bounds by Isotropic Discrepancy

- Restrict f to convex domains C, where $f|_C$ is of bounded variation

$$\begin{aligned}\left|\int_C f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1}\chi_C(x_i)f(x_i)\right| &\leq (V(f) + |f(1,\ldots,1)|)\, J(P_N) \\ &\leq (V(f) + |f(1,\ldots,1)|)\, 8sD^*(P_N)^{\frac{1}{s}}\end{aligned}$$

- Bound worse than the Monte Carlo rate for $s > 2$

Far Too Pessimistic Bounds by Isotropic Discrepancy

- Restrict f to convex domains C, where $f|_C$ is of bounded variation

$$\left| \int_C f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} \chi_C(x_i) f(x_i) \right| \leq (V(f) + |f(1,\ldots,1)|)\, J(P_N)$$

$$\leq (V(f) + |f(1,\ldots,1)|)\, 8sD^*(P_N)^{\frac{1}{s}}$$

- Bound worse than the Monte Carlo rate for $s > 2$

- Numerical experiments tell a different story...
  - see e.g. the experiments on the triangle discrepancy

- Justification by discrete density approximation
  - using low discrepancy sequences always is better

Far Too Pessimistic Bounds by Isotropic Discrepancy

- Restrict f to convex domains C, where $f|_C$ is of bounded variation

$$\left| \int_C f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} \chi_C(x_i) f(x_i) \right| \leq (V(f) + |f(1,\ldots,1)|)\, J(P_N)$$

$$\leq (V(f) + |f(1,\ldots,1)|)\, 8sD^*(P_N)^{\frac{1}{s}}$$

- Bound worse than the Monte Carlo rate for $s > 2$

- Numerical experiments tell a different story...
  - see e.g. the experiments on the triangle discrepancy

- Justification by discrete density approximation
  - using low discrepancy sequences always is better

- Which function class other than bounded variation ?

Convergence

- Quasi-Monte Carlo integration converges for Riemann-integrable functions

Convergence

- Quasi-Monte Carlo integration converges for Riemann-integrable functions
- Observed rate for discontinuous functions $\mathcal{O}\left(N^{-\frac{s+1}{2s}}\right)$

Convergence

- Quasi-Monte Carlo integration converges for Riemann-integrable functions
- Observed rate for discontinuous functions $\mathcal{O}\left(N^{-\frac{s+1}{2s}}\right)$
- Argument in ”Numerical Recipes”
  - **Weak assumption:**

    The behavior of low discrepancy samples at the border of characteristic sets is uncorrelated.
  - in fact true for jittered sampling [Mitchell]
  - generalized by Szirmay-Kalos

Convergence

- Quasi-Monte Carlo integration converges for Riemann-integrable functions
- Observed rate for discontinuous functions $\mathcal{O}\left(N^{-\frac{s+1}{2s}}\right)$
- Argument in ”Numerical Recipes”
  - **Weak assumption:**

    The behavior of low discrepancy samples at the border of characteristic sets is uncorrelated.
  - in fact true for jittered sampling [Mitchell]
  - generalized by Szirmay-Kalos
- Argument by [MC95]
  - **Weak assumption:**

    Rate of random sampling used as upper bound for low discrepancy sampling, i.e. it is assumed, that low discrepancy sampling deterministically (!) does not behave worse than random sampling.
  - there exist proofs for some special cases for $s = 2$

The Spirit of the Numerical Recipes' Argument

Proposition: *Using stratified sampling to integrate the characteristic function* χ_A *for some subset* $A \subset I^s$, $\lambda_s(A) > 0$, *for* $N = \prod_{j=1}^{s} N_j$ *and the axial subdivision into* N_j *equally spaced intervals, results in the convergence rate of* $\mathcal{O}\left(N^{-\frac{s+1}{2s}}\right)$.

The Spirit of the Numerical Recipes' Argument

Proposition: *Using stratified sampling to integrate the characteristic function χ_A for some subset $A \subset I^s$, $\lambda_s(A) > 0$, for $N = \prod_{j=1}^{s} N_j$ and the axial subdivision into N_j equally spaced intervals, results in the convergence rate of $\mathcal{O}\left(N^{-\frac{s+1}{2s}}\right)$.*

Proof:

- I^s partitioned into $N = \prod_{j=1}^{s} N_j$ voxels v_i, $\lambda_s(v_i) = \frac{1}{N}$, $1 \leq i \leq N$
- Jittered sampling for

$$\int_{I^s} \chi_A(x)dx \approx \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i|v_i)$$

The Spirit of the Numerical Recipes' Argument

Proposition: *Using stratified sampling to integrate the characteristic function* χ_A *for some subset* $A \subset I^s$, $\lambda_s(A) > 0$, *for* $N = \prod_{j=1}^{s} N_j$ *and the axial subdivision into* N_j *equally spaced intervals, results in the convergence rate of* $\mathcal{O}\left(N^{-\frac{s+1}{2s}}\right)$.

Proof:

- I^s partitioned into $N = \prod_{j=1}^{s} N_j$ voxels v_i, $\lambda_s(v_i) = \frac{1}{N}$, $1 \leq i \leq N$
- Jittered sampling for

$$\int_{I^s} \chi_A(x)dx \approx \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i|v_i)$$

- Three sets of voxel indices

$$\begin{aligned} V_i &= \{v_i | v_i \cap A = v_i\} \\ V_b &= \{v_i | \emptyset \neq v_i \cap A \neq v_i\} \\ V_o &= \{v_i | v_i \cap A = \emptyset\} \end{aligned}$$

The Spirit of the Numerical Recipes' Argument

Proposition: *Using stratified sampling to integrate the characteristic function χ_A for some subset $A \subset I^s$, $\lambda_s(A) > 0$, for $N = \prod_{j=1}^{s} N_j$ and the axial subdivision into N_j equally spaced intervals, results in the convergence rate of $\mathcal{O}\left(N^{-\frac{s+1}{2s}}\right)$.*

Proof:

- I^s partitioned into $N = \prod_{j=1}^{s} N_j$ voxels v_i, $\lambda_s(v_i) = \frac{1}{N}$, $1 \leq i \leq N$
- Jittered sampling for

$$\int_{I^s} \chi_A(x)dx \approx \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i|v_i)$$

- Three sets of voxel indices

$$\begin{aligned} V_i &= \{v_i | v_i \cap A = v_i\} \\ V_b &= \{v_i | \emptyset \neq v_i \cap A \neq v_i\} \\ V_o &= \{v_i | v_i \cap A = \emptyset\} \end{aligned}$$

- ***Assumption:*** $|V_i| \in \mathcal{O}(N)$
- ***Assumption:*** Dimension of the boundary $s-1 \Rightarrow |V_b| \in \mathcal{O}\left(N^{\frac{s-1}{s}}\right)$

- Random sample $x_i \in v_i \in V_b$ is Bernoulli random variable with

$$p_i = \frac{\lambda_s(A \cap v_i)}{\lambda_s(v_i)} \qquad \text{and} \qquad \sigma^2(\chi_{A \cap v_i}) \leq \frac{1}{4}$$

- Random sample $x_i \in v_i \in V_b$ is Bernoulli random variable with

$$p_i = \frac{\lambda_s(A \cap v_i)}{\lambda_s(v_i)} \qquad \text{and} \qquad \sigma^2(\chi_{A \cap v_i}) \leq \frac{1}{4}$$

- Then

$$\sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_A(x_i|v_i)\right) = \sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_{A\cap v_i}(x_i)\right)$$

- Random sample $x_i \in v_i \in V_b$ is Bernoulli random variable with

$$p_i = \frac{\lambda_s(A \cap v_i)}{\lambda_s(v_i)} \qquad \text{and} \qquad \sigma^2(\chi_{A \cap v_i}) \leq \frac{1}{4}$$

- Then

$$\sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_A(x_i|v_i)\right) = \sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_{A\cap v_i}(x_i)\right)$$

$$= \sigma^2\left(\frac{1}{N}\sum_{i\in V_i}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_o}\chi_{A\cap v_i}(x_i)\right)$$

- Random sample $x_i \in v_i \in V_b$ is Bernoulli random variable with

$$p_i = \frac{\lambda_s(A \cap v_i)}{\lambda_s(v_i)} \qquad \text{and} \qquad \sigma^2(\chi_{A \cap v_i}) \leq \frac{1}{4}$$

- Then

$$\sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_A(x_i|v_i)\right) = \sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_{A\cap v_i}(x_i)\right)$$

$$= \sigma^2\left(\frac{1}{N}\sum_{i\in V_i}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_o}\chi_{A\cap v_i}(x_i)\right)$$

$$= \sigma^2\left(\frac{1}{N}|V_i| + \frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i) + 0\right)$$

- Random sample $x_i \in v_i \in V_b$ is Bernoulli random variable with

$$p_i = \frac{\lambda_s(A \cap v_i)}{\lambda_s(v_i)} \qquad \text{and} \qquad \sigma^2(\chi_{A \cap v_i}) \leq \frac{1}{4}$$

- Then

$$\begin{aligned}
\sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_A(x_i|v_i)\right) &= \sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_{A\cap v_i}(x_i)\right) \\
&= \sigma^2\left(\frac{1}{N}\sum_{i\in V_i}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_o}\chi_{A\cap v_i}(x_i)\right) \\
&= \sigma^2\left(\frac{1}{N}|V_i| + \frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i) + 0\right) \\
&= \sigma^2\left(\frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i)\right)
\end{aligned}$$

- Random sample $x_i \in v_i \in V_b$ is Bernoulli random variable with

$$p_i = \frac{\lambda_s(A \cap v_i)}{\lambda_s(v_i)} \qquad \text{and} \qquad \sigma^2(\chi_{A \cap v_i}) \leq \frac{1}{4}$$

- Then

$$\sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_A(x_i|v_i)\right) = \sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_{A\cap v_i}(x_i)\right)$$

$$= \sigma^2\left(\frac{1}{N}\sum_{i\in V_i}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_o}\chi_{A\cap v_i}(x_i)\right)$$

$$= \sigma^2\left(\frac{1}{N}|V_i| + \frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i) + 0\right)$$

$$= \sigma^2\left(\frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i)\right) = \sum_{i\in V_b}\frac{\sigma^2(\chi_{A\cap v_i}(x_i))}{N^2}$$

- Random sample $x_i \in v_i \in V_b$ is Bernoulli random variable with

$$p_i = \frac{\lambda_s(A \cap v_i)}{\lambda_s(v_i)} \qquad \text{and} \qquad \sigma^2(\chi_{A \cap v_i}) \leq \frac{1}{4}$$

- Then

$$\begin{aligned}
\sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_A(x_i|v_i)\right) &= \sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_{A\cap v_i}(x_i)\right) \\
&= \sigma^2\left(\frac{1}{N}\sum_{i\in V_i}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_o}\chi_{A\cap v_i}(x_i)\right) \\
&= \sigma^2\left(\frac{1}{N}|V_i| + \frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i) + 0\right) \\
&= \sigma^2\left(\frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i)\right) = \sum_{i\in V_b}\frac{\sigma^2(\chi_{A\cap v_i}(x_i))}{N^2} \\
&\leq |V_b|\frac{\frac{1}{4}}{N^2} = cN^{\frac{s-1}{s}}N^{-2} = cN^{-\frac{s+1}{s}}
\end{aligned}$$

– By the Hölder inequality the error is expected to be

$$\left| \int_{I^s} \chi_A(x) dx - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i) \right| \leq \sqrt{c N^{-\frac{s+1}{s}}} \in \mathcal{O}(N^{-\frac{s+1}{2s}}) \qquad \textit{q.e.d.}$$

- By the Hölder inequality the error is expected to be

$$\left| \int_{I^s} \chi_A(x) dx - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i) \right| \leq \sqrt{c N^{-\frac{s+1}{s}}} \in \mathcal{O}(N^{-\frac{s+1}{2s}}) \qquad \textit{q.e.d.}$$

- **Note:**

$$\lim_{s \to \infty} N^{-\frac{s+1}{2s}} = N^{-\frac{1}{2}}$$

Error Control

- Determinism: Variance of estimate is zero !
  - no cheap error estimate from samples
  - no efficiency - complex analysis by information based complexity theory
  - quasi-Monte Carlo integration is ”biased” but ”consistent”

Error Control

- Determinism: Variance of estimate is zero !
  - no cheap error estimate from samples
  - no efficiency - complex analysis by information based complexity theory
  - quasi-Monte Carlo integration is ”biased” but ”consistent”

- Adaptive sampling by using low discrepancy sequences
  - convergence is rather smooth due to intrinsic stratification properties
  - choose fixed distance $\triangle N$ of samples
  - compare difference of averages all $\triangle N$ to a threshold
  - must be below the threshold T times

Error Control

- Determinism: Variance of estimate is zero !
  - no cheap error estimate from samples
  - no efficiency - complex analysis by information based complexity theory
  - quasi-Monte Carlo integration is ”biased” but ”consistent”

- Adaptive sampling by using low discrepancy sequences
  - convergence is rather smooth due to intrinsic stratification properties
  - choose fixed distance $\triangle N$ of samples
  - compare difference of averages all $\triangle N$ to a threshold
  - must be below the threshold T times

- The points ”know” where to fall

Error Control

- Determinism: Variance of estimate is zero !
  - no cheap error estimate from samples
  - no efficiency - complex analysis by information based complexity theory
  - quasi-Monte Carlo integration is ”biased” but ”consistent”

- Adaptive sampling by using low discrepancy sequences
  - convergence is rather smooth due to intrinsic stratification properties
  - choose fixed distance $\triangle N$ of samples
  - compare difference of averages all $\triangle N$ to a threshold
  - must be below the threshold T times

- The points ”know” where to fall

- **Consider local minima for $\triangle N$!**
  - e.g. (t, s)-sequences at $\triangle N = b^m$
  - e.g. Hammersley in $s = 2$

From Monte Carlo to Quasi-Monte Carlo Integration

- The basic algorithms transfer
  - integration
  - integro-approximation
  - Separation of main part and multilevel method of dependent tests

From Monte Carlo to Quasi-Monte Carlo Integration

- The basic algorithms transfer
  - integration
  - integro-approximation
  - Separation of main part and multilevel method of dependent tests

- Faster convergence by deterministic low discrepancy sampling
  - intrinsically stratified, Latin hypercube, regularized, antithetic, ...

From Monte Carlo to Quasi-Monte Carlo Integration

- The basic algorithms transfer
  - integration
  - integro-approximation
  - Separation of main part and multilevel method of dependent tests

- Faster convergence by deterministic low discrepancy sampling
  - intrinsically stratified, Latin hypercube, regularized, antithetic, ...

- The simulation of random variables becomes discrete density approximation
  - no independence required due to averaging
  - importance sampling carries over
  - rejection modeling impossible

From Monte Carlo to Quasi-Monte Carlo Integration

- The basic algorithms transfer
  - integration
  - integro-approximation
  - Separation of main part and multilevel method of dependent tests
- Faster convergence by deterministic low discrepancy sampling
  - intrinsically stratified, Latin hypercube, regularized, antithetic, ...
- The simulation of random variables becomes discrete density approximation
  - no independence required due to averaging
  - importance sampling carries over
  - rejection modeling impossible
- Adaptive sampling by difference comparison
- What about splitting ?

Efficient Design of Quasi-Monte Carlo Algorithms

- Write down the integral

Efficient Design of Quasi-Monte Carlo Algorithms

- Write down the integral
- Transform onto unit cube I^s

Efficient Design of Quasi-Monte Carlo Algorithms

- Write down the integral
- Transform onto unit cube I^s
- Separate the main part

Efficient Design of Quasi-Monte Carlo Algorithms

- Write down the integral
- Transform onto unit cube I^s
- Separate the main part
- Apply (multiple) importance sampling

Efficient Design of Quasi-Monte Carlo Algorithms

- Write down the integral
- Transform onto unit cube I^s
- Separate the main part
- Apply (multiple) importance sampling
- Use quasi-Monte Carlo points
  - sample size N
  - assigning dimensions

Efficient Design of Quasi-Monte Carlo Algorithms

- Write down the integral
- Transform onto unit cube I^s
- Separate the main part
- Apply (multiple) importance sampling
- Use quasi-Monte Carlo points
  - sample size N
  - assigning dimensions
- Use dependent splitting

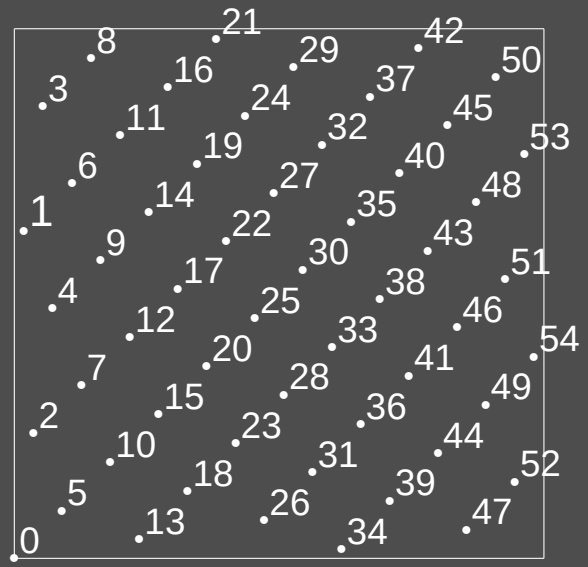
Quasi-Monte Carlo Integration using Lattice Points

- Originally developed for the class $E_\alpha(c)$ with $c > 0, \alpha > 1$, where

$$f \in E_\alpha(c) \Leftrightarrow |\hat{f}(h)| \leq \frac{c}{(\bar{h}_1 \cdots \bar{h}_s)^\alpha} \qquad \bar{h}_j := \max\{1, |h_j|\}, \vec{h} \in \mathbb{Z}^s$$

- Error bound

$$\left| \frac{1}{N} \sum_{i=0}^{N-1} f\left(\frac{i}{N}\vec{g}\right) - \int_{I^s} f(x)dx \right| \leq \sum_{\vec{h}\cdot\vec{g} \equiv 0 (\bmod N), \vec{h} \neq 0} \frac{1}{(\bar{h}_1 \cdots \bar{h}_s)^\alpha}$$



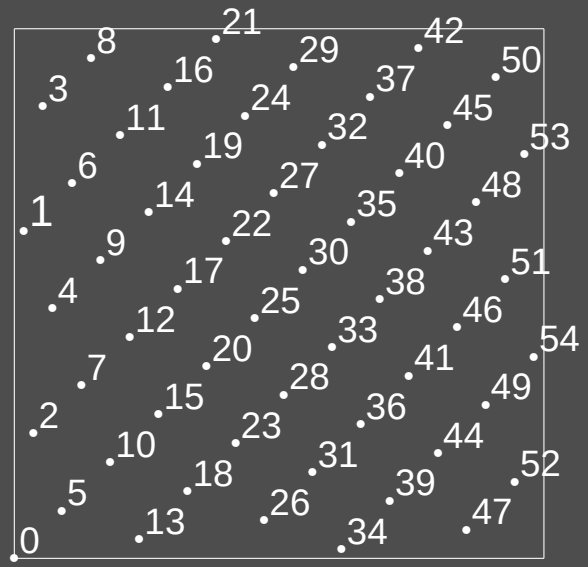
Quasi-Monte Carlo Integration using Lattice Points

- Originally developed for the class $E_\alpha(c)$ with $c > 0, \alpha > 1$, where

$$f \in E_\alpha(c) \Leftrightarrow |\hat{f}(h)| \leq \frac{c}{(\bar{h}_1 \cdots \bar{h}_s)^\alpha} \qquad \bar{h}_j := \max\{1, |h_j|\}, \vec{h} \in \mathbb{Z}^s$$

- Error bound

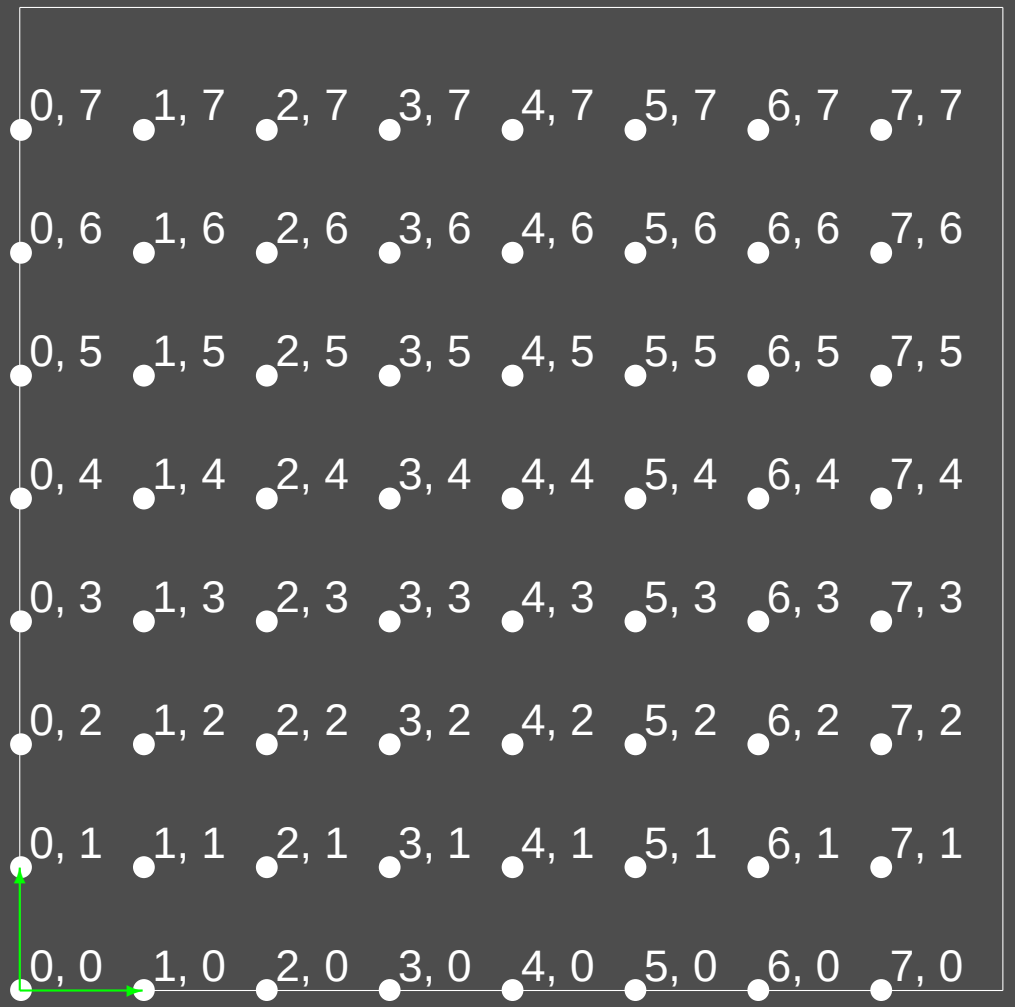
$$\left| \frac{1}{N} \sum_{i=0}^{N-1} f\left(\frac{i}{N}\vec{g}\right) - \int_{I^s} f(x)dx \right| \leq \sum_{\vec{h}\cdot\vec{g} \equiv 0 (\bmod N), \vec{h} \neq 0} \frac{1}{(\bar{h}_1 \cdots \bar{h}_s)^\alpha}$$



- Generalized to class of bounded variation

Curse of Dimension from Regular Grids

- Lattices of rank s with $N = n^s$ points from tensor product approach



- $\mathcal{O}\left(n^s \log n\right)$ for s fast Fourier transforms

Fourier Transform on Rank-1 Lattices

- Choice of wave vectors

$$K_N := \{\vec{k}_0, \ldots, \vec{k}_{N-1}\} \subset \mathbb{Z}^s$$

such that

$$\vec{k}_m \in Z_m := \{\vec{k} \in \mathbb{Z}^s \mid \vec{k}^T \cdot \vec{g} \equiv m \pmod{N}\}$$

Fourier Transform on Rank-1 Lattices

- Choice of wave vectors

$$K_N := \{\vec{k}_0, \ldots, \vec{k}_{N-1}\} \subset \mathbb{Z}^s$$

such that

$$\vec{k}_m \in Z_m := \{\vec{k} \in \mathbb{Z}^s \mid \vec{k}^T \cdot \vec{g} \equiv m \pmod{N}\}$$

since then

$$\vec{k}_m^T \cdot \vec{x}_n = \vec{k}_m^T \cdot \frac{n}{N}\vec{g} = (m + l_m N)\frac{n}{N}$$

Fourier Transform on Rank-1 Lattices

- Choice of wave vectors

  $$K_N := \{\vec{k}_0, \ldots, \vec{k}_{N-1}\} \subset \mathbb{Z}^s$$

  such that

  $$\vec{k}_m \in Z_m := \{\vec{k} \in \mathbb{Z}^s \mid \vec{k}^T \cdot \vec{g} \equiv m \pmod N\}$$

  since then

  $$\vec{k}_m^T \cdot \vec{x}_n = \vec{k}_m^T \cdot \frac{n}{N}\vec{g} = (m + l_m N)\frac{n}{N}$$

- Evaluate

  $$\vec{f}(\vec{x}_n) = \sum_{\vec{k} \in K_N} \hat{\vec{f}}(\vec{k}) e^{2\pi i \vec{k}^T \cdot \vec{x}_n} = \sum_{m=0}^{N-1} \hat{\vec{f}}(\vec{k}_m) e^{2\pi i \vec{k}_m^T \cdot \vec{x}_n}$$

Fourier Transform on Rank-1 Lattices

- Choice of wave vectors

  $$K_N := \{\vec{k}_0, \ldots, \vec{k}_{N-1}\} \subset \mathbb{Z}^s$$

  such that

  $$\vec{k}_m \in Z_m := \{\vec{k} \in \mathbb{Z}^s \mid \vec{k}^T \cdot \vec{g} \equiv m \pmod{N}\}$$

  since then

  $$\vec{k}_m^T \cdot \vec{x}_n = \vec{k}_m^T \cdot \frac{n}{N}\vec{g} = (m + l_m N)\frac{n}{N}$$

- Evaluate

  $$\begin{aligned} \vec{f}(\vec{x}_n) &= \sum_{\vec{k} \in K_N} \hat{\vec{f}}(\vec{k}) e^{2\pi i \vec{k}^T \cdot \vec{x}_n} = \sum_{m=0}^{N-1} \hat{\vec{f}}(\vec{k}_m) e^{2\pi i \vec{k}_m^T \cdot \vec{x}_n} \\ &= \sum_{m=0}^{N-1} \hat{\vec{f}}(\vec{k}_m) e^{2\pi i (m\frac{n}{N} + l_m n)} \end{aligned}$$

Fourier Transform on Rank-1 Lattices

- Choice of wave vectors

$$K_N := \{\vec{k}_0, \ldots, \vec{k}_{N-1}\} \subset \mathbb{Z}^s$$

such that

$$\vec{k}_m \in Z_m := \{\vec{k} \in \mathbb{Z}^s \mid \vec{k}^T \cdot \vec{g} \equiv m \pmod{N}\}$$

since then

$$\vec{k}_m^T \cdot \vec{x}_n = \vec{k}_m^T \cdot \frac{n}{N}\vec{g} = (m + l_m N)\frac{n}{N}$$

- Evaluate

$$\begin{aligned}
\vec{f}(\vec{x}_n) &= \sum_{\vec{k} \in K_N} \hat{\vec{f}}(\vec{k}) e^{2\pi i \vec{k}^T \cdot \vec{x}_n} = \sum_{m=0}^{N-1} \hat{\vec{f}}(\vec{k}_m) e^{2\pi i \vec{k}_m^T \cdot \vec{x}_n} \\
&= \sum_{m=0}^{N-1} \hat{\vec{f}}(\vec{k}_m) e^{2\pi i (m\frac{n}{N} + l_m n)} \\
&= \sum_{m=0}^{N-1} \hat{\vec{f}}(\vec{k}_m) e^{2\pi i m \frac{n}{N}}
\end{aligned}$$

by **one-dimensional** Fourier transform $\Rightarrow$ **way to break curse of dimension !**

Determining the Wave Vectors

- Many possible choices for

$$\vec{k}_m \in Z_m := \{\vec{k} \in \mathbb{Z}^s \mid \vec{k}^T \cdot \vec{g} \equiv m \pmod{N}\}$$

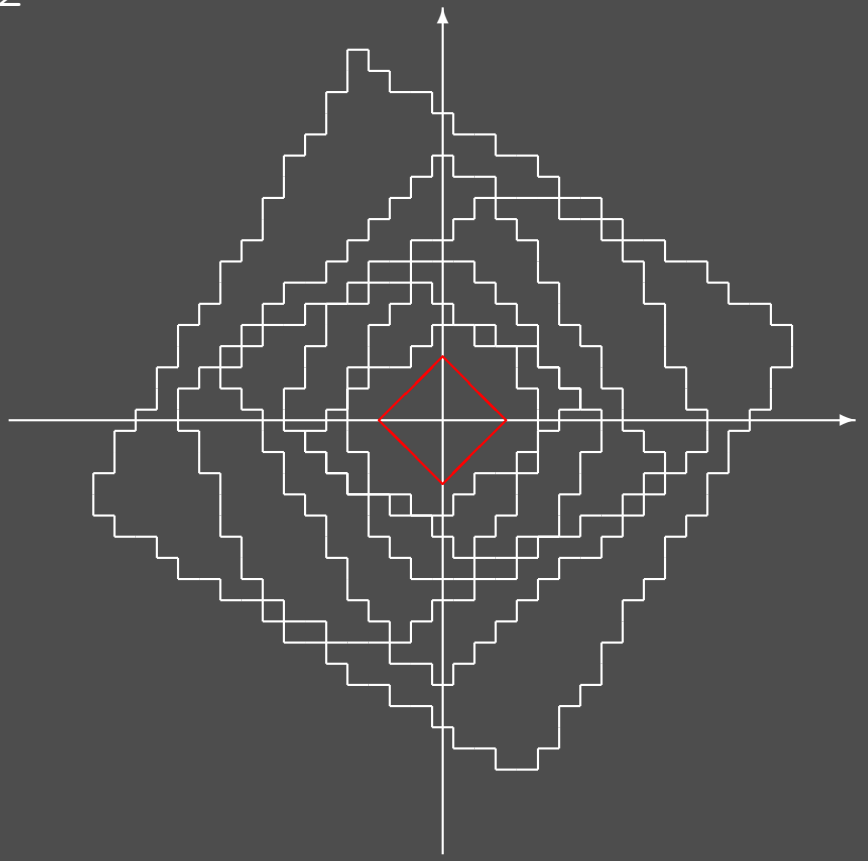
Determining the Wave Vectors

- Many possible choices for

$$\vec{k}_m \in Z_m := \{\vec{k} \in \mathbb{Z}^s \mid \vec{k}^T \cdot \vec{g} \equiv m \pmod{N}\}$$

- Choose largest waves first

$$\|\vec{k}_m\|_2 = \min_{\vec{k} \in Z_m} \|\vec{k}\|_2.$$



- Enumerate along lines of constant $\|\cdot\|_1$-norm

Summary

- Quasi-Monte Carlo simpler and faster than Monte Carlo integration
- Most Monte Carlo techniques transfer
- However, no rejection sampling !
- Works fine on L^2, too
  - justification by discrete density approximation
- Breaks curse of dimension even for discrete Fourier transform

Summary

- Quasi-Monte Carlo simpler and faster than Monte Carlo integration
- Most Monte Carlo techniques transfer
- However, no rejection sampling !
- Works fine on L^2, too
  - justification by discrete density approximation
- Breaks curse of dimension even for discrete Fourier transform

- **Use whenever you can write the problem as an integral**

Beyond Monte Carlo

- Day 1: Monte Carlo
- Day 2: Quasi-Monte Carlo points
- Day 3: Quasi-Monte Carlo integration
- **Day 4: Monte Carlo extensions of quasi-Monte Carlo**
- Day 5: Applications to computer graphics

Day 4: Monte Carlo Extensions of Quasi-Monte Carlo

- Random field synthesis on good lattice points
- Randomized quasi-Monte Carlo integration
- Randomized replications
- Restricted randomized replications

Periodic Random Field Synthesis on Good Lattice Points

- Applications of Periodic Random Fields $\vec{f_\omega}(\vec{x}) = \vec{f_\omega}(\vec{x} + \vec{z})$ for $\vec{z} \in \mathbb{Z}^s$ (Period 1)
  - height fields: Waves, terrain
  - caustics
  - turbulent wind fields

Periodic Random Field Synthesis on Good Lattice Points

- Applications of Periodic Random Fields $\vec{f_\omega}(\vec{x}) = \vec{f_\omega}(\vec{x} + \vec{z})$ for $\vec{z} \in \mathbb{Z}^s$ (Period 1)
  - height fields: Waves, terrain
  - caustics
  - turbulent wind fields
- Typical procedure

1. Realize Gaussian noise

$$\vec{N_\omega}(\vec{k}) \sim (\mathcal{N}(0,1) \times i\mathcal{N}(0,1))^d$$

Periodic Random Field Synthesis on Good Lattice Points

- Applications of Periodic Random Fields $\vec{f_\omega}(\vec{x}) = \vec{f_\omega}(\vec{x} + \vec{z})$ for $\vec{z} \in \mathbb{Z}^s$ (Period 1)
  - height fields: Waves, terrain
  - caustics
  - turbulent wind fields
- Typical procedure

1. Realize Gaussian noise

$$\vec{N}_\omega(\vec{k}) \sim (\mathcal{N}(0,1) \times i\mathcal{N}(0,1))^d$$

2. Filter noise by spectrum S of phenomenon

$$\vec{\tilde{f}}_\omega(\vec{k}) = S(\vec{k})\vec{N}_\omega(\vec{k})$$

Periodic Random Field Synthesis on Good Lattice Points

- Applications of Periodic Random Fields $\vec{f_\omega}(\vec{x}) = \vec{f_\omega}(\vec{x} + \vec{z})$ for $\vec{z} \in \mathbb{Z}^s$ (Period 1)
  - height fields: Waves, terrain
  - caustics
  - turbulent wind fields
- Typical procedure

1. Realize Gaussian noise

$$\vec{N}_\omega(\vec{k}) \sim (\mathcal{N}(0,1) \times i\mathcal{N}(0,1))^d$$

2. Filter noise by spectrum S of phenomenon

$$\vec{\hat{f}}_\omega(\vec{k}) = S(\vec{k})\vec{N}_\omega(\vec{k})$$

3. Band limited evaluation by fast Fourier transform

$$\vec{f_\omega}(\vec{x}) = \sum_{\vec{k} \in K_N} \vec{\hat{f}}_\omega(\vec{k}) e^{2\pi i \vec{k}^T \cdot \vec{x}}$$

Fourier Transform on Rank-1 Lattices

- Choice of wave vectors $K_N := \{\vec{k}_0, \ldots, \vec{k}_{N-1}\} \subset \mathbb{Z}^s$

  such that

  $$\vec{k}_m \in Z_m := \{\vec{k} \in \mathbb{Z}^s \mid \vec{k}^T \cdot \vec{g} \equiv m \pmod N\}$$

  hence with $x_n = \frac{n}{N}\vec{g}$

  $$\vec{k}_m^T \cdot \vec{x}_n = \vec{k}_m^T \cdot \frac{n}{N}\vec{g} = (m + l_m N)\frac{n}{N}$$

- By **one-dimensional** Fourier transform evaluate

$$\begin{aligned} \vec{f}(\vec{x}_n) &= \sum_{\vec{k} \in K_N} \vec{\hat{f}}_\omega(\vec{k}) e^{2\pi i \vec{k}^T \cdot \vec{x}_n} = \sum_{m=0}^{N-1} \vec{\hat{f}}_\omega(\vec{k}_m) e^{2\pi i \vec{k}_m^T \cdot \vec{x}_n} \\ &= \sum_{m=0}^{N-1} \vec{\hat{f}}_\omega(\vec{k}_m) e^{2\pi i (m\frac{n}{N} + l_m n)} \\ &= \sum_{m=0}^{N-1} \vec{\hat{f}}_\omega(\vec{k}_m) e^{2\pi i m \frac{n}{N}} \end{aligned}$$

Application: Ocean Wave Simulation

- **Ocean height field synthesis**

1. Realize Gaussian noise random field $\xi_{r,m}, \xi_{i,m} \sim \mathcal{N}(0, 1)$

Application: Ocean Wave Simulation

- **Ocean height field synthesis**

1. Realize Gaussian noise random field $\xi_{r,m}, \xi_{i,m} \sim \mathcal{N}(0, 1)$

2. Fourier coefficients by filtering with Philipps spectrum $P_h(k_m)$

$$\hat{h}_\omega(\vec{k}_m, t) = \sqrt{\frac{P_h(k_m)}{2}} \left((\xi_{r,m} + i\xi_{i,m}) e^{i\omega(k_m)t} + (\xi_{r,m} - i\xi_{i,m}) e^{-i\omega(k_m)t} \right)$$

Application: Ocean Wave Simulation

- **Ocean height field synthesis**

1. Realize Gaussian noise random field $\xi_{r,m}, \xi_{i,m} \sim \mathcal{N}(0,1)$

2. Fourier coefficients by filtering with Philipps spectrum $P_h(k_m)$

$$\hat{h}_\omega(\vec{k}_m, t) = \sqrt{\frac{P_h(k_m)}{2}} \left((\xi_{r,m} + i\xi_{i,m}) e^{i\omega(k_m)t} + (\xi_{r,m} - i\xi_{i,m}) e^{-i\omega(k_m)t} \right)$$

3. Height field $h_\omega : \mathbb{R}^3 \to \mathbb{R}$ and normals by $\nabla h_\omega : \mathbb{R}^3 \to \mathbb{R}^3$

$$\begin{aligned} h_\omega(\vec{x}_n, t) &= \sum_{m=0}^{N-1} \hat{h}_\omega(\vec{k}_m, t) e^{2\pi i m \frac{n}{N}} \\ \nabla h_\omega(\vec{x}_n, t) &= \sum_{m=0}^{N-1} 2\pi i \vec{k}_m \hat{h}_\omega(\vec{k}_m, t) e^{2\pi i m \frac{n}{N}} \end{aligned}$$

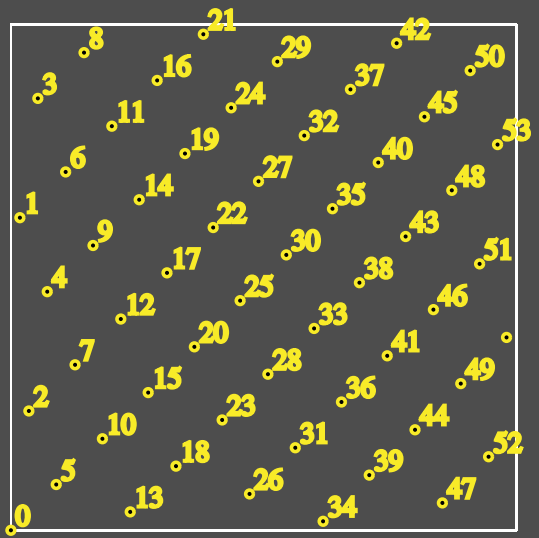
$\Rightarrow \dim \vec{x}_n = 2$, but evaluation by **one-dimensional** fast Fourier transform

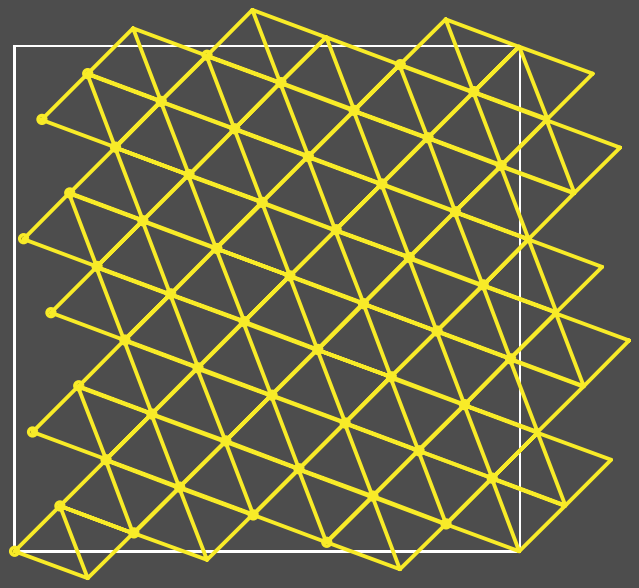
Example: Ocean Waves on Fibonacci Rank-1 Lattices

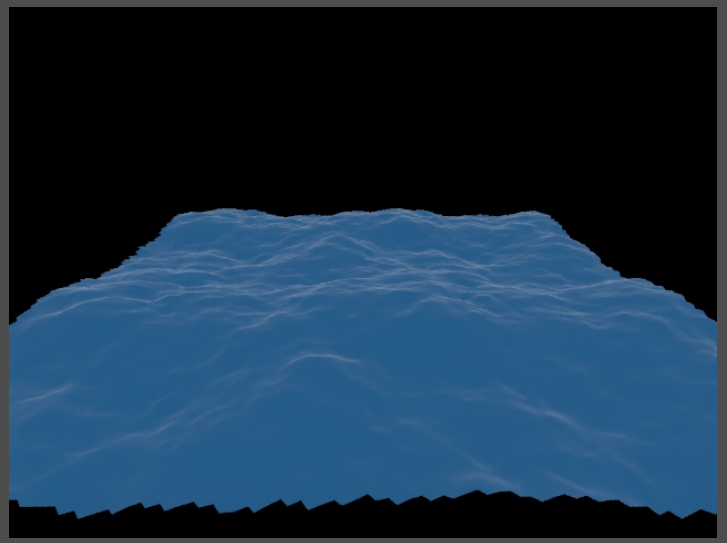
- Fibonacci numbers: $F_1 = F_2 = 1$, $F_k = F_{k-1} + F_{k-2}$ for $k > 2$
- **Fibonacci lattice** by generator vector $\vec{g} = (1, F_{k-1})$ at $N = F_k$ points

$$\vec{x}_n := \frac{n}{F_k}(1, F_{k-1})$$

  - Low discrepancy
- Example: $N = F_{10} = 55$, $\vec{x}_n := \frac{n}{55}(1, 34)$

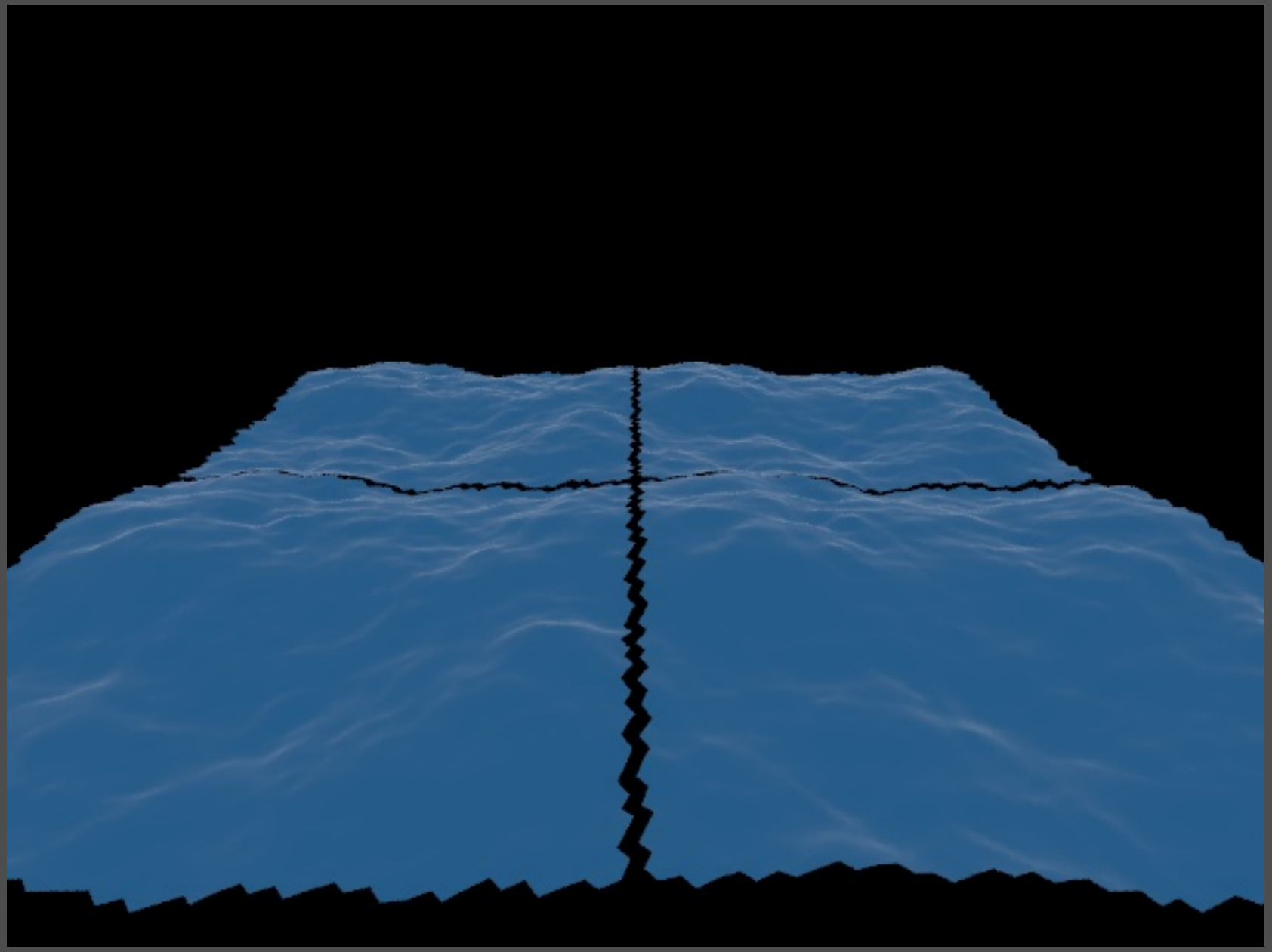




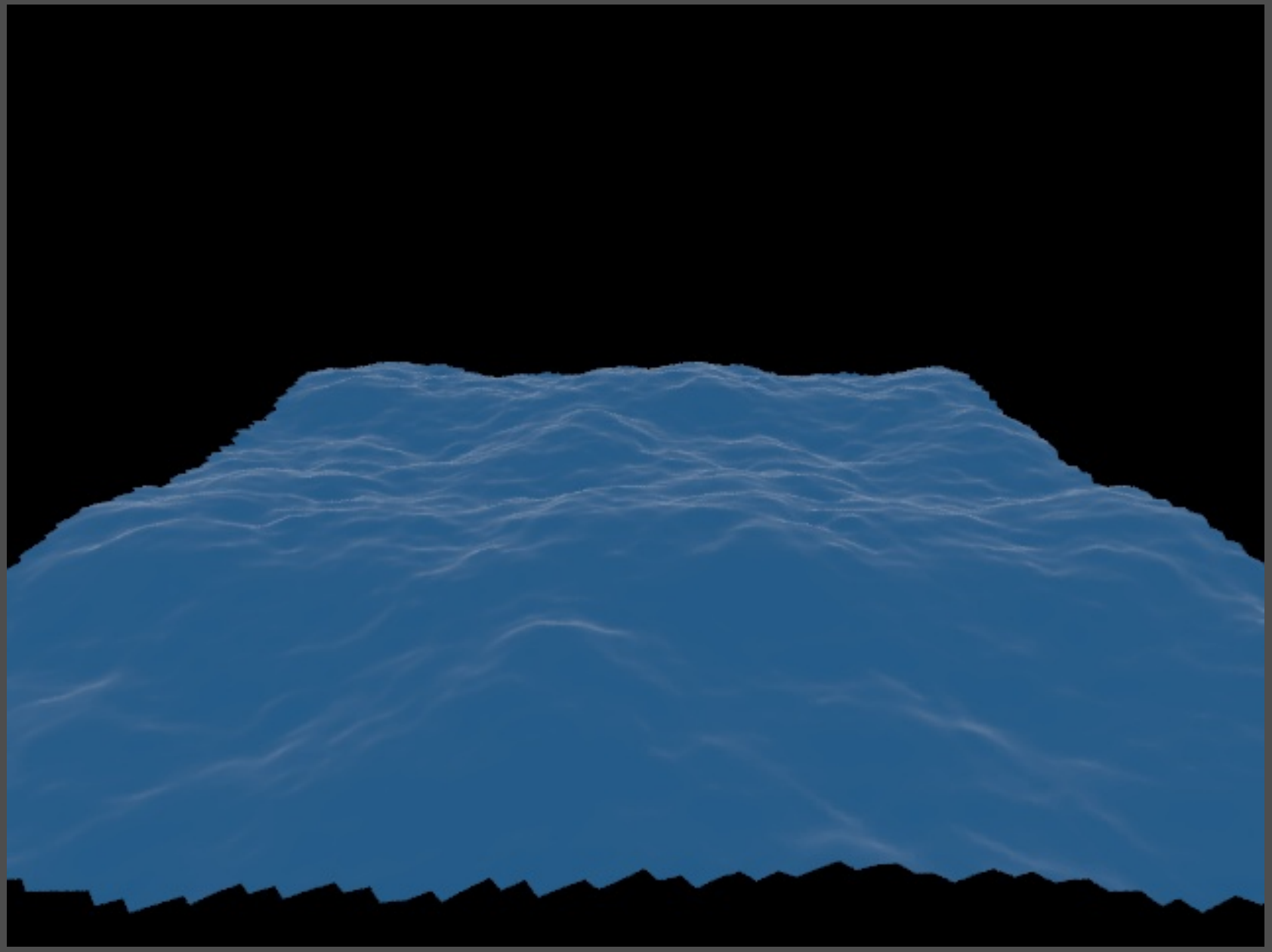


- Barycentric interpolation on periodic Delauney triangulation

Periodic Tiling



Periodic Tiling



Breaking the Curse of Dimension

- Point set $P_N = \{x_0, \ldots, x_{N-1}\}$
- **Monte Carlo Integration:** Random points P_N

$$\textbf{Prob}\left(\left\{\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| < \frac{3}{\sqrt{N}}\sigma(f)\right\}\right) \approx 0.997$$

  - slow
  - **cheap error estimate**
  - **easy math for L^2**
- **Quasi-Monte Carlo Integration:** Quasi-Monte Carlo points P_N

$$\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| < D^*(P_N)V(f)$$

  - **fast**
  - no error estimate
  - heavy math for BV

Breaking the Curse of Dimension

- Point set $P_N = \{x_0, \ldots, x_{N-1}\}$
- **Monte Carlo Integration:** Random points P_N

$$\textbf{Prob}\left(\left\{\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| < \frac{3}{\sqrt{N}}\sigma(f)\right\}\right) \approx 0.997$$

  - slow
  - **cheap error estimate**
  - **easy math for L^2**
- **Quasi-Monte Carlo Integration:** Quasi-Monte Carlo points P_N

$$\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| < D^*(P_N)V(f)$$

  - **fast**
  - no error estimate
  - heavy math for BV
- Combine and take the best !
- Price: A little bit of convergence, problems of random number generators

Randomized Quasi-Monte Carlo Integration

- Randomized replications of a QMC point set $A := \{A_0, \ldots, A_{n-1}\}$

  $X_k := \{X_{k,0}, \ldots, X_{k,n-1}\}$ for $1 \leq k \leq r$

  such that

  1. **Uniformity:** $X_{k,i} \sim U[0,1)^s$ for fixed i
  2. **Equidistribution:** $X_1, \ldots, X_r$ are low-discrepancy point sets with probability one

Randomized Quasi-Monte Carlo Integration

- Randomized replications of a QMC point set $A := \{A_0, \ldots, A_{n-1}\}$

  $X_k := \{X_{k,0}, \ldots, X_{k,n-1}\}$ for $1 \leq k \leq r$

  such that

  1. **Uniformity:** $X_{k,i} \sim U[0,1)^s$ for fixed i
  2. **Equidistribution:** $X_1, \ldots, X_r$ are low-discrepancy point sets with probability one

- Monte Carlo estimate

$$I_{r,n} f := \frac{1}{r} \sum_{k=1}^{r} \frac{1}{n} \sum_{i=0}^{n-1} f(X_{k,i})$$

Randomized Quasi-Monte Carlo Integration

- Randomized replications of a QMC point set $A := \{A_0, \ldots, A_{n-1}\}$

  $X_k := \{X_{k,0}, \ldots, X_{k,n-1}\}$ for $1 \leq k \leq r$

  such that

  1. **Uniformity:** $X_{k,i} \sim U[0,1)^s$ for fixed i
  2. **Equidistribution:** $X_1, \ldots, X_r$ are low-discrepancy point sets with probability one

- Monte Carlo estimate

  $$I_{r,n}f := \frac{1}{r}\sum_{k=1}^{r}\frac{1}{n}\sum_{i=0}^{n-1} f(X_{k,i})$$

  with error estimate

  $$\sigma^2(I_{r,n}f) \approx \frac{1}{r(r-1)}\sum_{k=1}^{r}\left(\frac{1}{n}\sum_{i=0}^{n-1} f(X_{k,i}) - I_{r,n}f\right)^2$$

Randomized Quasi-Monte Carlo Integration

- Randomized replications of a QMC point set $A := \{A_0, \ldots, A_{n-1}\}$

  $X_k := \{X_{k,0}, \ldots, X_{k,n-1}\}$ for $1 \le k \le r$

  such that

  1. **Uniformity:** $X_{k,i} \sim U[0,1)^s$ for fixed i
  2. **Equidistribution:** $X_1, \ldots, X_r$ are low-discrepancy point sets with probability one

- Monte Carlo estimate

  $$I_{r,n}f := \frac{1}{r}\sum_{k=1}^{r}\frac{1}{n}\sum_{i=0}^{n-1} f(X_{k,i})$$

  with error estimate

  $$\sigma^2(I_{r,n}f) \approx \frac{1}{r(r-1)}\sum_{k=1}^{r}\left(\frac{1}{n}\sum_{i=0}^{n-1} f(X_{k,i}) - I_{r,n}f\right)^2$$

- Presmoothing of the integrand by correlated sampling

Randomized Replications

- Random bijections

$$R_\omega : I^s \to I^s$$

  - in fact dependent sampling replication heuristics

Randomized Replications

- Random bijections

$$R_\omega : I^s \to I^s$$

  - in fact dependent sampling replication heuristics

- Cranley-Patterson rotations
  - originally designed for error estimation with lattice points
  - very simple

Randomized Replications

- Random bijections

  $$R_\omega : I^s \to I^s$$

  – in fact dependent sampling replication heuristics

- Cranley-Patterson rotations
  – originally designed for error estimation with lattice points
  – very simple

- Owen-Scrambling
  – designed for (t, m, s)-nets and (t, s)-sequences in base b
  – advanced

Randomized Replications by Cranley-Patterson Rotations

- Random shifts on the torus I^s applied to A

$$X_{k,i}^{(j)} := A_i^{(j)} + U_k^{(j)} \bmod 1 \text{ for } 1 \leq j \leq s$$

Randomized Replications by Cranley-Patterson Rotations

- Random shifts on the torus I^s applied to A

$$X_{k,i}^{(j)} := A_i^{(j)} + U_k^{(j)} \bmod 1 \text{ for } 1 \leq j \leq s$$

- Originally A was a lattice of low discrepancy

- **Note:** Cranley-Patterson rotations work with any arbitrary point set A
  - still unbiased Monte Carlo scheme

Randomized Replications by Cranley-Patterson Rotations

- Random shifts on the torus I^s applied to A

$$X_{k,i}^{(j)} := A_i^{(j)} + U_k^{(j)} \bmod 1 \text{ for } 1 \leq j \leq s$$

- Originally A was a lattice of low discrepancy

- **Note:** Cranley-Patterson rotations work with any arbitrary point set A
  - still unbiased Monte Carlo scheme
  - especially for (t, s)-sequences and (t, m, s)-nets
    - however discrepancy can be affected due to shifting

Randomized Replications by Cranley-Patterson Rotations

- Random shifts on the torus I^s applied to A

$$X_{k,i}^{(j)} := A_i^{(j)} + U_k^{(j)} \bmod 1 \text{ for } 1 \leq j \leq s$$

- Originally A was a lattice of low discrepancy

- **Note:** Cranley-Patterson rotations work with any arbitrary point set A
  - still unbiased Monte Carlo scheme
  - especially for (t, s)-sequences and (t, m, s)-nets
    - however discrepancy can be affected due to shifting
  - example: Padded replications sampling
    - pad A by low dimensional point sets, apply random shifts
    - exploit problem structure, e.g. in transport problems
    - cheaper point sets than quasi-Monte Carlo points in high dimensions

Randomized Replications by Owen-Scrambling

- Scramble (t, m, s)-nets and (t, s)-sequences in base b
- Algorithm: Start with $H = I^s$ and for each axis
  1. slice H into b equally sized volumes $H_1, H_2, \ldots, H_b$ along the axis
  2. randomly permute these volume
  3. for each H_h recursively repeat the procedure with $H = H_h$

Randomized Replications by Owen-Scrambling

- Scramble (t, m, s)-nets and (t, s)-sequences in base b
- Algorithm: Start with $H = I^s$ and for each axis
  1. slice H into b equally sized volumes $H_1, H_2, \ldots, H_b$ along the axis
  2. randomly permute these volume
  3. for each H_h recursively repeat the procedure with $H = H_h$
- Algorithm gets finite by finite precision of computation, i.e. digital constructions
- Net and sequence parameters remain untouched
  - contrary to random shifts by Cranley-Patterson

Randomized Replications by Owen-Scrambling

- Scramble (t, m, s)-nets and (t, s)-sequences in base b
- Algorithm: Start with $H = I^s$ and for each axis
  1. slice H into b equally sized volumes $H_1, H_2, \ldots, H_b$ along the axis
  2. randomly permute these volume
  3. for each H_h recursively repeat the procedure with $H = H_h$
- Algorithm gets finite by finite precision of computation, i.e. digital constructions
- Net and sequence parameters remain untouched
  - contrary to random shifts by Cranley-Patterson
- Much faster convergence for $N > s^s$

$$\mathcal{O}\left(\frac{\log^{\frac{s-1}{2}} N}{N^{\frac{3}{2}}}\right)$$

due to extinction effects by full stratification

Replication by Scrambling

- Unit square $[0, 1)^2$

Replication by Scrambling

- Bit 1 of x

Replication by Scrambling

- Bit 2 of x

Replication by Scrambling

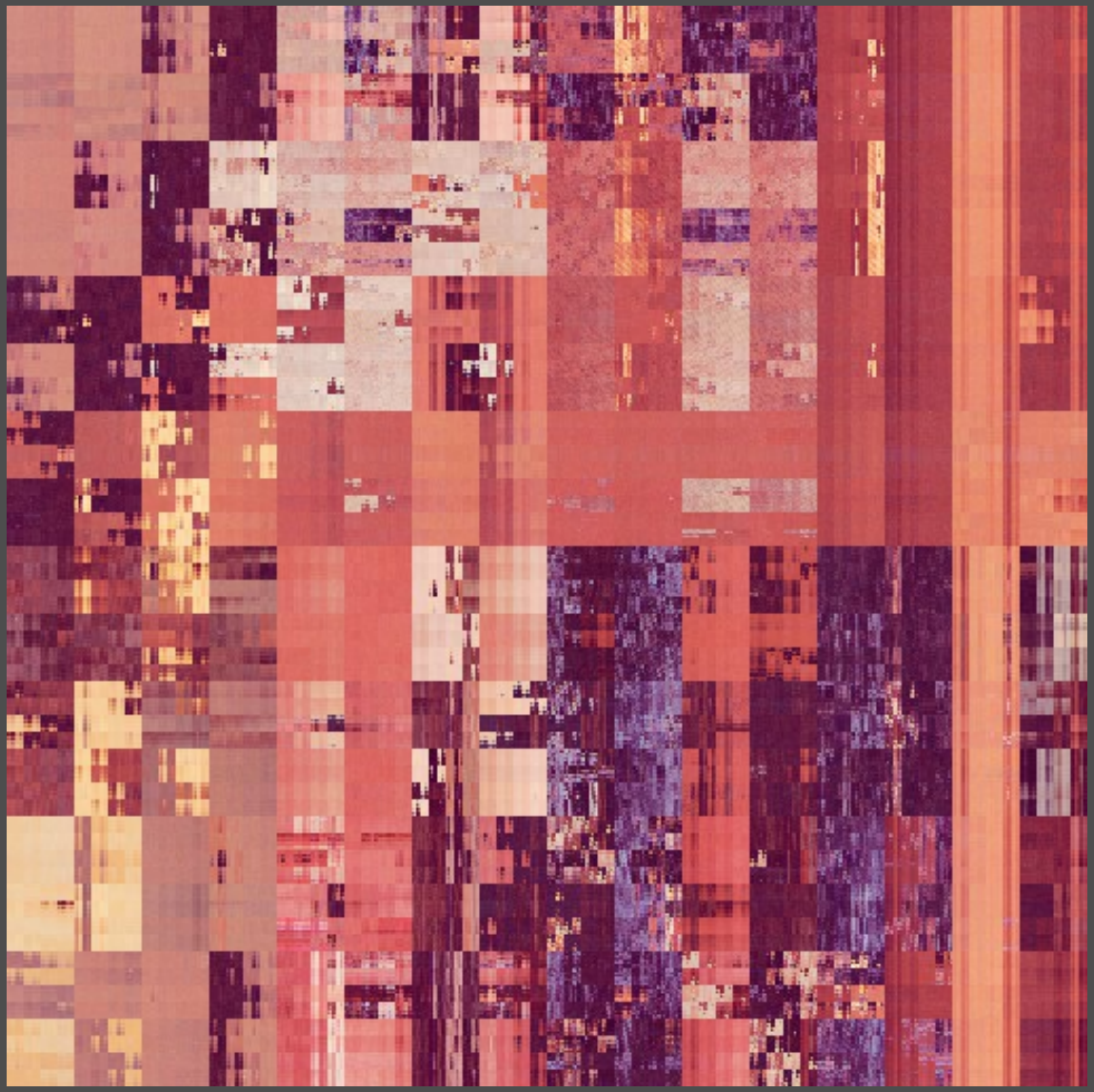
- Bit 3 of x

Replication by Scrambling

- All bits of x

Replication by Scrambling

- All bits of x and y



Formalization of Scrambling

- Given a digital (t, m, s)-net $A = \{A_0, \ldots, A_{N-1}\}$ in base b with components

$$A_i^{(j)} = \sum_{k=1}^{M} a_{i,k}^{(j)} \cdot b^{-k} =_b 0.a_{i,1}^{(j)} a_{i,2}^{(j)} \ldots a_{i,M}^{(j)}$$

Formalization of Scrambling

- Given a digital (t, m, s)-net $A = \{A_0, \ldots, A_{N-1}\}$ in base b with components

$$A_i^{(j)} = \sum_{k=1}^{M} a_{i,k}^{(j)} \cdot b^{-k} =_b 0.a_{i,1}^{(j)} a_{i,2}^{(j)} \ldots a_{i,M}^{(j)}$$

- A **scrambled replicate $\mathbf{X}$ of $\mathbf{A}$** is obtained by

$$X_i^{(j)} = \sum_{k=1}^{M} x_{i,k}^{(j)} \cdot b^{-k} =_b 0.x_{i,1}^{(j)} x_{i,2}^{(j)} x_{i,3}^{(j)} \cdots x_{i,M}^{(j)}$$

Formalization of Scrambling

- Given a digital (t, m, s)-net $A = \{A_0, \ldots, A_{N-1}\}$ in base b with components

$$A_i^{(j)} = \sum_{k=1}^{M} a_{i,k}^{(j)} \cdot b^{-k} =_b 0.a_{i,1}^{(j)} a_{i,2}^{(j)} \ldots a_{i,M}^{(j)}$$

- A **scrambled replicate $\mathbf{X}$ of $\mathbf{A}$** is obtained by

$$X_i^{(j)} = \sum_{k=1}^{M} x_{i,k}^{(j)} \cdot b^{-k} =_b 0.x_{i,1}^{(j)} x_{i,2}^{(j)} x_{i,3}^{(j)} \cdots x_{i,M}^{(j)}$$

where

$$\begin{aligned}
x_{i,1}^{(j)} &:= \pi^{(j)}\left(a_{i,1}^{(j)}\right) \\
x_{i,2}^{(j)} &:= \pi^{(j)}_{a_{i,1}^{(j)}}\left(a_{i,2}^{(j)}\right) \\
&\vdots \\
x_{i,M}^{(j)} &:= \pi^{(j)}_{a_{i,1}^{(j)}, a_{i,2}^{(j)}, \ldots, a_{i,M-1}^{(j)}}\left(a_{i,M}^{(j)}\right)
\end{aligned}$$

- Independent random permutations $\pi^{(j)} \in S_b$
- Permutation depends on the $k-1$ leading digits of $A_i^{(j)} \Rightarrow$ permutation tree

Efficient Implementation of Scrambling

- **Main ideas for efficient scrambling:**
  - keep only one path of the permutation tree in memory
  - traverse permutation tree paths that way, that each permutation is used only once

Efficient Implementation of Scrambling

- **Main ideas for efficient scrambling:**
  - keep only one path of the permutation tree in memory
  - traverse permutation tree paths that way, that each permutation is used only once

- Implies reordering of the points that should be scrambled
  - sorting the components

$$A^{(j)} = \{A_0^{(j)}, \ldots, A_{N-1}^{(j)}\} \rightarrow A_{\sigma_j(0)}^{(j)} \leq \ldots \leq A_{\sigma_j(N-1)}^{(j)}$$

  - in this order scramble the components

    $\Rightarrow$ each branch of the permutation tree is traversed at most once
  - undo the sorting using the inverse permutation σ_j^{-1}

Example: Scrambled $(0, m, 2)$*-Nets in Base* $b = 2$

- $N = 2^m$ points $A = \{A_0, \ldots, A_{N-1}\}$
- The components correspond to the inverse permutations $\sigma_j^{-1}(i) = N \cdot A_i^{(j)}$
  - e.g. Hammersley: $\sigma_0^{-1}(i) = 2^m \cdot \frac{i}{N}$ and $\sigma_1^{-1}(i) = 2^m \cdot \Phi_2(i)$
- Random permutations on $\mathbb{Z}_2$ are random bit flips and can be vectorized
  - i.e. applying a path of permutation means XORing the bit vector of bit permutations

Example: Scrambled $(0, m, 2)$*-Nets in Base* $b = 2$

- $N = 2^m$ points $A = \{A_0, \ldots, A_{N-1}\}$
- The components correspond to the inverse permutations $\sigma_j^{-1}(i) = N \cdot A_i^{(j)}$
  - e.g. Hammersley: $\sigma_0^{-1}(i) = 2^m \cdot \frac{i}{N}$ and $\sigma_1^{-1}(i) = 2^m \cdot \Phi_2(i)$
- Random permutations on $\mathbb{Z}_2$ are random bit flips and can be vectorized
  - i.e. applying a path of permutation means XORing the bit vector of bit permutations

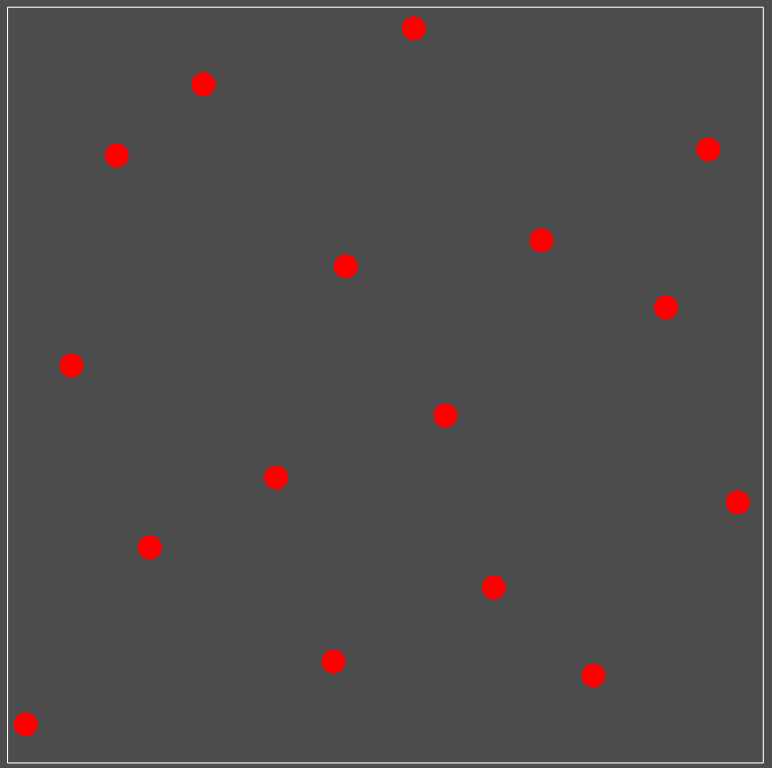
- Scrambling the component j:
  - start out with a random bit vector and save it in $X^{(j)}_{\sigma_j^{-1}(0)}$
  - permutation tree traversal by enumerating $i = 1, \ldots, 2^m - 1$
    - detect were tree ramifies: Number f of leading shared digits of $i - 1$ and i
    - XOR a bit vector with f leading zeros followed by a 1 filled by random bits
      $\equiv$ change the branch and choose new random permutations π
    - store result in $X^{(j)}_{\sigma_j^{-1}(i)}$

Implementation: Scrambled Hammersley Point Set

```
N = 1 << m;

Digits = get_32_random_bits();
P(0, 0) = (double) Digits / (double) 0x100000000L;

Digits2 = get_32_random_bits();
P(0, 1) = (double) Digits2 / (double) 0x100000000L;

for(i = 1; i < N; i++)
{
   Difference = (i - 1) ^ i;

   for(Bits = 0; Difference; Bits++)
      Difference >>= 1;

   Shift = Log - Bits;

   Digits ^= (0x80000000 | get_31_random_bits()) >> Shift;
   P(i, 0) = (double) Digits / (double) 0x100000000L;

   Digits2 ^= (0x80000000 | get_31_random_bits()) >> Shift;
   P((int) ((double) N * Φ2(i)), 1) = (double) Digits2
            / (double) 0x100000000L;
}
```

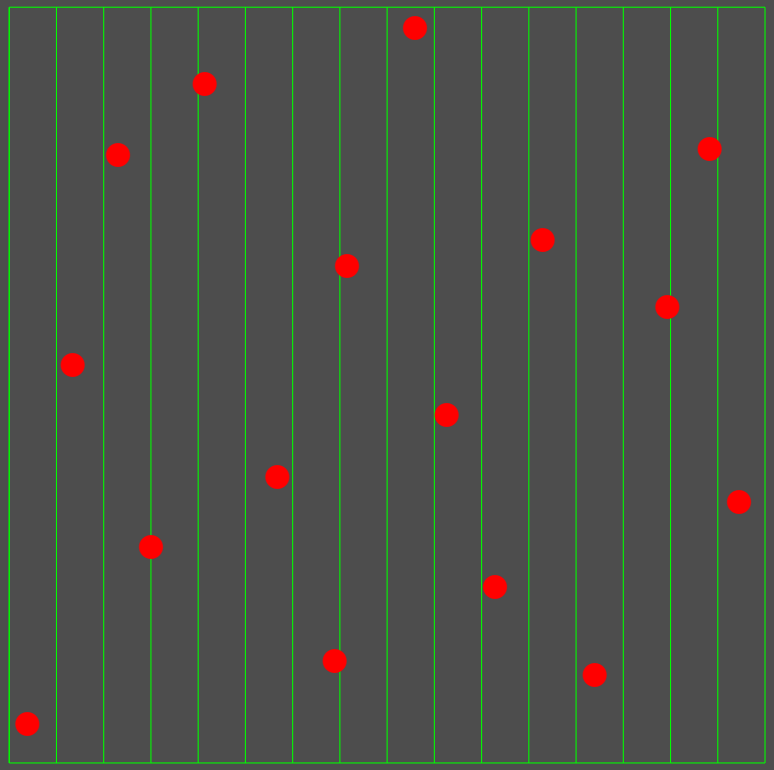
Example: Instance of a Randomly Scrambled $(0, 4, 2)$*-Net*

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...



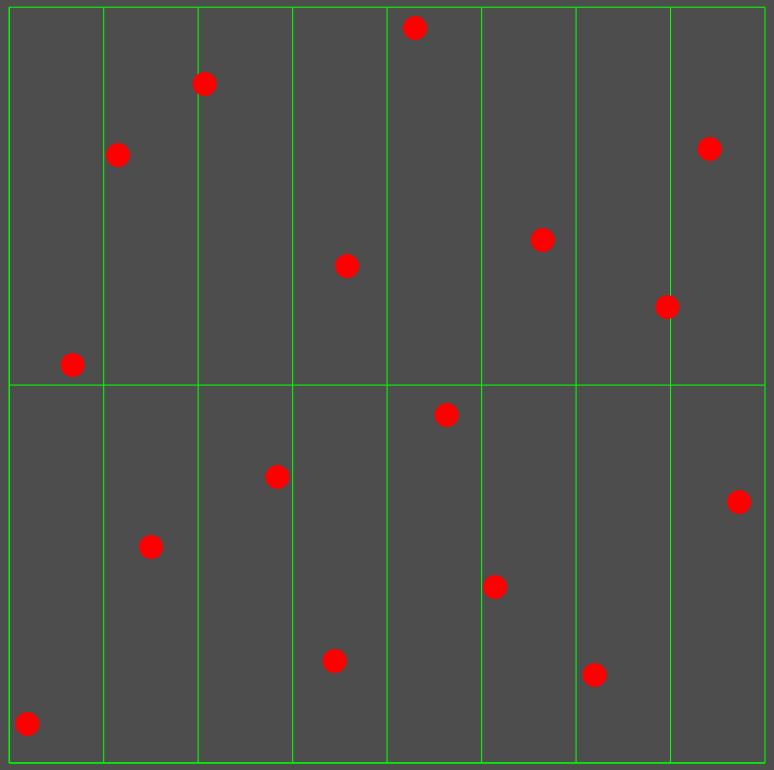
Example: Instance of a Randomly Scrambled $(0, 4, 2)$*-Net*

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...



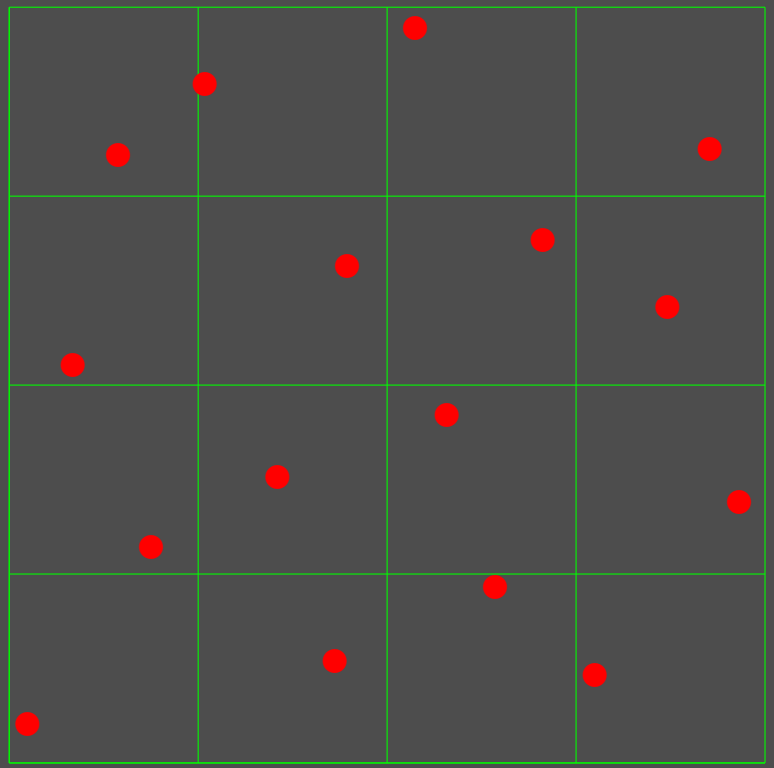
Example: Instance of a Randomly Scrambled $(0, 4, 2)$*-Net*

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...



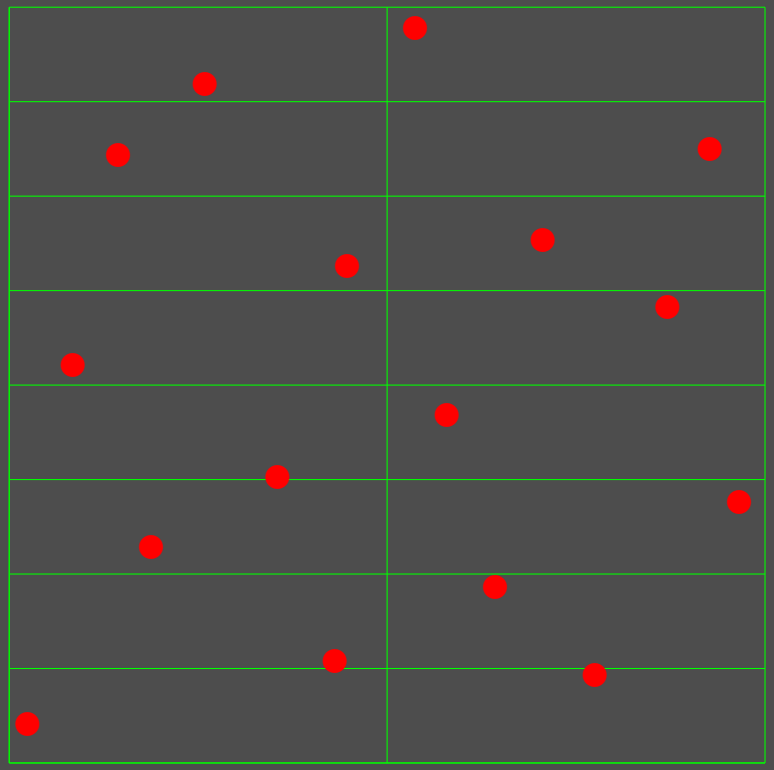
Example: Instance of a Randomly Scrambled $(0, 4, 2)$*-Net*

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...



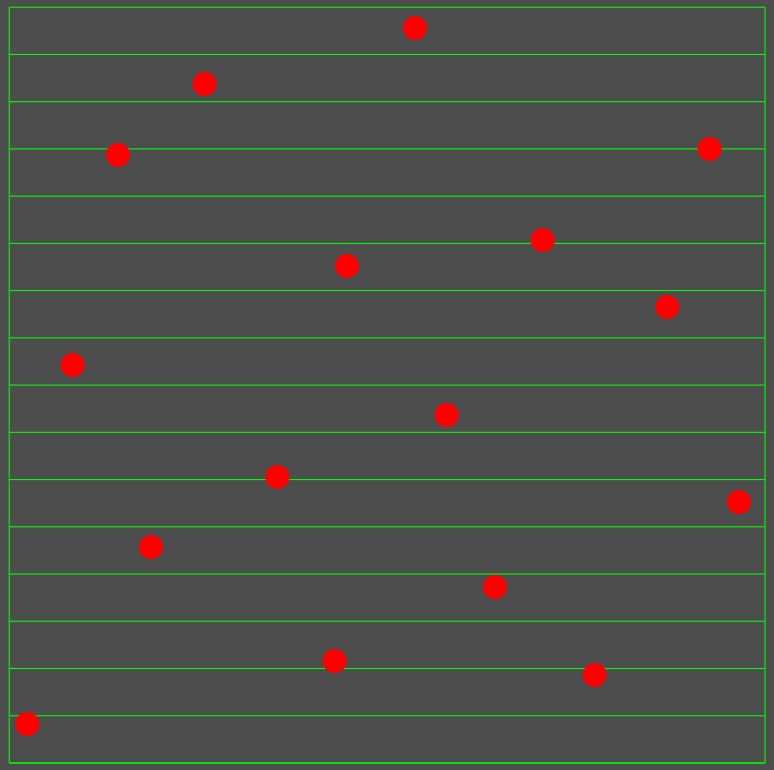
Example: Instance of a Randomly Scrambled $(0, 4, 2)$*-Net*

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...



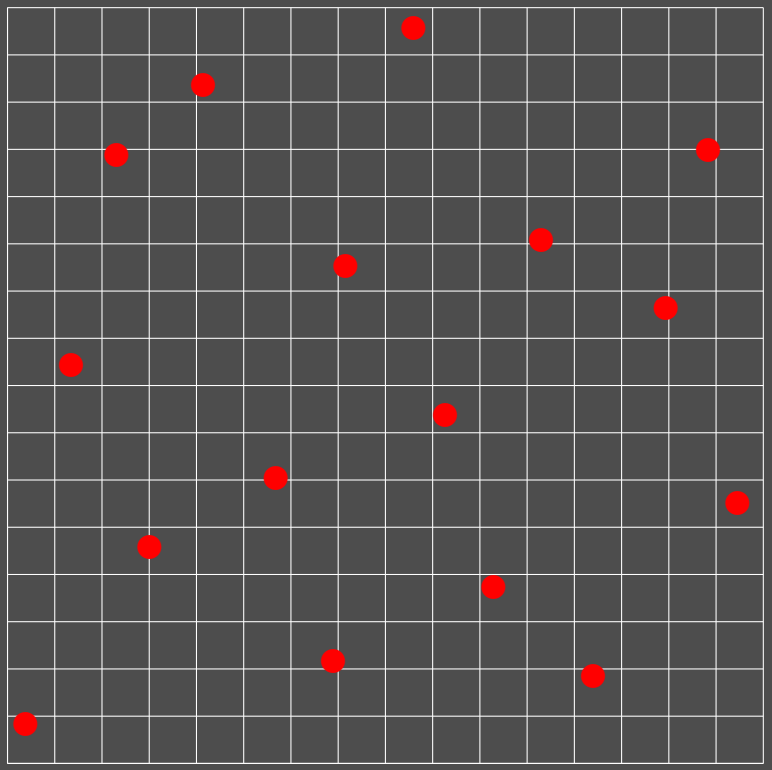
Example: Instance of a Randomly Scrambled $(0, 4, 2)$*-Net*

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...



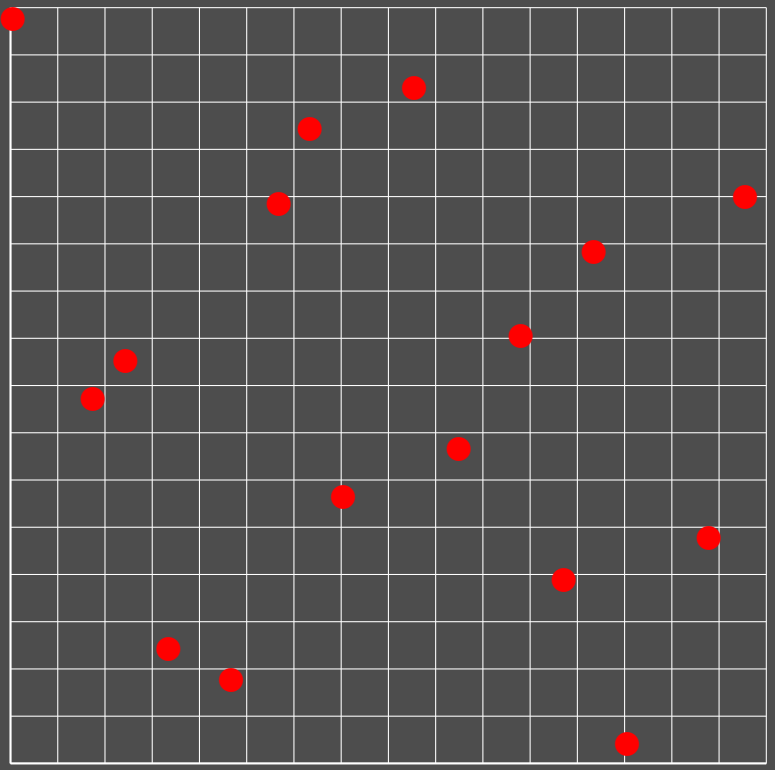
Example: Instance of a Randomly Scrambled $(0, 4, 2)$*-Net*

- All instances are of low discrepancy
- Not all instances are equally good...



Another Instance of a Randomly Scrambled $(0, 4, 2)$*-Net*

- All instances are of low discrepancy
- Not all instances are equally good...



Trajectory Splitting and Dependent Sampling

- Increase efficiency by splitting

$$\frac{1}{N}\sum_{i=0}^{N-1} f(x_i, y_i) \approx \int_{I^{s1}} \int_{I^{s2}} f(x, y) dx dy$$

Trajectory Splitting and Dependent Sampling

- Increase efficiency by splitting

$$\frac{1}{N}\sum_{i=0}^{N-1} f(x_i, y_i) \approx \int_{I^{s1}}\int_{I^{s2}} f(x,y)dxdy \approx \frac{1}{Ns}\sum_{i=0}^{N-1}\sum_{j=0}^{s-1} f(x_i, y_{i,j})$$

depending on the correlation coefficient of $f(\xi, \eta)$ and $f(\xi, \eta')$

Trajectory Splitting and Dependent Sampling

- Increase efficiency by splitting

$$\frac{1}{N}\sum_{i=0}^{N-1} f(x_i, y_i) \approx \int_{I^{s1}}\int_{I^{s2}} f(x,y)dxdy \approx \frac{1}{Ns}\sum_{i=0}^{N-1}\sum_{j=0}^{s-1} f(x_i, y_{i,j})$$

  depending on the correlation coefficient of $f(\xi, \eta)$ and $f(\xi, \eta')$

- Exploit smoothness by correlated sampling

$$\sum_{j=1}^{M}\frac{1}{N_j}\sum_{i=0}^{N_j-1} f_j(x_{i,j}) \approx \sum_{j=1}^{M}\int_{I^s} f_j(x)dx$$

Trajectory Splitting and Dependent Sampling

- Increase efficiency by splitting

$$\frac{1}{N}\sum_{i=0}^{N-1} f(x_i, y_i) \approx \int_{I^{s1}}\int_{I^{s2}} f(x,y)dxdy \approx \frac{1}{Ns}\sum_{i=0}^{N-1}\sum_{j=0}^{s-1} f(x_i, y_{i,j})$$

  depending on the correlation coefficient of $f(\xi, \eta)$ and $f(\xi, \eta')$

- Exploit smoothness by correlated sampling

$$\begin{aligned}\sum_{j=1}^{M}\frac{1}{N_j}\sum_{i=0}^{N_j-1} f_j(x_{i,j}) &\approx \sum_{j=1}^{M}\int_{I^s} f_j(x)dx \\ &= \int_{I^s}\sum_{j=1}^{M} f_j(x)dx \approx \frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=1}^{M} f_j(x_i)\end{aligned}$$

  e.g. separation of the main part

Trajectory Splitting by Dependent Sampling

- Integrals invariant under Cranley-Patterson rotation by $z_j \in I^{s2}$

$$\begin{array}{rcl} R_j : I^{s2} & \rightarrow & I^{s2} \\ y & \mapsto & (y + z_j) \bmod 1 \end{array} \quad \Rightarrow \quad \int_{I^{s2}} g(y)dy = \int_{I^{s2}} g(R_j(y))dy$$

Trajectory Splitting by Dependent Sampling

- Integrals invariant under Cranley-Patterson rotation by $z_j \in I^{s2}$

$$\begin{array}{rcl} R_j : I^{s2} & \rightarrow & I^{s2} \\ y & \mapsto & (y + z_j) \bmod 1 \end{array} \quad \Rightarrow \quad \int_{I^{s2}} g(y)dy = \int_{I^{s2}} g(R_j(y))dy$$

- Presmoothing of selected dimensions by replication

$$\int_{I^{s1}} \int_{I^{s2}} f(x,y)dydx \;=\; \int_{I^{s1}} \int_{I^{s2}} \frac{1}{M} \sum_{j=0}^{M-1} f(x, R_j(y))dydx$$

Trajectory Splitting by Dependent Sampling

- Integrals invariant under Cranley-Patterson rotation by $z_j \in I^{s2}$

$$\begin{array}{rcl} R_j : I^{s2} & \rightarrow & I^{s2} \\ y & \mapsto & (y + z_j) \bmod 1 \end{array} \quad \Rightarrow \quad \int_{I^{s2}} g(y)dy = \int_{I^{s2}} g(R_j(y))dy$$

- Presmoothing of selected dimensions by replication

$$\begin{aligned} \int_{I^{s1}} \int_{I^{s2}} f(x,y)dydx &= \int_{I^{s1}} \int_{I^{s2}} \frac{1}{M} \sum_{j=0}^{M-1} f(x, R_j(y))dydx \\ &\approx \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{M} \sum_{j=0}^{M-1} f(x_i, R_j(y_i)) \\ &= \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{M} \sum_{j=0}^{M-1} f(x_i, (y_i + z_j) \bmod 1) \end{aligned}$$

  - global quadrature rule $P_{N,s_1+s_2} = (x_i, y_i)_{i=0}^{N-1}$

Trajectory Splitting by Dependent Sampling

- Integrals invariant under Cranley-Patterson rotation by $z_j \in I^{s2}$

$$\begin{array}{rcl} R_j : I^{s2} & \rightarrow & I^{s2} \\ y & \mapsto & (y + z_j) \bmod 1 \end{array} \quad \Rightarrow \quad \int_{I^{s2}} g(y)dy = \int_{I^{s2}} g(R_j(y))dy$$

- Presmoothing of selected dimensions by replication

$$\begin{aligned} \int_{I^{s1}} \int_{I^{s2}} f(x,y)dydx &= \int_{I^{s1}} \int_{I^{s2}} \frac{1}{M} \sum_{j=0}^{M-1} f(x, R_j(y))dydx \\ &\approx \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{M} \sum_{j=0}^{M-1} f(x_i, R_j(y_i)) \\ &= \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{M} \sum_{j=0}^{M-1} f(x_i, (y_i + z_j) \bmod 1) \end{aligned}$$

  - global quadrature rule $P_{N,s_1+s_2} = (x_i, y_i)_{i=0}^{N-1}$
  - local quadrature rule $P_{M,s_2} = (z_j)_{j=0}^{M-1}$

Trajectory Splitting by Dependent Sampling

- Integrals invariant under Cranley-Patterson rotation by $z_j \in I^{s2}$

$$\begin{aligned} R_j : I^{s2} &\rightarrow I^{s2} \\ y &\mapsto (y + z_j) \bmod 1 \end{aligned} \quad \Rightarrow \quad \int_{I^{s2}} g(y)dy = \int_{I^{s2}} g(R_j(y))dy$$

- Presmoothing of selected dimensions by replication

$$\begin{aligned} \int_{I^{s1}} \int_{I^{s2}} f(x,y)dydx &= \int_{I^{s1}} \int_{I^{s2}} \frac{1}{M} \sum_{j=0}^{M-1} f(x, R_j(y))dydx \\ &\approx \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{M} \sum_{j=0}^{M-1} f(x_i, R_j(y_i)) \\ &= \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{M} \sum_{j=0}^{M-1} f(x_i, (y_i + z_j) \bmod 1) \end{aligned}$$

  - global quadrature rule $P_{N,s_1+s_2} = (x_i, y_i)_{i=0}^{N-1}$
  - local quadrature rule $P_{M,s_2} = (z_j)_{j=0}^{M-1}$

$\Rightarrow$ Trajectories split by dependent sampling

Further Randomization Techniques

- Padding quasi-Monte Carlo points for high dimensions
  - by random numbers
  - by Latin hypercube samples

Further Randomization Techniques

- Padding quasi-Monte Carlo points for high dimensions
  - by random numbers
  - by Latin hypercube samples

- Jittered quasi-Monte Carlo point sets
  - Latin hypercube samples, however deterministic permutation

    Note: Rate of randomly permuted Latin hypercube samples does not apply !
  - e.g. $(0, m, 2)$-net with jitter of size b^{-m}

Further Randomization Techniques

- Padding quasi-Monte Carlo points for high dimensions
  - by random numbers
  - by Latin hypercube samples

- Jittered quasi-Monte Carlo point sets
  - Latin hypercube samples, however deterministic permutation

    Note: Rate of randomly permuted Latin hypercube samples does not apply !
  - e.g. $(0, m, 2)$-net with jitter of size b^{-m}

- Latin supercube sampling
  - biased
  - unbiased if used for decorrelating padded replications sampling

Summary

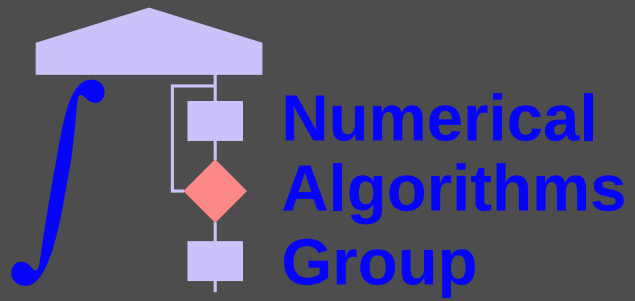
- Random field synthesis on good lattice points
- Randomized quasi-Monte Carlo integration
  - error estimate
  - L^2
  - almost as fast as pure quasi-Monte Carlo integration
  - concept of randomized replications
- Dependent splitting

Our Research

- Monte Carlo methods
- Quasi-Monte Carlo methods (***mental ray***)
- Randomized quasi-Monte Carlo methods
- Quantum complexity

Visit us at

www.uni-kl.de/AG-Heinrich



Acknowledgements

- Peter Schröder, CalTech MultiRes Group, Pasadena CA, USA
- Rolf Herken, mental images, Berlin, Germany

See you at SIGGRAPH 2001...