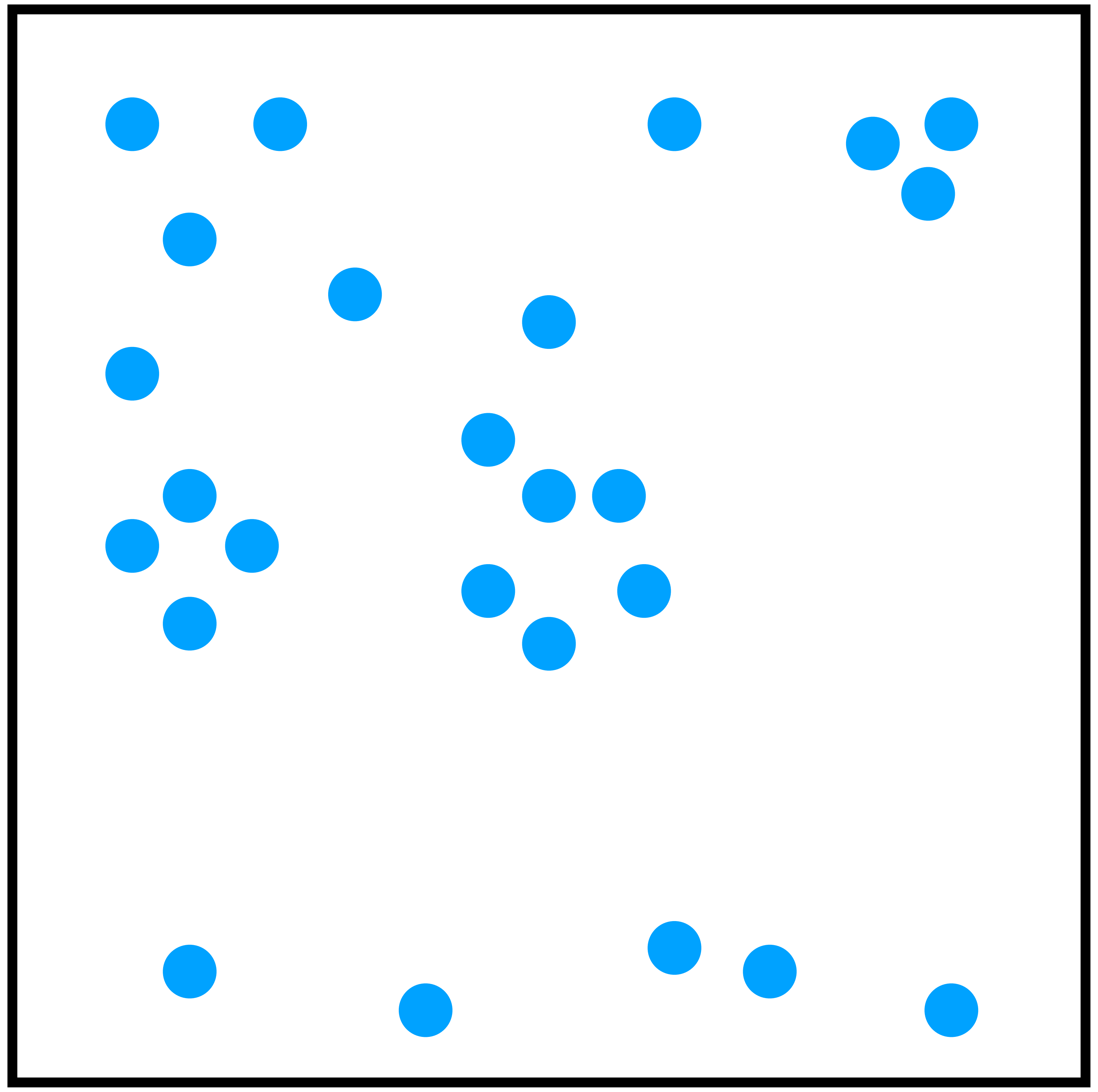
Stratification 1

UCSD CSE 272
Advanced Image Synthesis

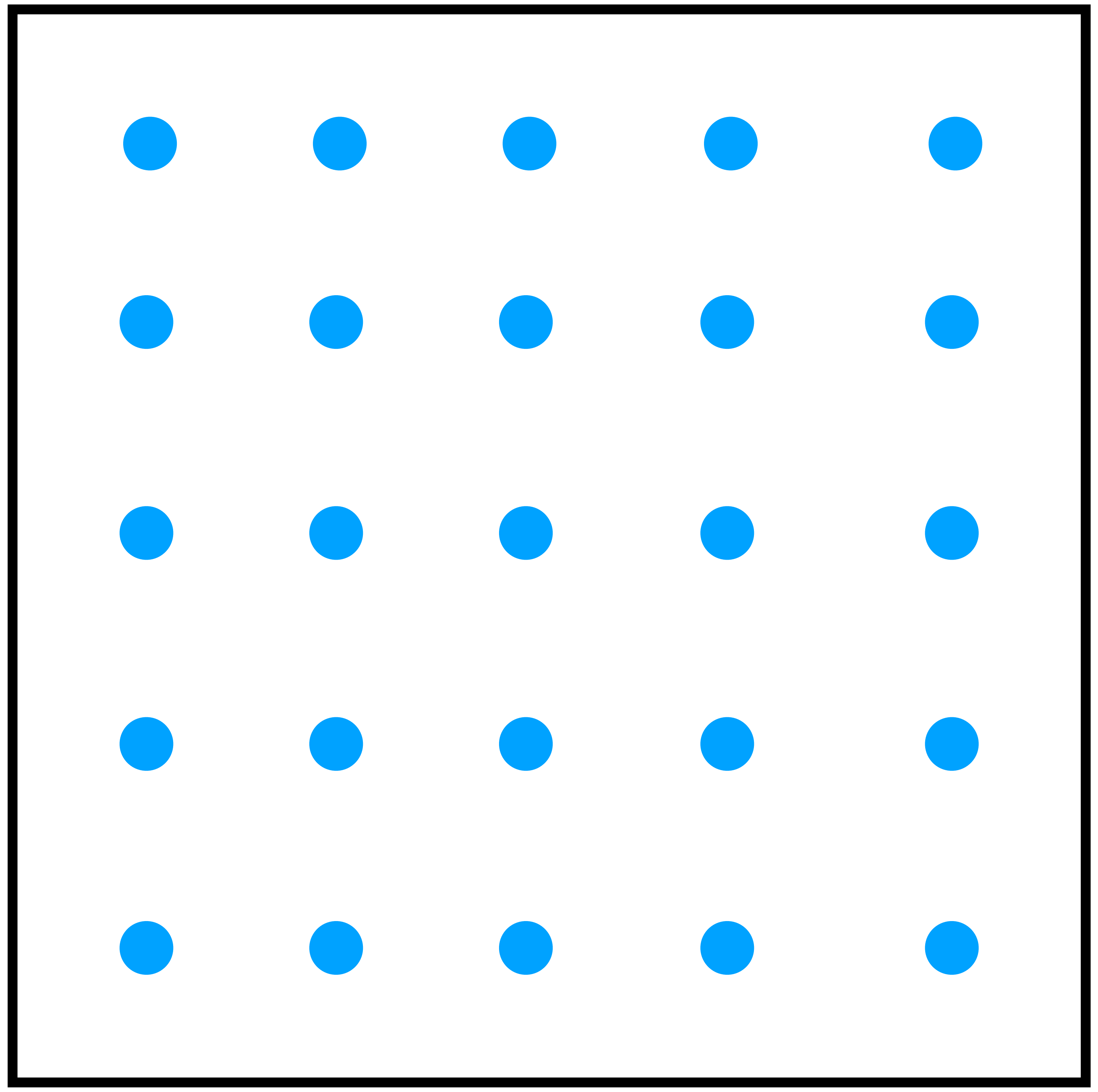
Tzu-Mao Li

with slides & images from Wojciech Jarosz & Gurprit Singh

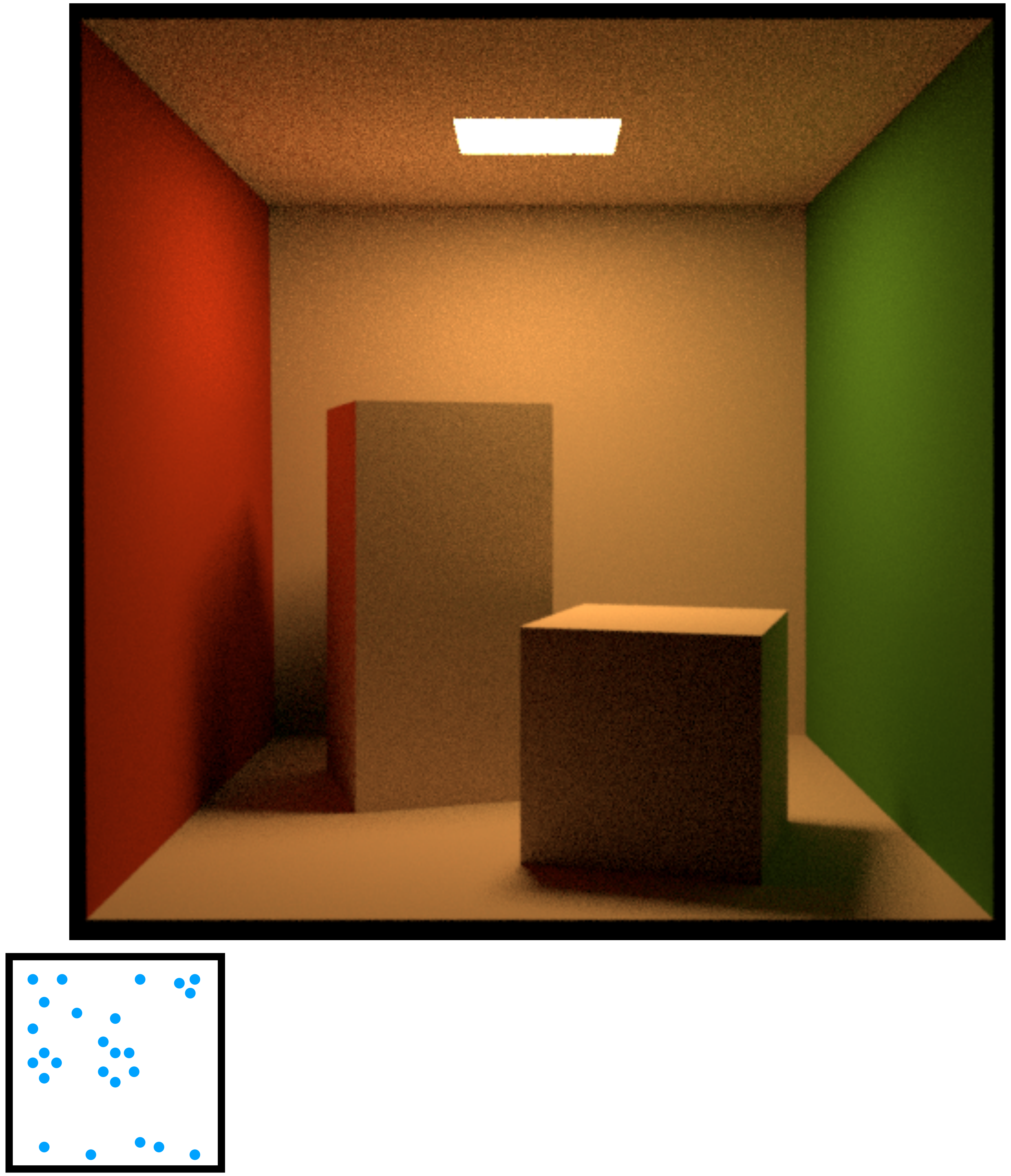
Sampling pattern matters

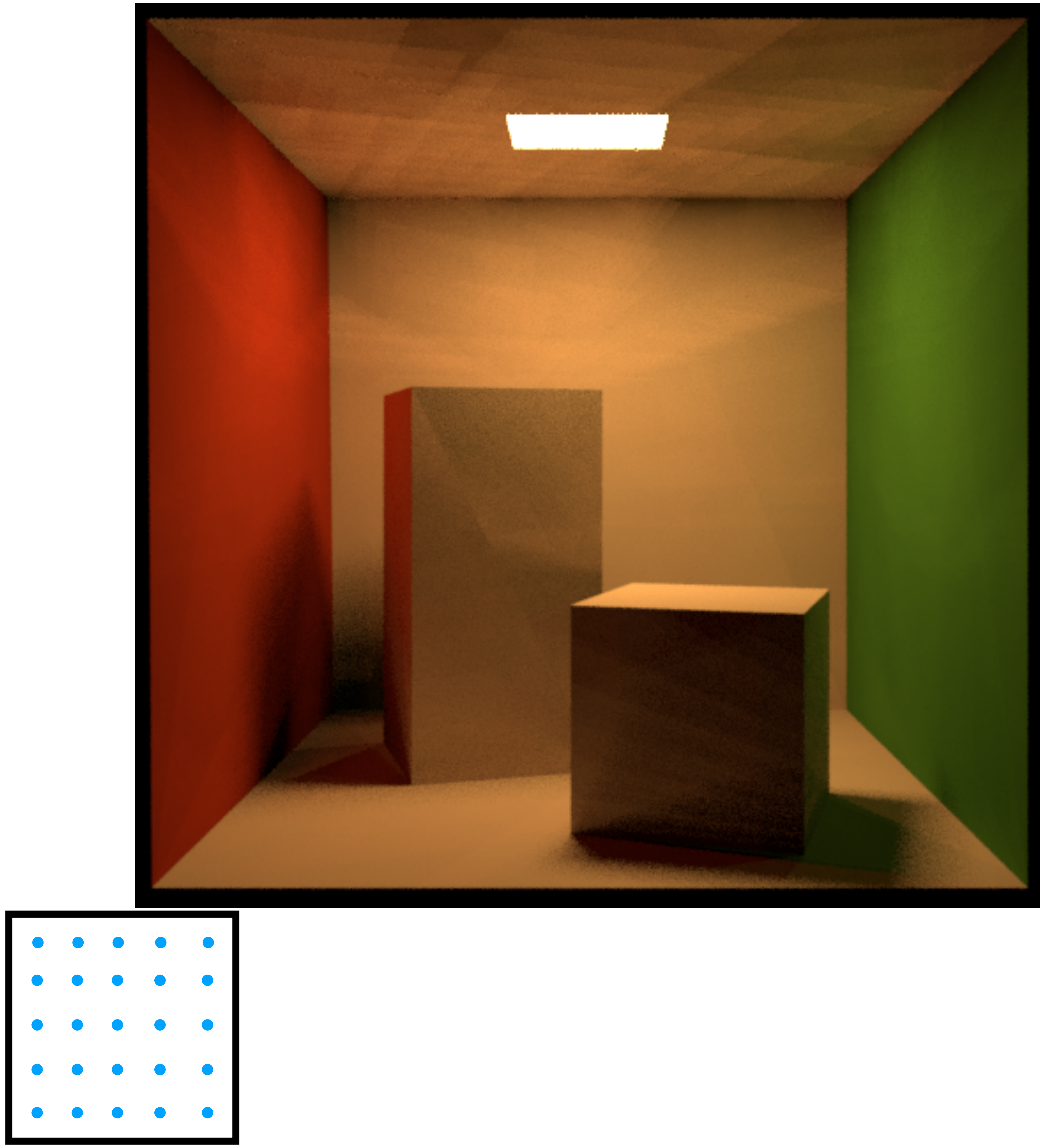


which one
is better?

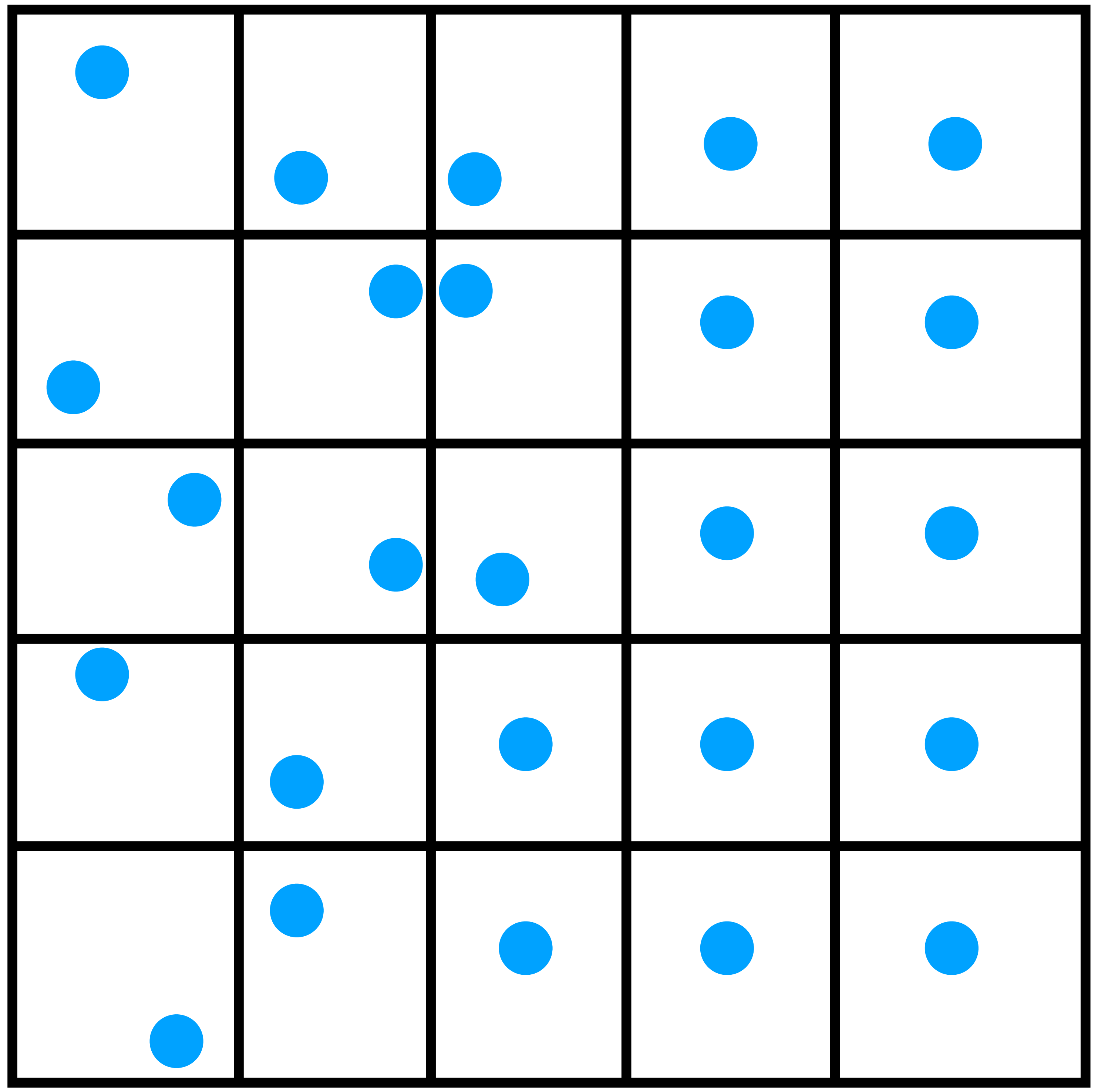


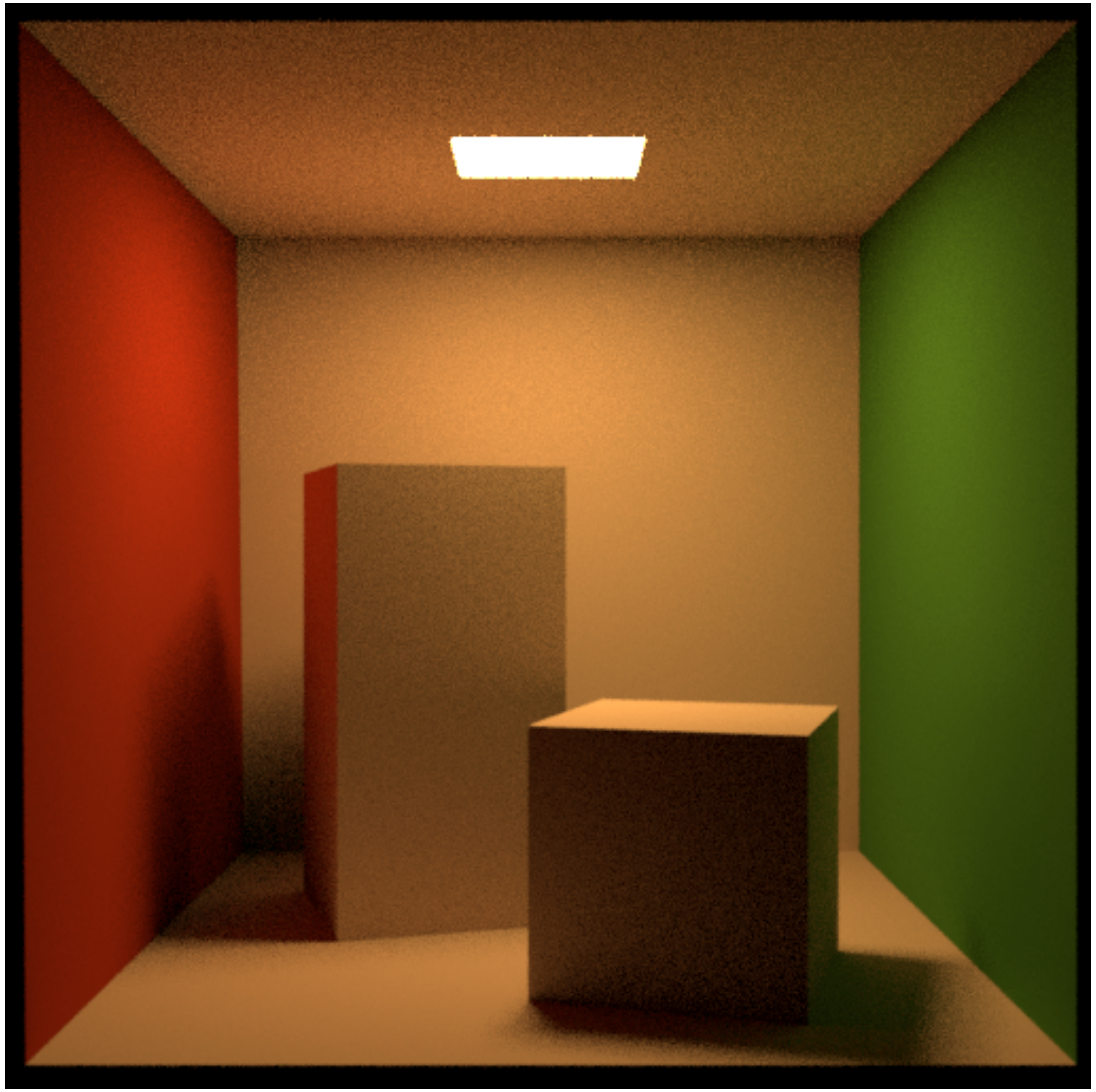
Noise v.s. aliasing trade-offs



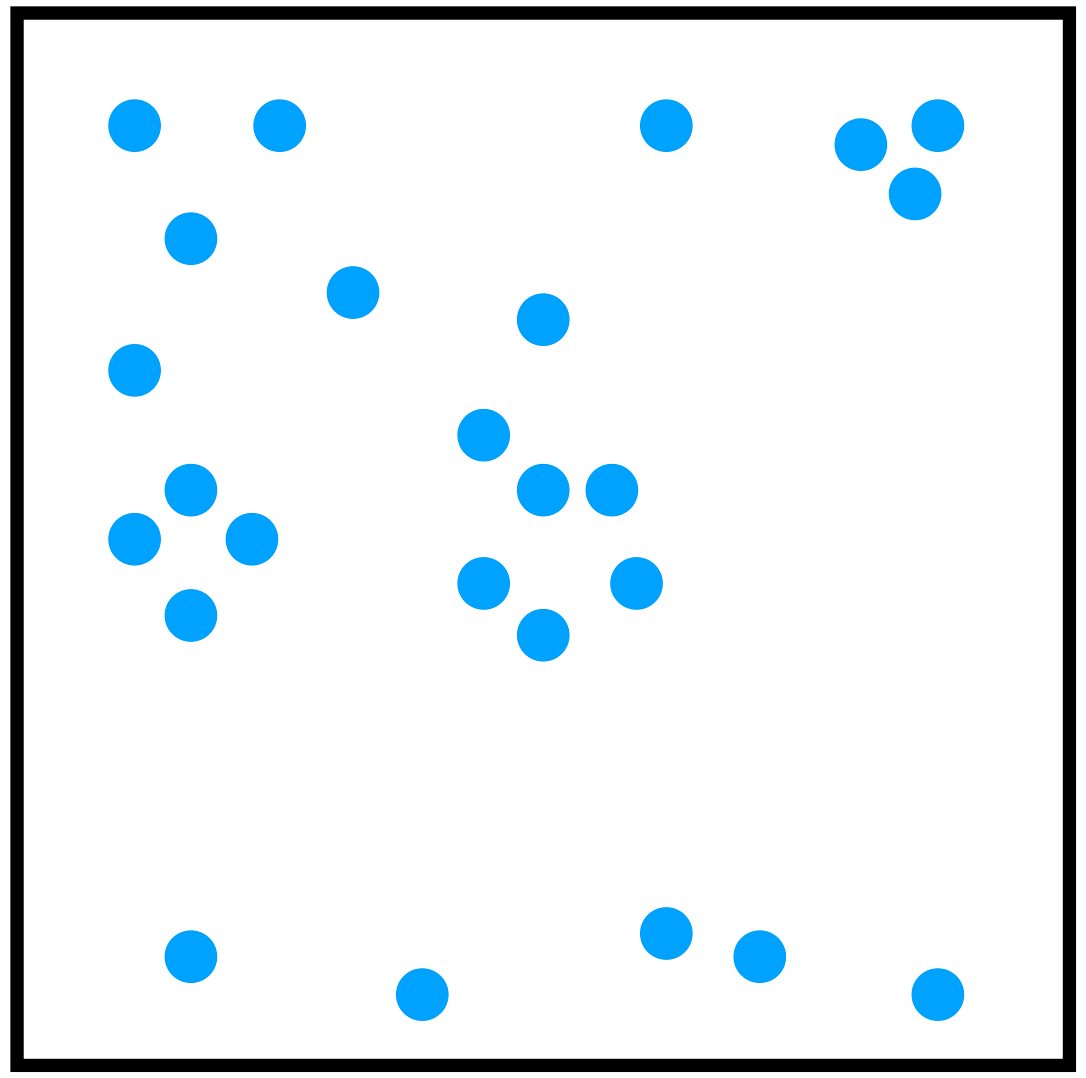


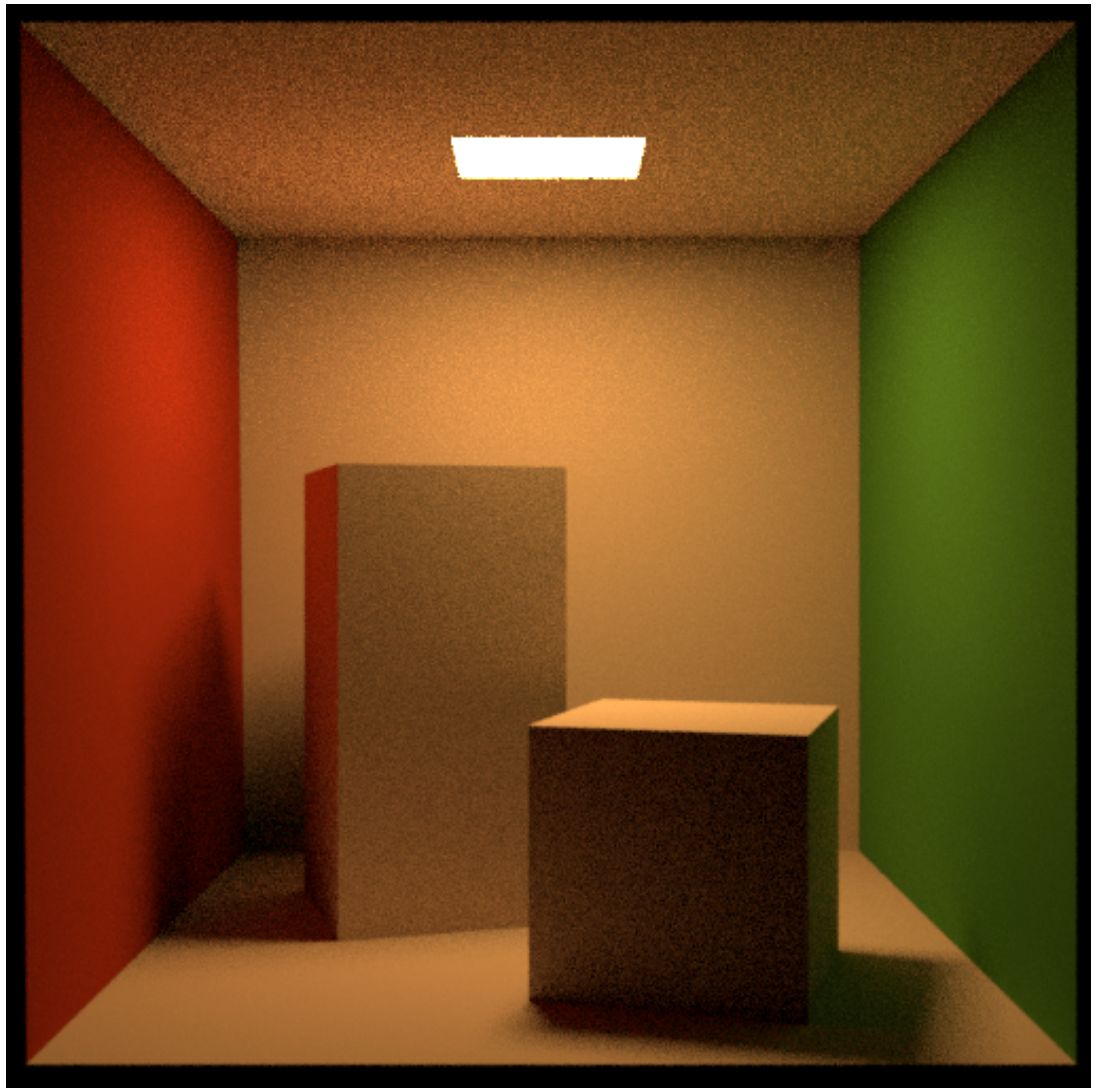
A middle ground?

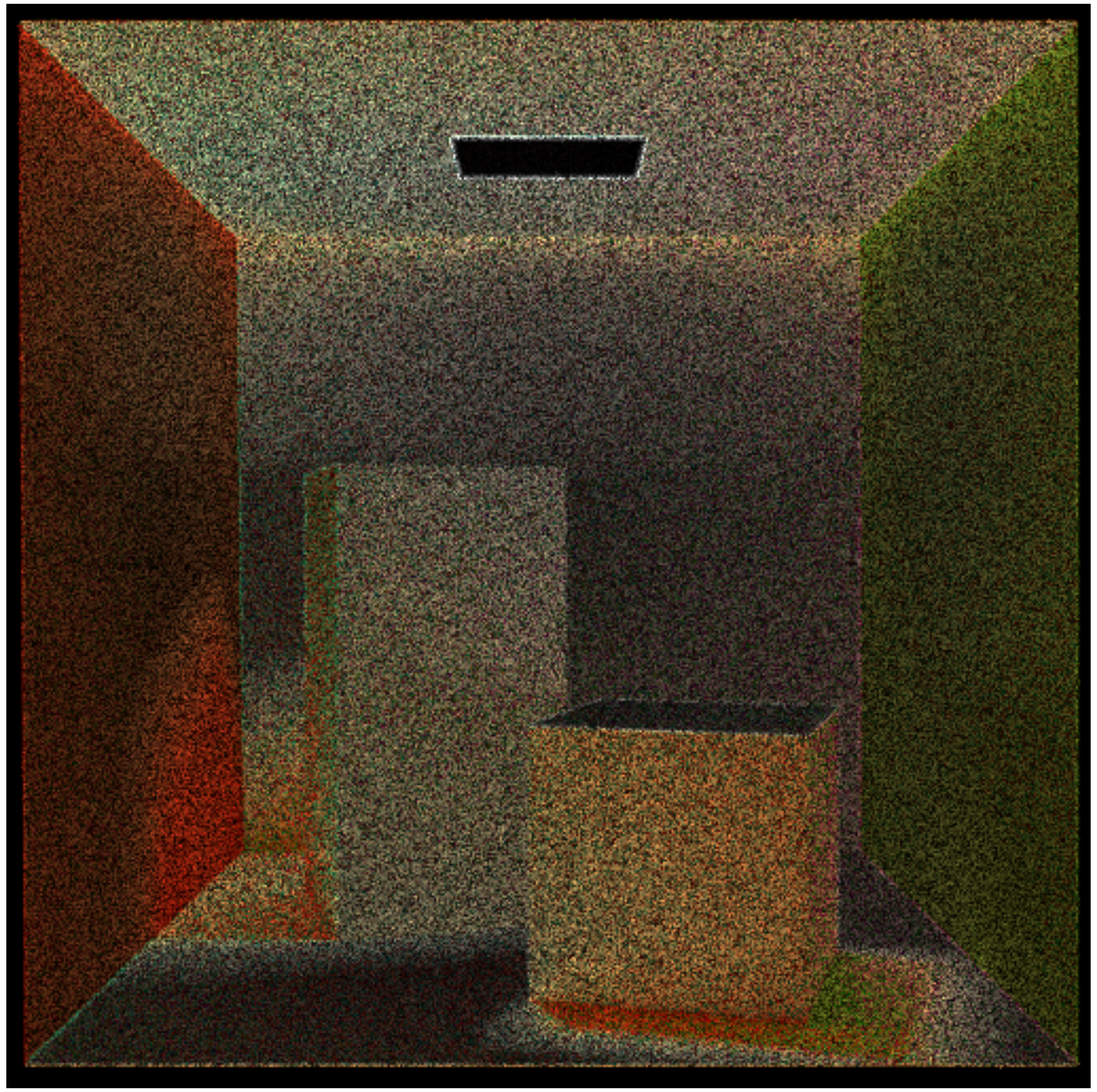




Comparison

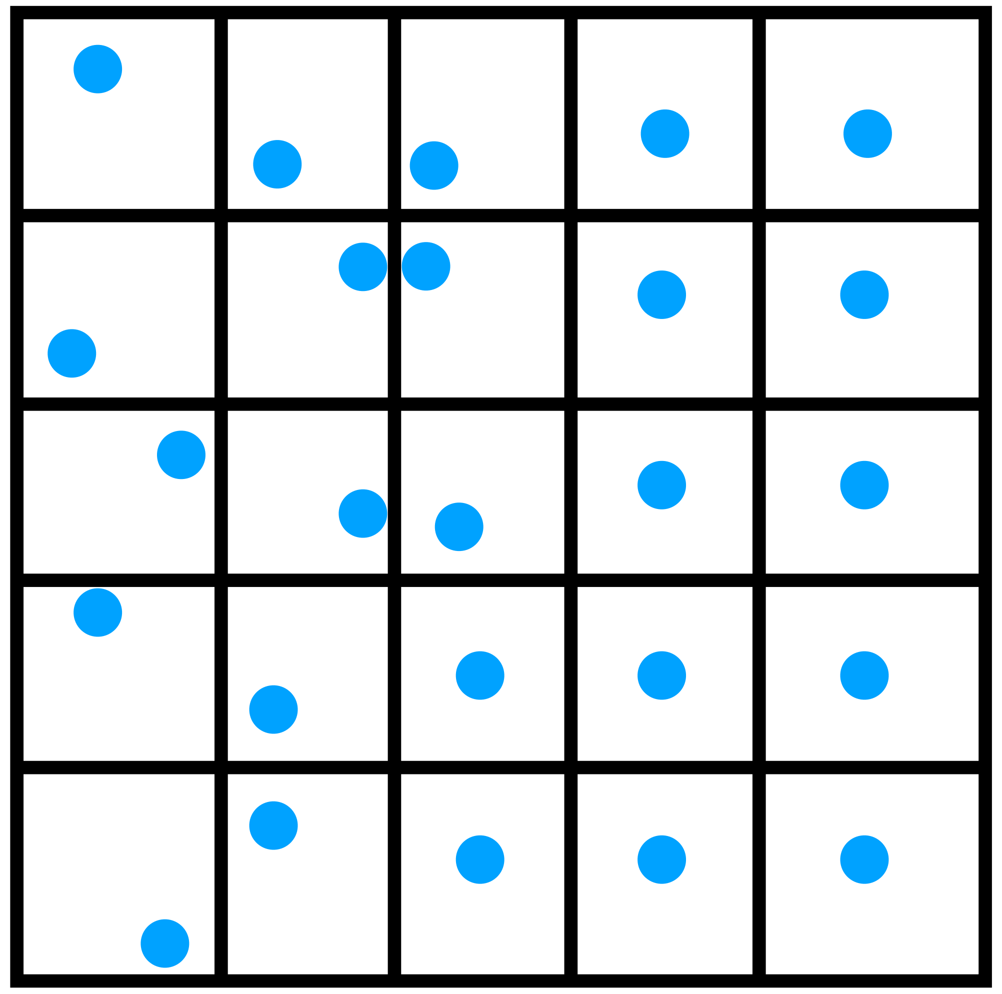


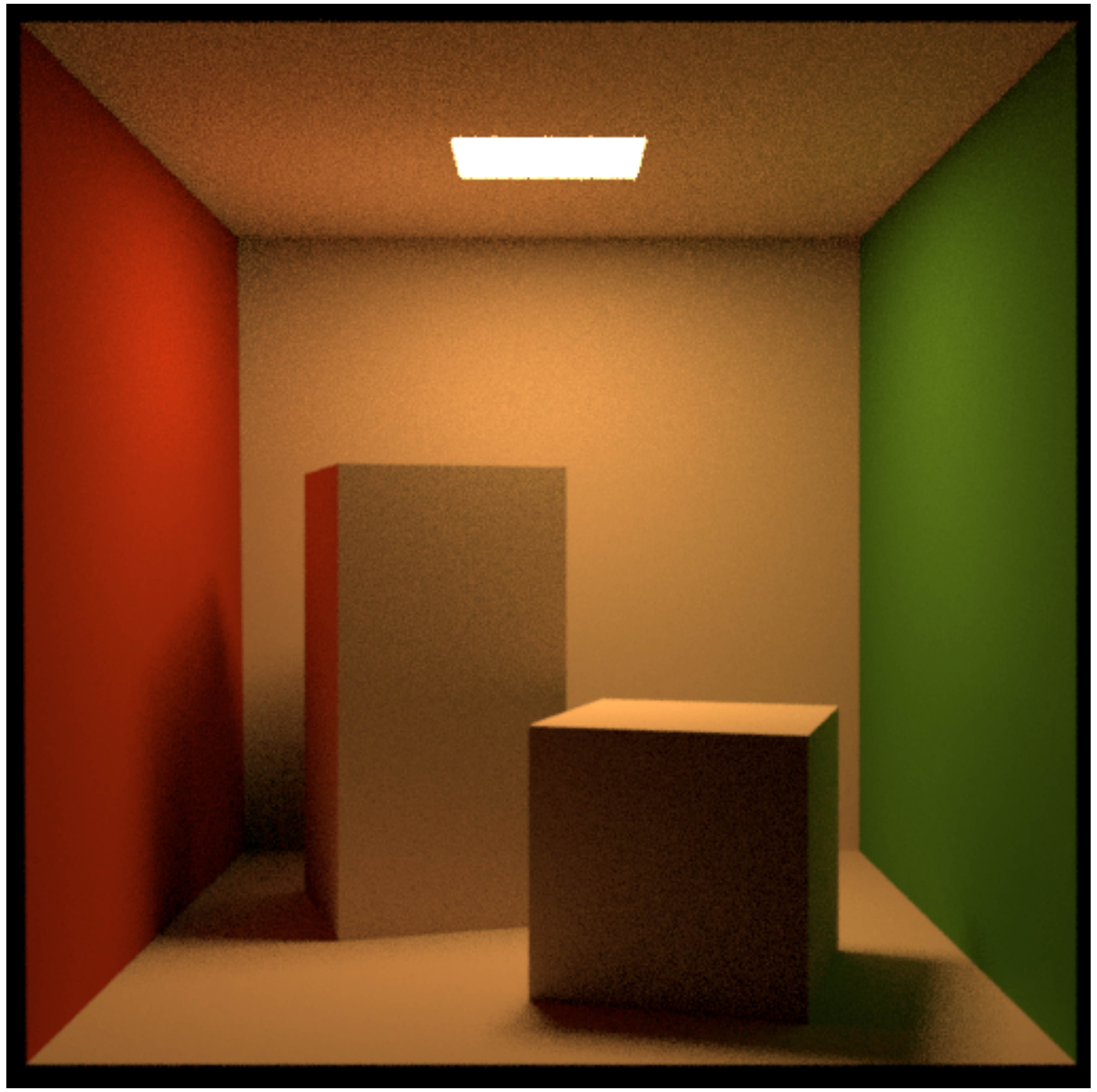


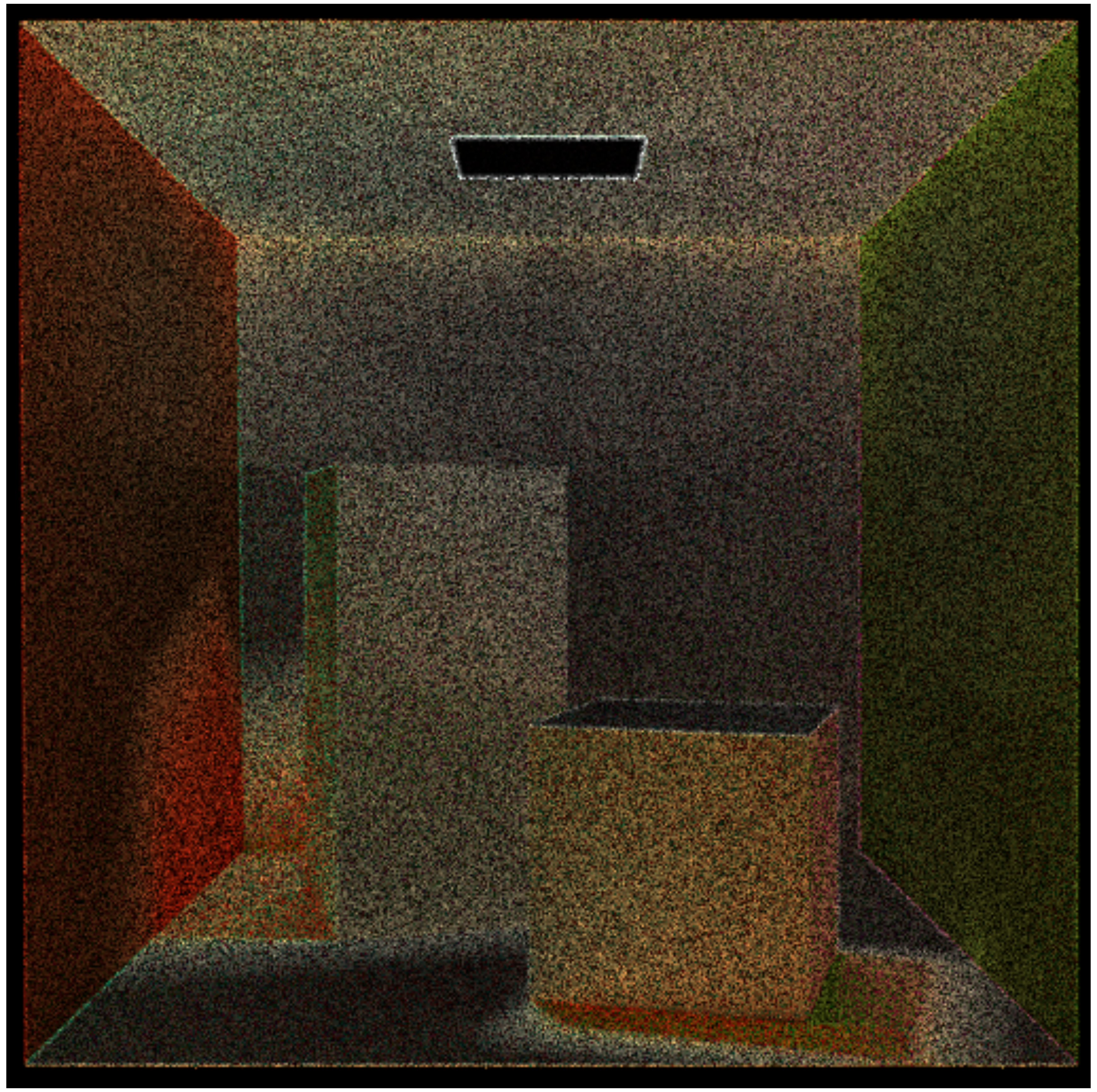


per pixel (relative) error

Comparison







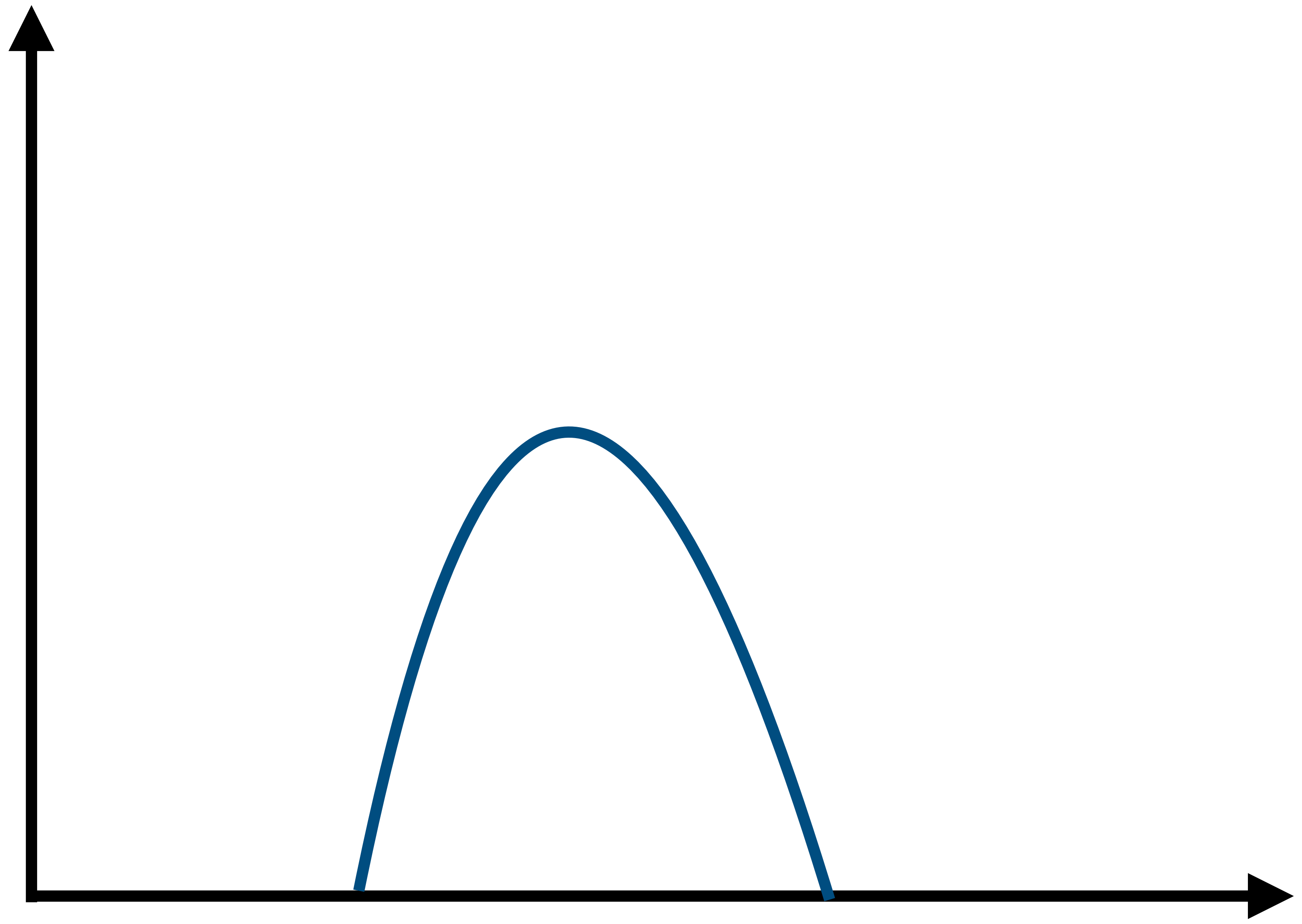
per pixel (relative) error

Questions

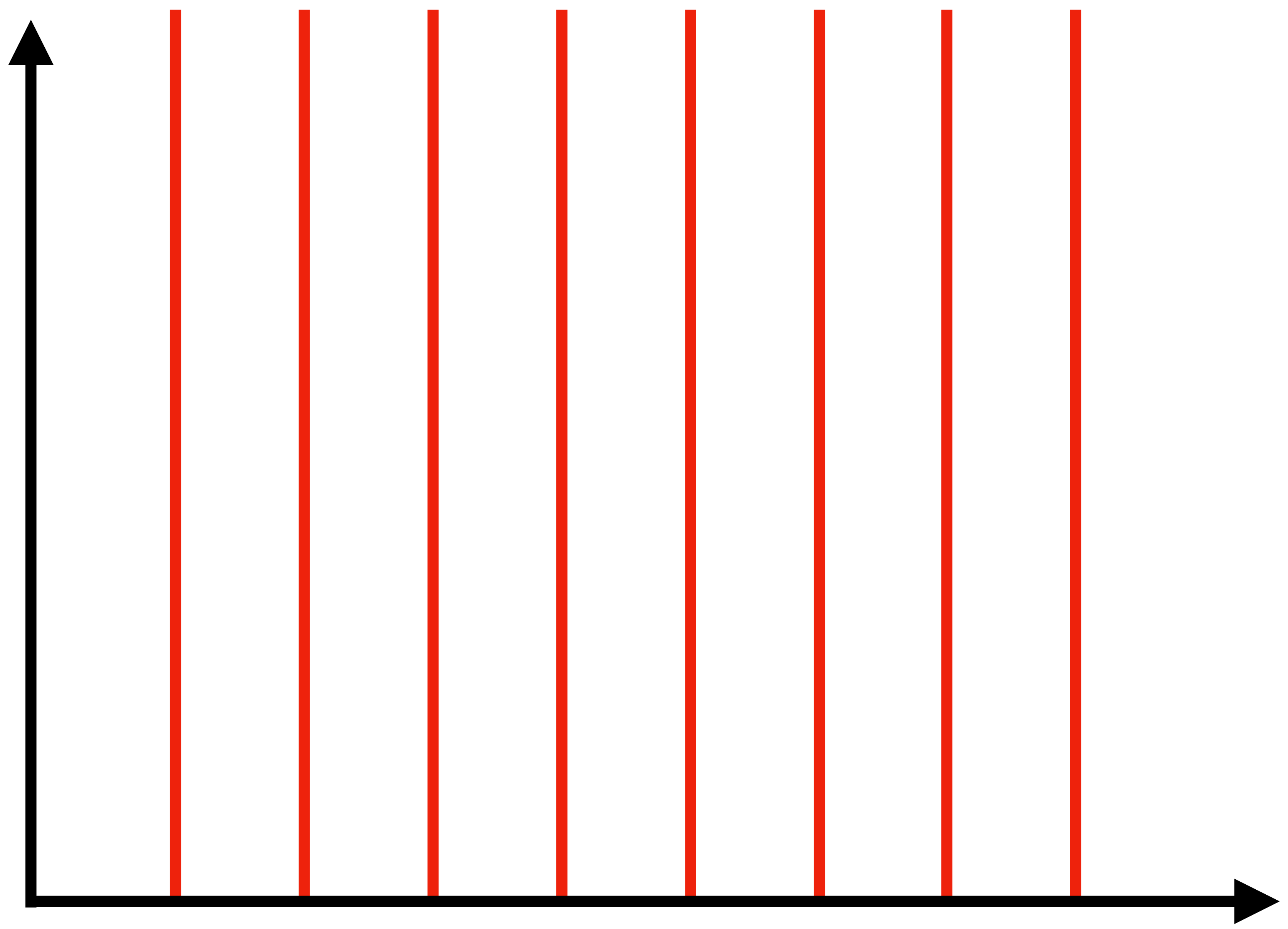
- Are there other ways to stratify?
- How do we generalize this to high-dimensional space?
- What are the mathematical tools we have for analyzing these patterns?
- Pros and cons between different patterns?

Frequency analysis of Monte Carlo integration

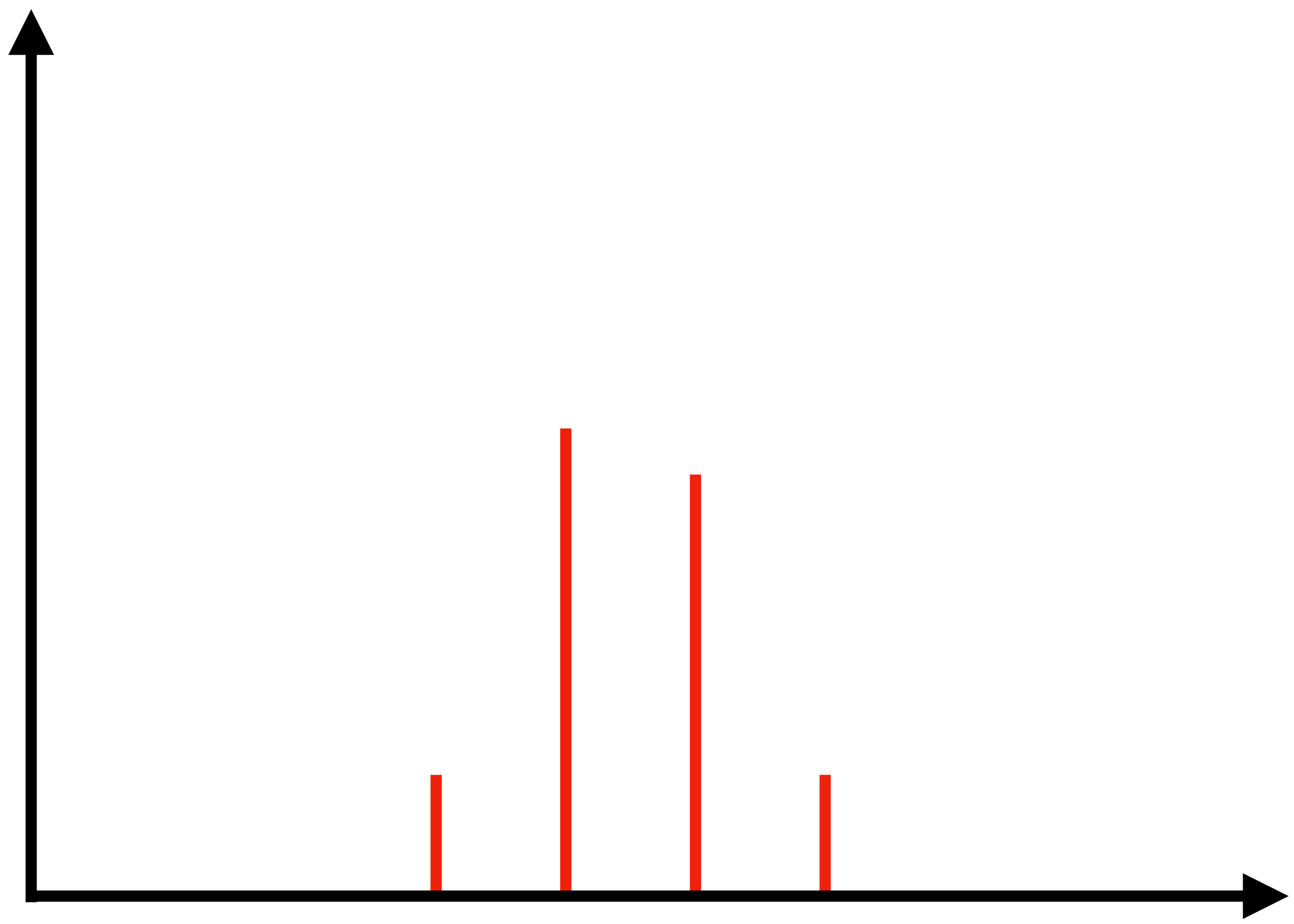
integrand f



sampling pattern S



Sf (multiplication)



$$\int f(x)\mathrm{d}x \approx \frac{1}{N}\sum_{i=0}^{N} f(x_i) = \int f(x)S(x)\mathrm{d}x \qquad S(x) = \sum \delta(x - x_i)$$

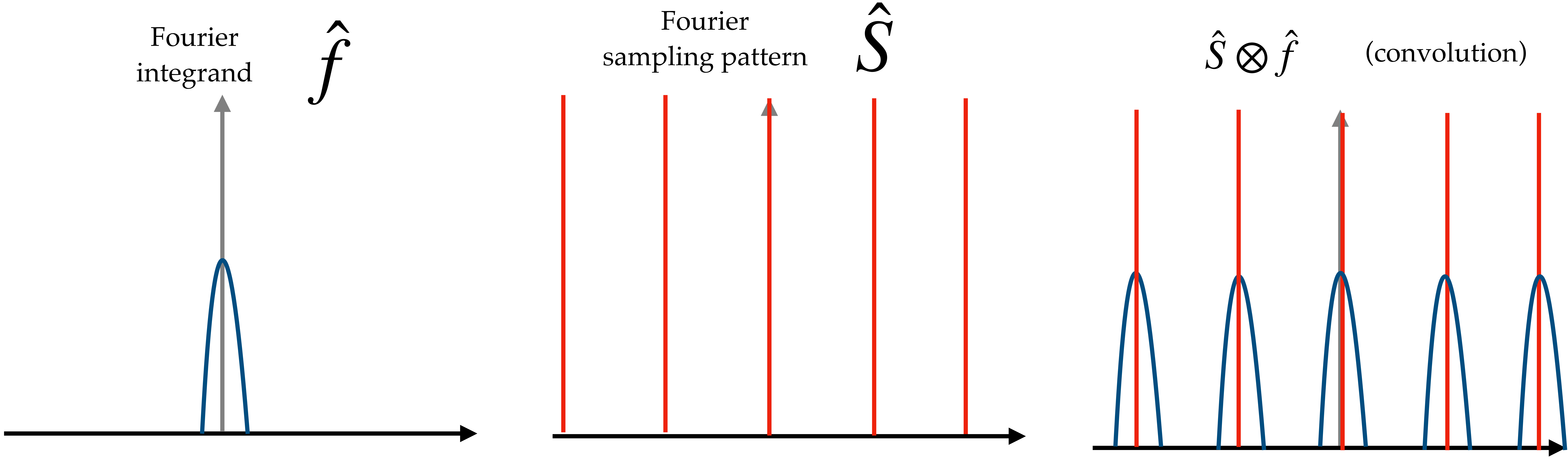
2011

A Frequency Analysis of Monte-Carlo and other Numerical Integration Schemes

Frédo Durand
MIT CSAIL

Frequency analysis of Monte Carlo integration

- numerical integration = taking DC of the convolution between sampling patterns & integrand in frequency domain



$$\int f(x)S(x)\mathrm{d}x = \hat{f} \otimes \hat{S}(0)$$

2011

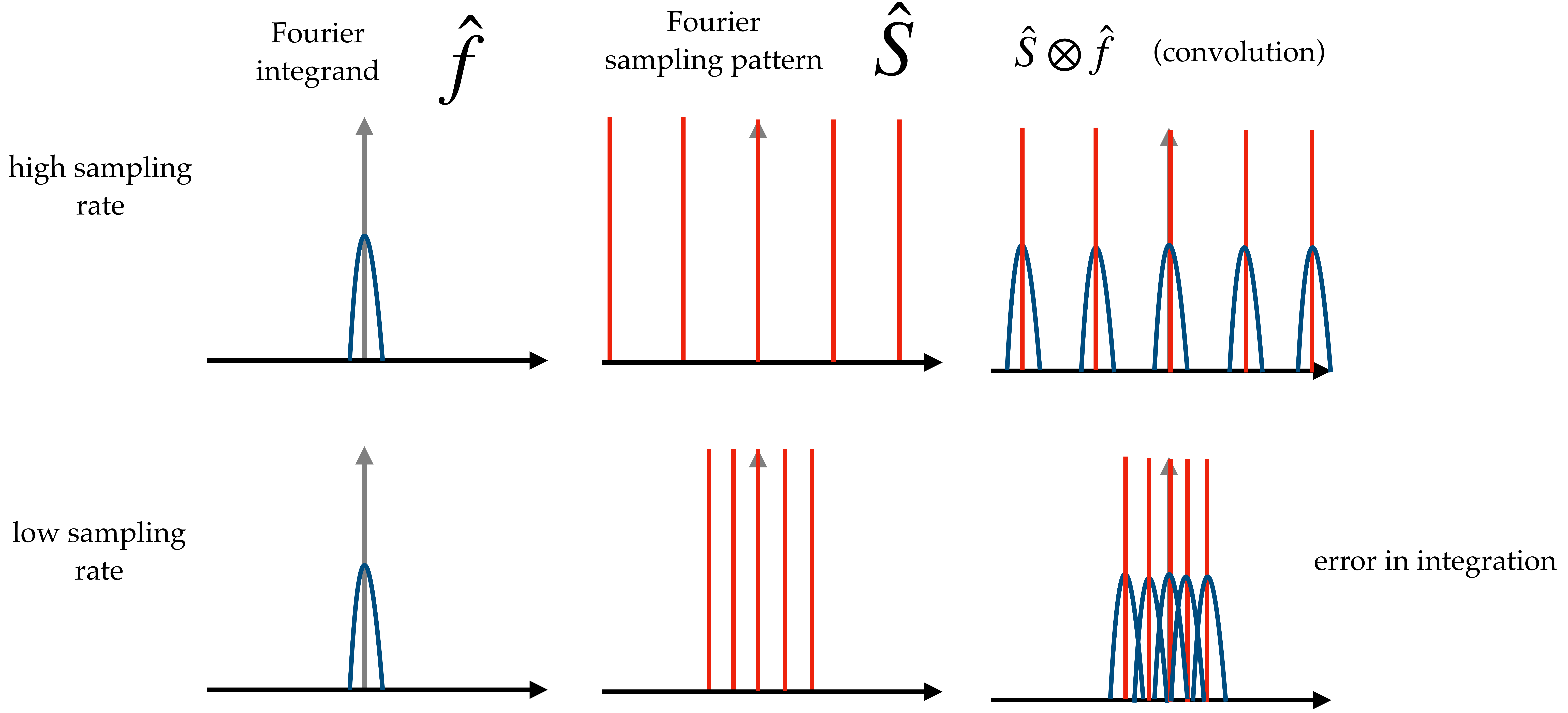
A Frequency Analysis of Monte-Carlo and other Numerical Integration Schemes

Frédo Durand
MIT CSAIL

Frequency analysis of Monte Carlo integration

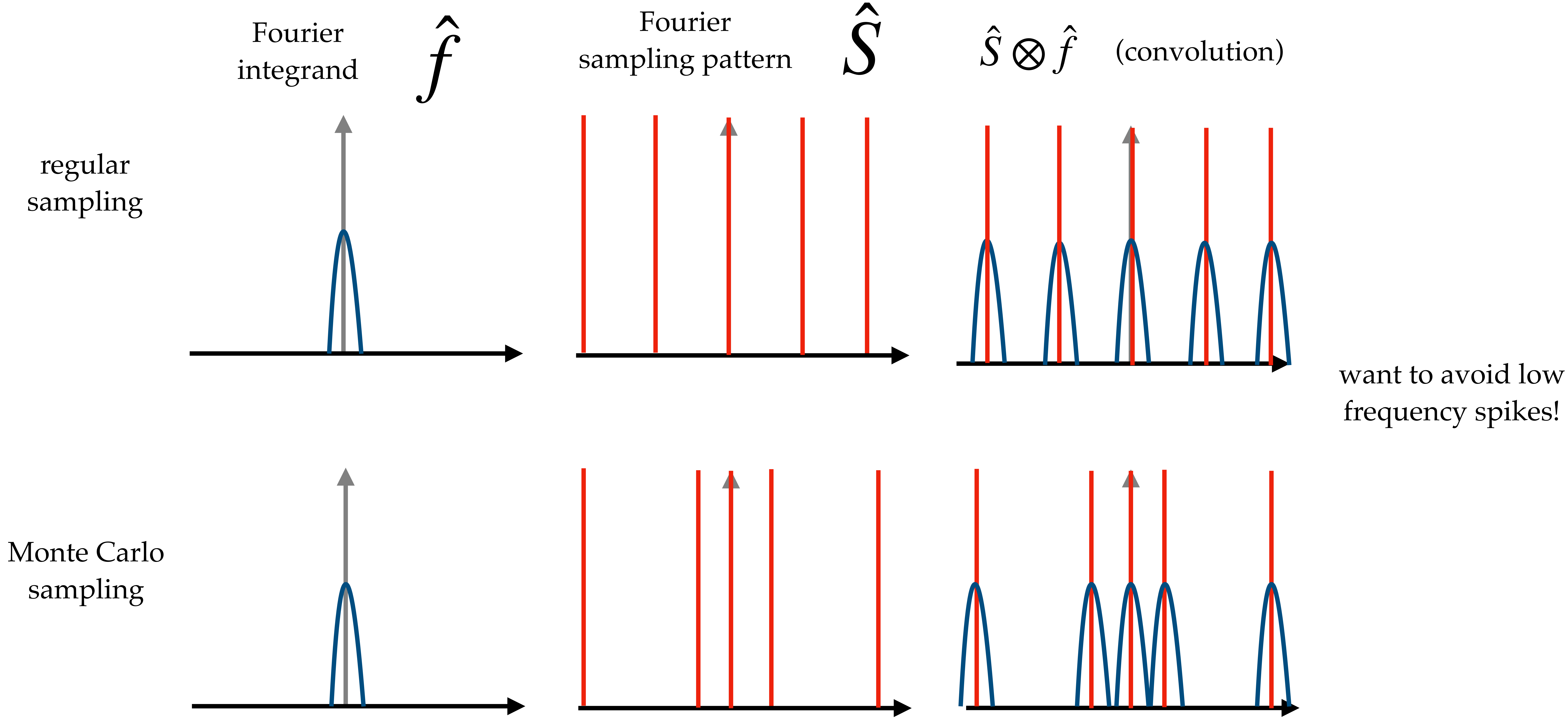
- numerical integration = taking DC of the convolution between sampling patterns & integrand in frequency domain

quiz: when will we have perfect reconstruction?



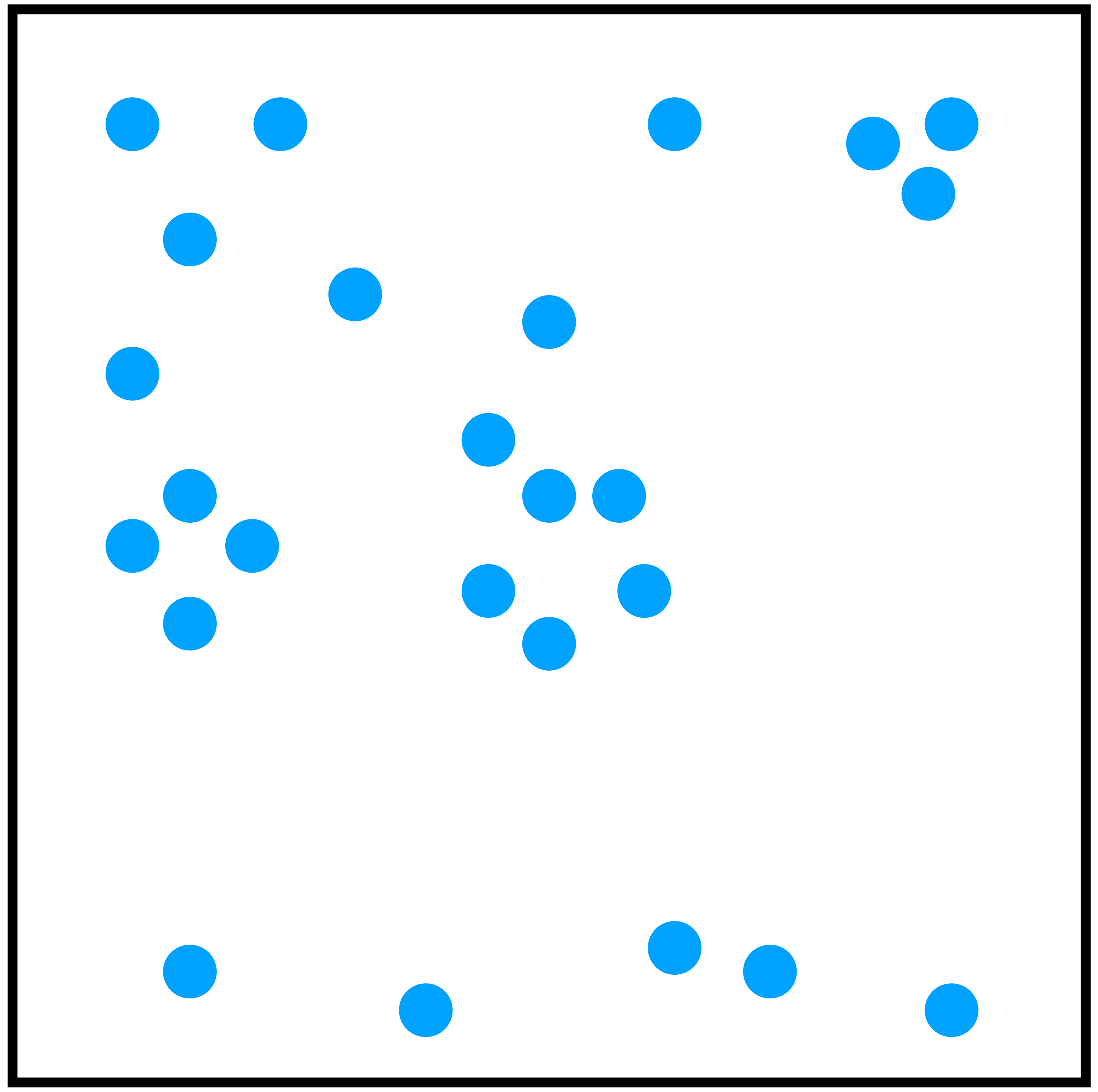
Frequency analysis of Monte Carlo integration

- numerical integration = taking DC of the convolution between sampling patterns & integrand in frequency domain



Observation: S is a random variable

$$\int f(x)S(x)\mathrm{d}x = \hat{f} \otimes \hat{S}(0)$$



Bias-variance analysis in Fourier domain

$$F = \int f(x)\mathrm{d}x$$

mean square error = bias^2 + variance

$$F_{\text{est}} = \int f(x)S(x)\mathrm{d}x = \hat{f} \otimes \hat{S}(0)$$

$$E\left[\left(F_{\text{est}} - F\right)^2\right] = E\left[F_{\text{est}} - F\right]^2 + \mathrm{Var}\left[F_{\text{est}} - F\right]$$

Bias-variance analysis in Fourier domain

$$\text{bias} = \hat{f}(0) - \int \hat{f}^*(\omega) E[\hat{S}(\omega)] \mathrm{d}\omega$$

$$F = \int f(x) S(x) \mathrm{d}x = \hat{f} \otimes \hat{S}(0)$$

$$\text{variance} = \int \left| \hat{f}(\omega) \right|^2 E\left[\left| \hat{S}(\omega) \right|^2 \right] \mathrm{d}\omega$$

(slightly simplified)

2013

Fourier Analysis of Stochastic Sampling Strategies for Assessing Bias and Variance in Integration

Kartic Subr*
University College London

Jan Kautz†
University College London

Bias-variance analysis in Fourier domain

$$\text{bias} = \hat{f}(0) - \int \hat{f}^*(\omega) E[\hat{S}(\omega)] \mathrm{d}\omega$$

$$F = \int f(x) S(x) \mathrm{d}x = \hat{f} \otimes \hat{S}(0)$$

for many random samplers, $E[\hat{S}(\omega)] = 0$ iff $\omega \neq 0$

$$\text{variance} = \int \left| \hat{f}(\omega) \right|^2 E\left[\left| \hat{S}(\omega) \right|^2 \right] \mathrm{d}\omega$$

(slightly simplified)

2013

Fourier Analysis of Stochastic Sampling Strategies for Assessing Bias and Variance in Integration

Kartic Subr[*]
University College London

Jan Kautz[†]
University College London

Bias-variance analysis in Fourier domain

$$\text{bias} = \hat{f}(0) - \int \hat{f}^*(\omega) E[\hat{S}(\omega)] \mathrm{d}\omega$$

$$F = \int f(x) S(x) \mathrm{d}x = \hat{f} \otimes \hat{S}(0)$$

for many random samplers, $E[\hat{S}(\omega)] = 0$ iff $\omega \neq 0$

$$\text{variance} = \int \left| \hat{f}(\omega) \right|^2 E\left[\left| \hat{S}(\omega) \right|^2 \right] \mathrm{d}\omega$$

(slightly simplified)

the expected power spectrum of
the sampling pattern $E[\hat{S}^2]$ is the key!!

2013

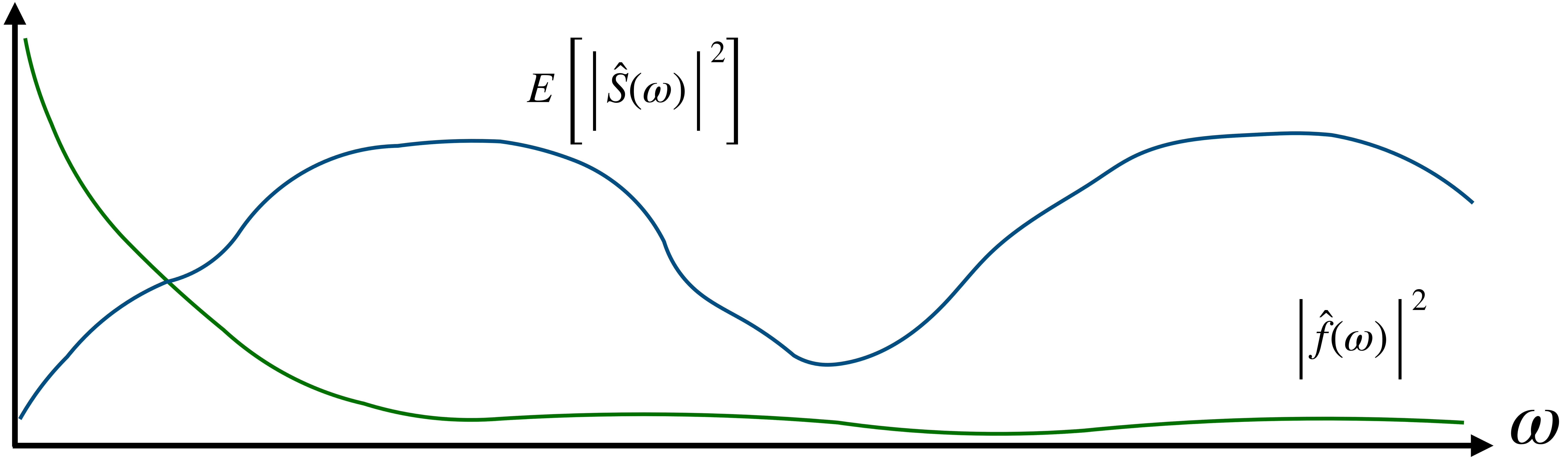
Fourier Analysis of Stochastic Sampling Strategies for Assessing Bias and Variance in Integration

Kartic Subr[*]
University College London

Jan Kautz[†]
University College London

Variance analysis = multiplication of power spectrums

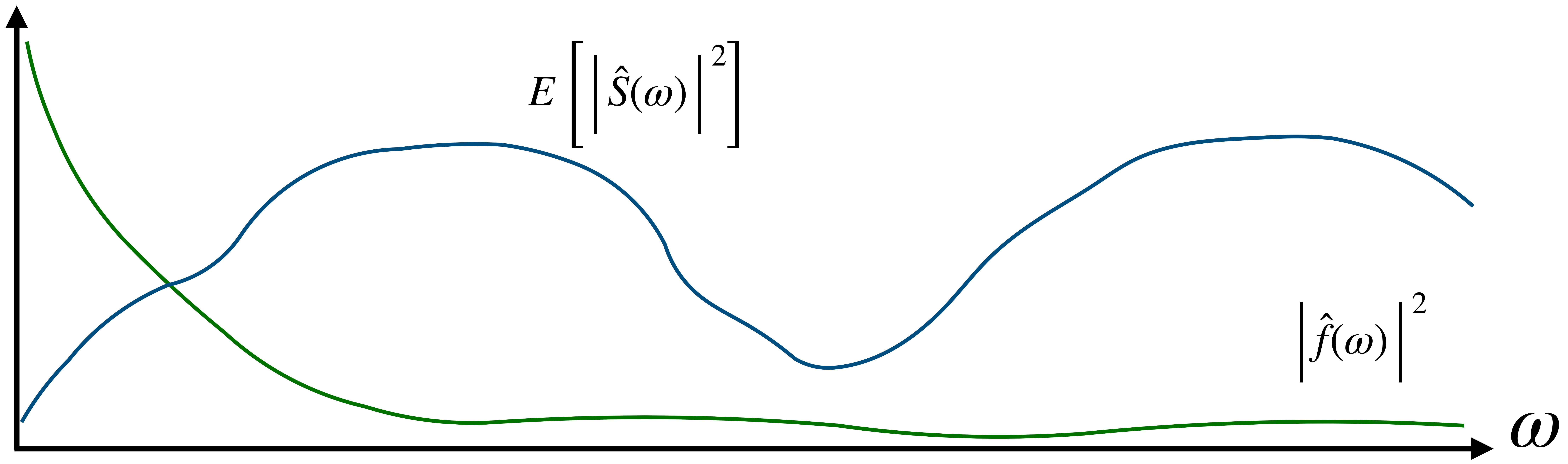
- natural signals/integrands usually have energy concentrated at low frequencies
    - **quiz**: what $E[\hat{S}^2]$ will lead to low variance?



$$\text{variance} = \int \left|\hat{f}(\omega)\right|^2 E\left[\left|\hat{S}(\omega)\right|^2\right] \mathrm{d}\omega$$

Variance analysis = multiplication of power spectrums

- natural signals/integrands usually have energy concentrated at low frequencies
  - sampling patterns with small low frequency energy are better!!



$$\text{variance} = \int \left|\hat{f}(\omega)\right|^2 E\left[\left|\hat{S}(\omega)\right|^2\right] \mathrm{d}\omega$$

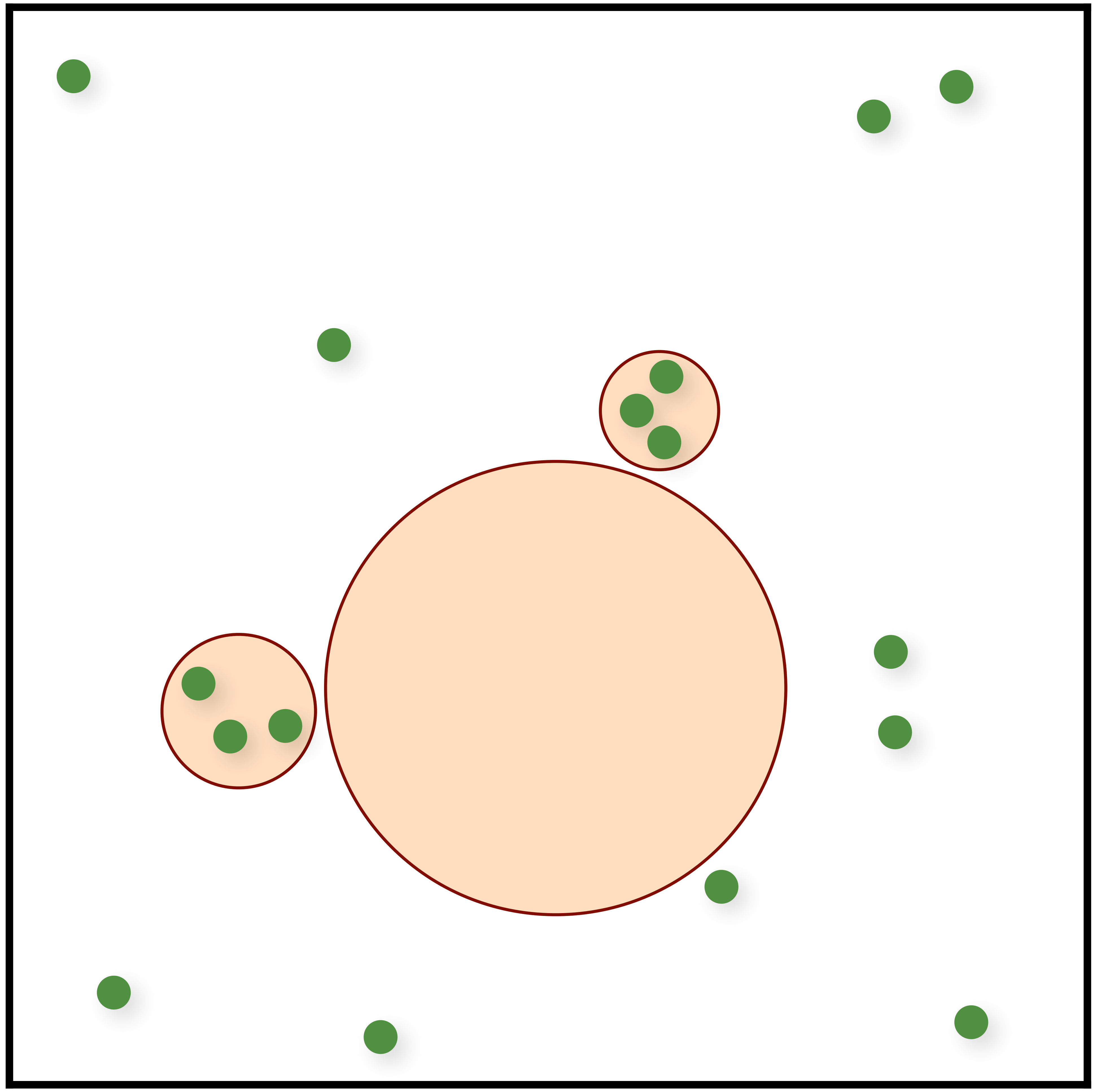
Let’s look at different sampling patterns!

slides heavily borrowed from Wojciech Jarosz
https://cs.dartmouth.edu/~wjarosz/publications/subr16fourier.html

Independent random sampling

```
for (int k = 0; k < num; k++)
{
    samples(k).x = randf();
    samples(k).y = randf();
}
```

quiz: pros and cons?



Independent random sampling

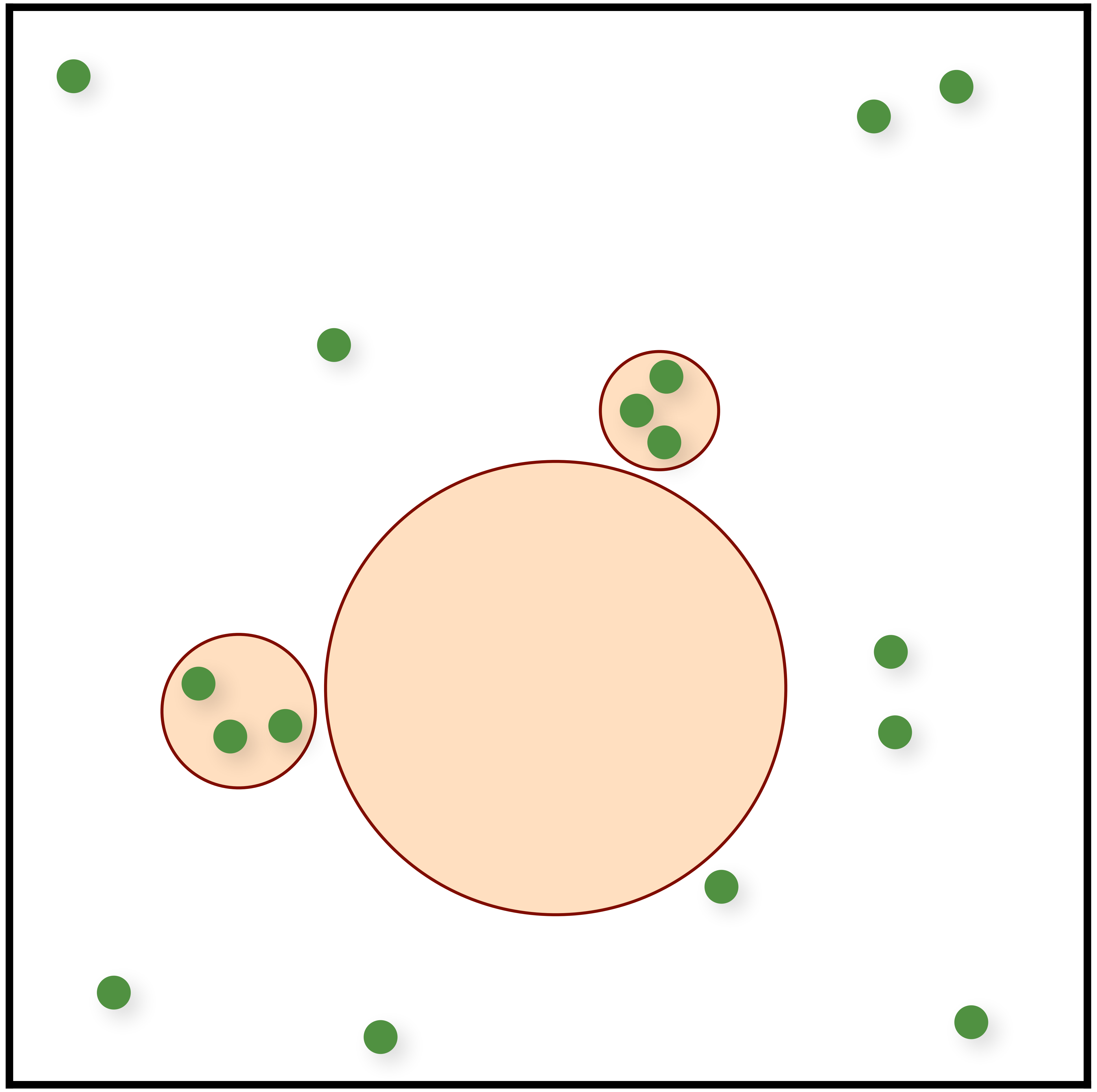
```
for (int k = 0; k < num; k++)
{
    samples(k).x = randf();
    samples(k).y = randf();
}
```

✔ Trivially extends to higher dimensions

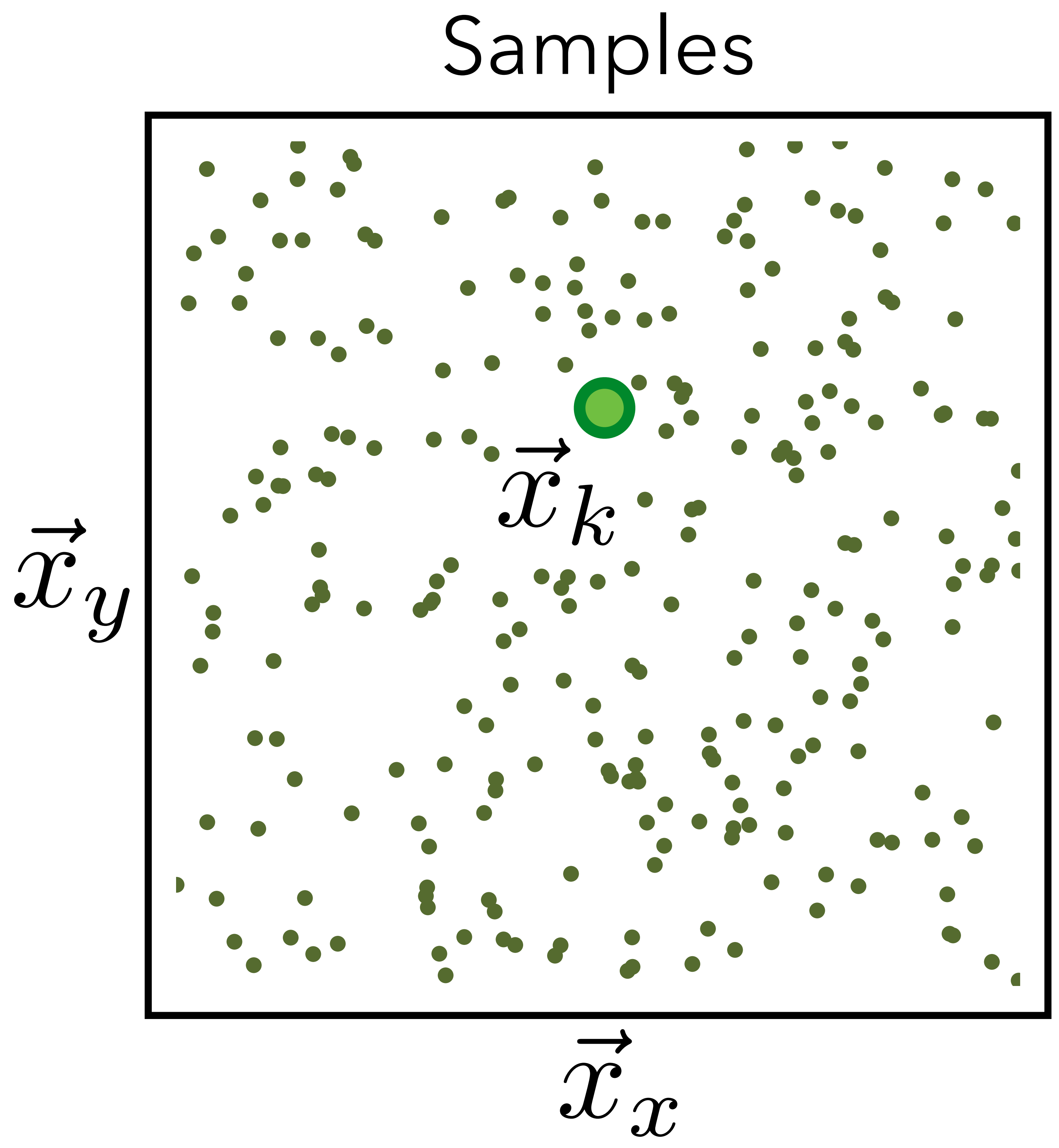
✔ Trivially progressive and memory-less

✘ Big gaps

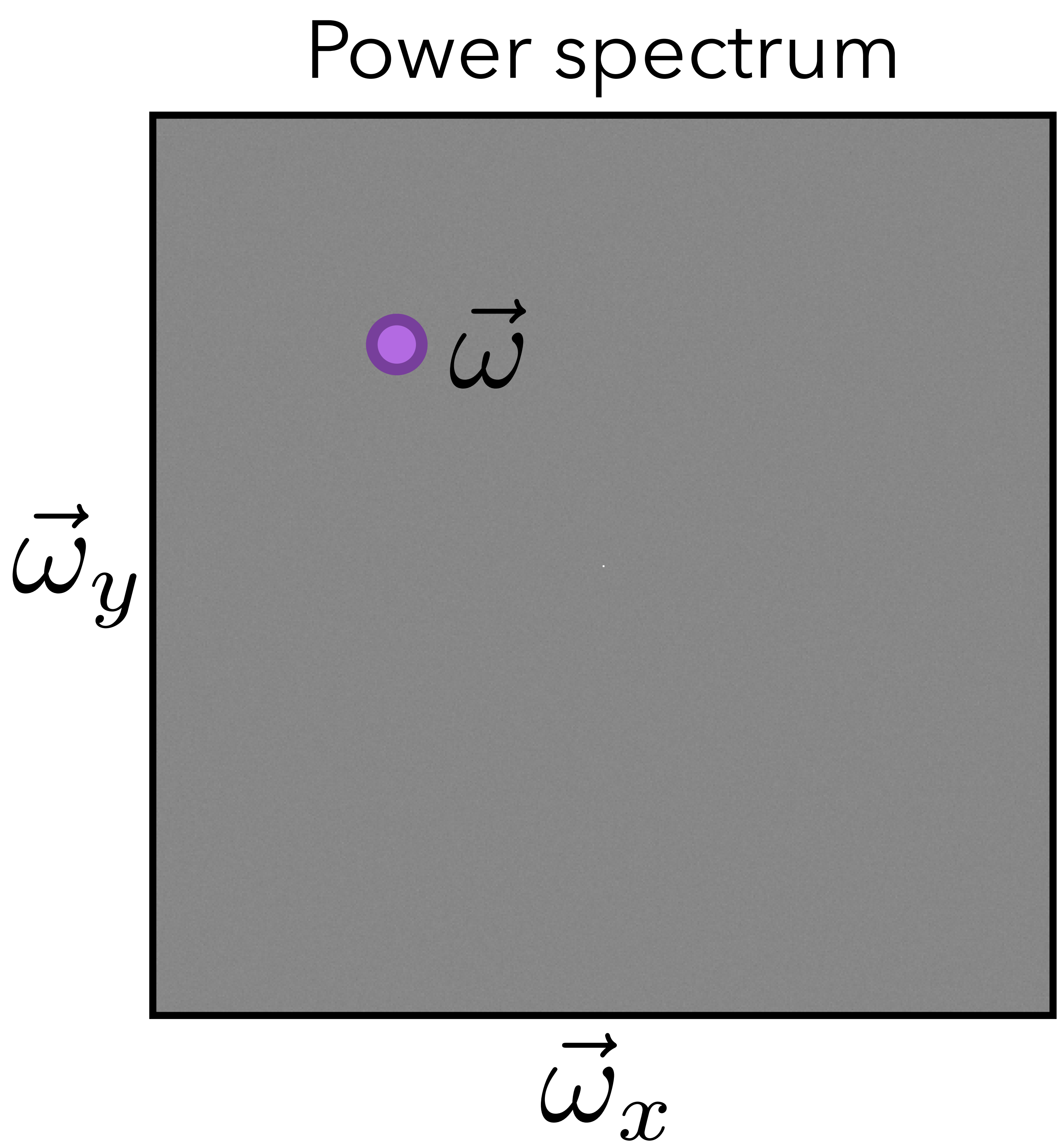
✘ Clumping



Frequency analysis of independent random sampling



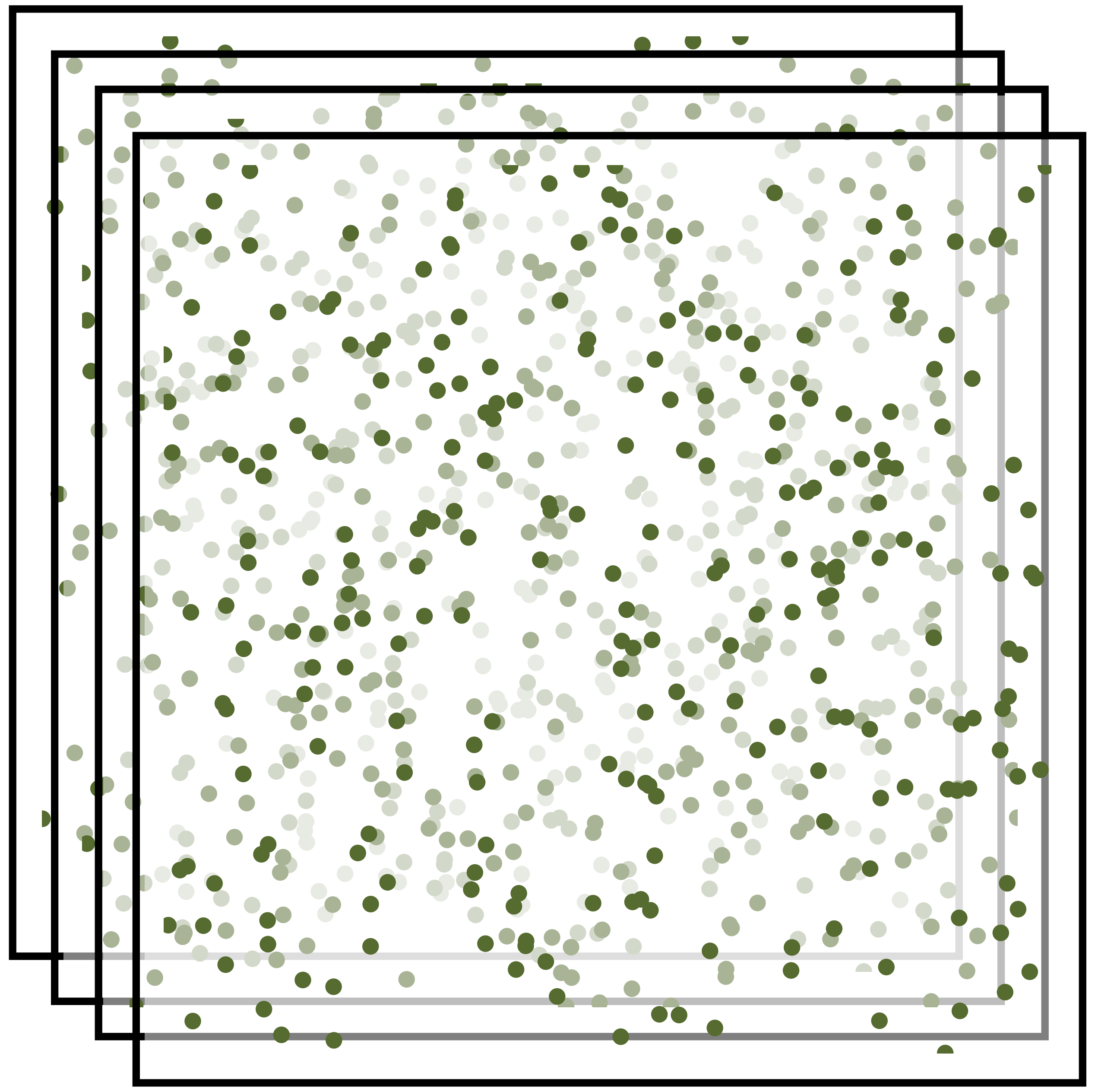
$$\frac{1}{N}\sum_{k=1}^{N}\delta(|\vec{x}-\vec{x}_k|)$$



$$\left|\frac{1}{N}\sum_{k=1}^{N}\mathrm{e}^{-2\,\pi\,\imath\,(\vec{\omega}\cdot\vec{x}_k)}\right|^2$$

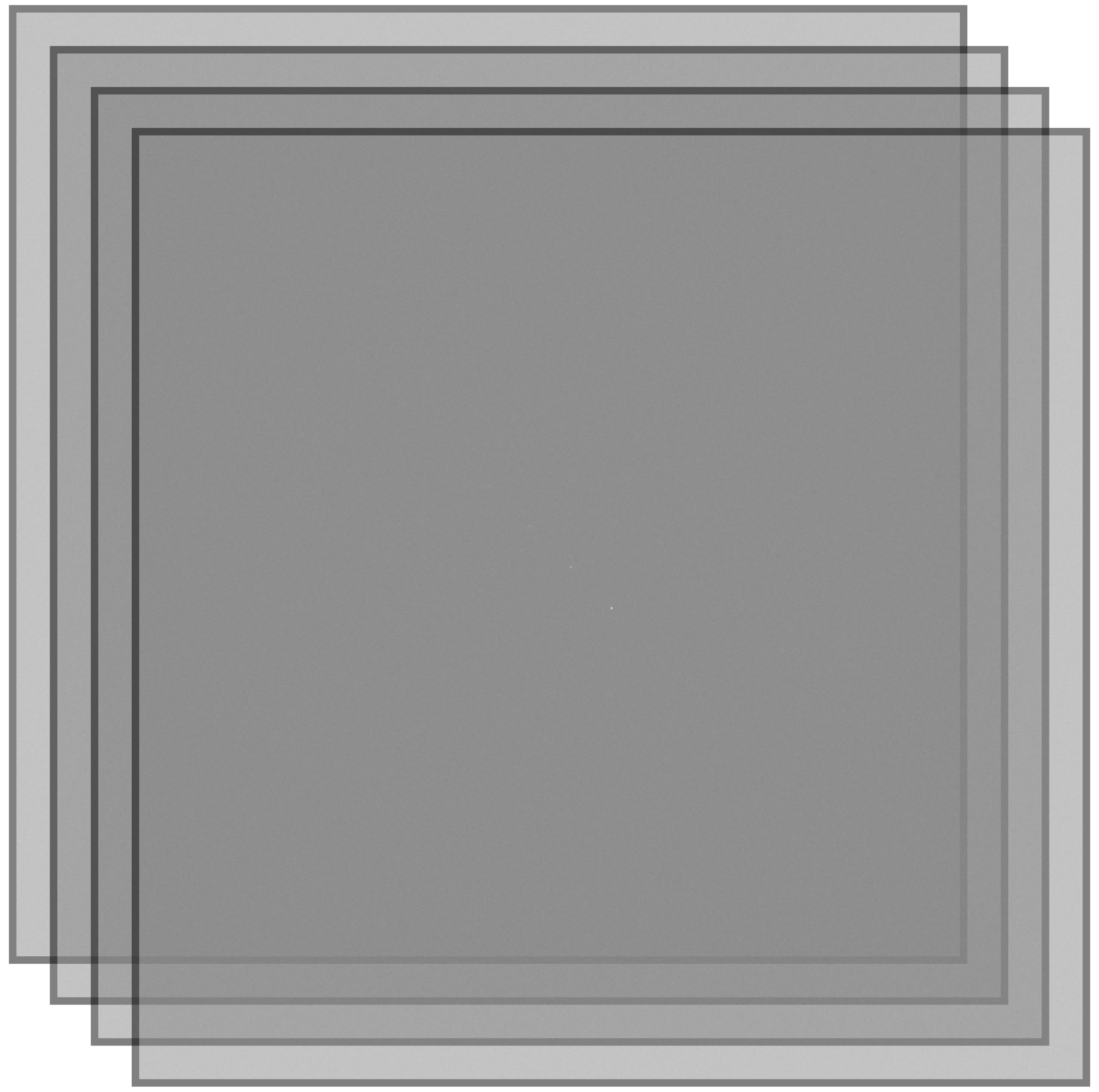
Frequency analysis of independent random sampling

Many sample set realizations



$$\frac{1}{N}\sum_{k=1}^{N}\delta(|\vec{x}-\vec{x}_k|)$$

Expected power spectrum



$$\mathrm{E}\left[\left|\frac{1}{N}\sum_{k=1}^{N}\mathrm{e}^{-2\,\pi\,\imath\,(\vec{\omega}\cdot\vec{x}_k)}\right|^2\right]$$

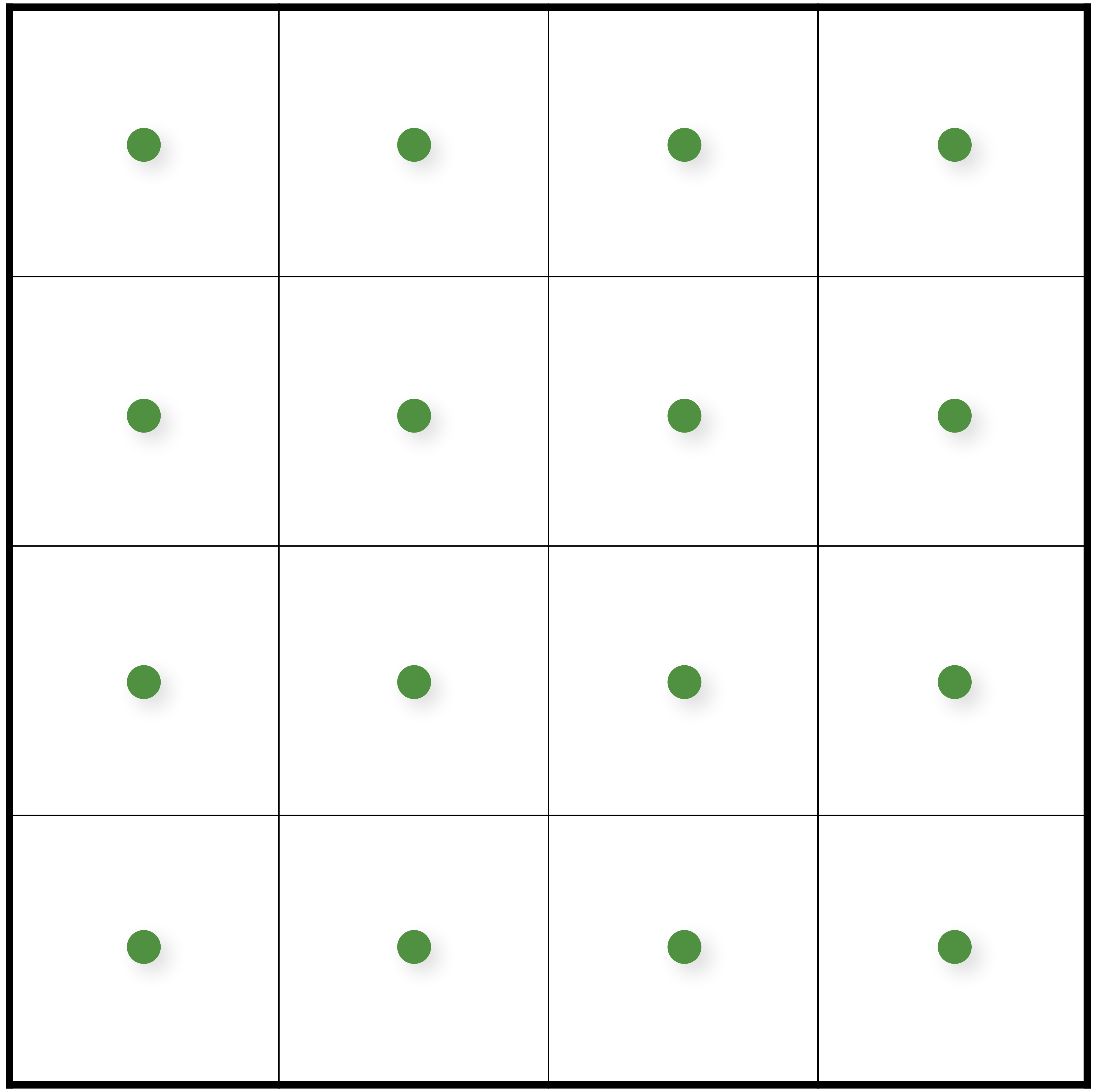
Useful to visualize the radial mean of expected power spectrum

$$\frac{1}{N}\sum_{k=1}^{N}\delta(|\vec{x}-\vec{x}_k|) \quad \mathrm{E}\left[\left|\frac{1}{N}\sum_{k=1}^{N} \mathrm{e}^{-2\pi\imath(\vec{\omega}\cdot\vec{x}_k)}\right|^2\right]$$

Regular sampling: high bias, zero variance

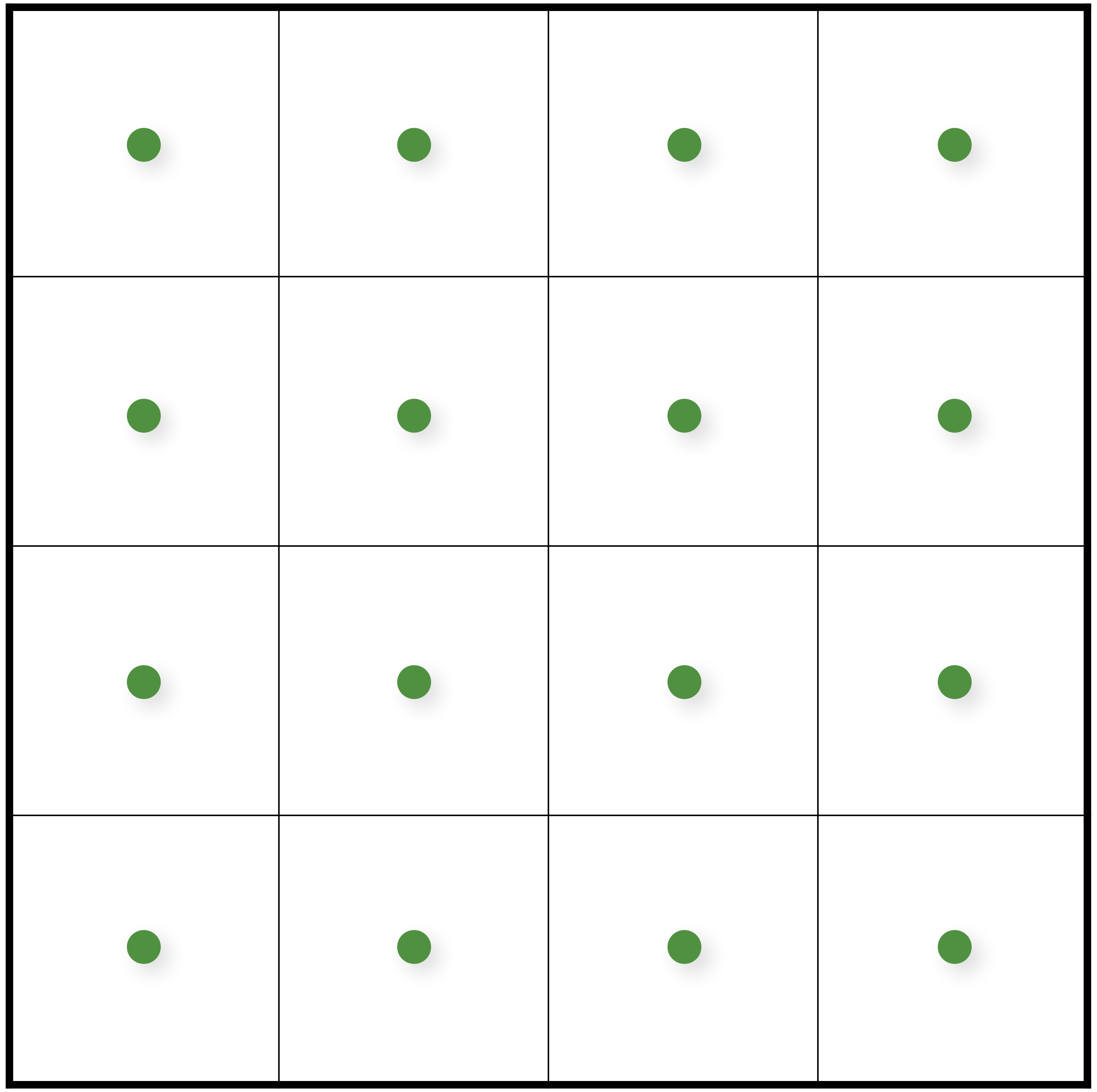
```
for (uint i = 0; i < numX; i++)
    for (uint j = 0; j < numY; j++)
    {
        samples(i,j).x = (i + 0.5)/numX;
        samples(i,j).y = (j + 0.5)/numY;
    }
```

quiz: pros and cons?



Regular sampling: high bias, zero variance

```
for (uint i = 0; i < numX; i++)
    for (uint j = 0; j < numY; j++)
    {
        samples(i,j).x = (i + 0.5)/numX;
        samples(i,j).y = (j + 0.5)/numY;
    }
```



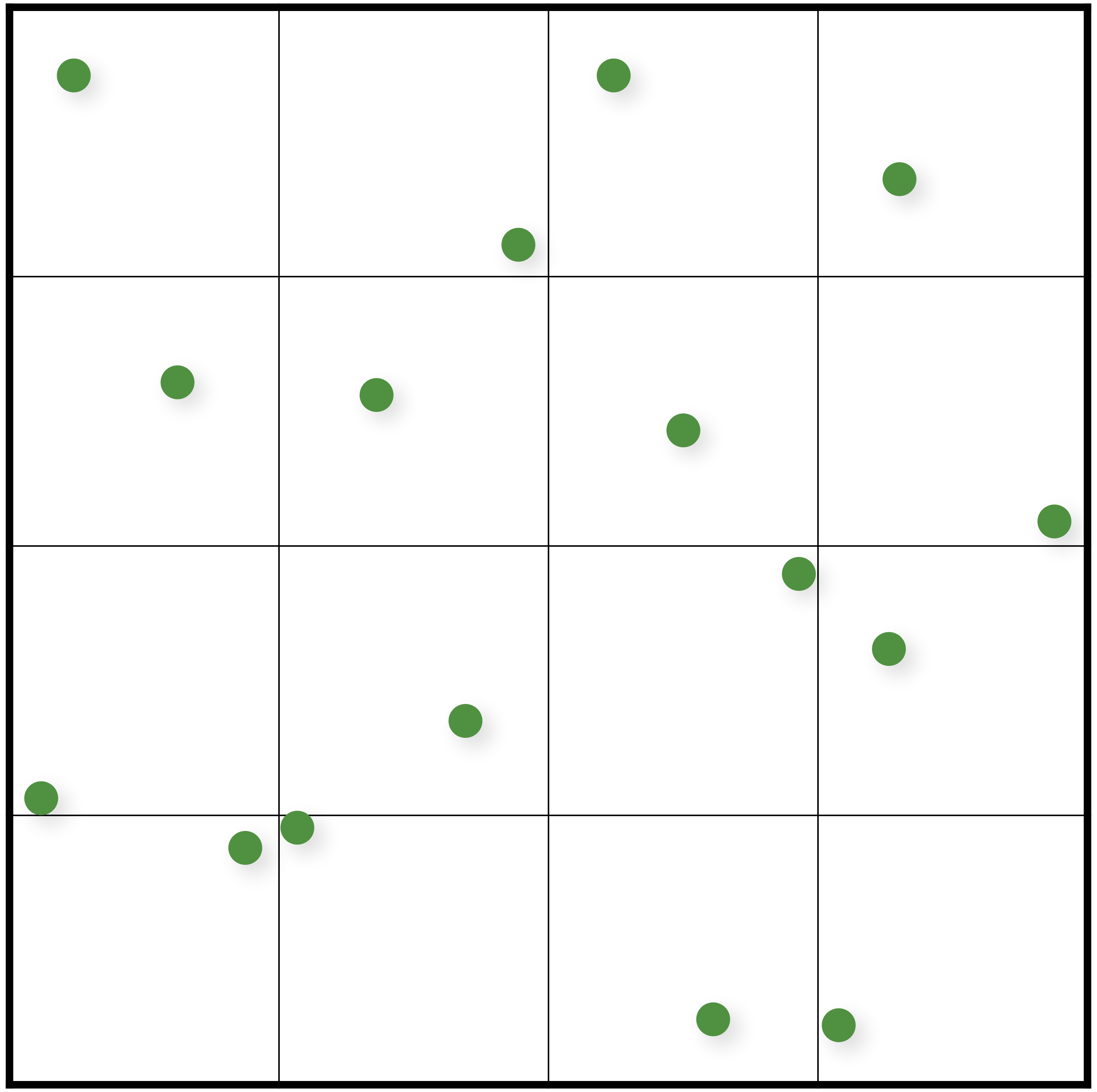
✔ Extends to higher dimensions, but…

✘ Curse of dimensionality

✘ Aliasing

Jittered / stratified sampling: zero bias, low variance

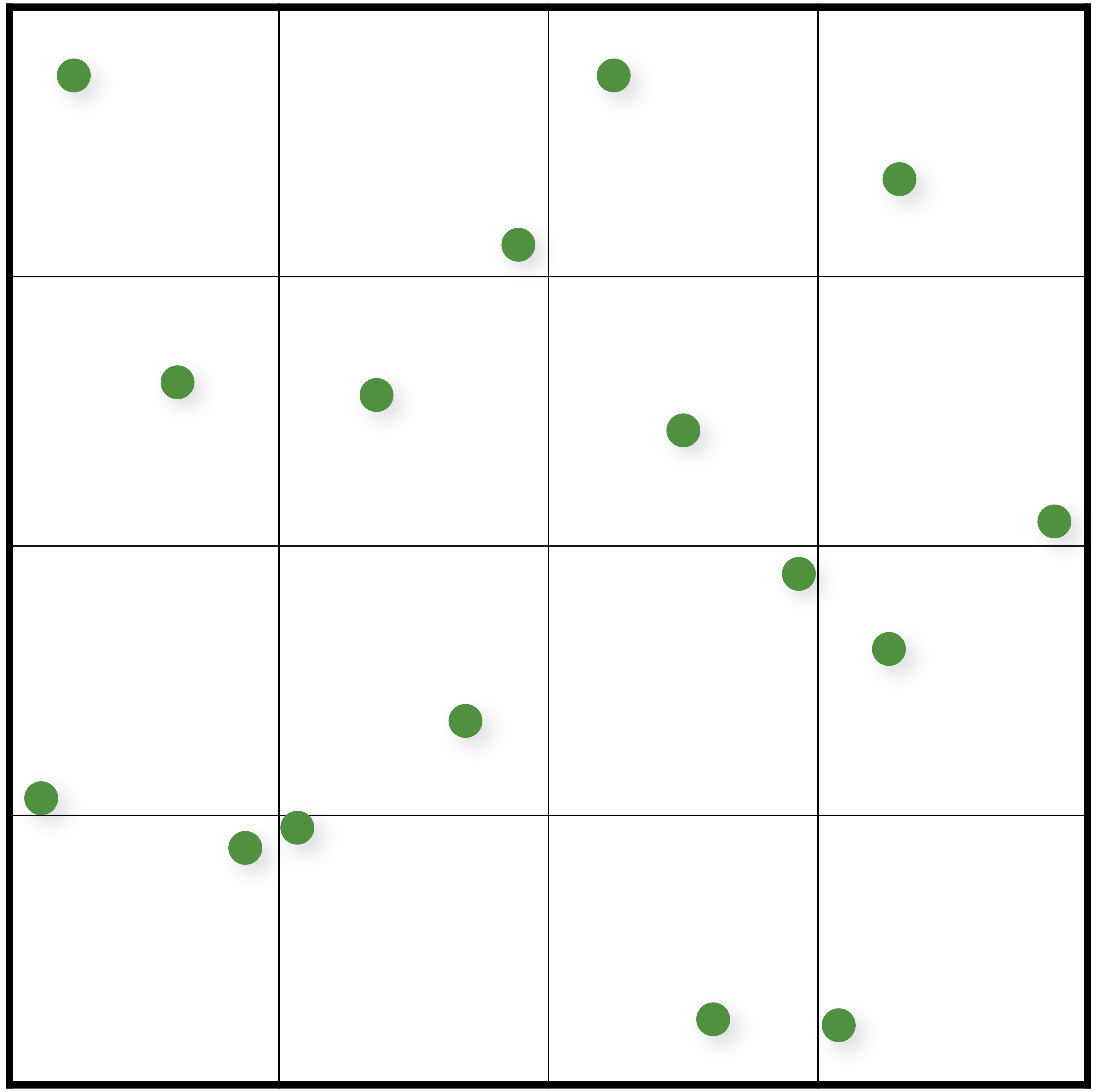
```
for (uint i = 0; i < numX; i++)
    for (uint j = 0; j < numY; j++)
    {
        samples(i,j).x = (i + randf())/numX;
        samples(i,j).y = (j + randf())/numY;
    }
```



quiz: pros and cons?

Jittered / stratified sampling: zero bias, low variance

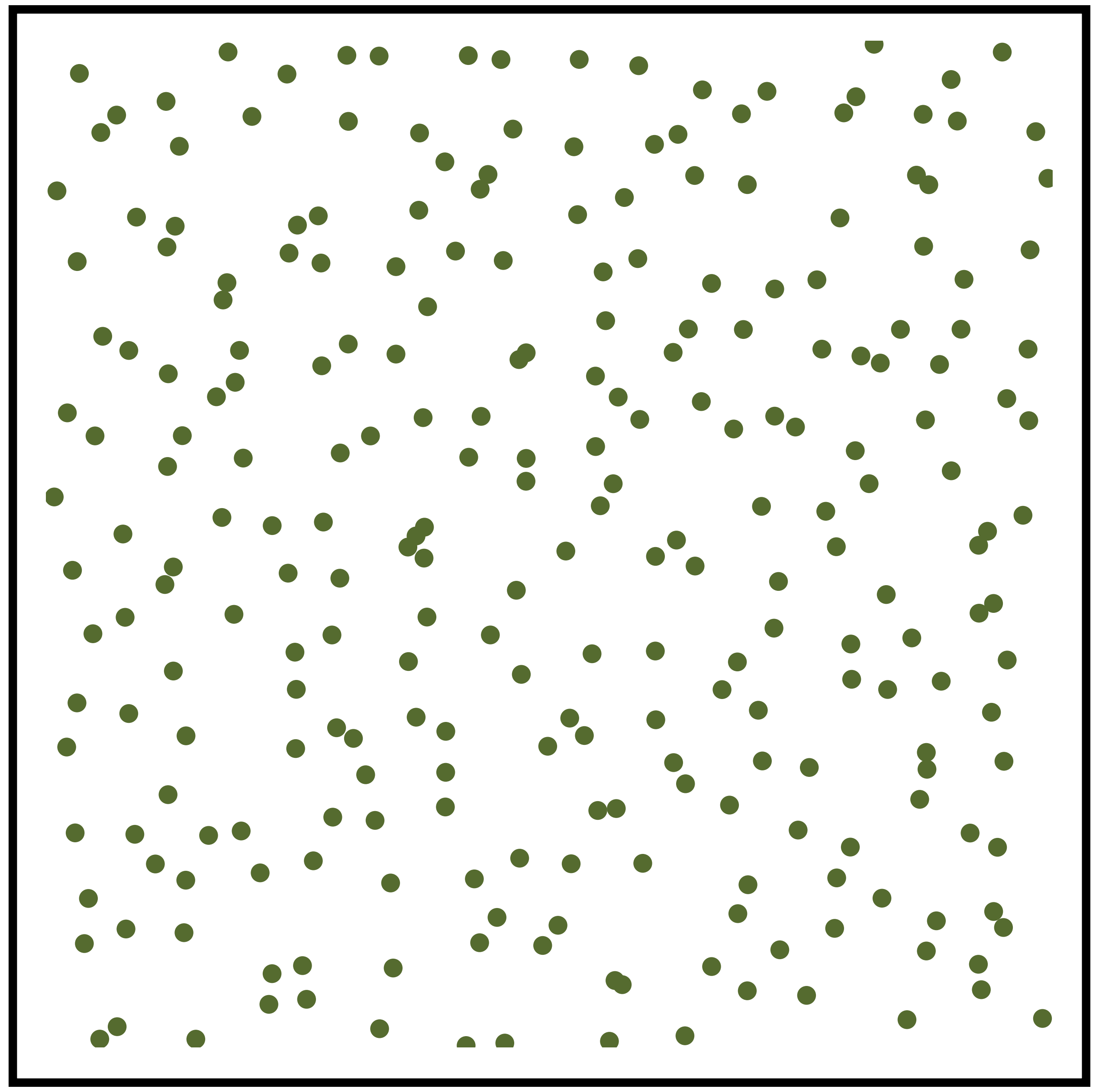
```
for (uint i = 0; i < numX; i++)
    for (uint j = 0; j < numY; j++)
    {
        samples(i,j).x = (i + randf())/numX;
        samples(i,j).y = (j + randf())/numY;
    }
```



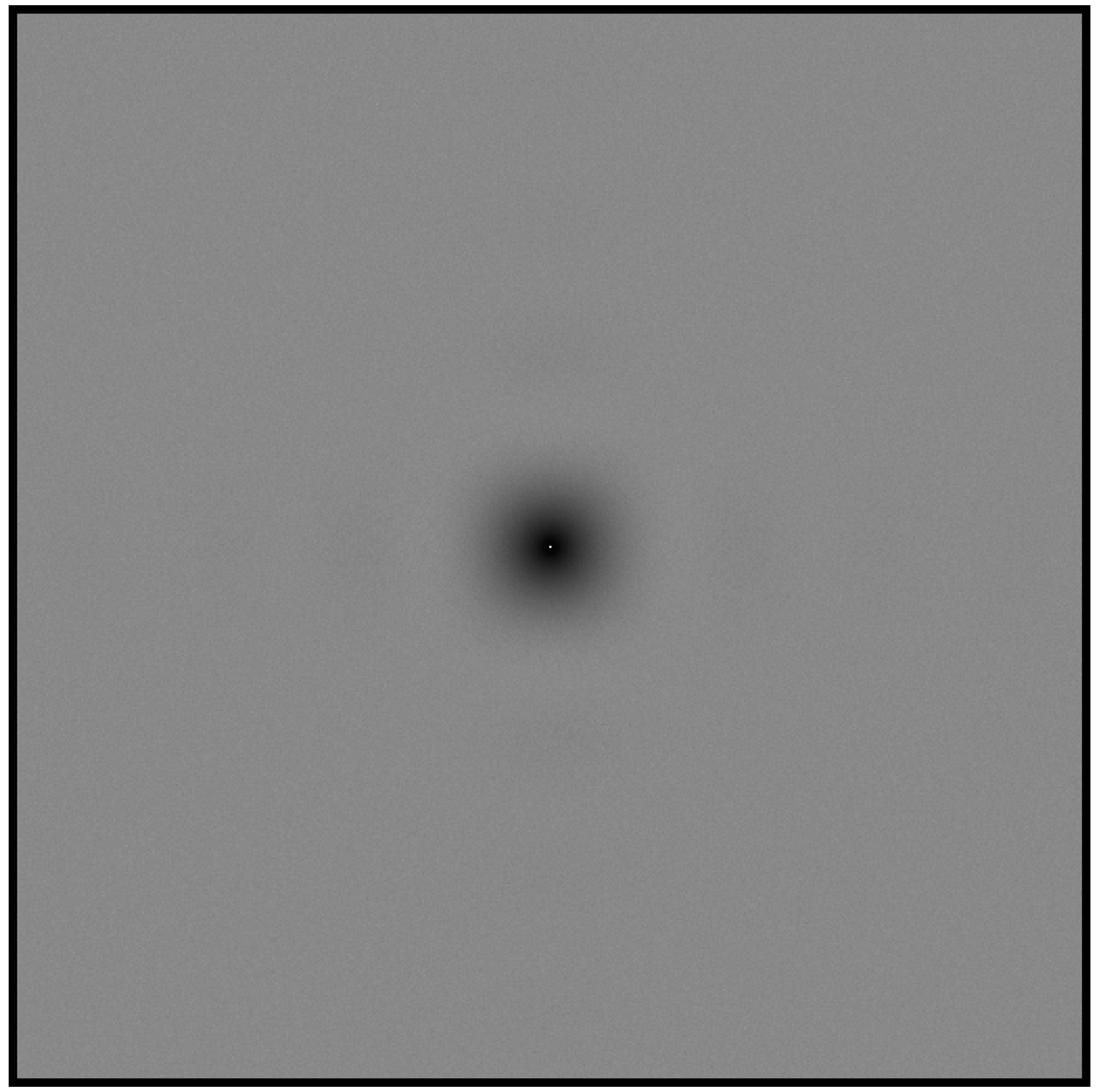
- ✔ Provably cannot increase variance
- ✔ Extends to higher dimensions, but…
- ✘ Curse of dimensionality
- ✘ Not progressive

Power spectrum of jittered sampling

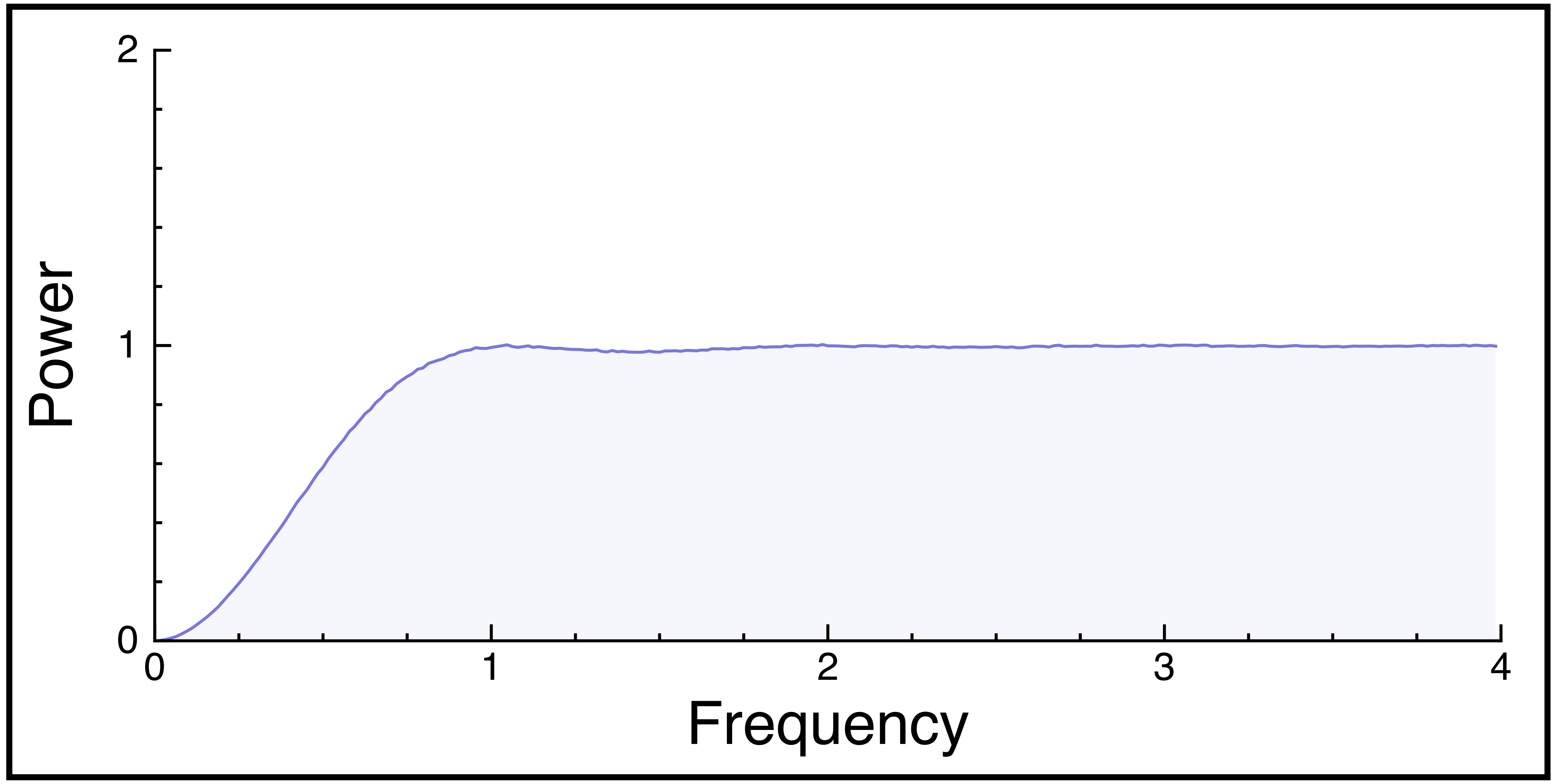
Samples



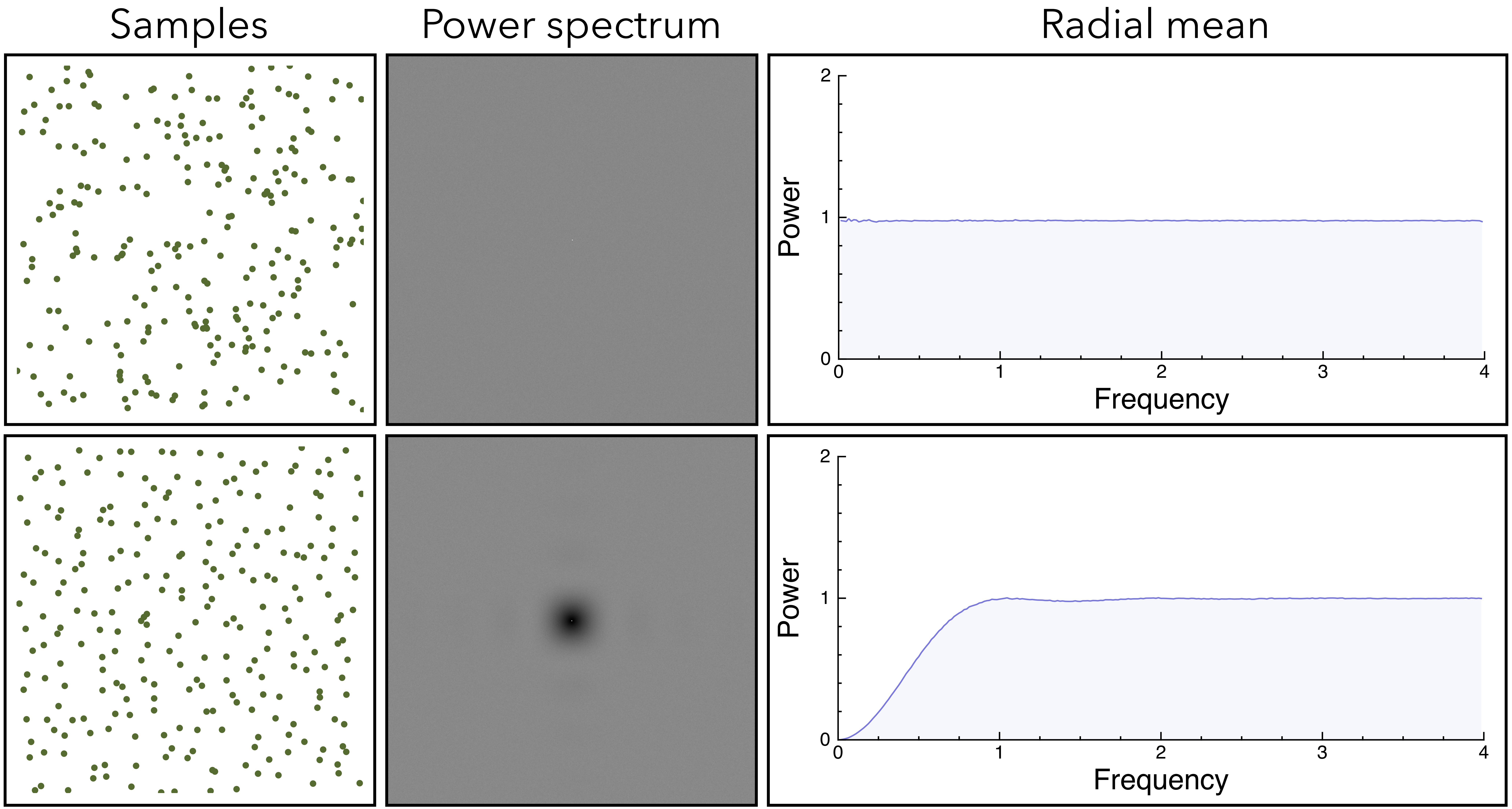
Expected power spectrum



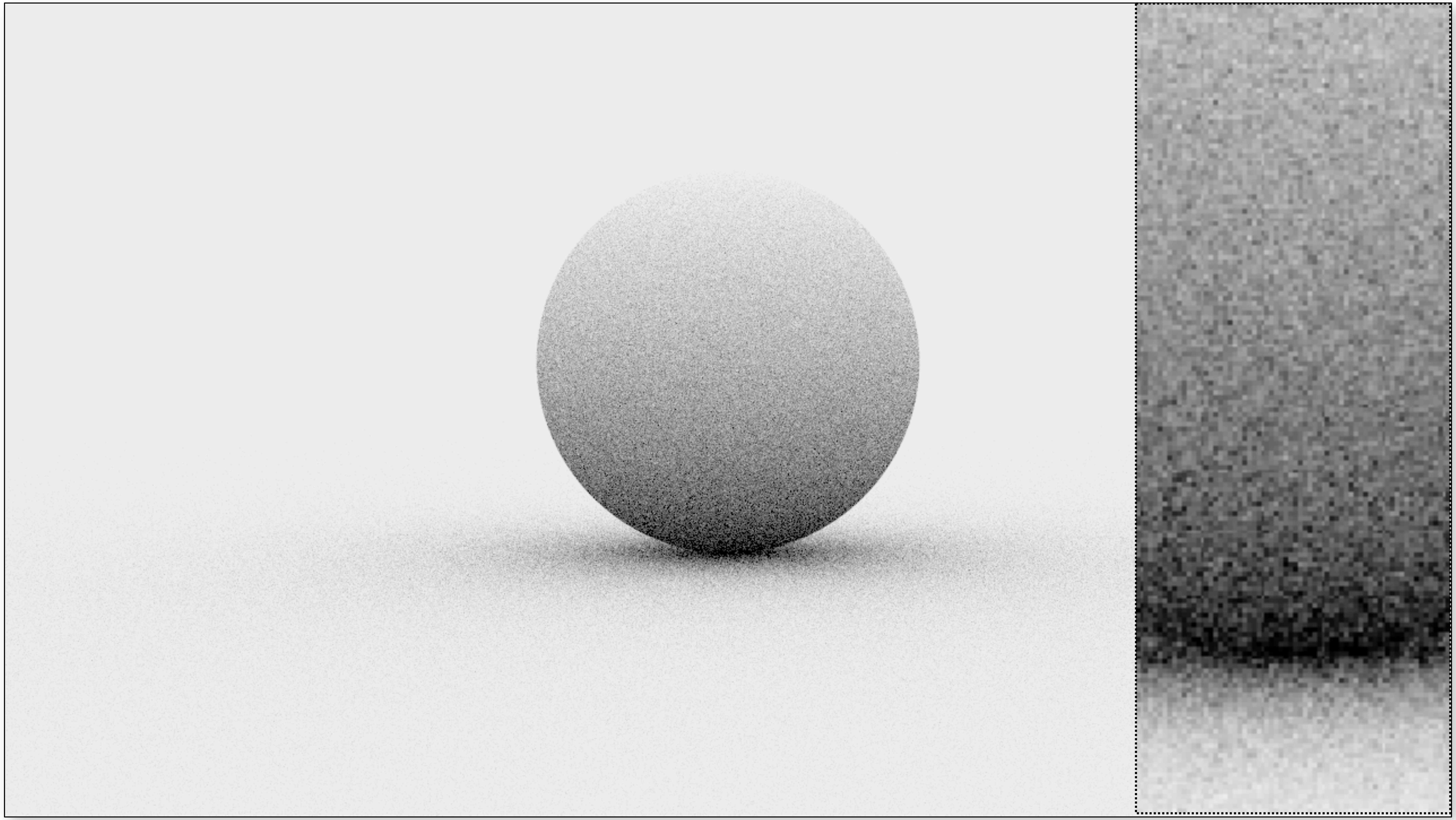
Radial mean



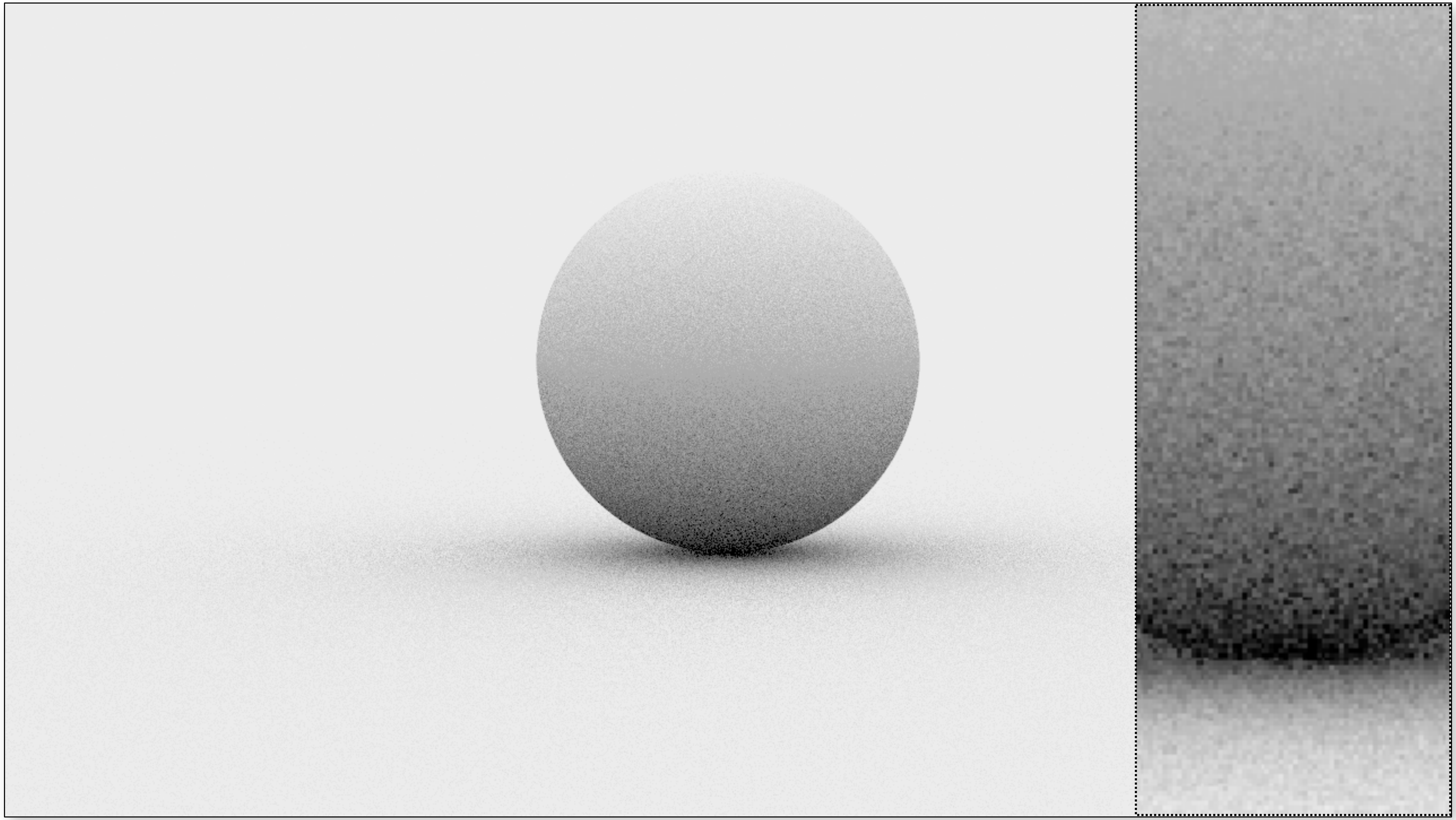
Random sampling vs jittered sampling



Random sampling (16 samples per pixel)



Jittered sampling (16 samples per pixel)



High-dimensional stratification is hard

Stratification requires O(N^d) samples

- e.g. pixel (2D) + lens (2D) + time (1D) = 5D
  - splitting 2 times in 5D = 2^5 = 32 samples
  - splitting 3 times in 5D = 3^5 = 243 samples!

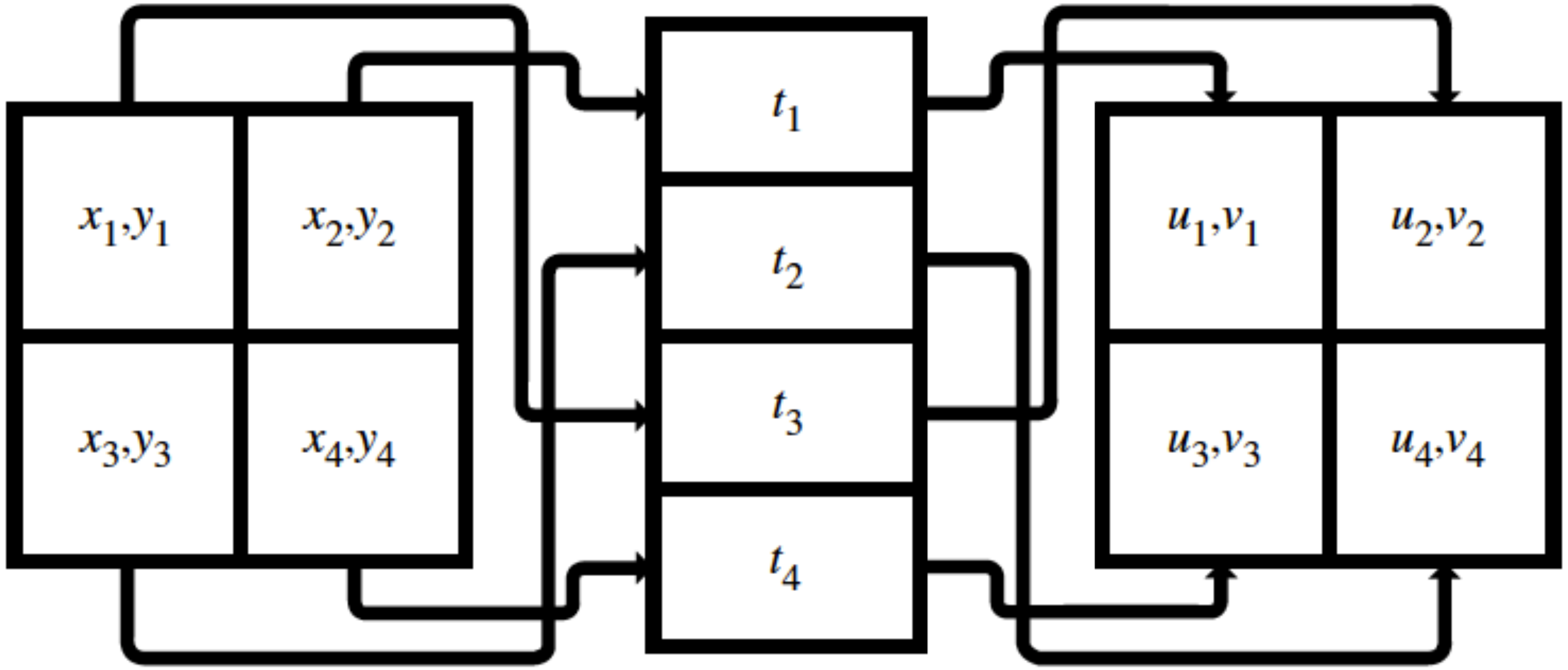
Inconvenient for large d

- cannot select sample count with fine granularity

Uncorrelated Jitter [Cook 1986]

Compute stratified samples in sub-dimensions

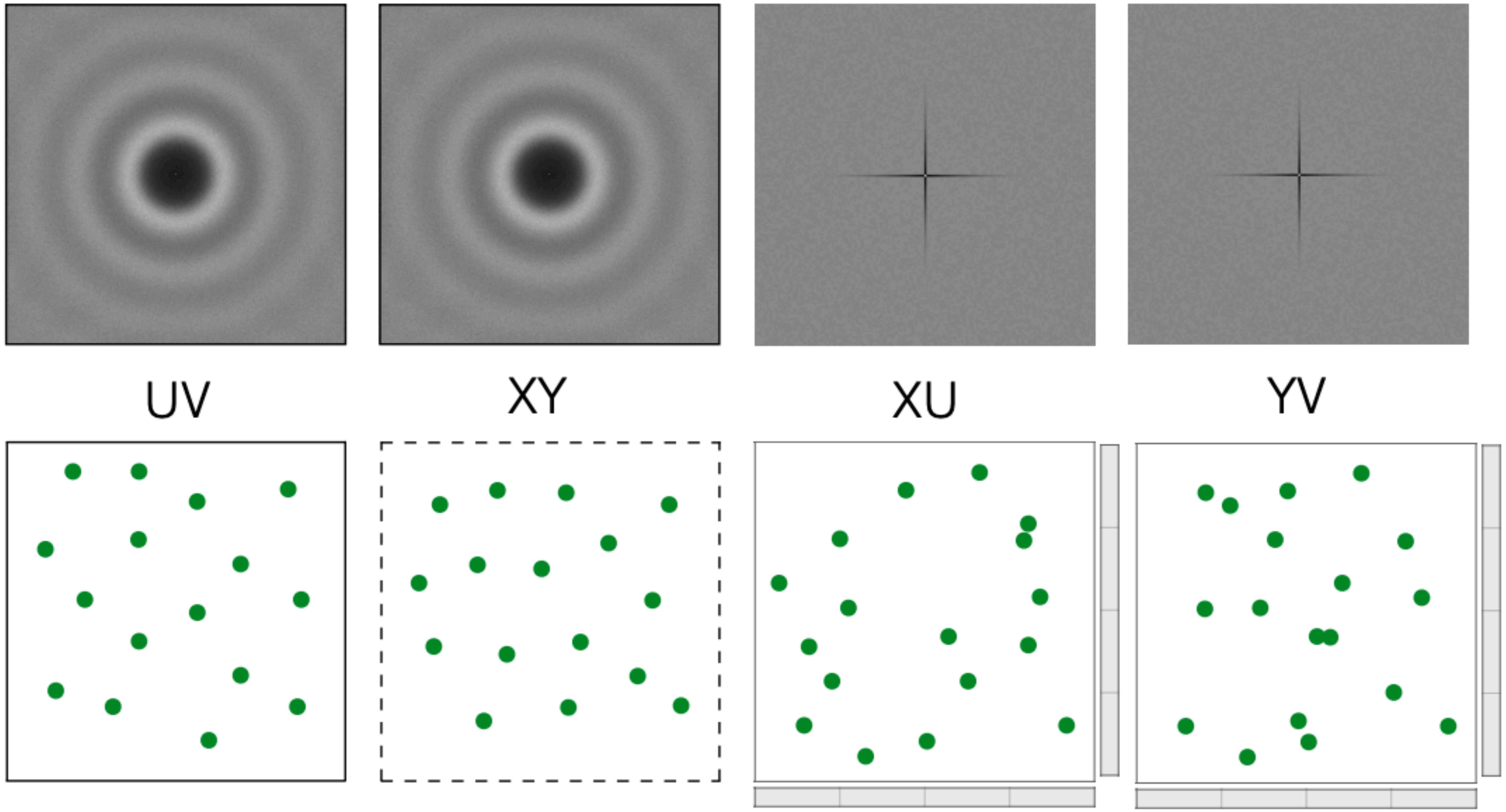
- 2D jittered (x,y) for pixel
- 2D jittered (u,v) for lens
- 1D jittered (t) for time
- combine dimensions in random order



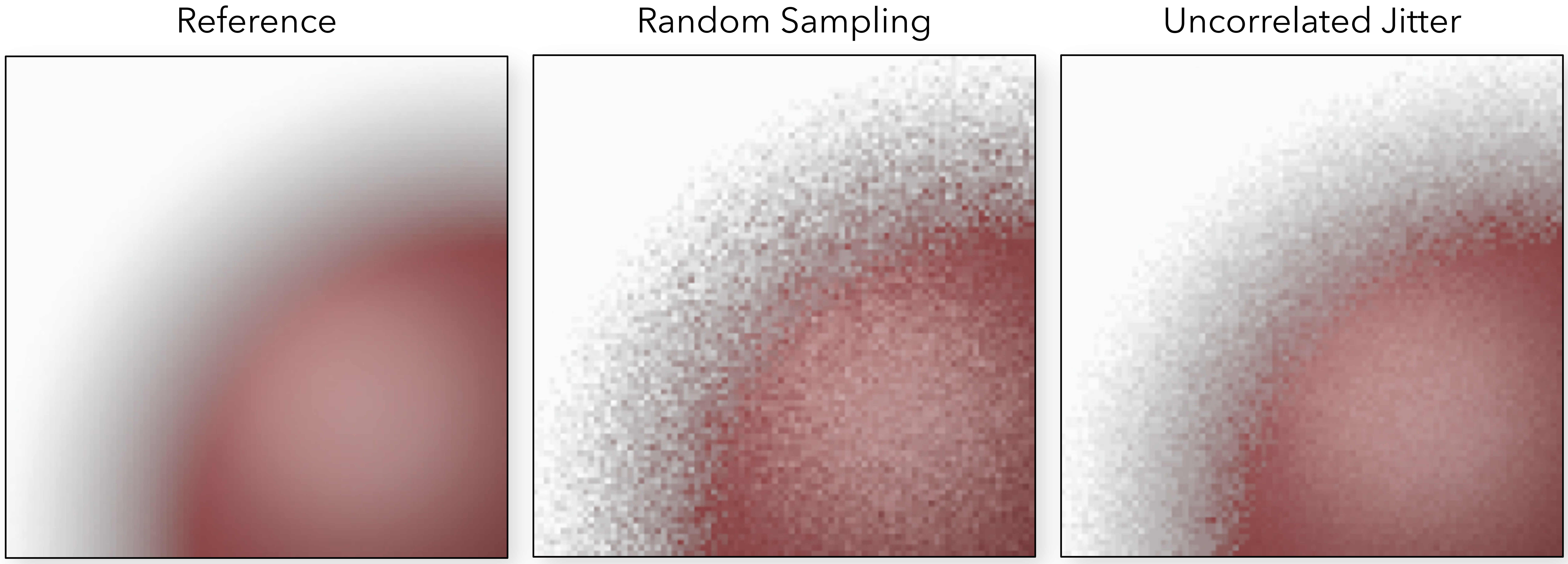
Stochastic Sampling in Computer Graphics

ROBERT L. COOK
Pixar

Not all dimensions are well stratified with uncorrelated jitter



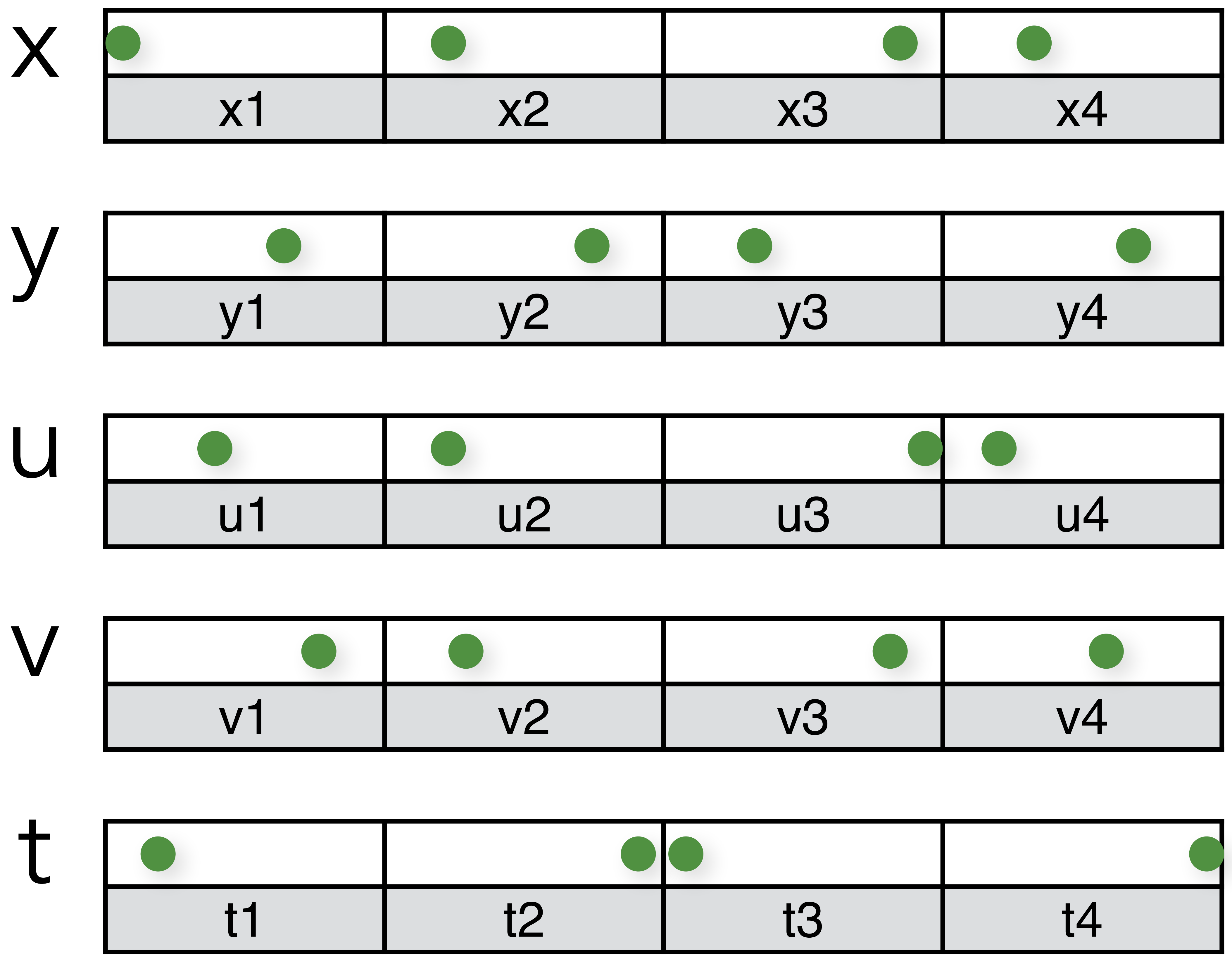
4D integral with uncorrelated jitter



Uncorrelated jitter is a special case of Latin hypercube sampling

Stratify samples in each dimension separately

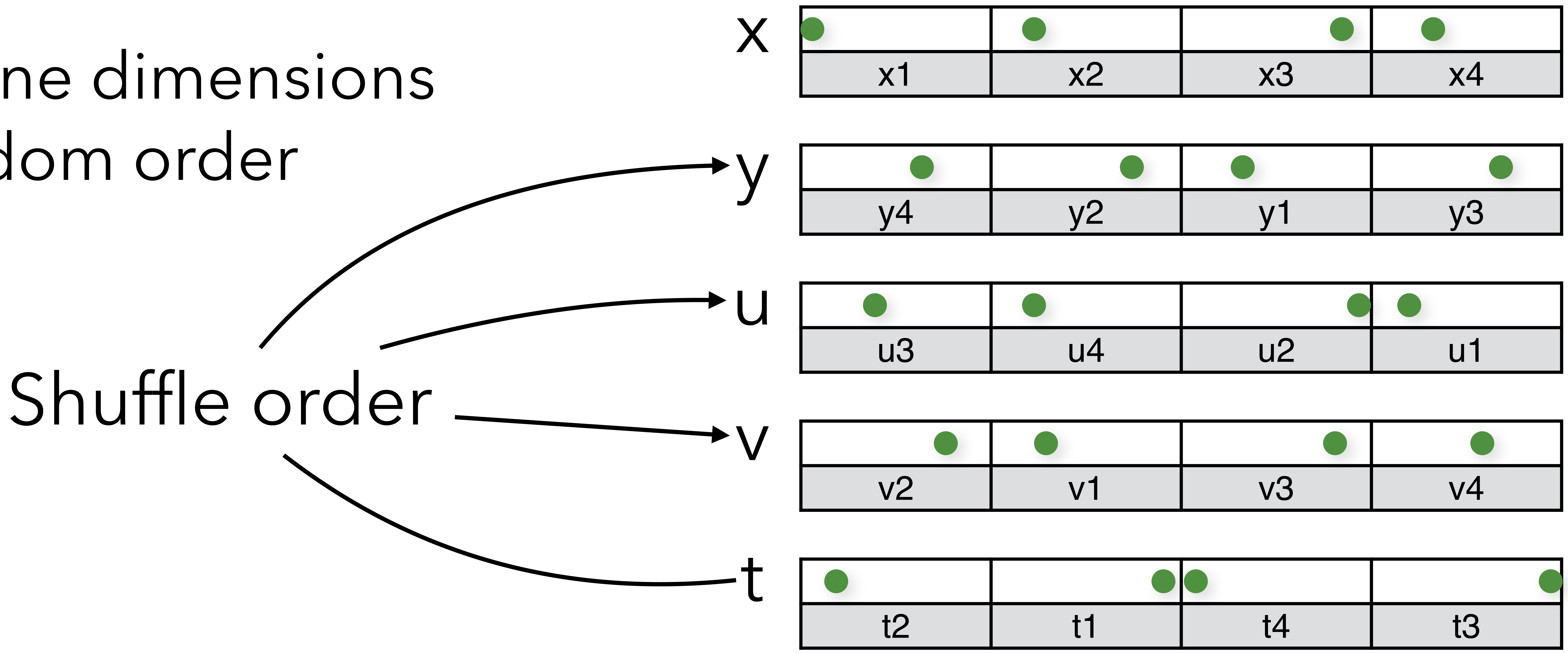
- for 5D: 5 separate 1D jittered point sets
- combine dimensions in random order



Uncorrelated jitter is a special case of Latin hypercube sampling

Stratify samples in each dimension separately

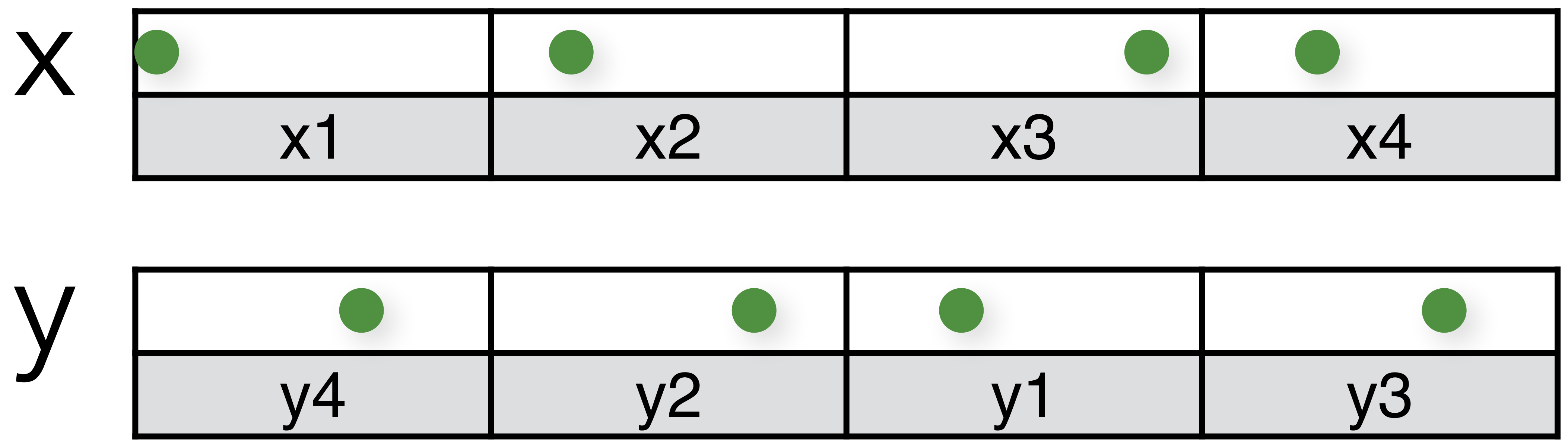
- for 5D: 5 separate 1D jittered point sets
- combine dimensions in random order



N-Rook: 2D version of Latin hypercube

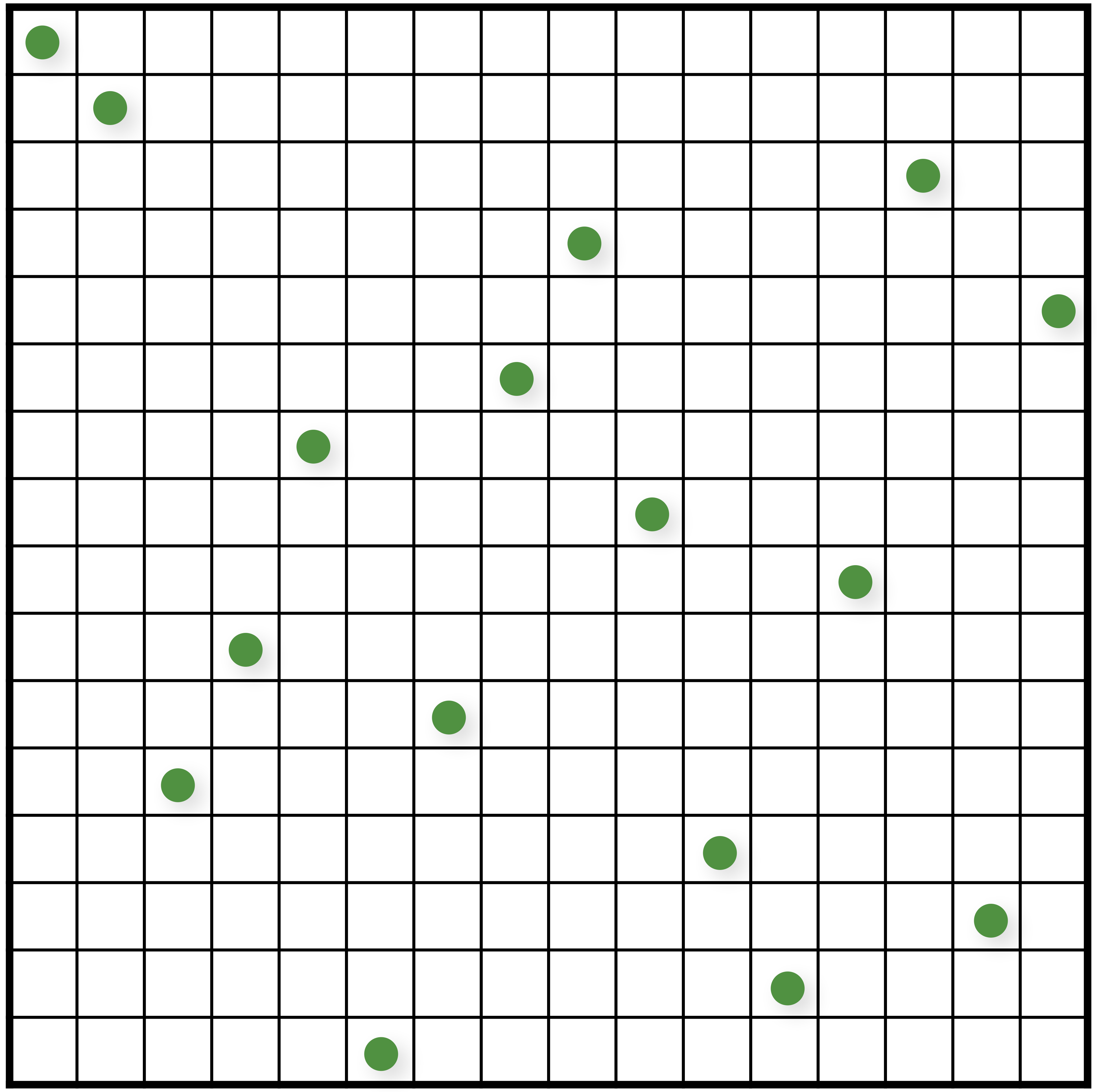
Stratify samples in each dimension separately

- for **2D**: **2** separate 1D jittered point sets
- combine dimensions
  in random order



Latin-Hypercube (N-Rooks) Sampling

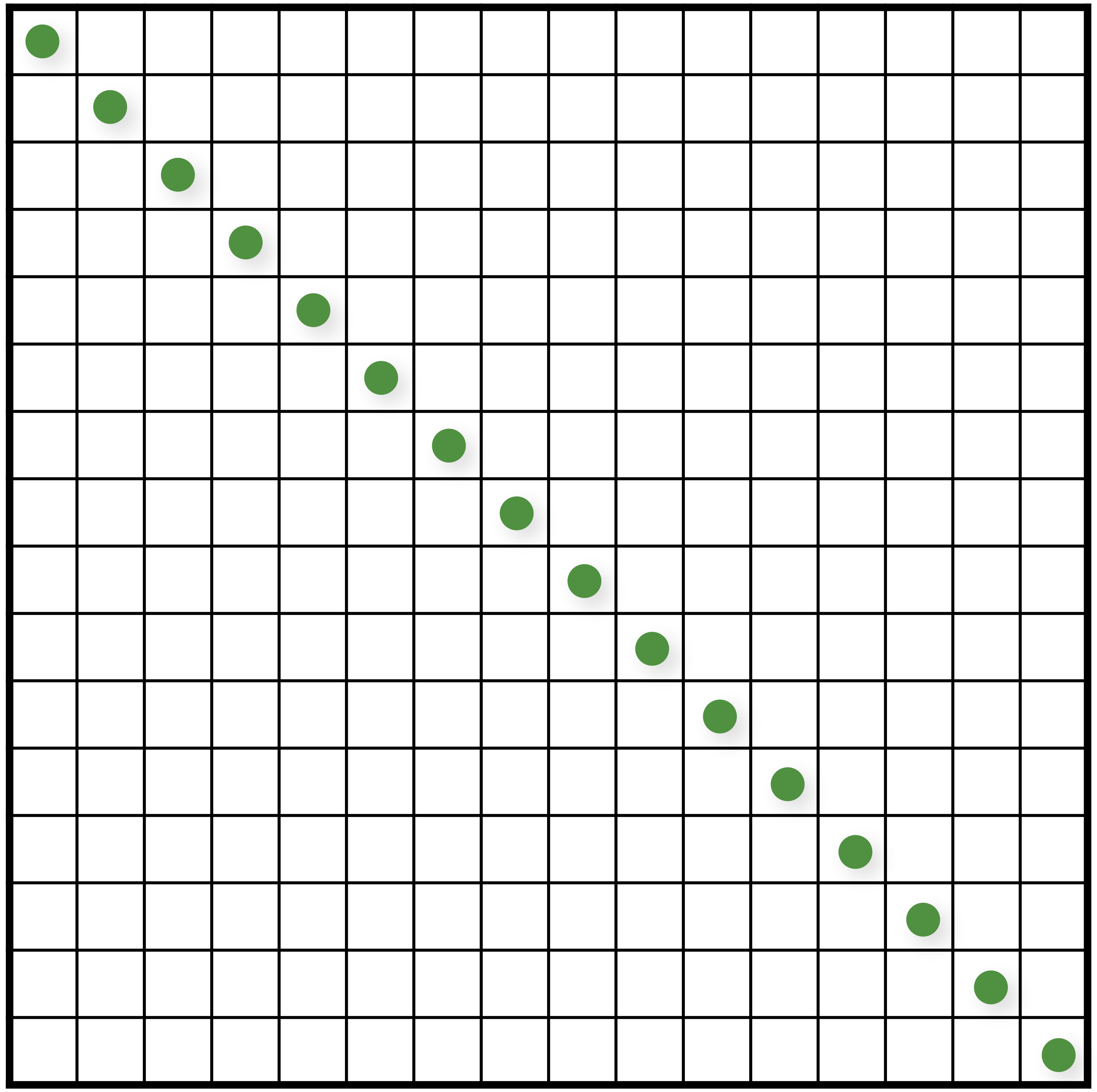
[Shirley 91]



Latin-Hypercube (N-Rooks) Sampling

```
// initialize the diagonal
for (uint d = 0; d < numDimensions; d++)
    for (uint i = 0; i < numS; i++)
        samples(d,i) = (i + randf())/numS;
```

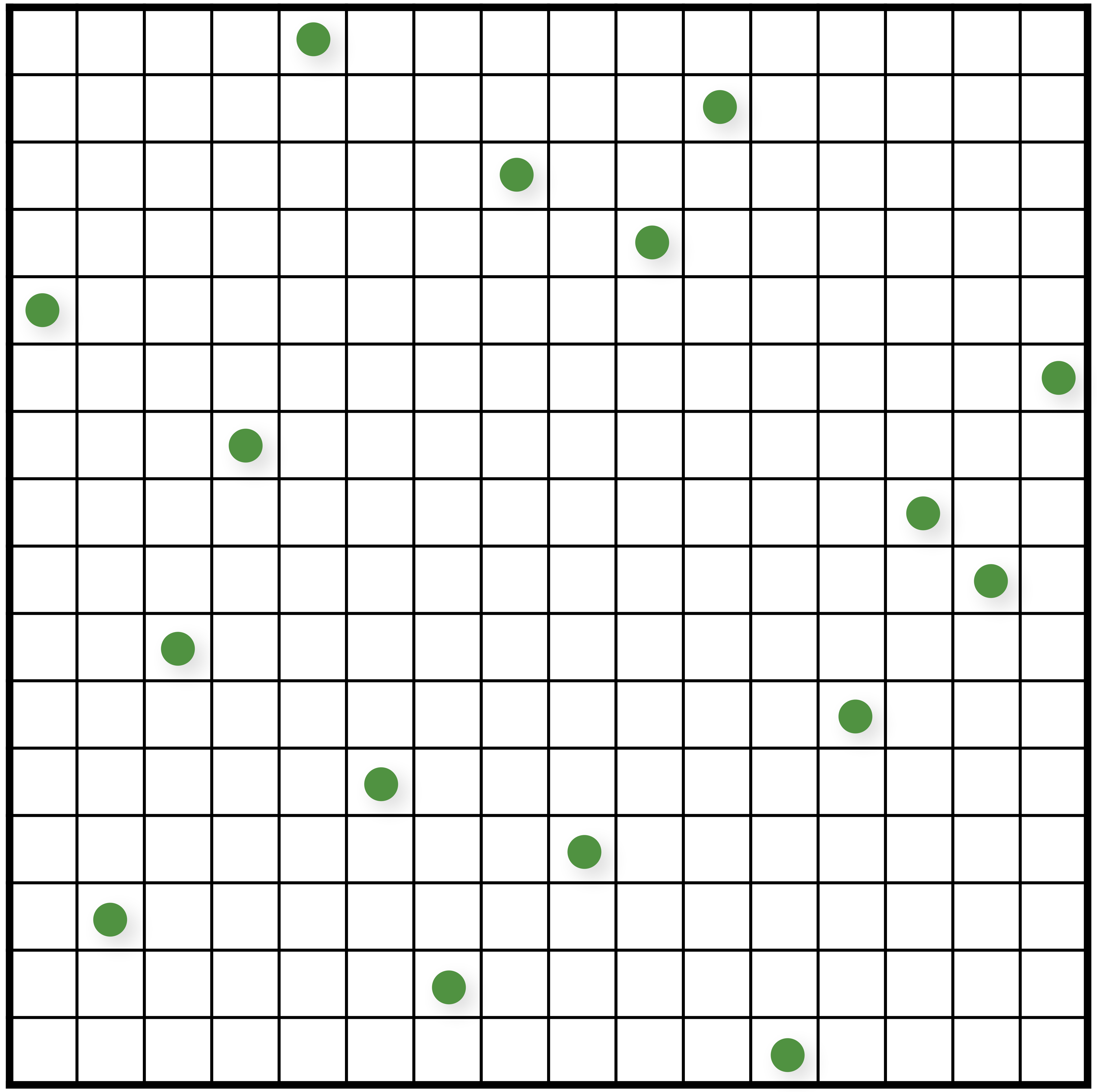
```
// shuffle each dimension independently
for (uint d = 0; d < numDimensions; d++)
    shuffle(samples(d,:));
```



Initialize

Latin-Hypercube (N-Rooks) Sampling

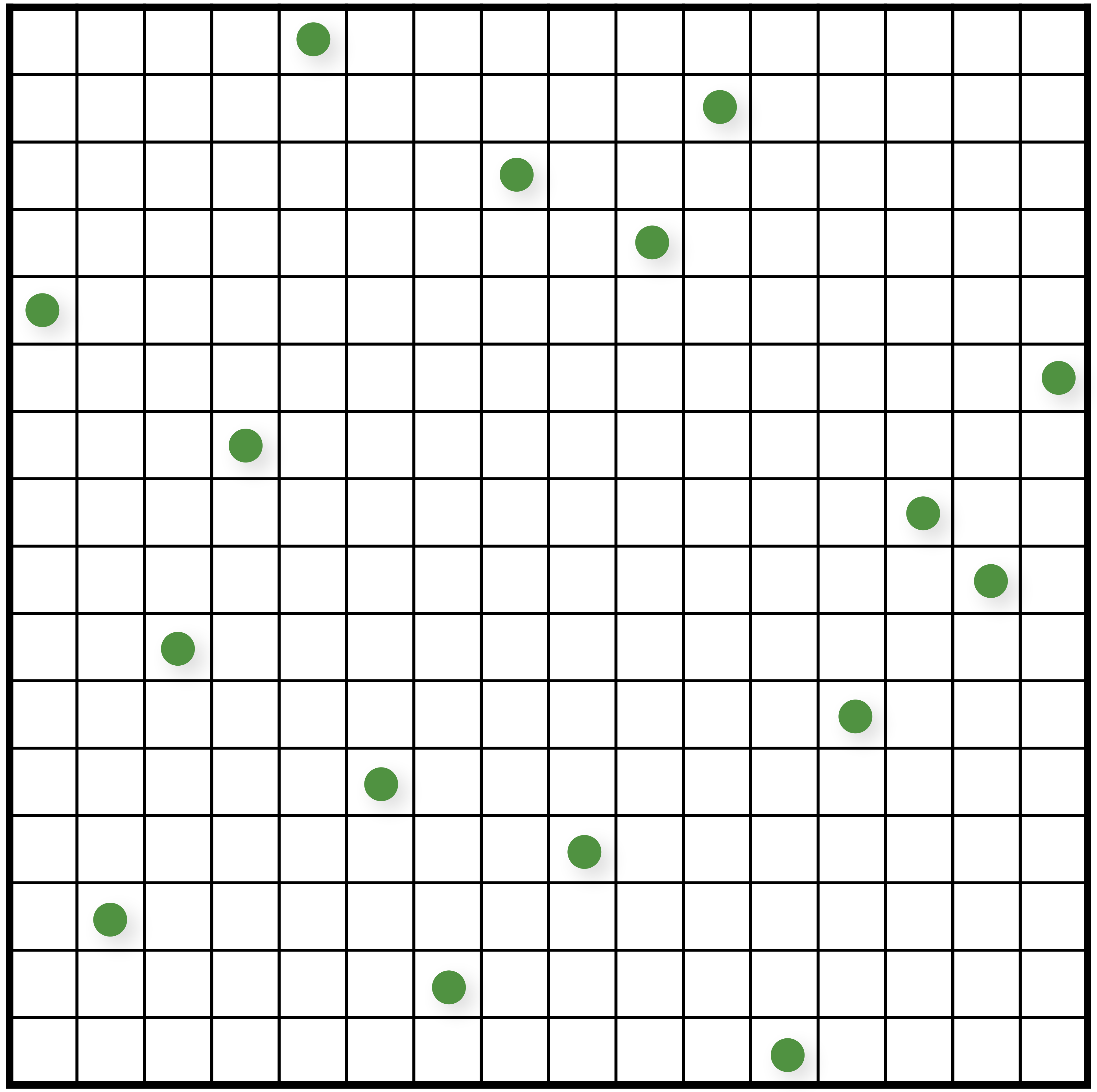
```
// initialize the diagonal
for (uint d = 0; d < numDimensions; d++)
    for (uint i = 0; i < numS; i++)
        samples(d,i) = (i + randf())/numS;

// shuffle each dimension independently
for (uint d = 0; d < numDimensions; d++)
    shuffle(samples(d,:));
```



Shuffle rows

Latin-Hypercube (N-Rooks) Sampling

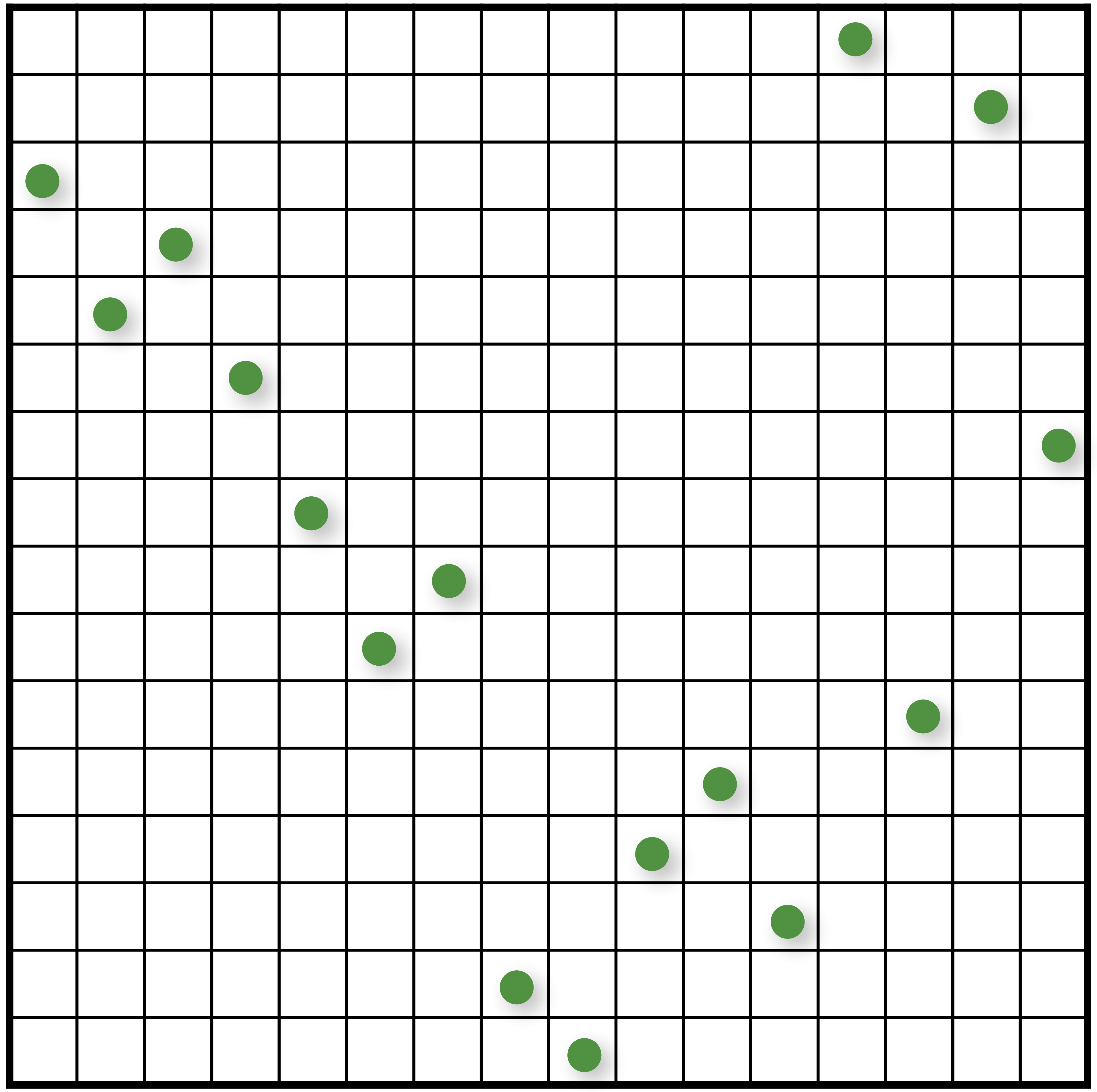
```
// initialize the diagonal
for (uint d = 0; d < numDimensions; d++)
    for (uint i = 0; i < numS; i++)
        samples(d,i) = (i + randf())/numS;

// shuffle each dimension independently
for (uint d = 0; d < numDimensions; d++)
    shuffle(samples(d,:));
```



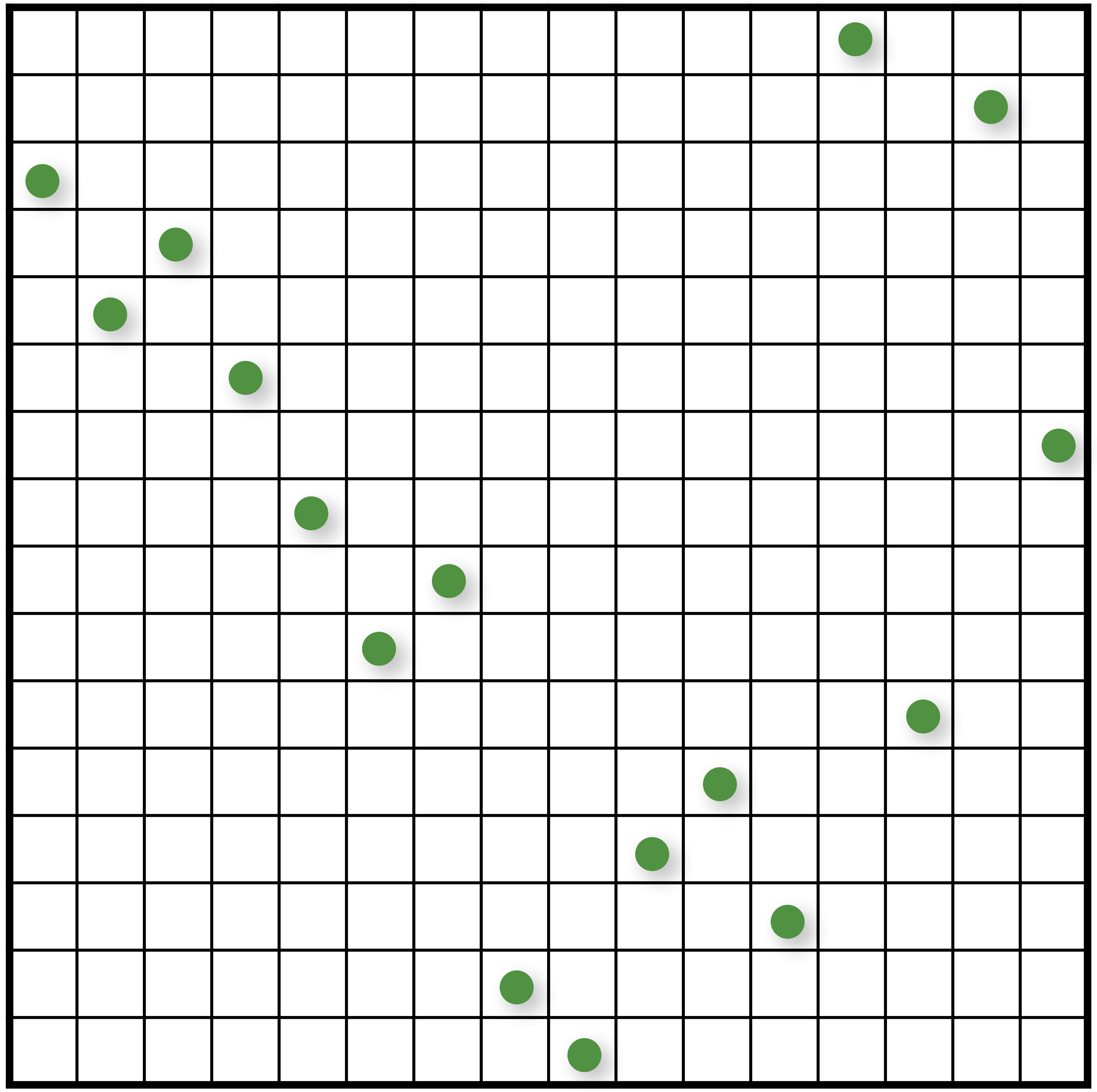
Shuffle columns

Latin-Hypercube (N-Rooks) Sampling

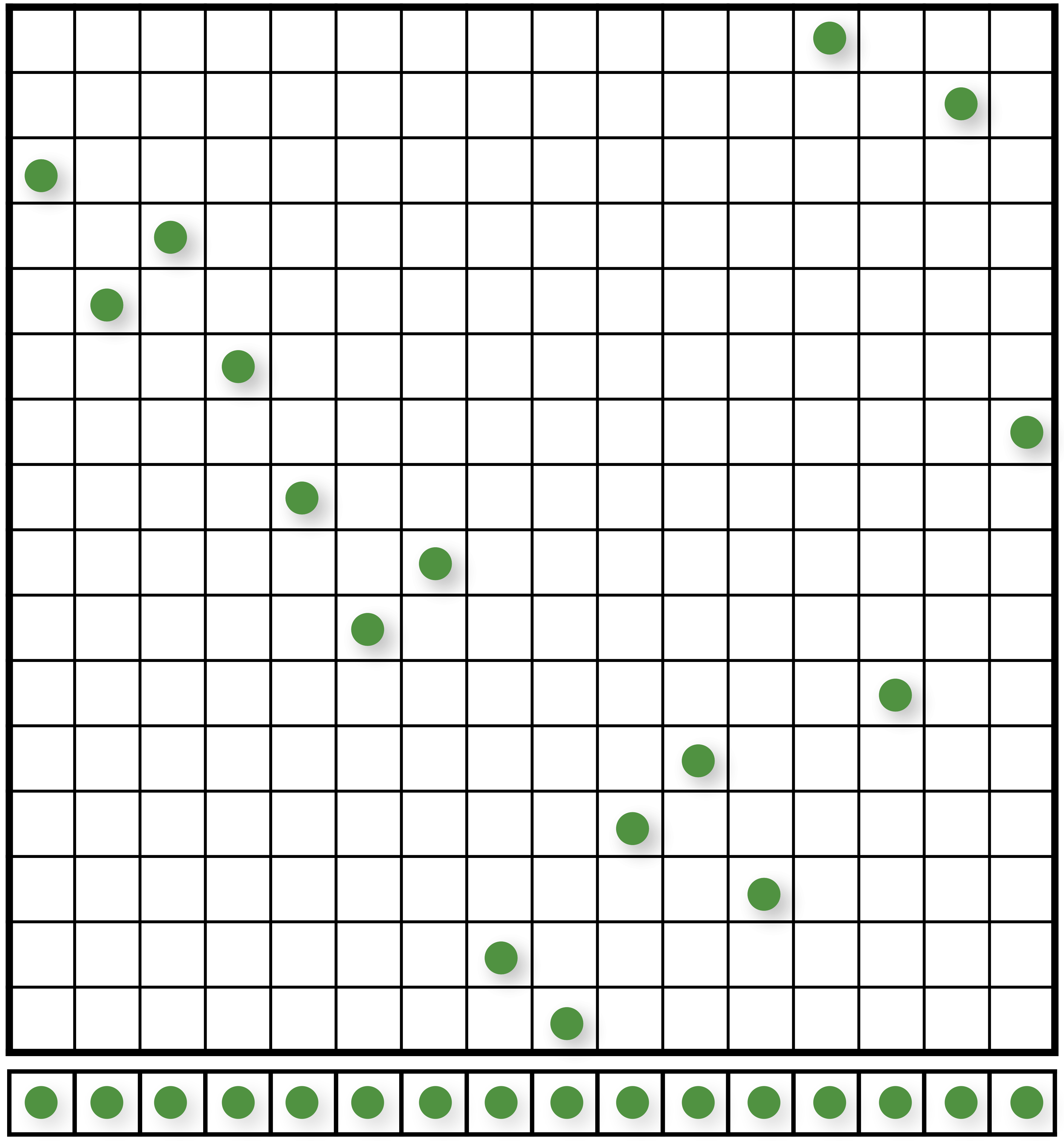
```
// initialize the diagonal
for (uint d = 0; d < numDimensions; d++)
    for (uint i = 0; i < numS; i++)
        samples(d,i) = (i + randf())/numS;

// shuffle each dimension independently
for (uint d = 0; d < numDimensions; d++)
    shuffle(samples(d,:));
```



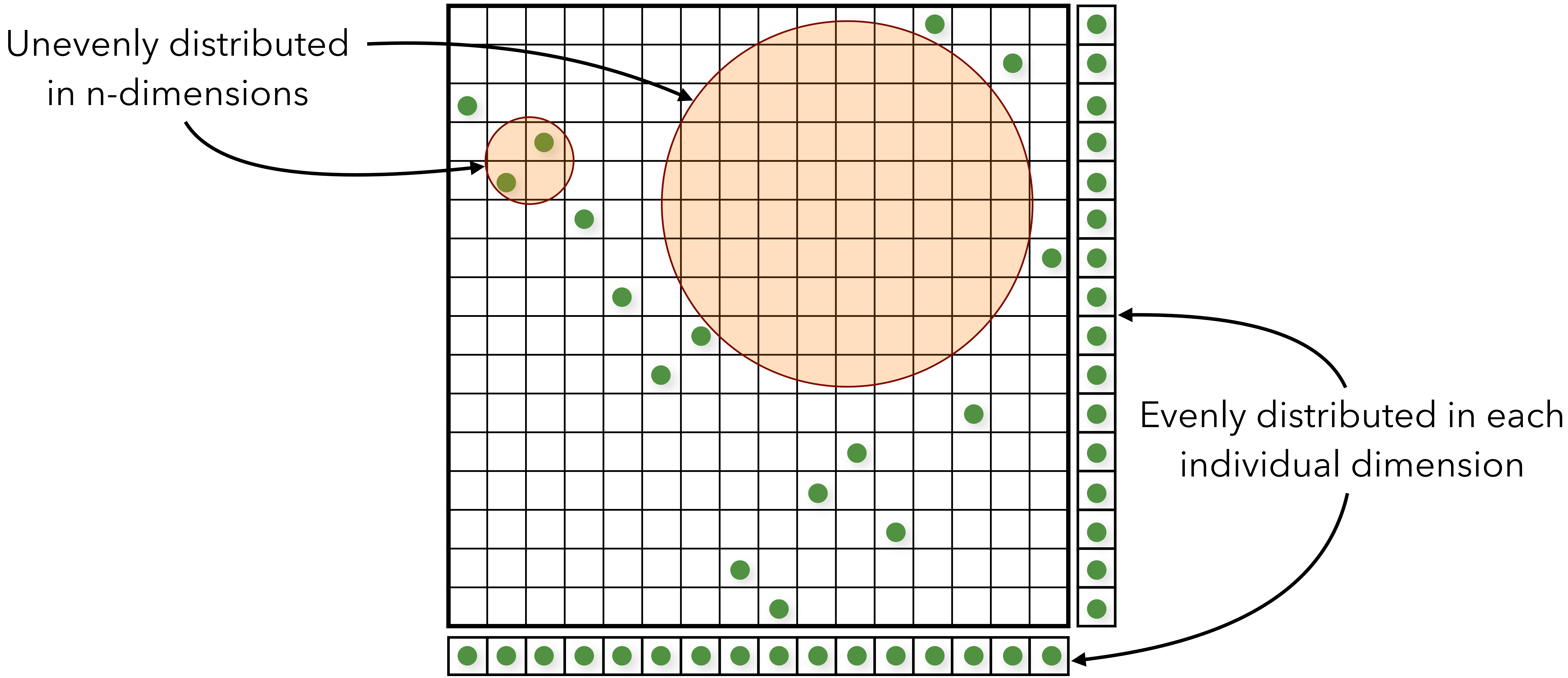
Latin-Hypercube (N-Rooks) Sampling: good 1D projections, gaps in 2D



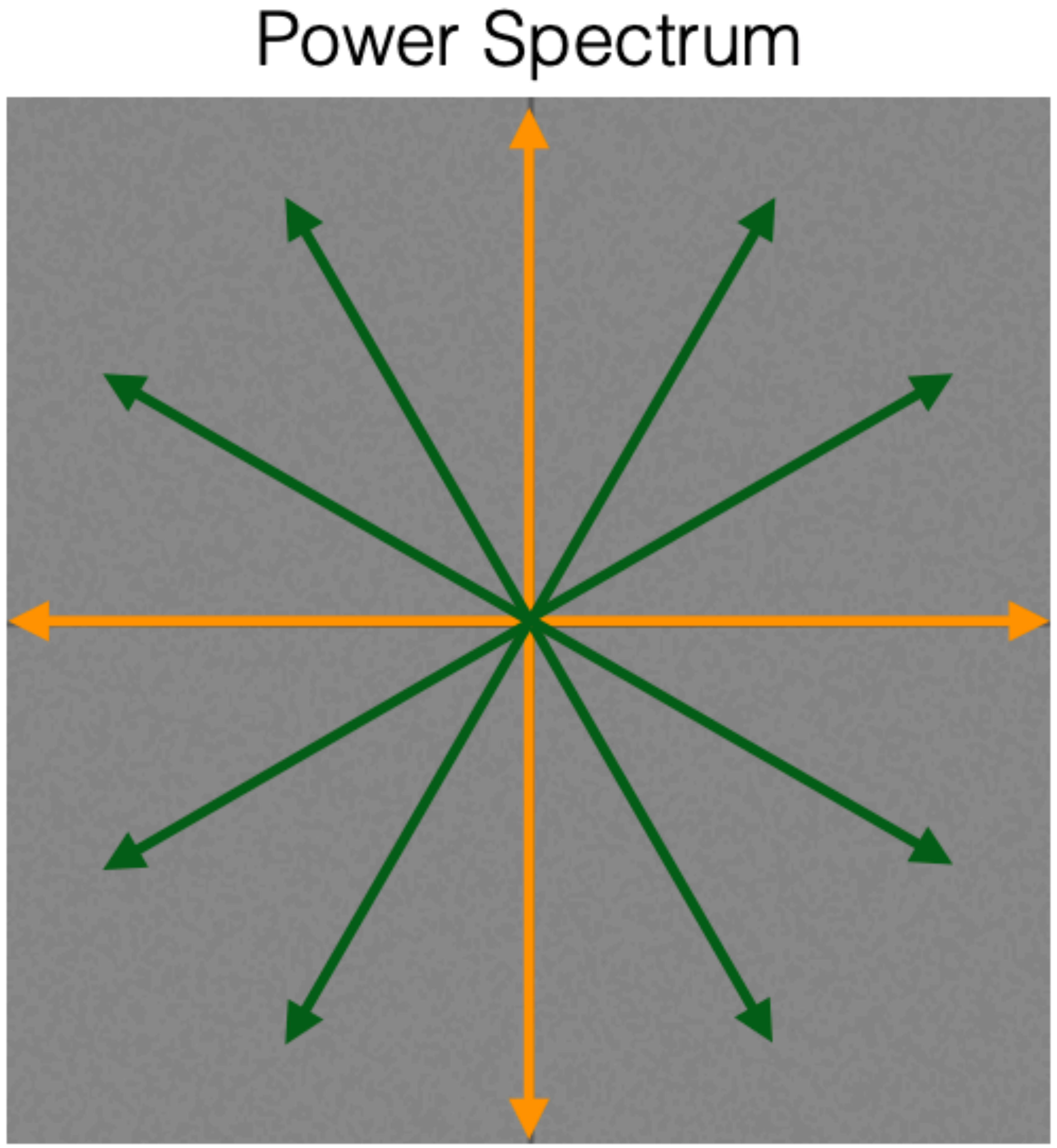
Latin-Hypercube (N-Rooks) Sampling: good 1D projections, gaps in 2D

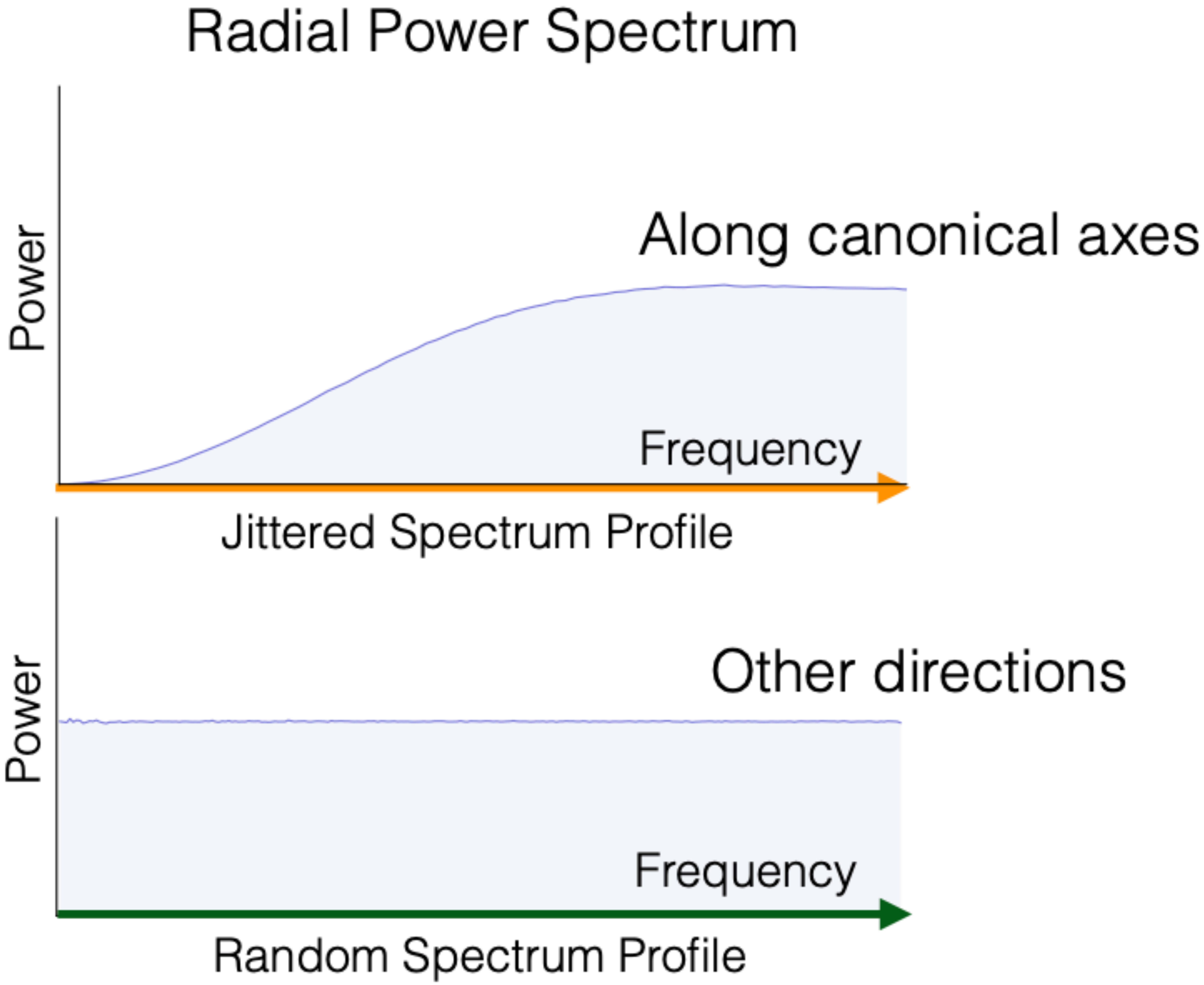


Latin-Hypercube (N-Rooks) Sampling: good 1D projections, gaps in 2D

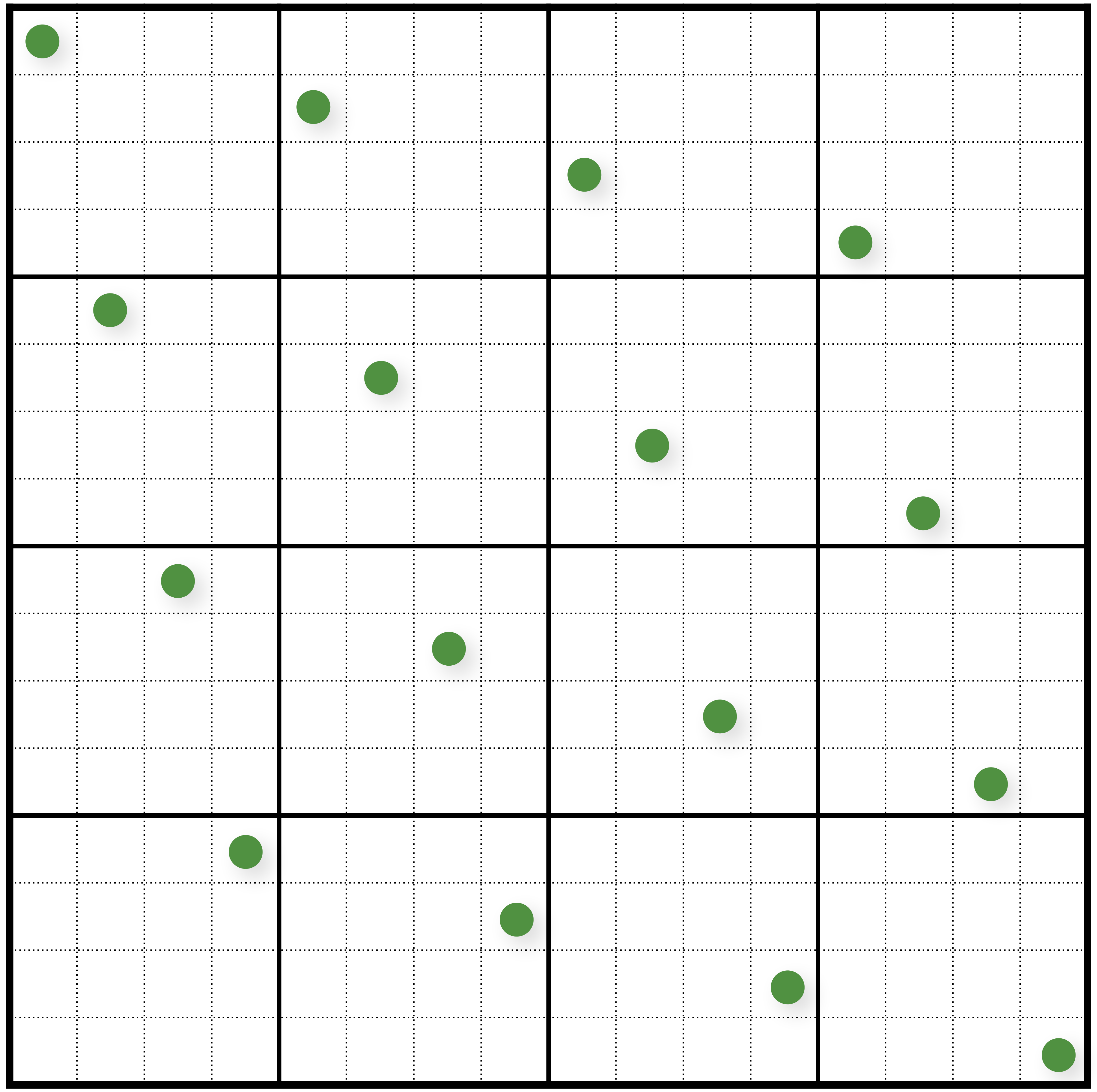


Power spectrum of N-Rooks sampling

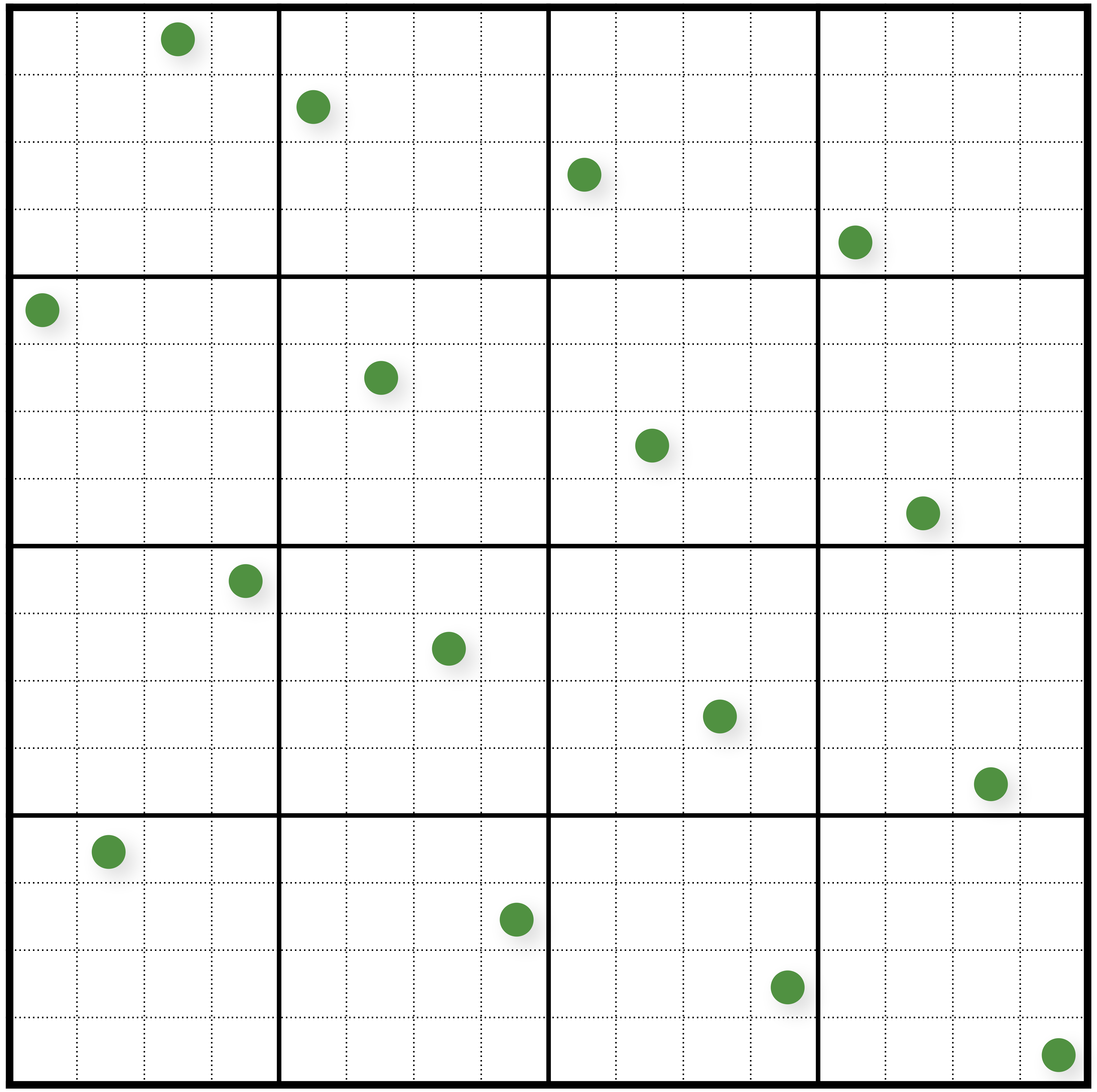




Multi-jittered sampling [Chiu 1994]

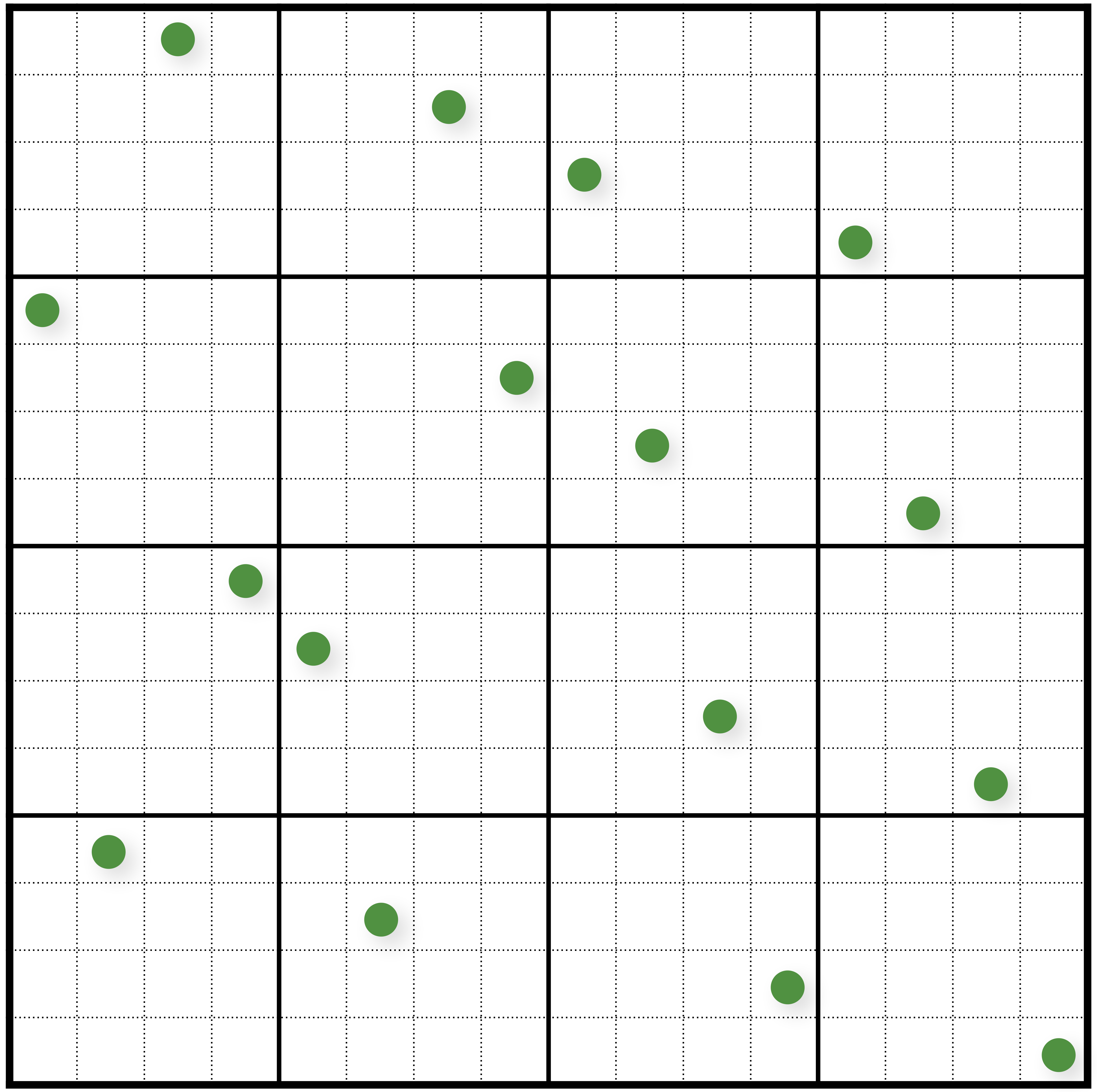


Multi-jittered sampling [Chiu 1994]



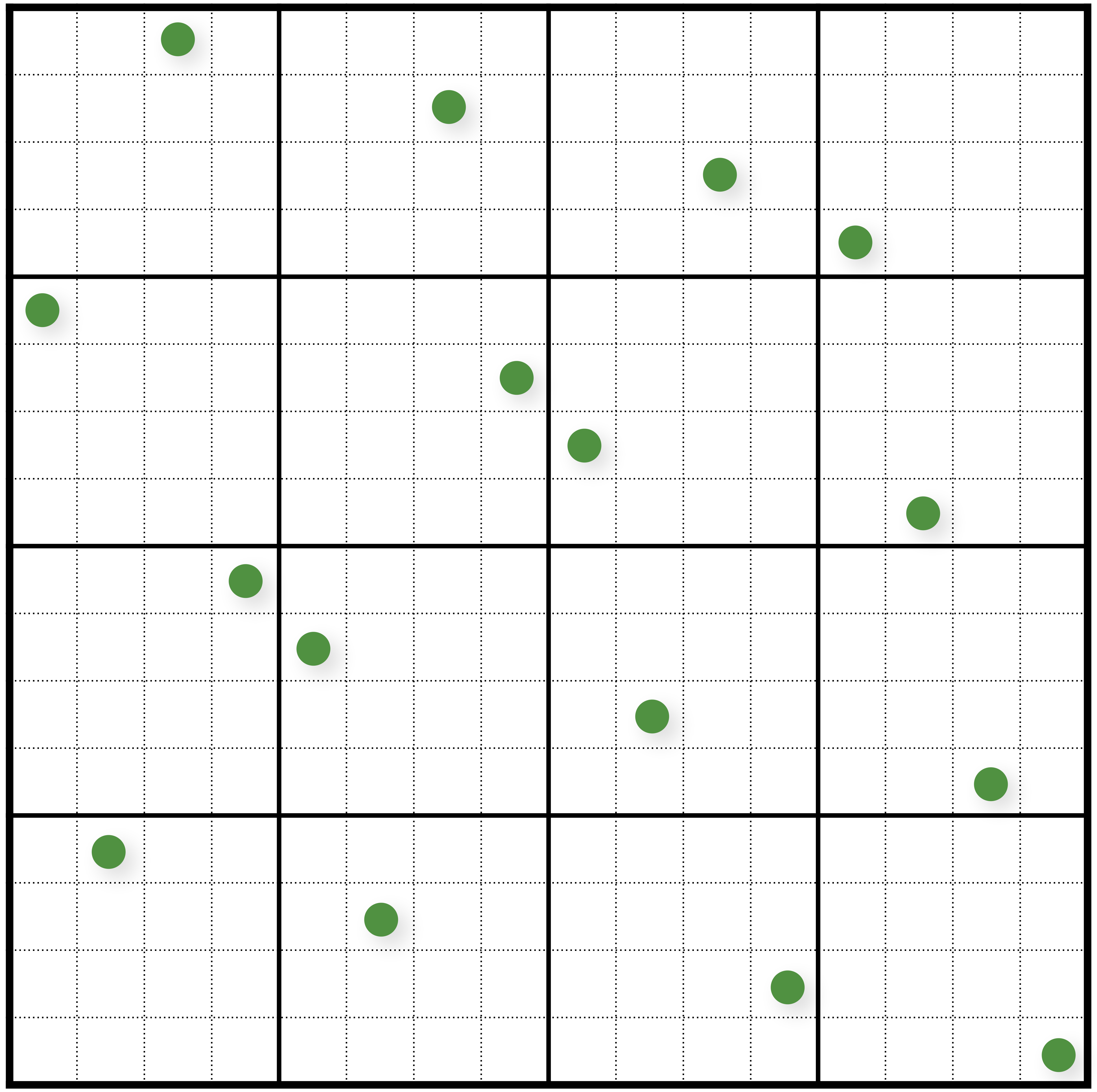
Shuffle x-coords

Multi-jittered sampling [Chiu 1994]



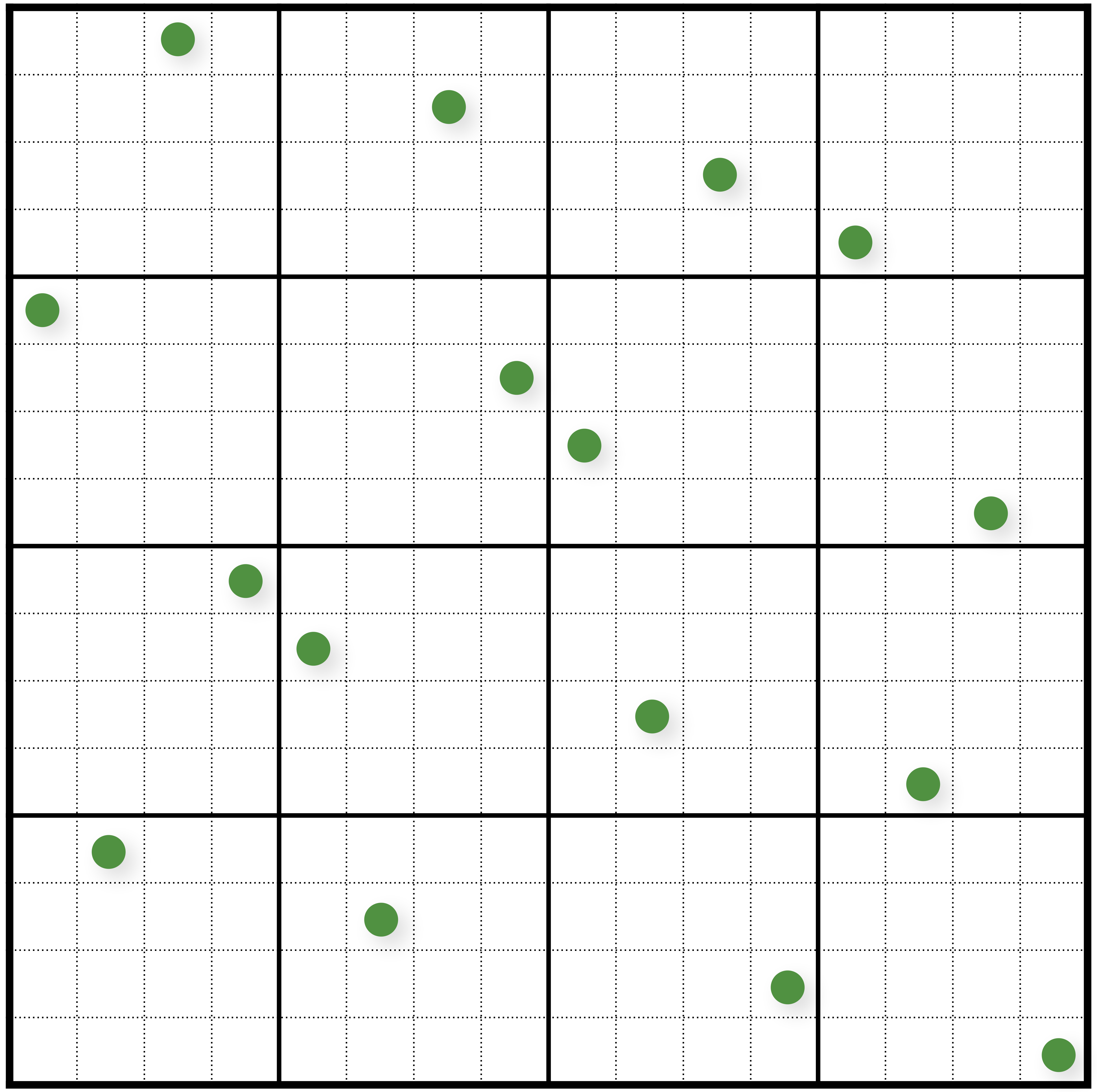
Shuffle x-coords

Multi-jittered sampling [Chiu 1994]



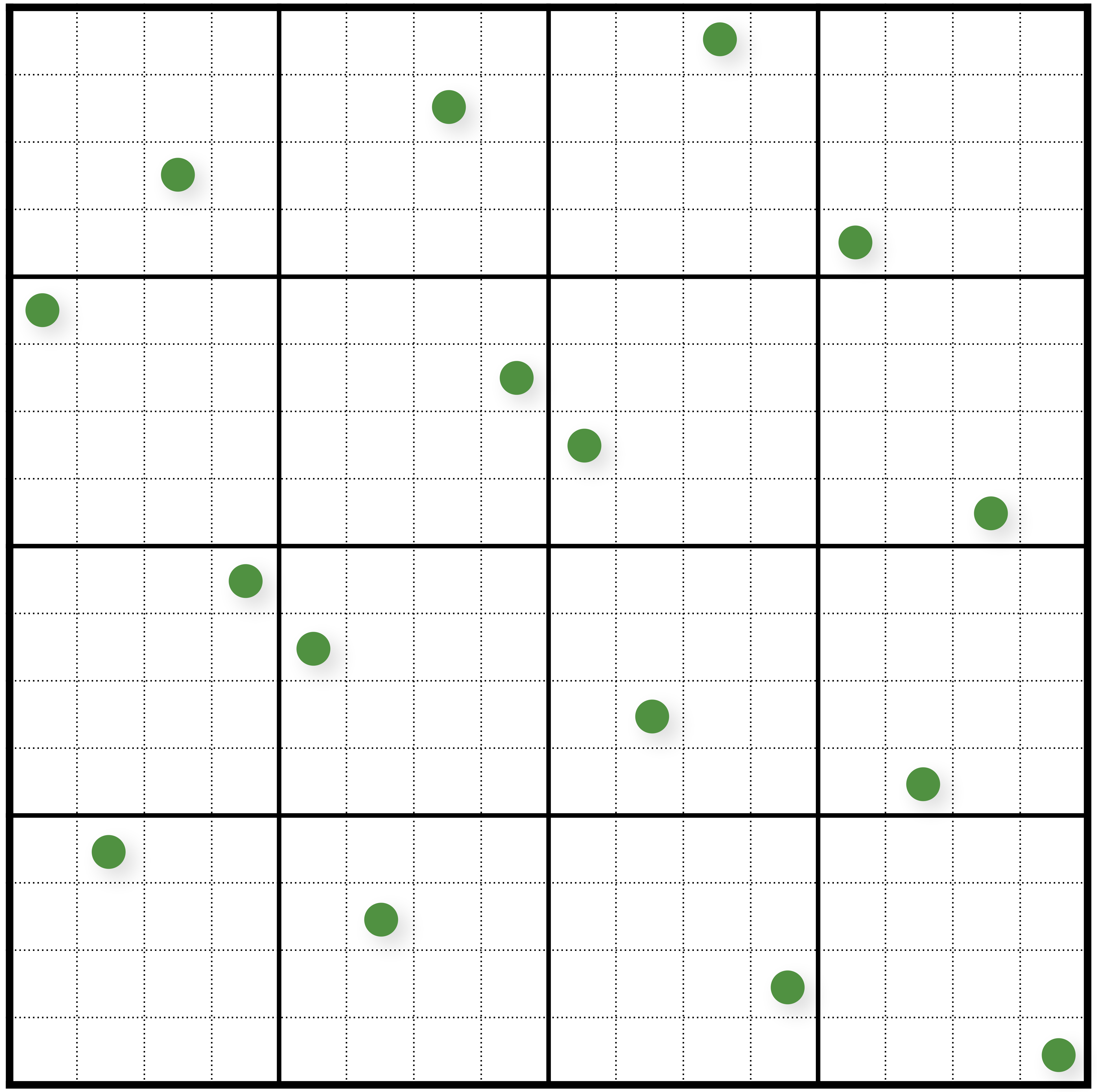
Shuffle x-coords

Multi-jittered sampling [Chiu 1994]



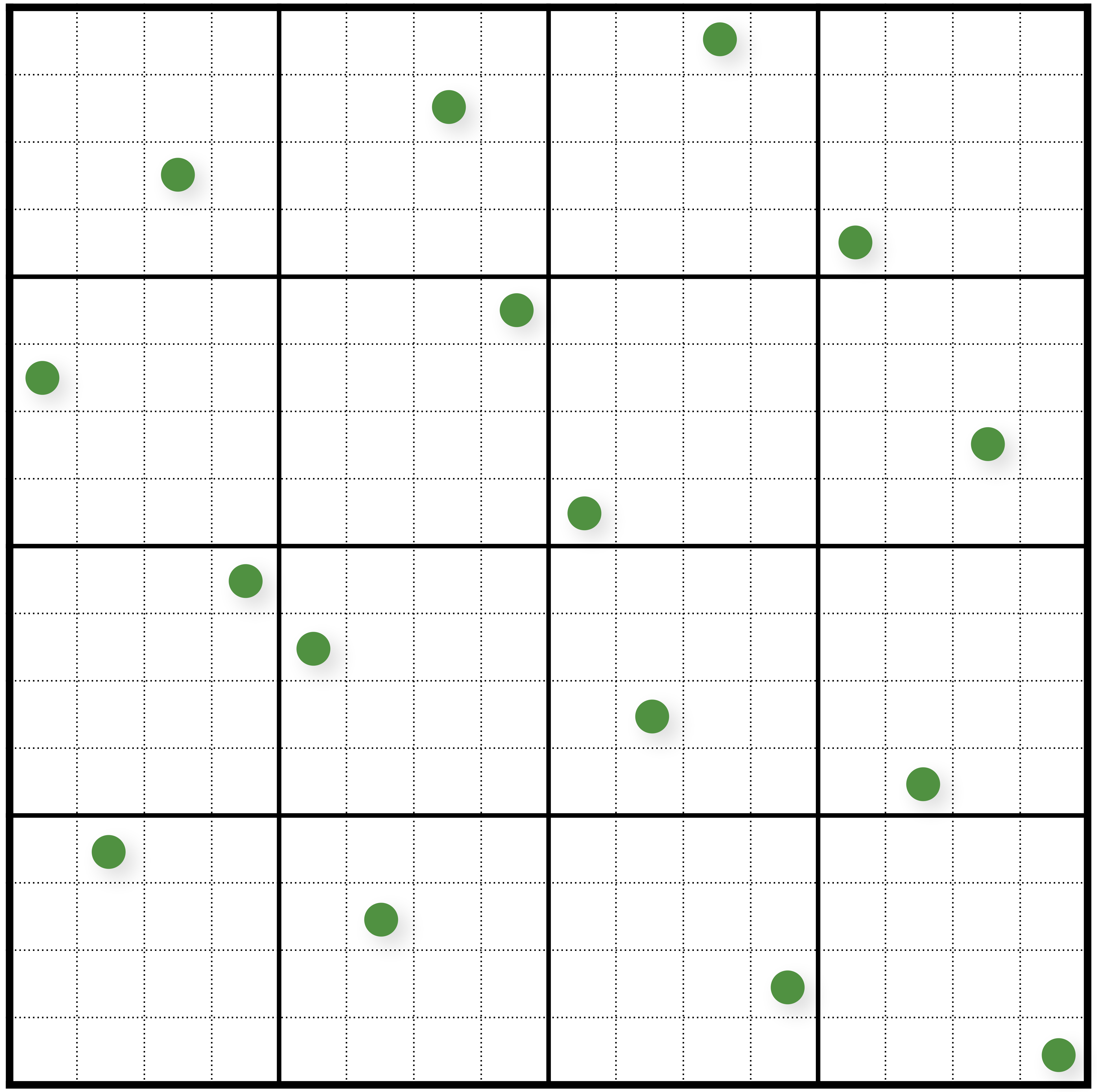
Shuffle x-coords

Multi-jittered sampling [Chiu 1994]



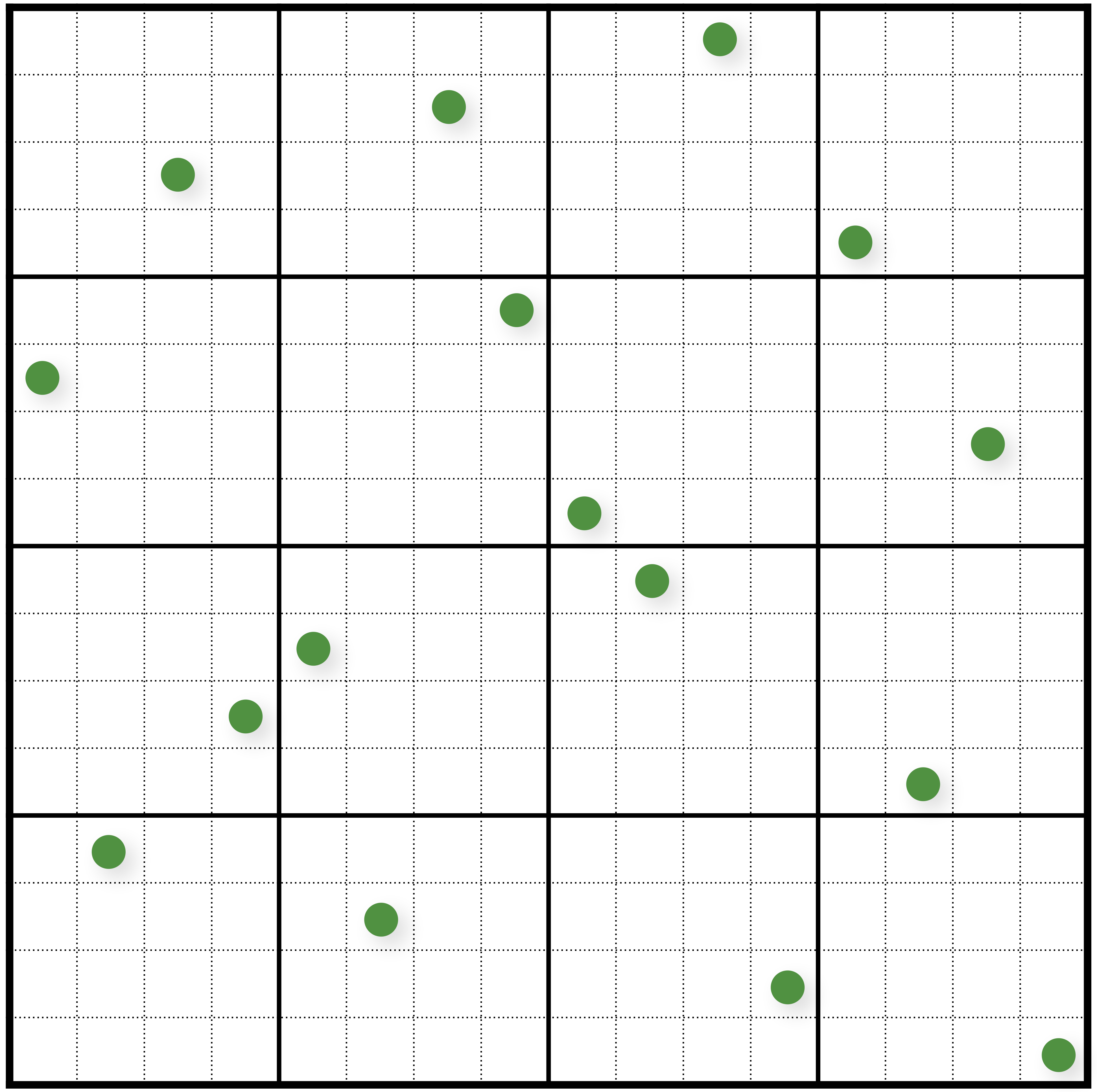
Shuffle y-coords

Multi-jittered sampling [Chiu 1994]



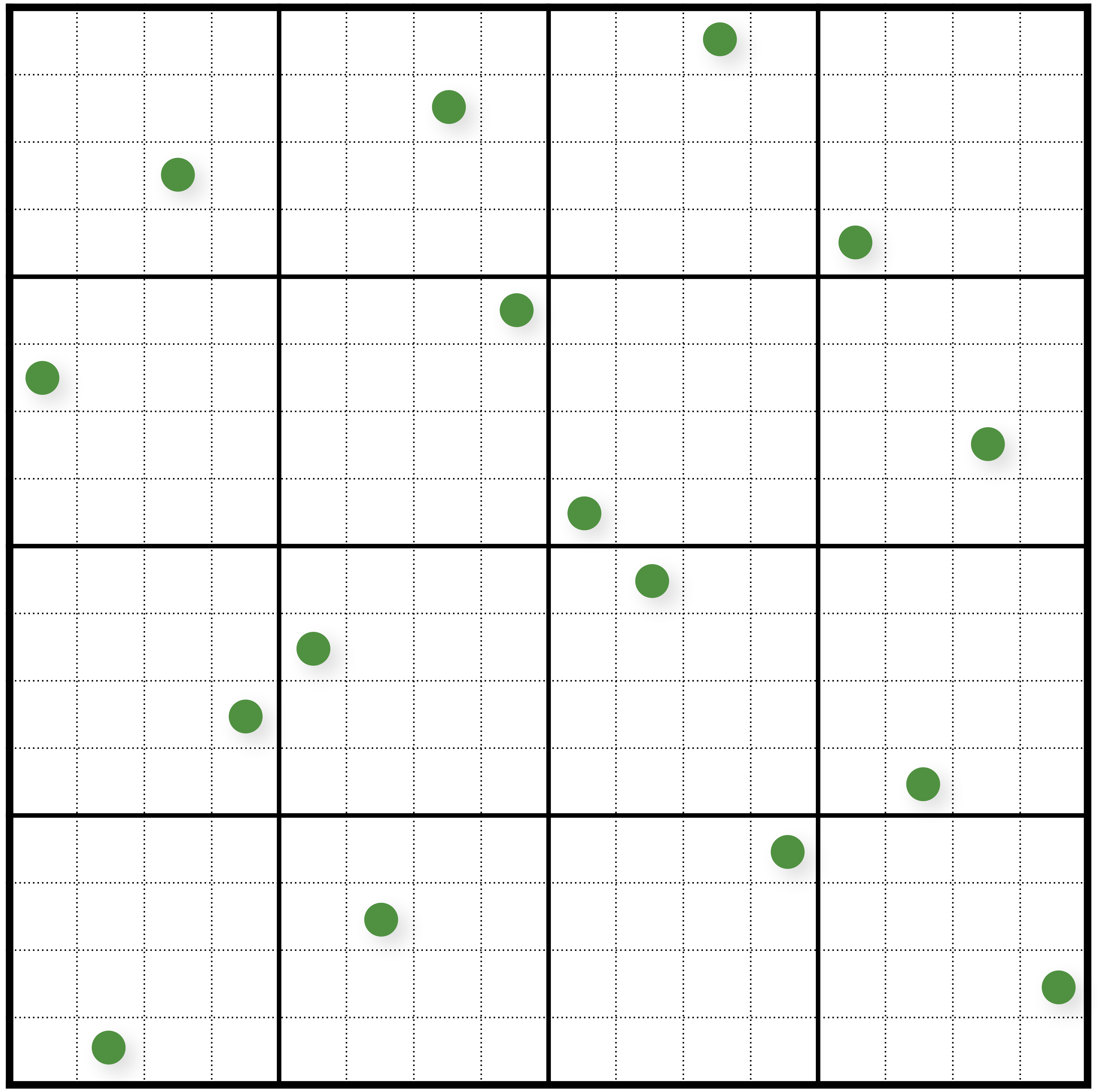
Shuffle y-coords

Multi-jittered sampling [Chiu 1994]



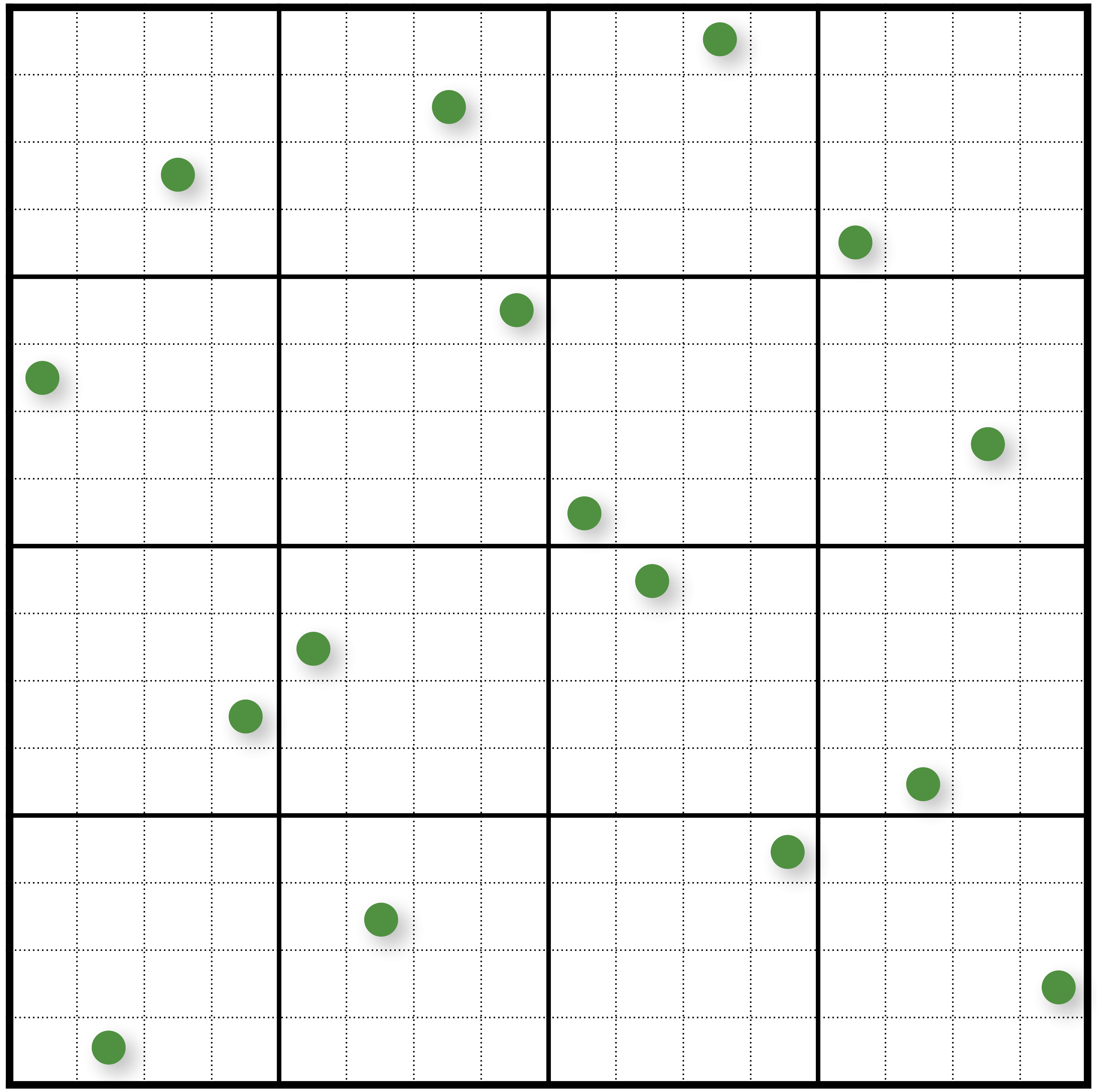
Shuffle y-coords

Multi-jittered sampling [Chiu 1994]

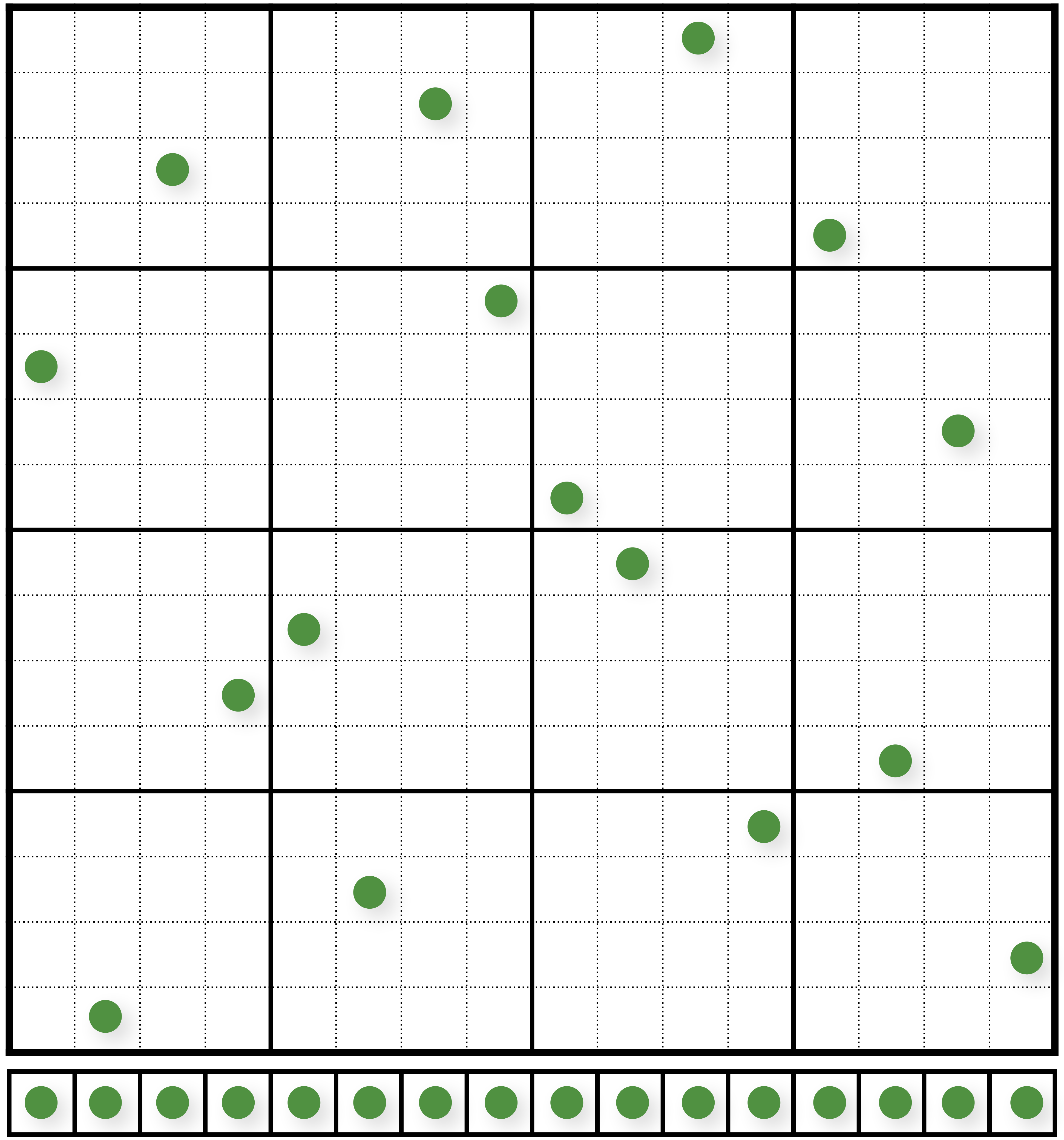


Shuffle y-coords

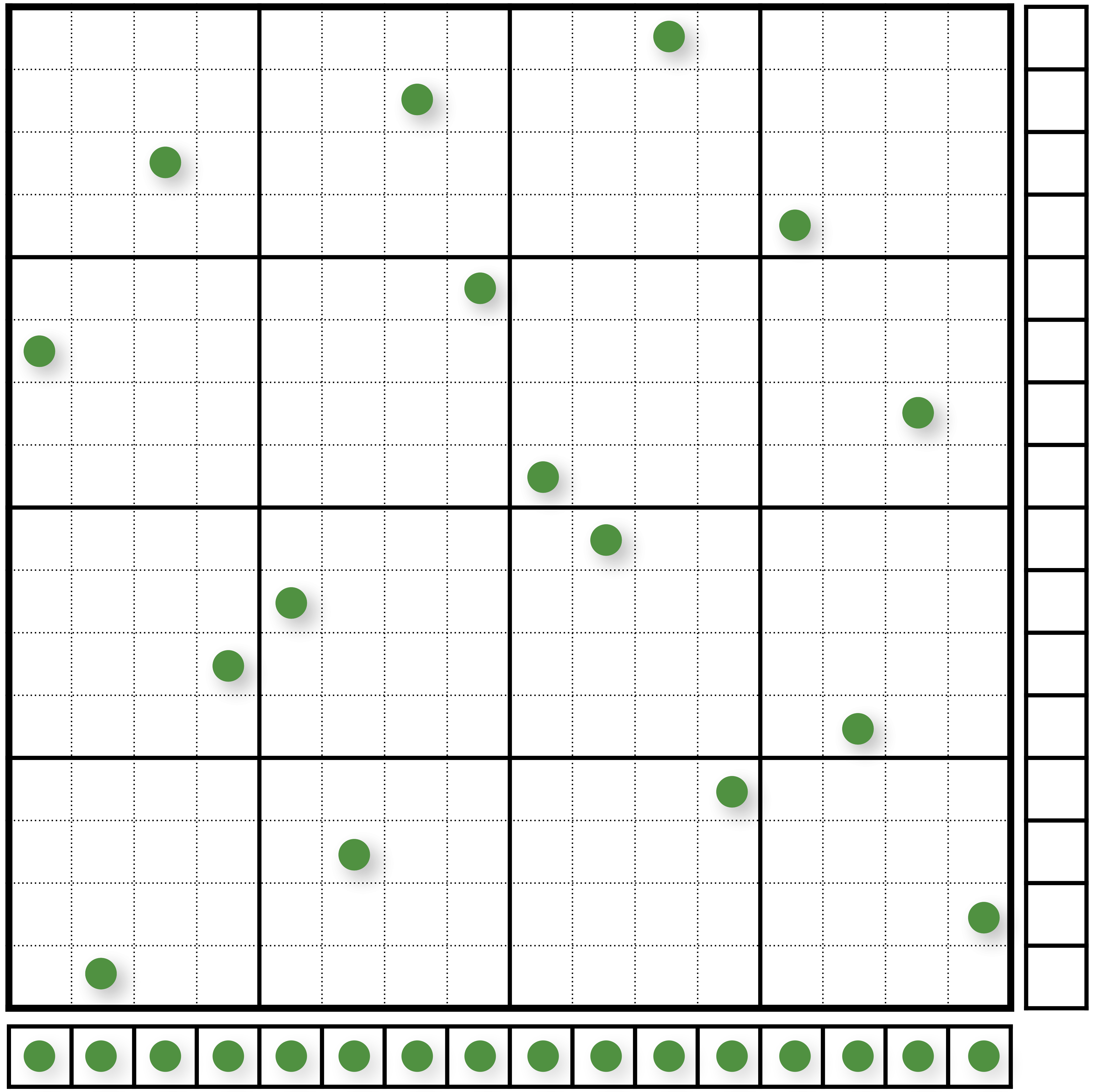
Multi-jittered sampling [Chiu 1994]



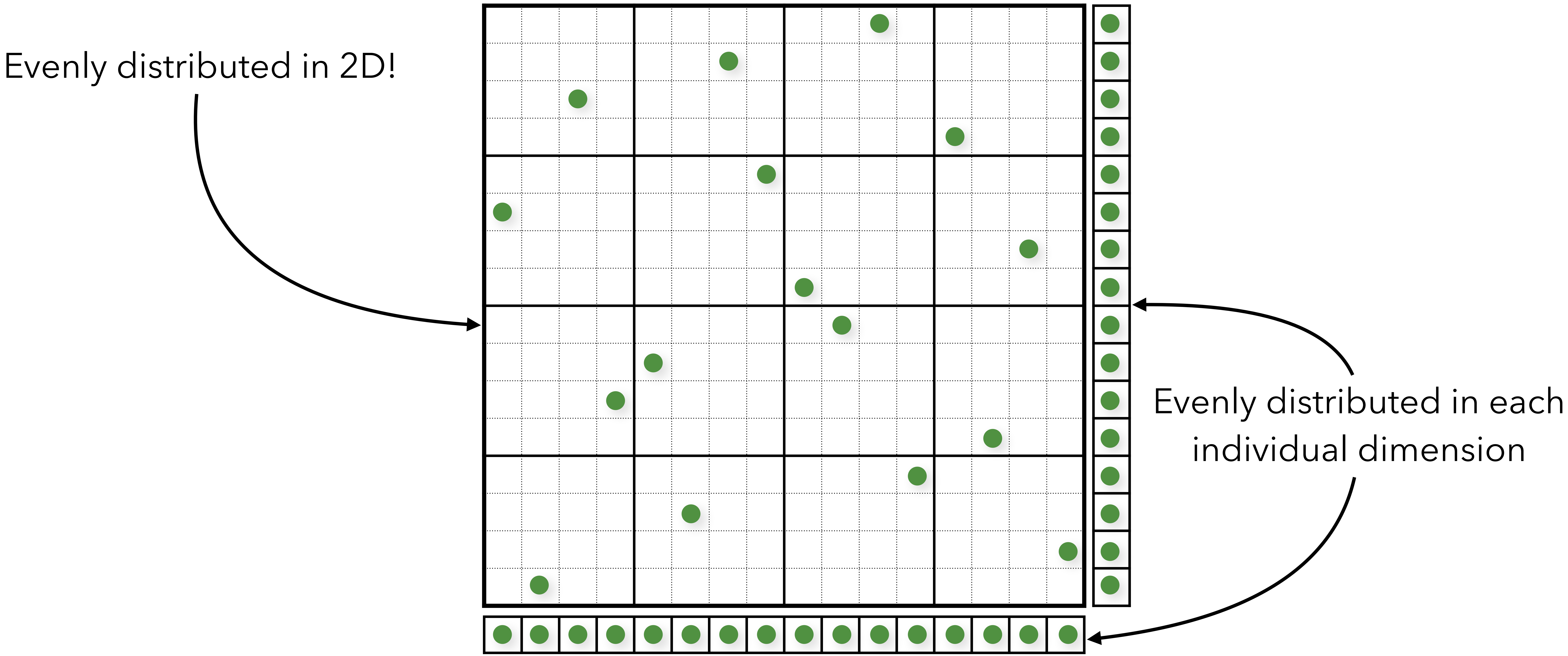
Multi-jittered sampling [Chiu 1994]



Multi-jittered sampling [Chiu 1994]

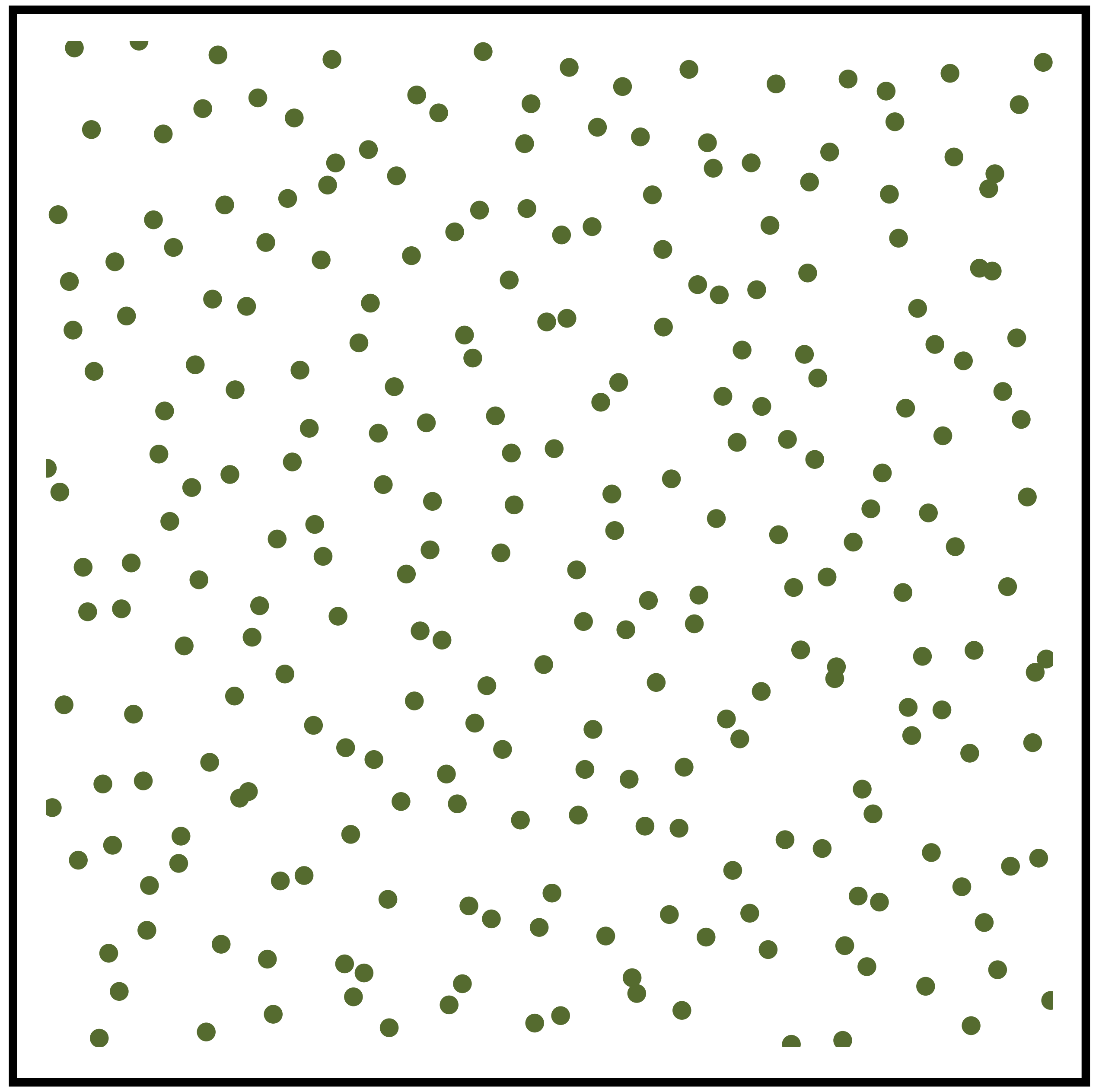


Multi-jittered sampling [Chiu 1994]

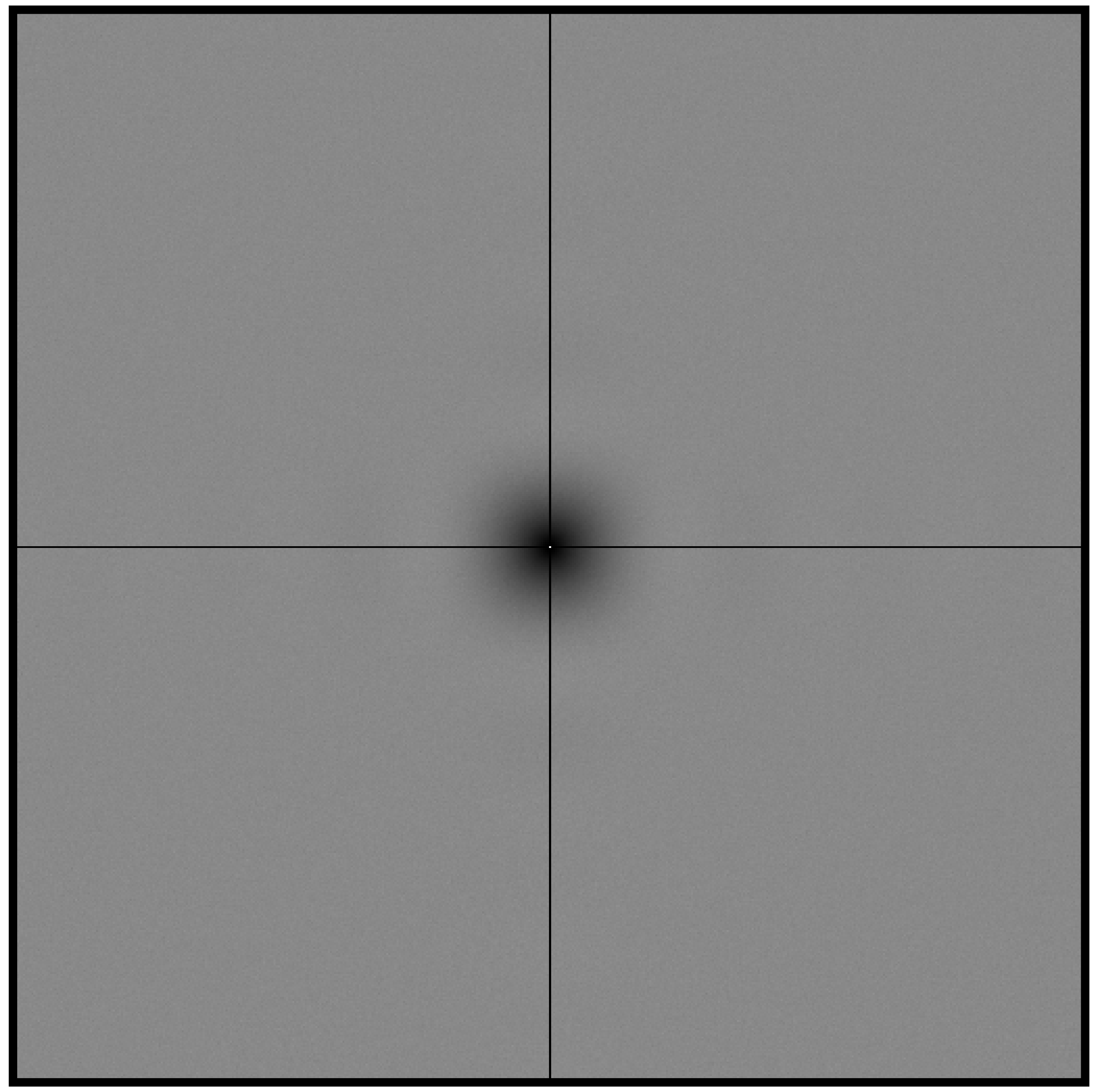


Power spectrum of multi-jittered sampling

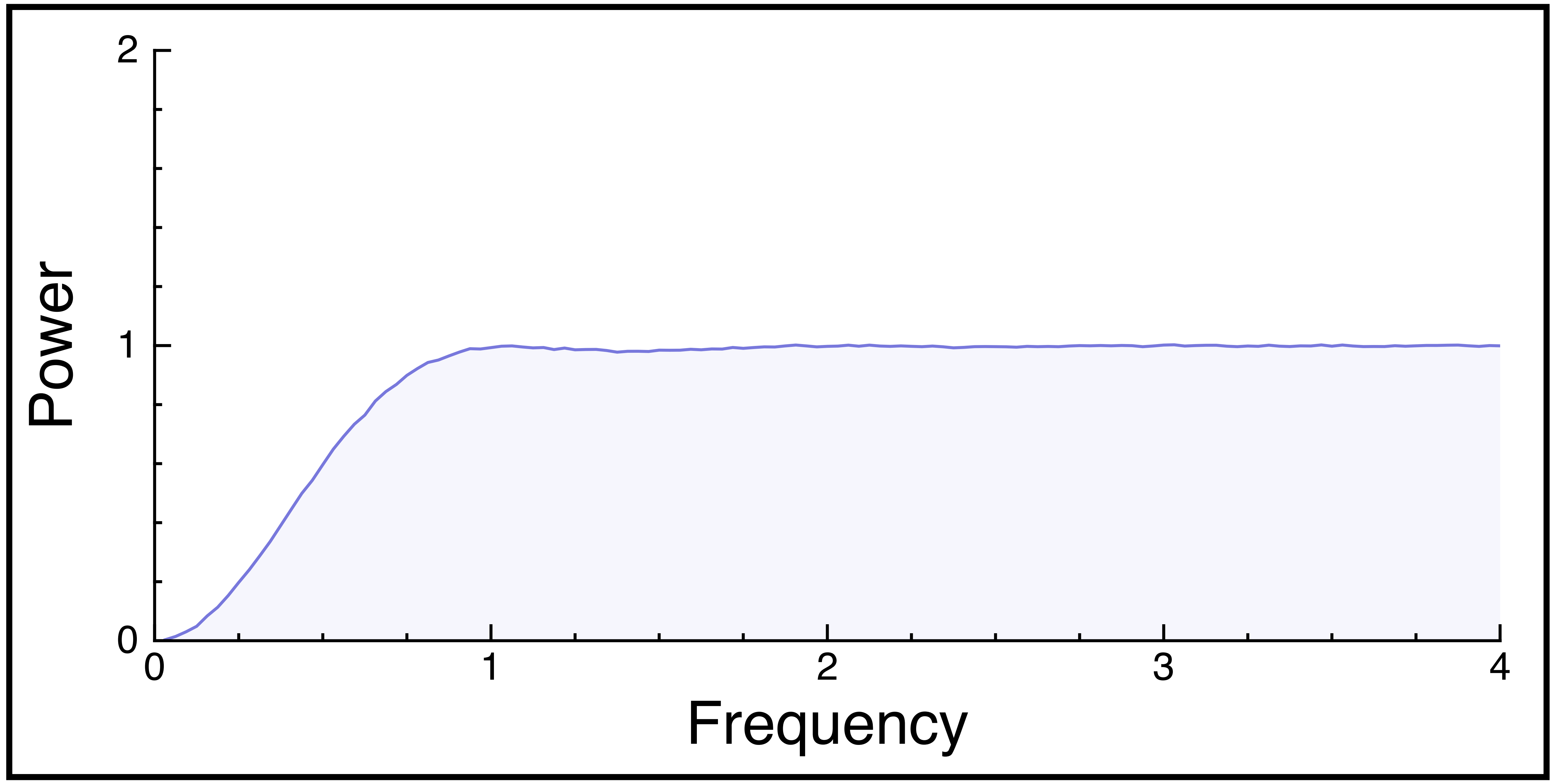
Samples



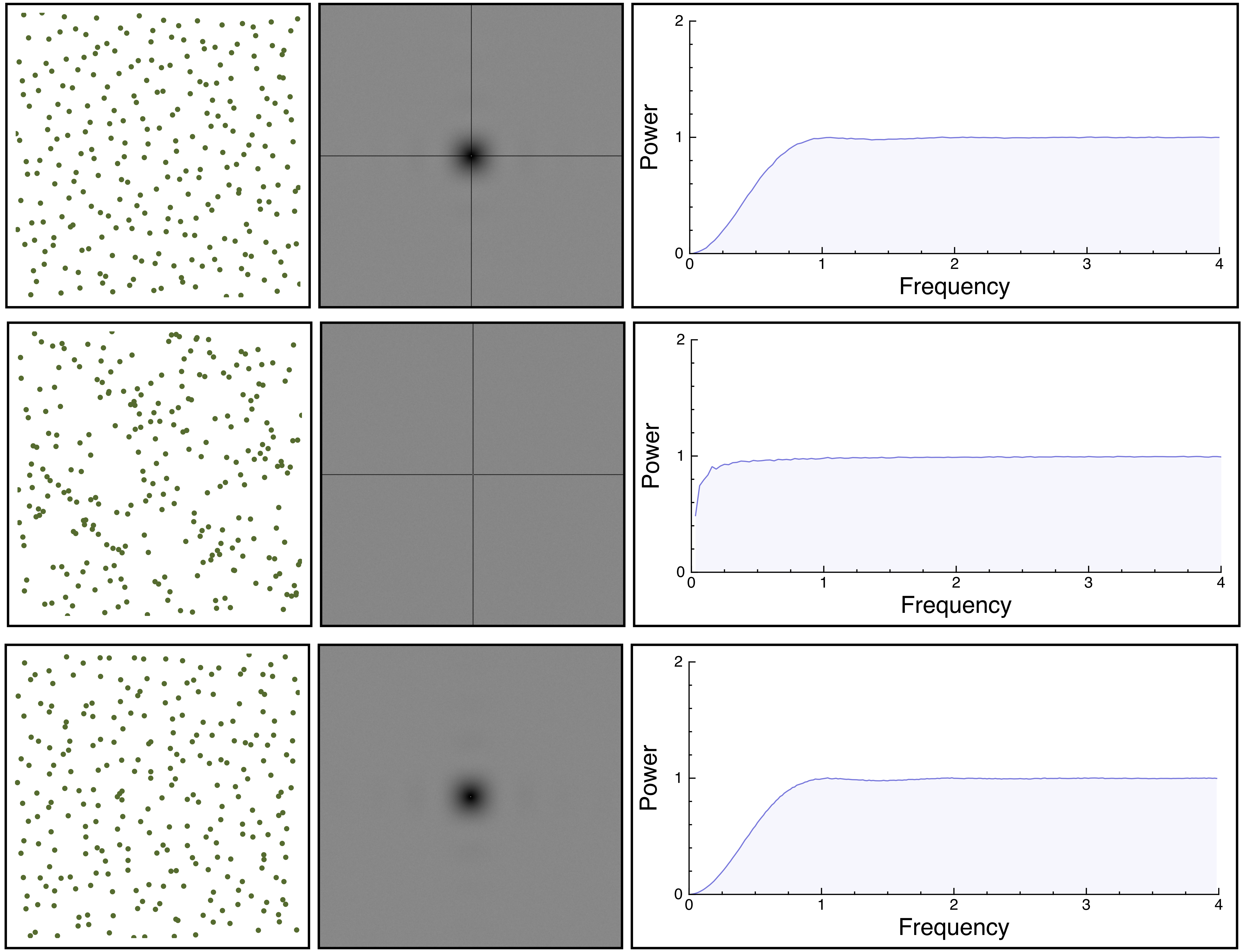
Expected power spectrum



Radial mean



Multi-jittered vs N-Rooks vs jittered

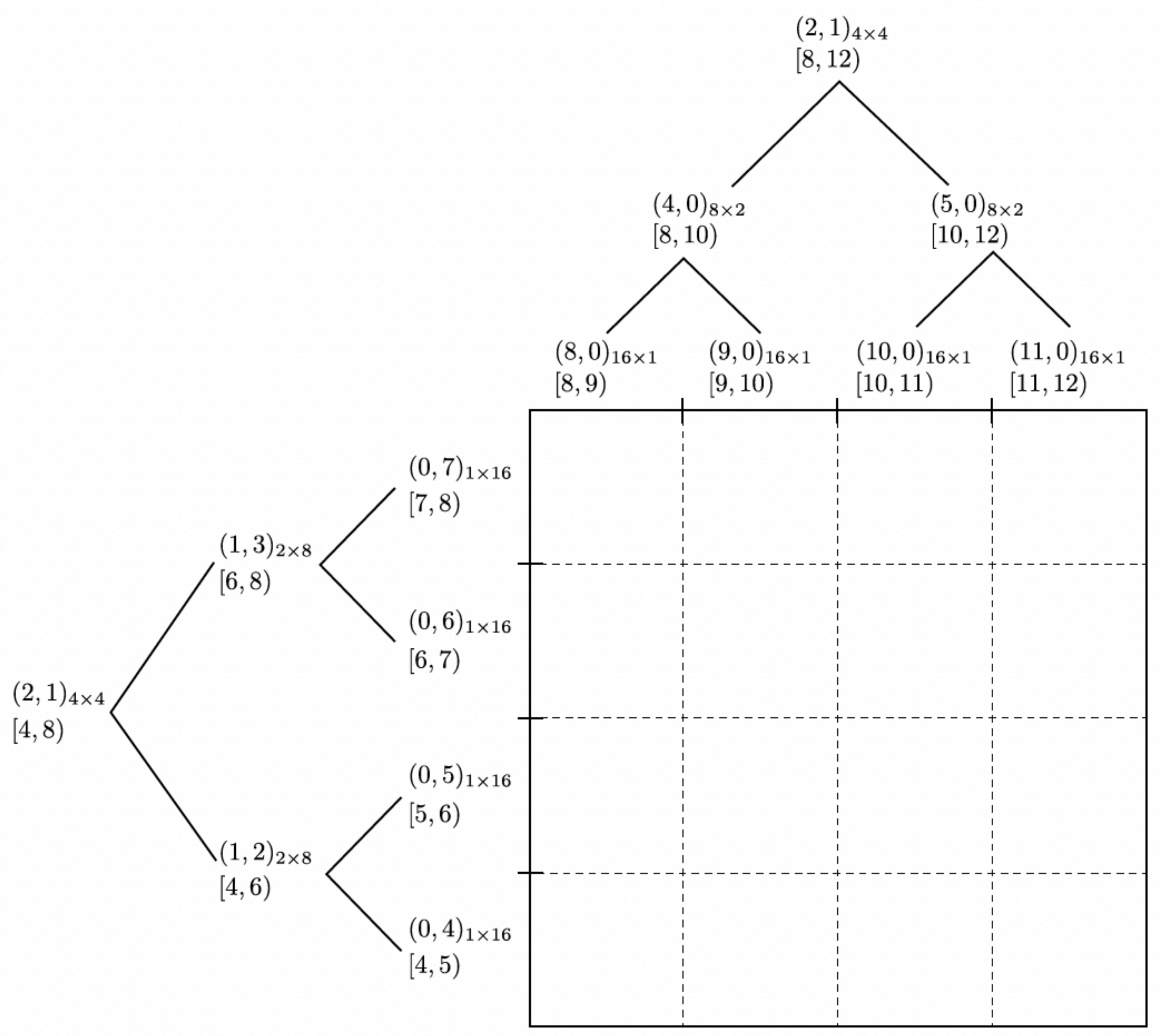


quiz: what is the difference between jittered & multi-jittered?

Progressive multi-jittered sampling

probably the best sampling pattern we discussed today!

- don’t need to know the number of samples in advance!
- idea: keep track of which strata is occupied by previous samples using trees (O(sqrt(N)))



Efficient Generation of Points that Satisfy Two-Dimensional Elementary Intervals

Matt Pharr
NVIDIA Research

2019

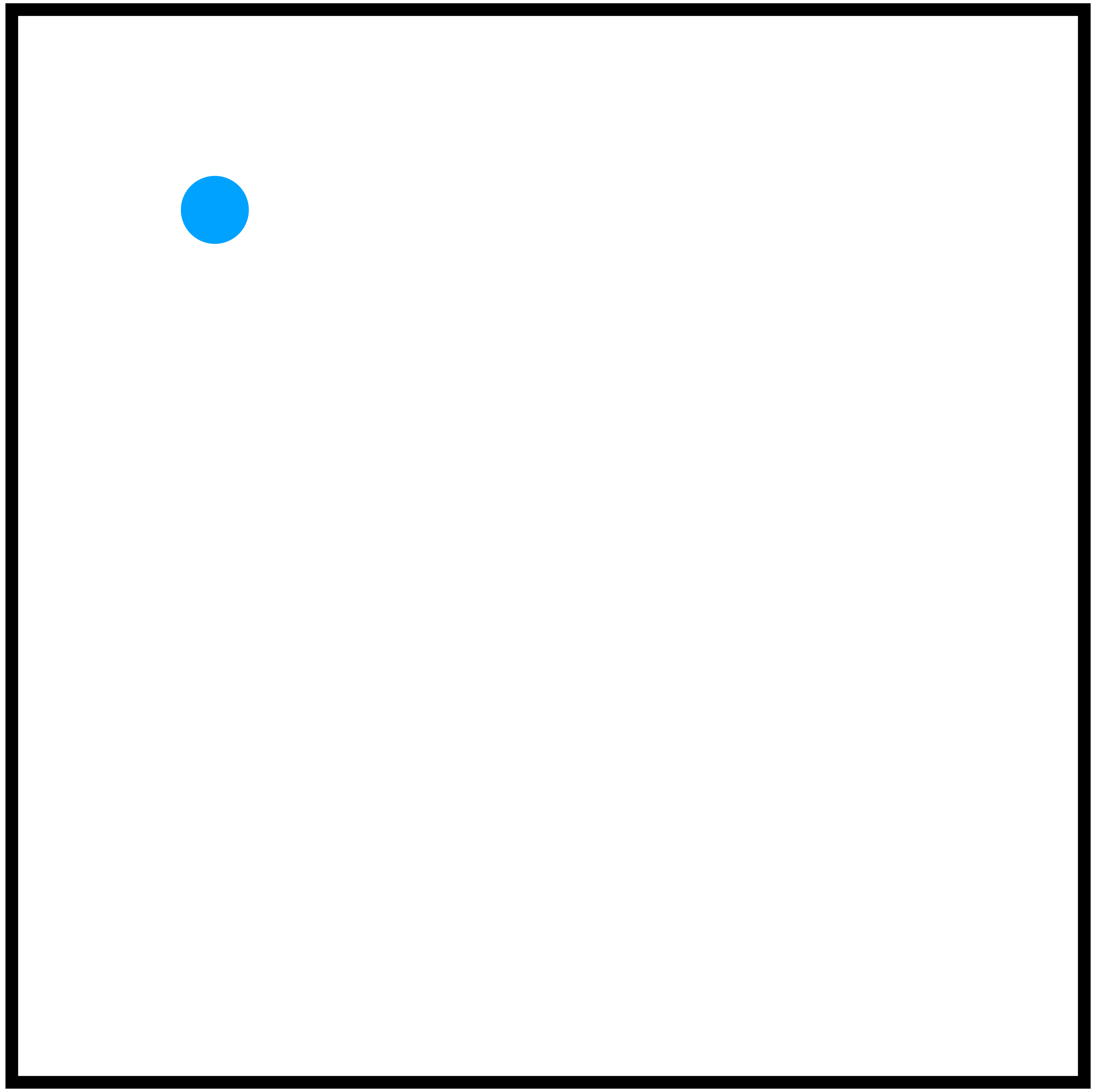
Progressive Multi-Jittered Sample Sequences

Per Christensen Andrew Kensler Charlie Kilpatrick
Pixar Animation Studios

2018

Progressive multi-jittered sampling

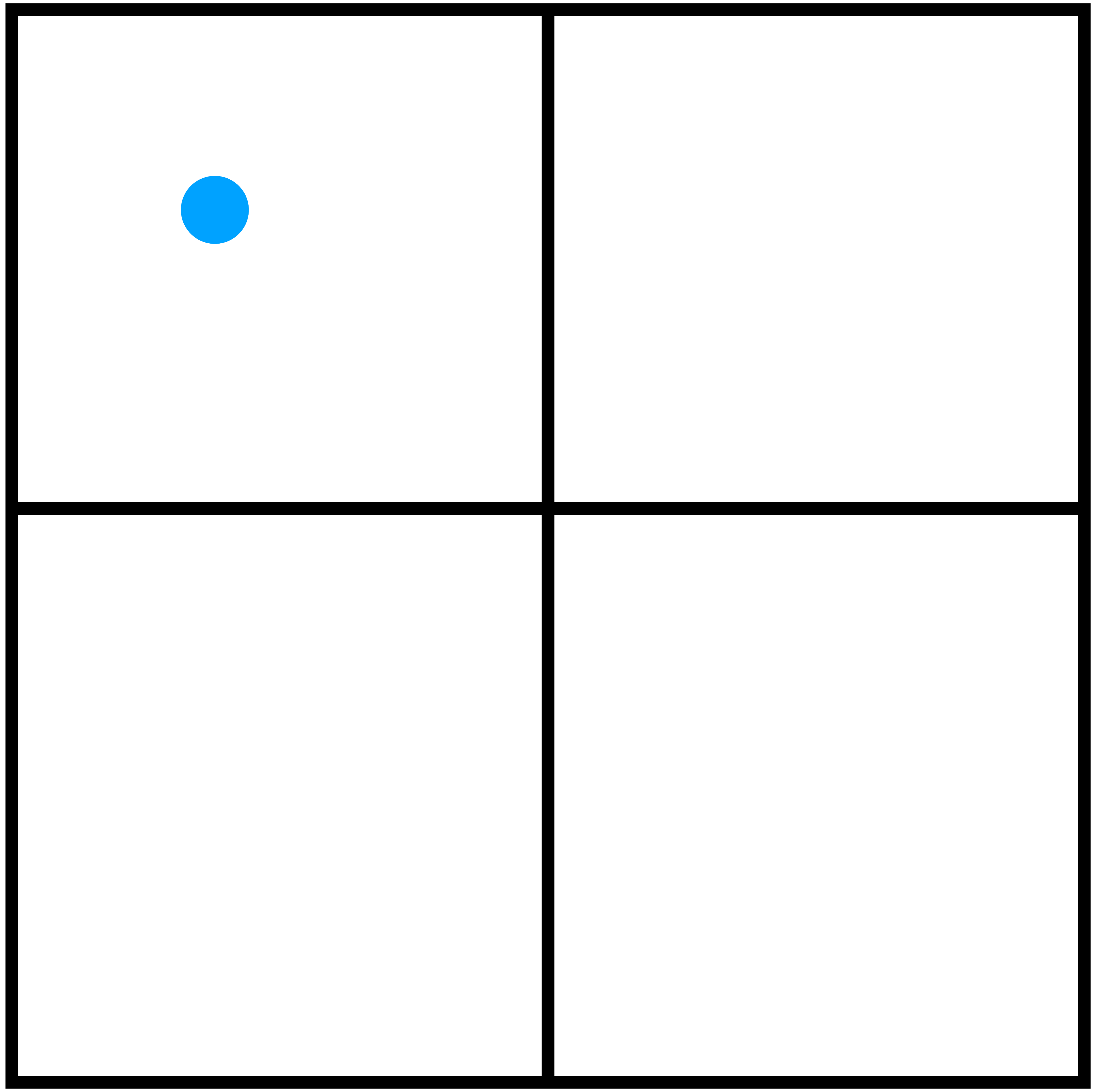
first sample: randomly place in the unit square



Progressive multi-jittered sampling

first sample: randomly place in the unit square

divide the unit square into 4 quadrants

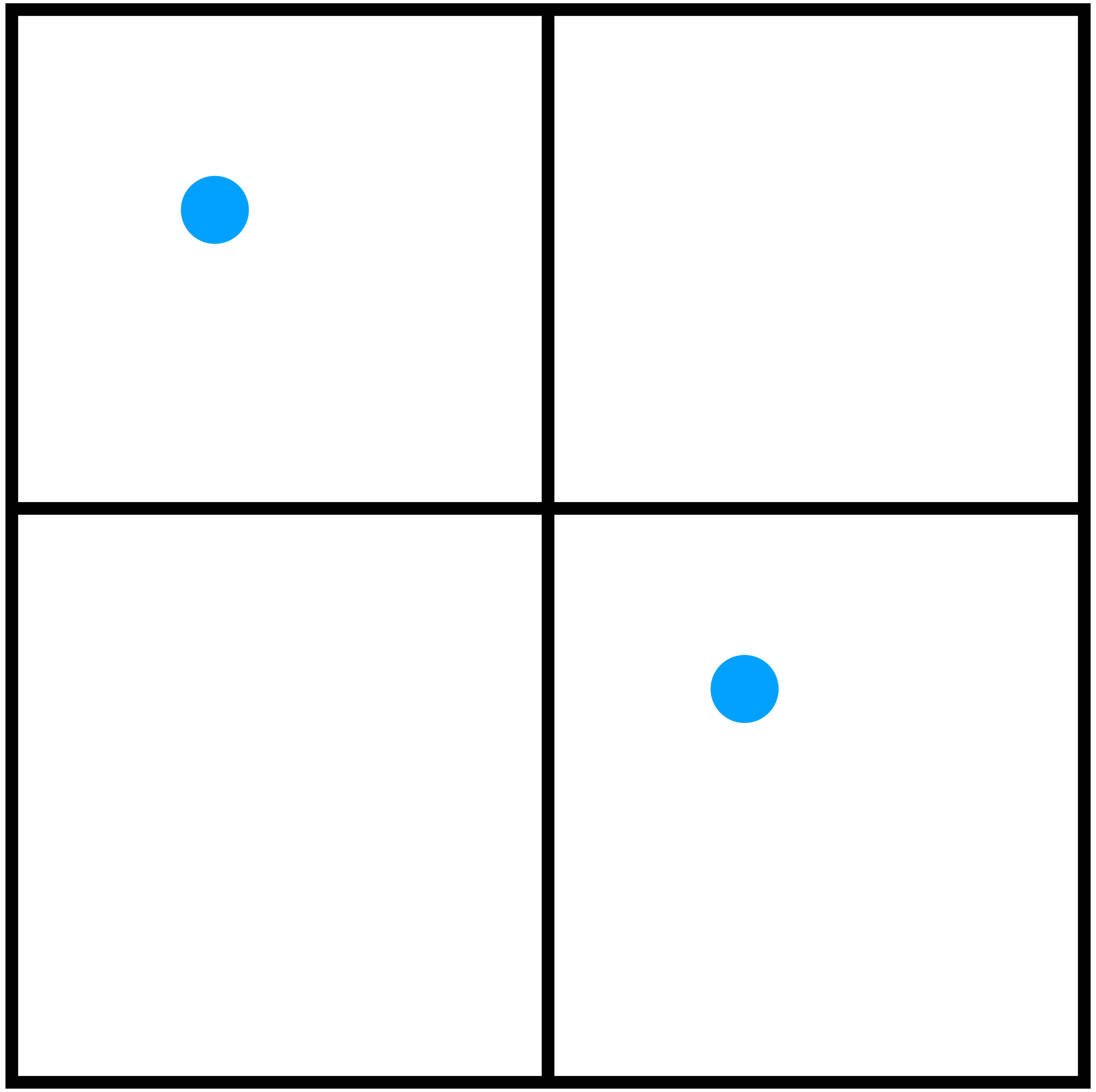


Progressive multi-jittered sampling

first sample: randomly place in the unit square

divide the unit square into 4 quadrants

place the second sample at the diagonally opposite quadrant



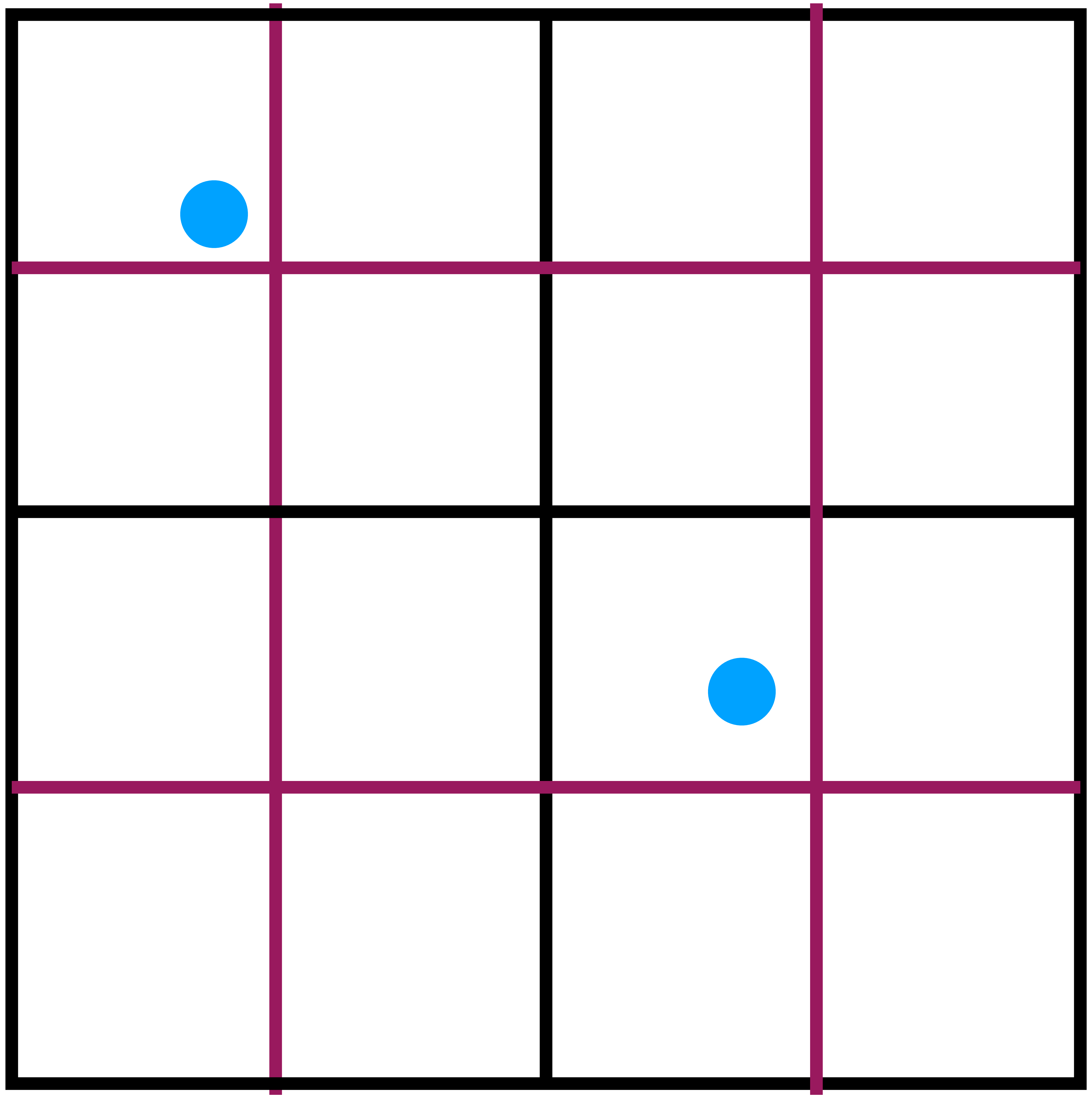
Progressive multi-jittered sampling

first sample: randomly place in the unit square

divide the unit square into 4 quadrants

place the second sample at the
diagonally opposite quadrant

divide the unit square into 16 regions



Progressive multi-jittered sampling

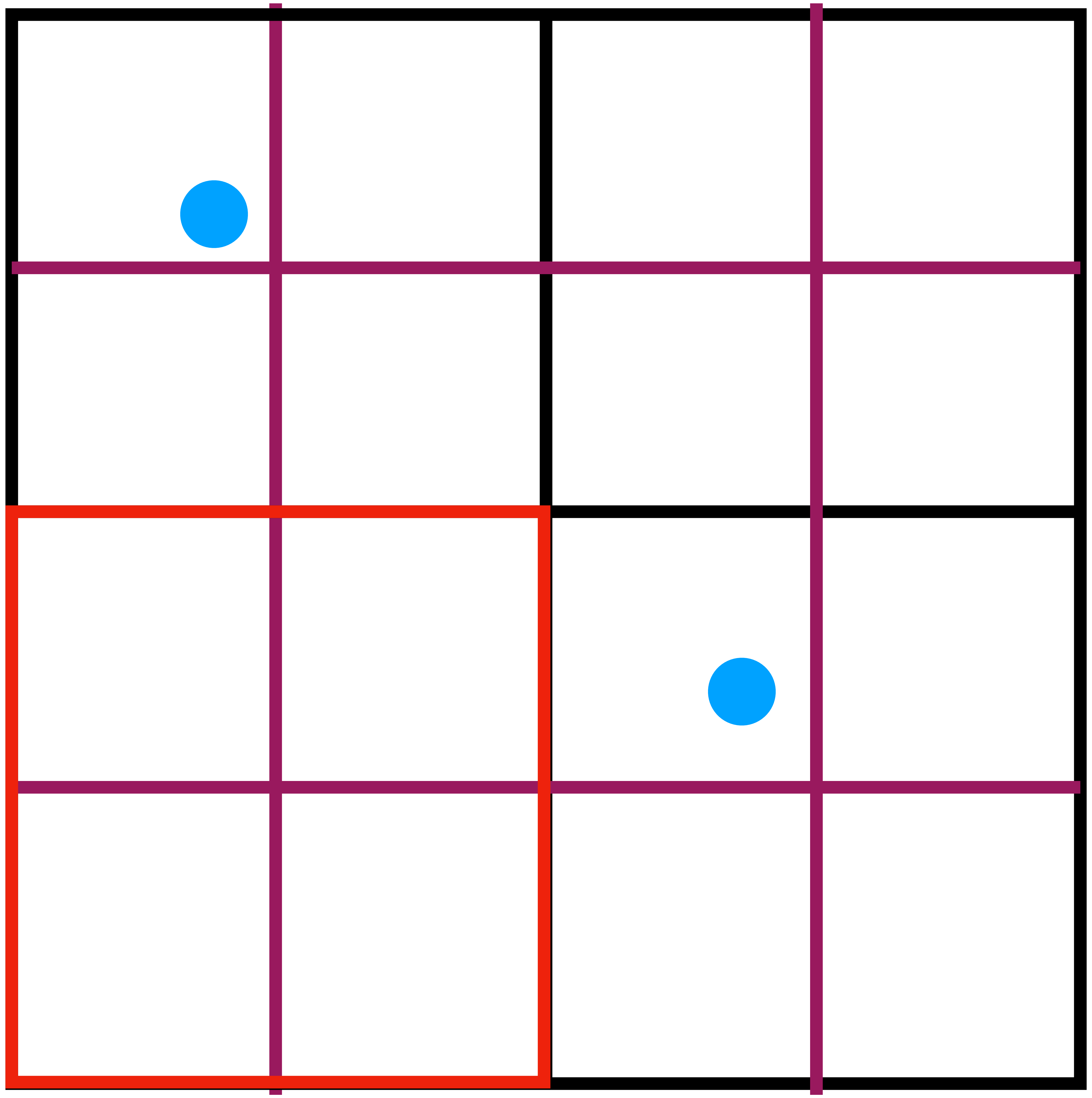
first sample: randomly place in the unit square

divide the unit square into 4 quadrants

place the second sample at the diagonally opposite quadrant

divide the unit square into 16 regions

choose an empty quadrant



Progressive multi-jittered sampling

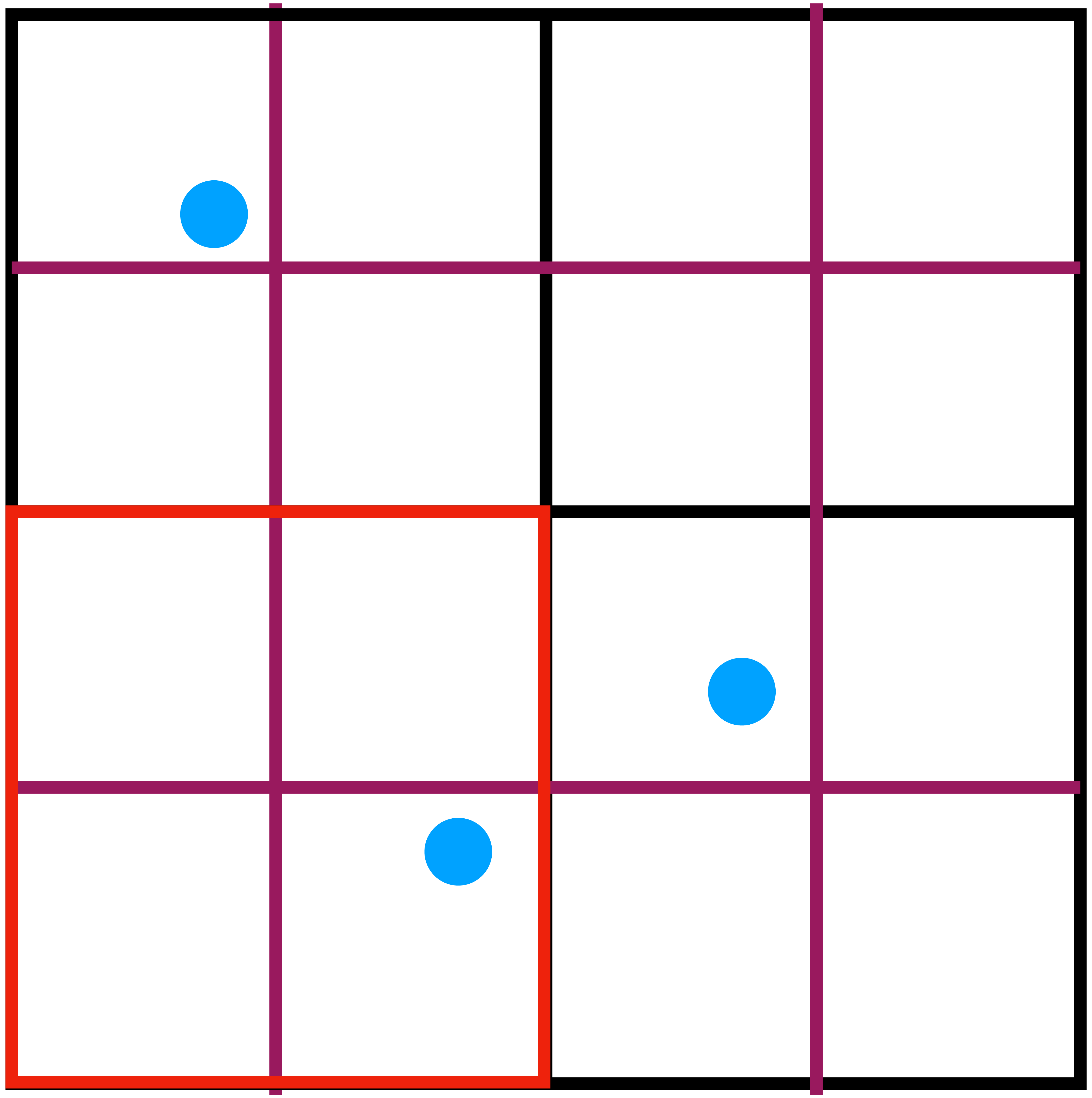
first sample: randomly place in the unit square

divide the unit square into 4 quadrants

place the second sample at the
diagonally opposite quadrant

divide the unit square into 16 regions

choose an empty quadrant, place a sample
that follows the N-rook rule



Progressive multi-jittered sampling

first sample: randomly place in the unit square

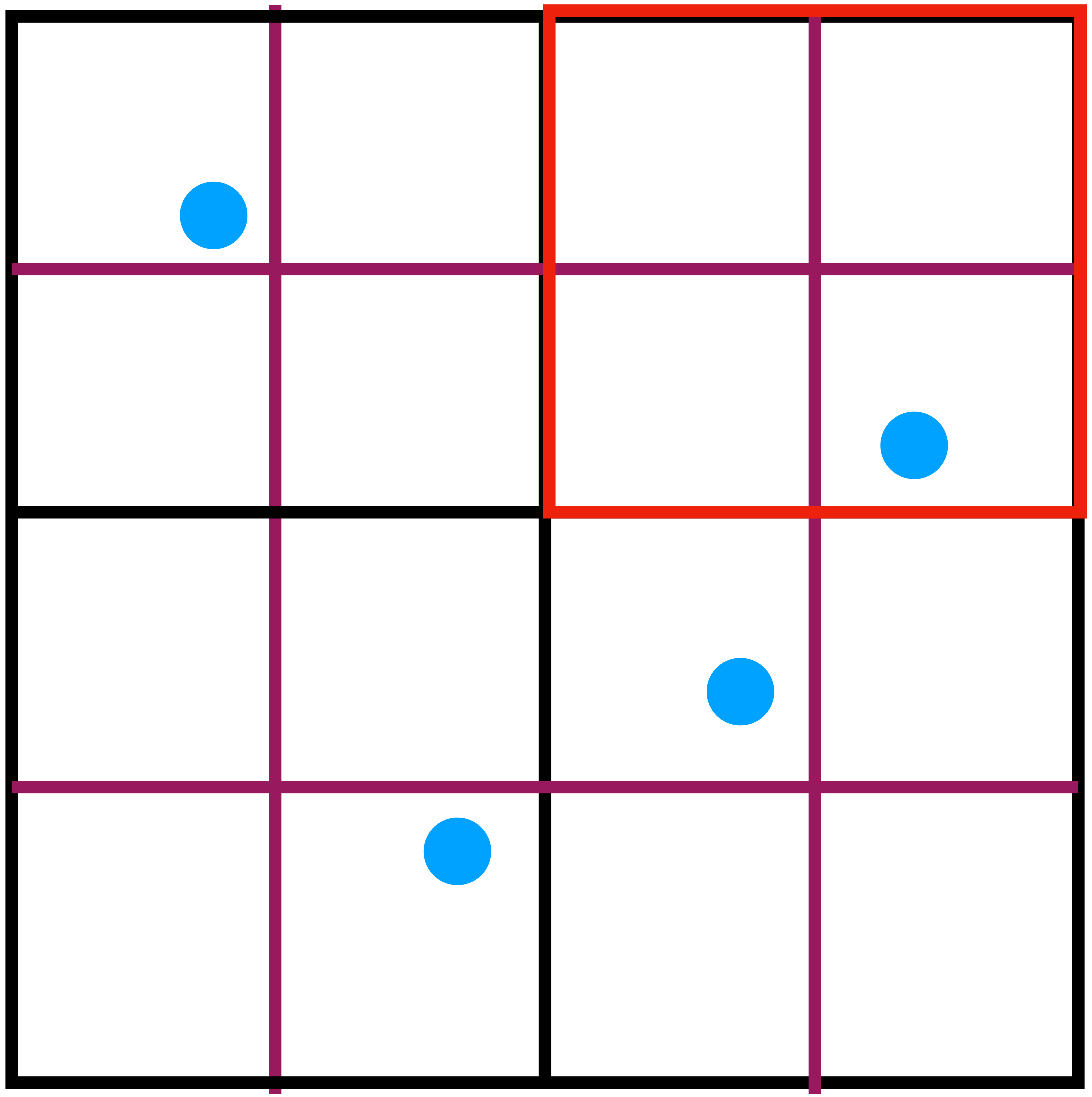
divide the unit square into 4 quadrants

place the second sample at the diagonally opposite quadrant

divide the unit square into 16 regions

choose an empty quadrant, place a sample that follows the N-rook rule

place a sample at the diagonally opposite quadrant, following the N-rook rule



Progressive multi-jittered sampling

first sample: randomly place in the unit square

divide the unit square into 4 quadrants

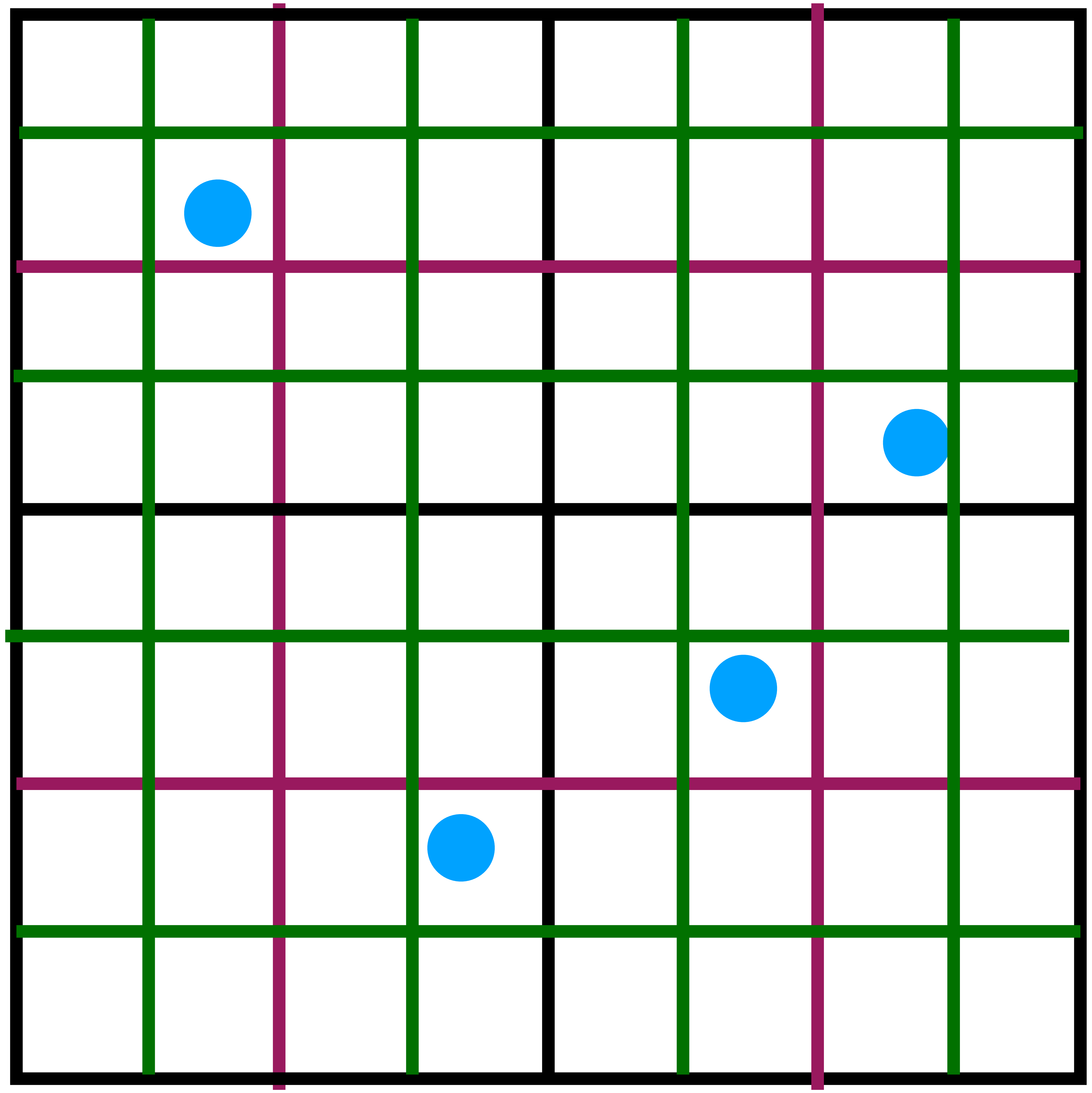
place the second sample at the diagonally opposite quadrant

divide the unit square into 16 regions

choose an empty quadrant, place a sample that follows the N-rook rule

place a sample at the diagonally opposite quadrant, following the N-rook rule

repeat



Progressive multi-jittered sampling

first sample: randomly place in the unit square

divide the unit square into 4 quadrants

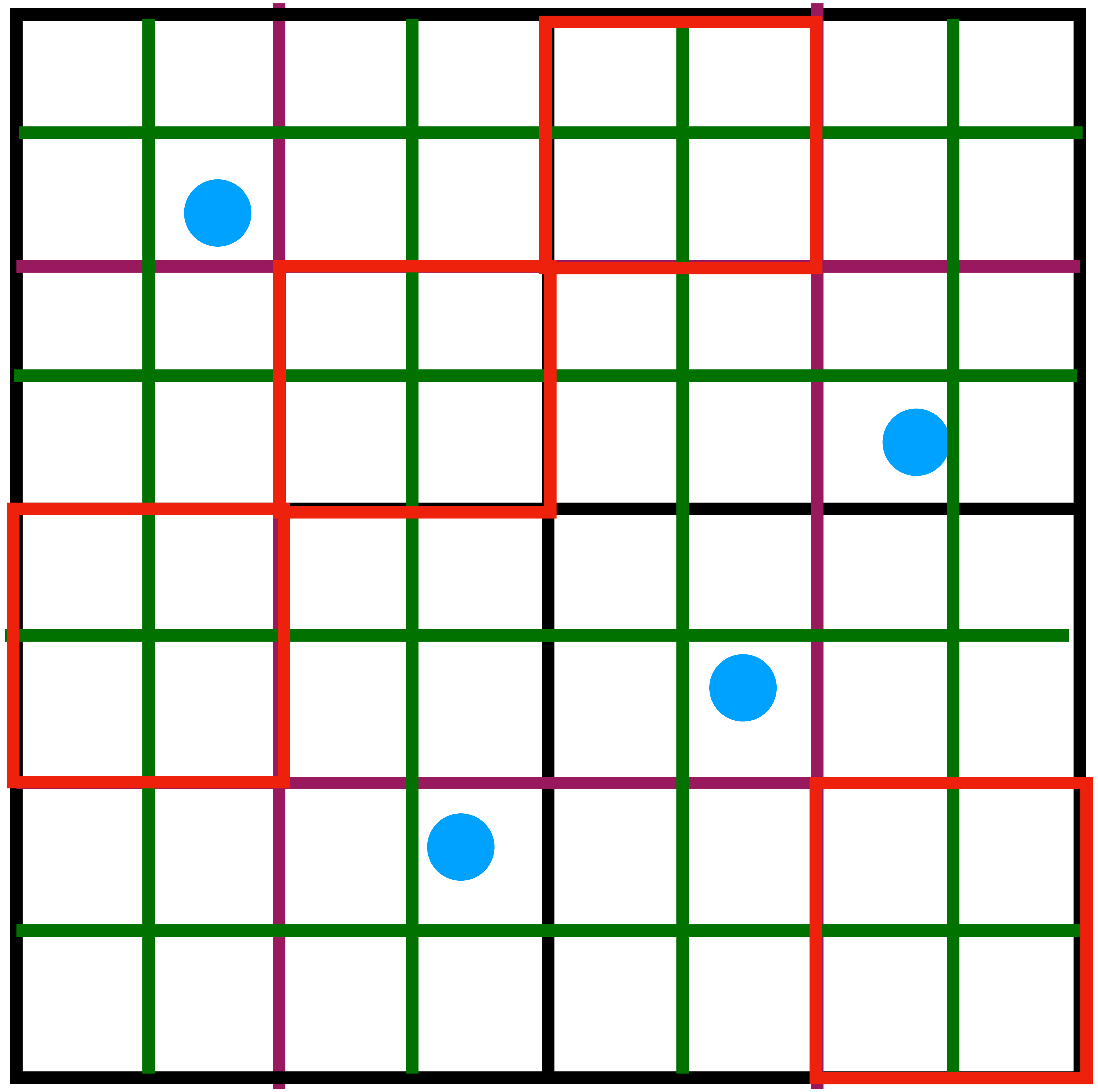
place the second sample at the diagonally opposite quadrant

divide the unit square into 16 regions

choose an empty quadrant, place a sample that follows the N-rook rule

place a sample at the diagonally opposite quadrant, following the N-rook rule

repeat



Progressive multi-jittered sampling

first sample: randomly place in the unit square

divide the unit square into 4 quadrants

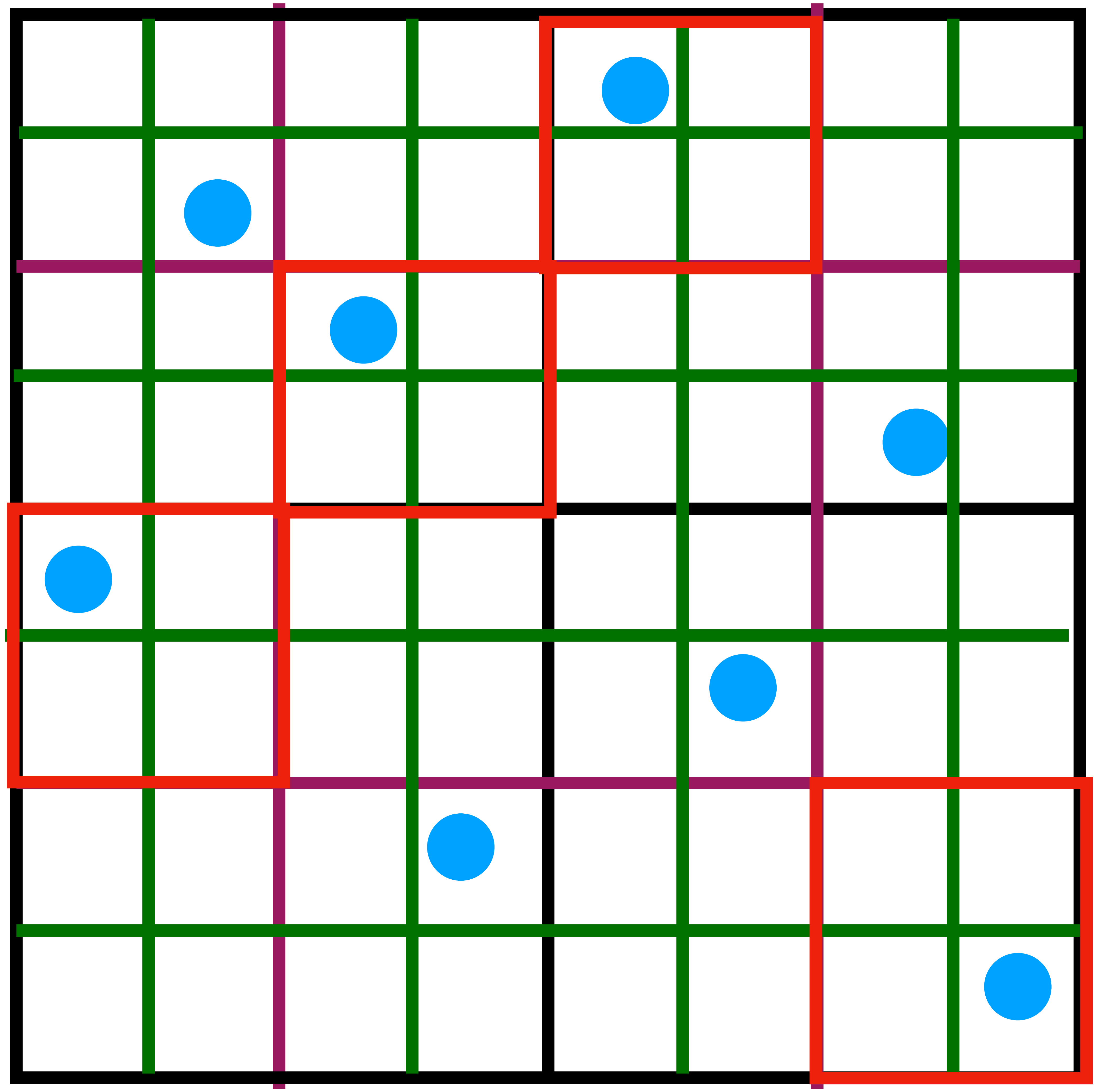
place the second sample at the diagonally opposite quadrant

divide the unit square into 16 regions

choose an empty quadrant, place a sample that follows the N-rook rule

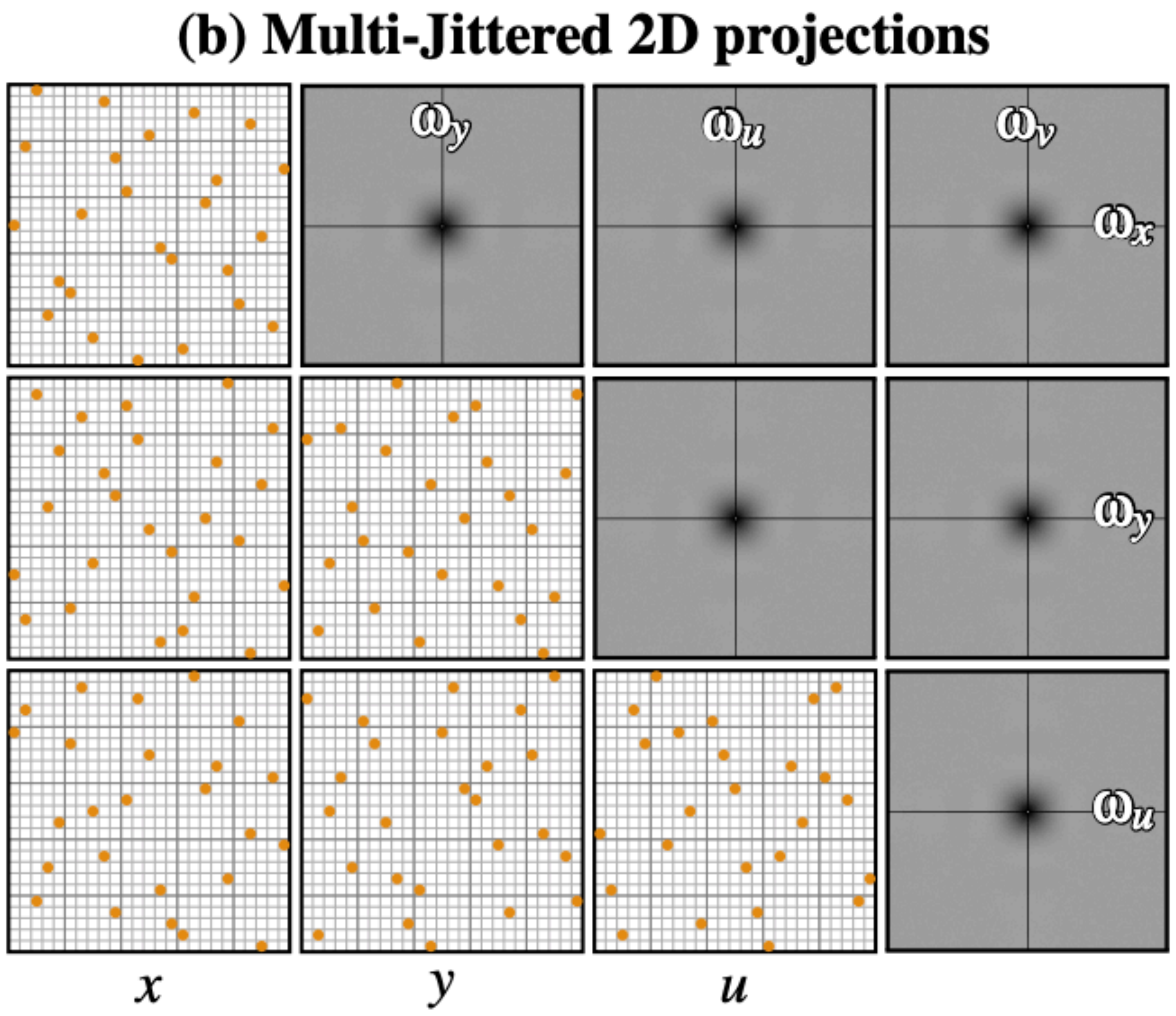
place a sample at the diagonally opposite quadrant, following the N-rook rule

repeat



Orthogonal array sampling

- stratify in all 2D projections
- need to know no. of samples in advance currently



2019

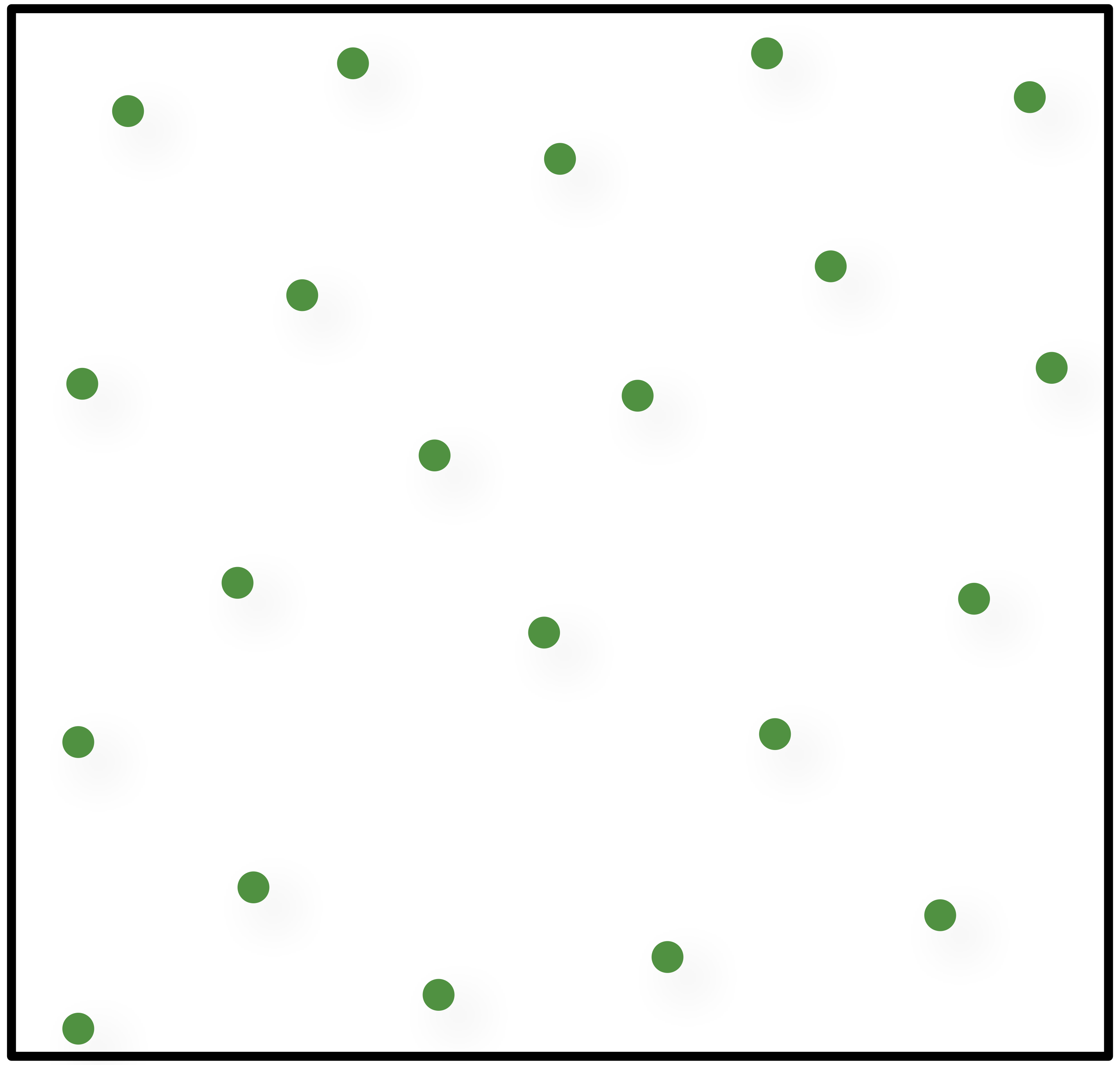
Orthogonal Array Sampling for Monte Carlo Rendering

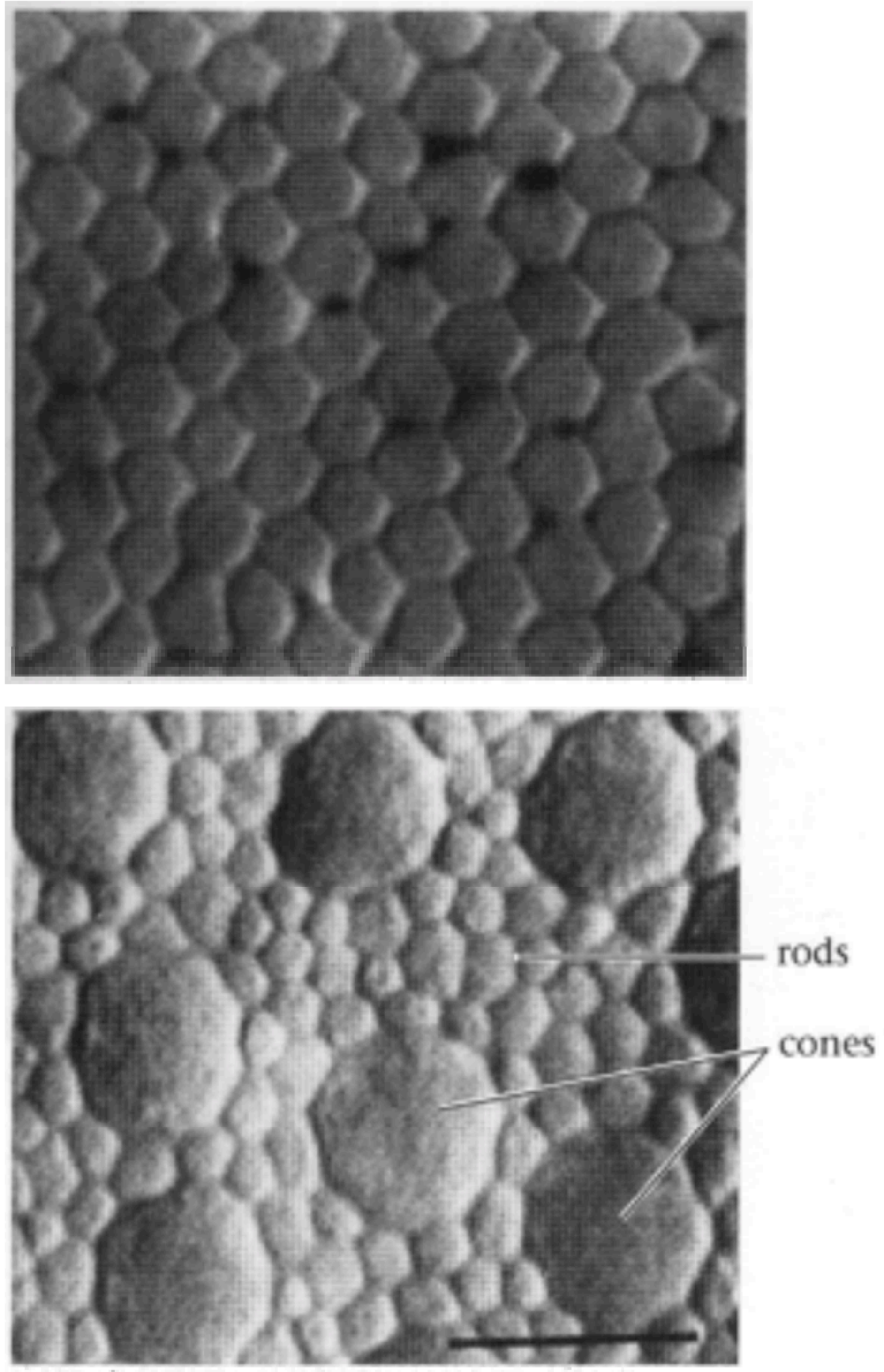
Wojciech Jarosz[1] Afnan Enayet[1] Andrew Kensler[2] Charlie Kilpatrick[2] Per Christensen[2]

[1]Dartmouth College [2]Pixar Animation Studios

Poisson-disk/blue noise sampling

- human eyes' sampling pattern!





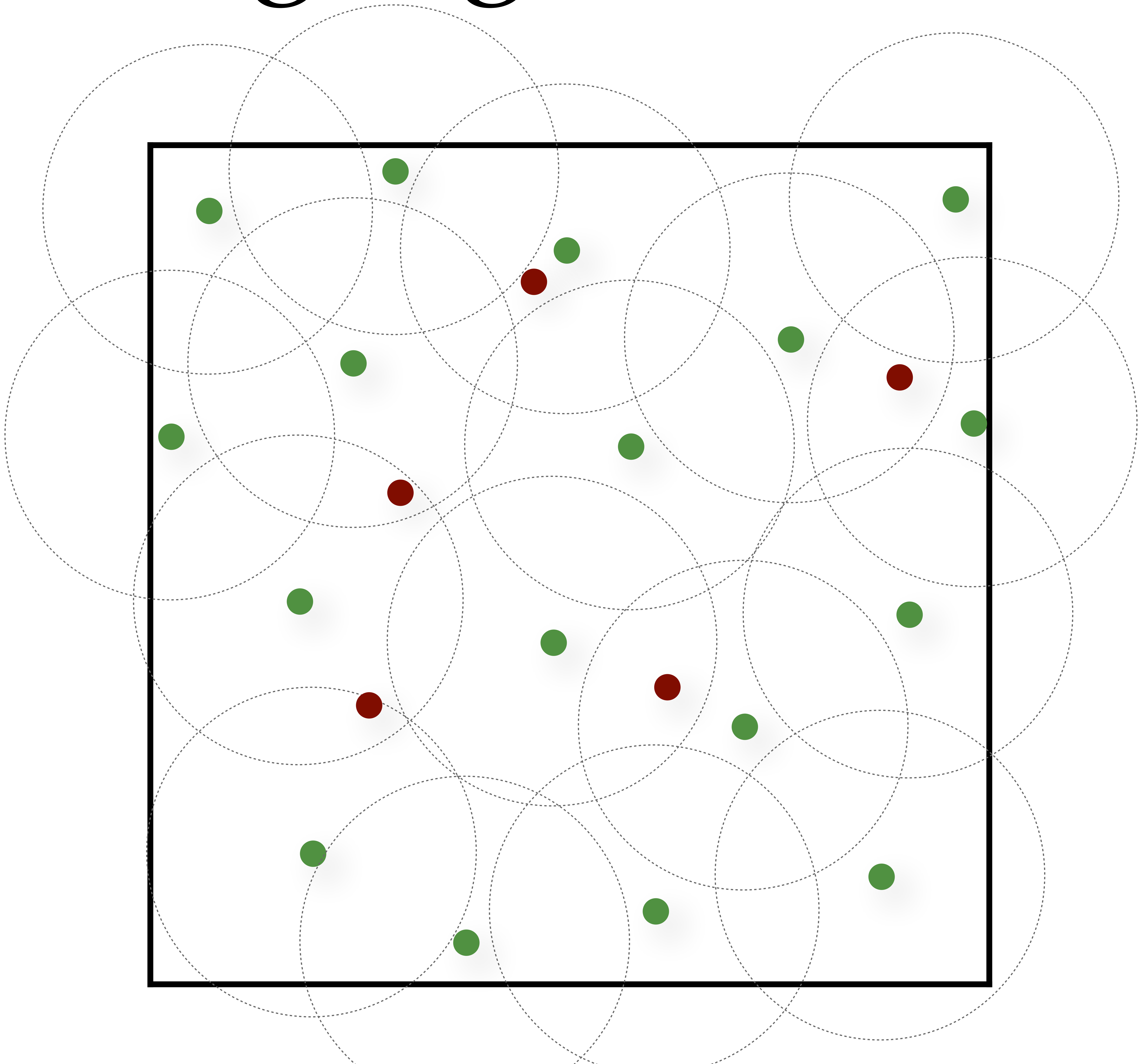
Spectral Consequences of Photoreceptor Sampling in the Rhesus Retina

JOHN I. YELLOTT, JR

SCIENCE • 22 Jul 1983 • Vol 221, Issue 4608 • pp. 382-385 • DOI: 10.1126/science.6867716

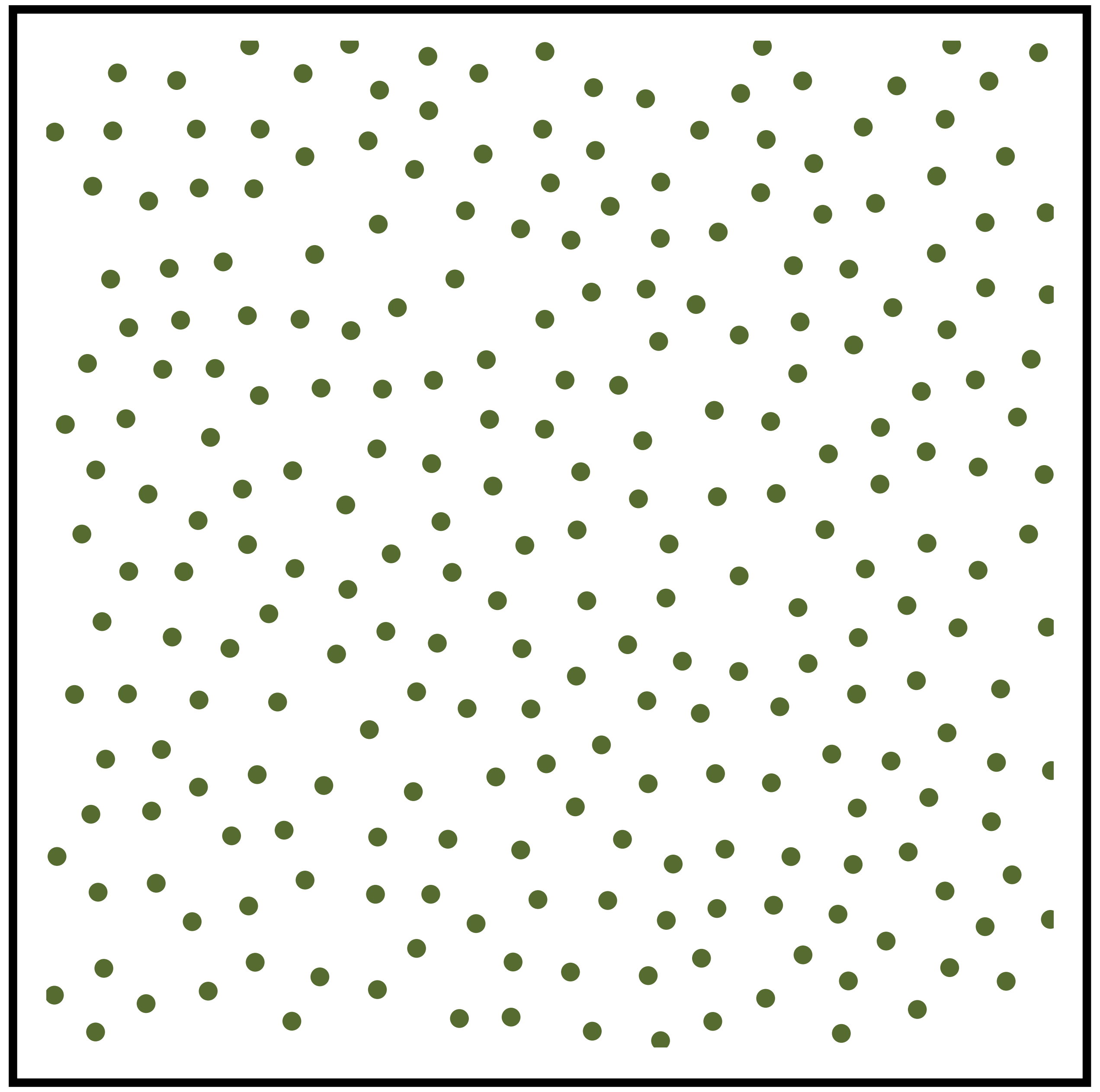
https://www.csie.ntu.edu.tw/~cyy/courses/rendering/16fall/lectures/handouts/chap05_color_radiometry.pdf

Dart throwing algorithm [Cook 1986]

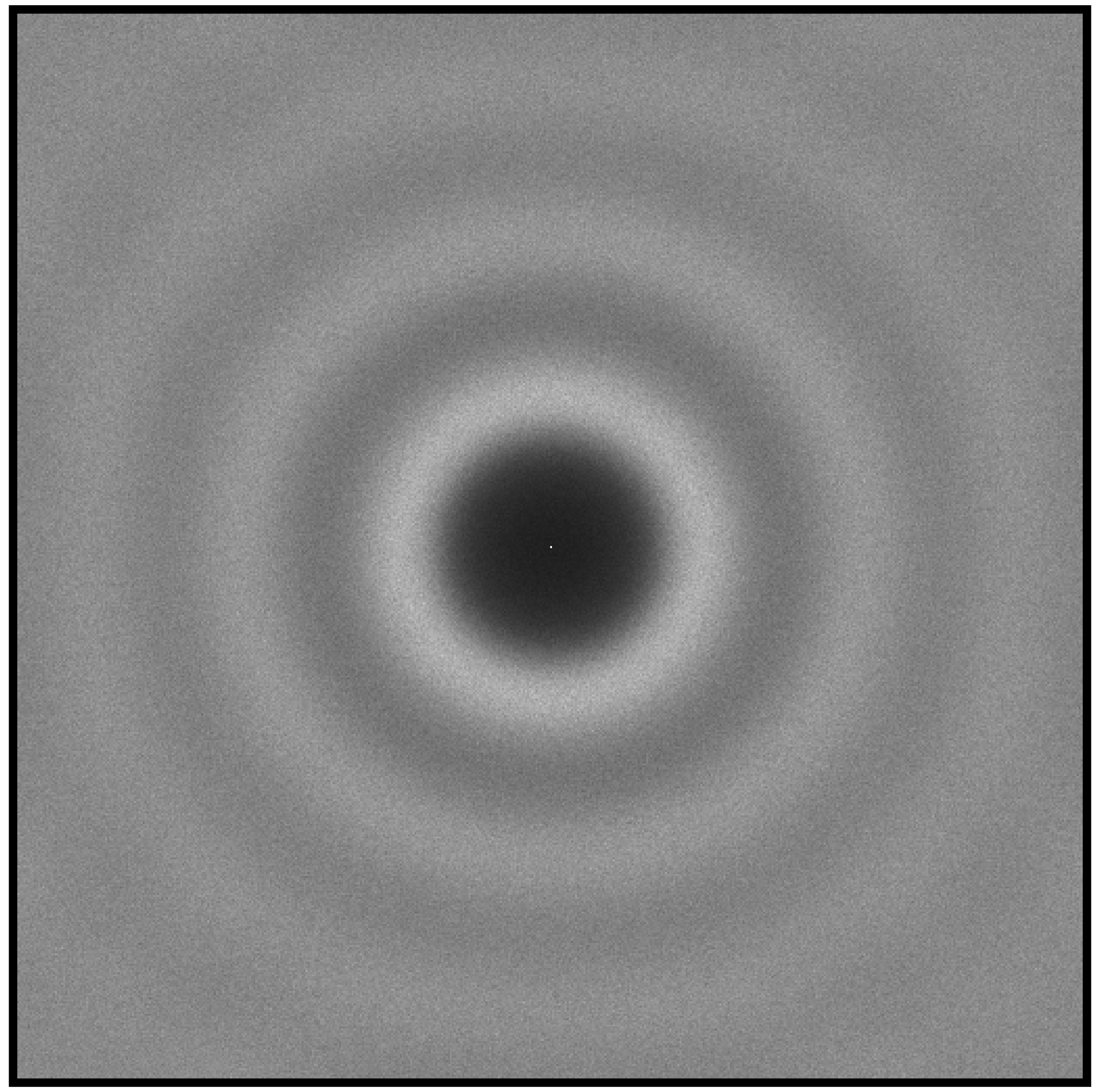


Power spectrum of Poisson disk

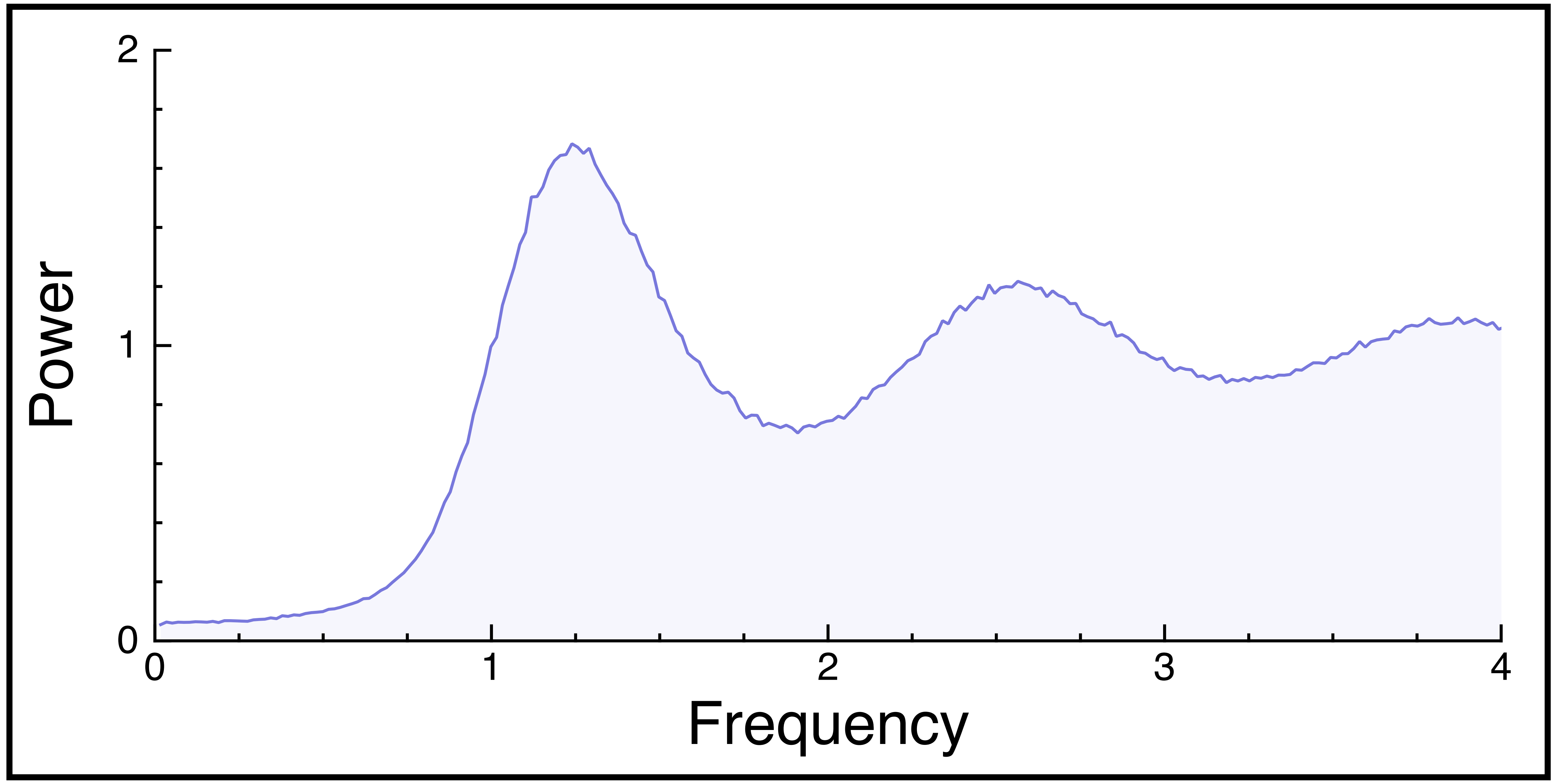
Samples



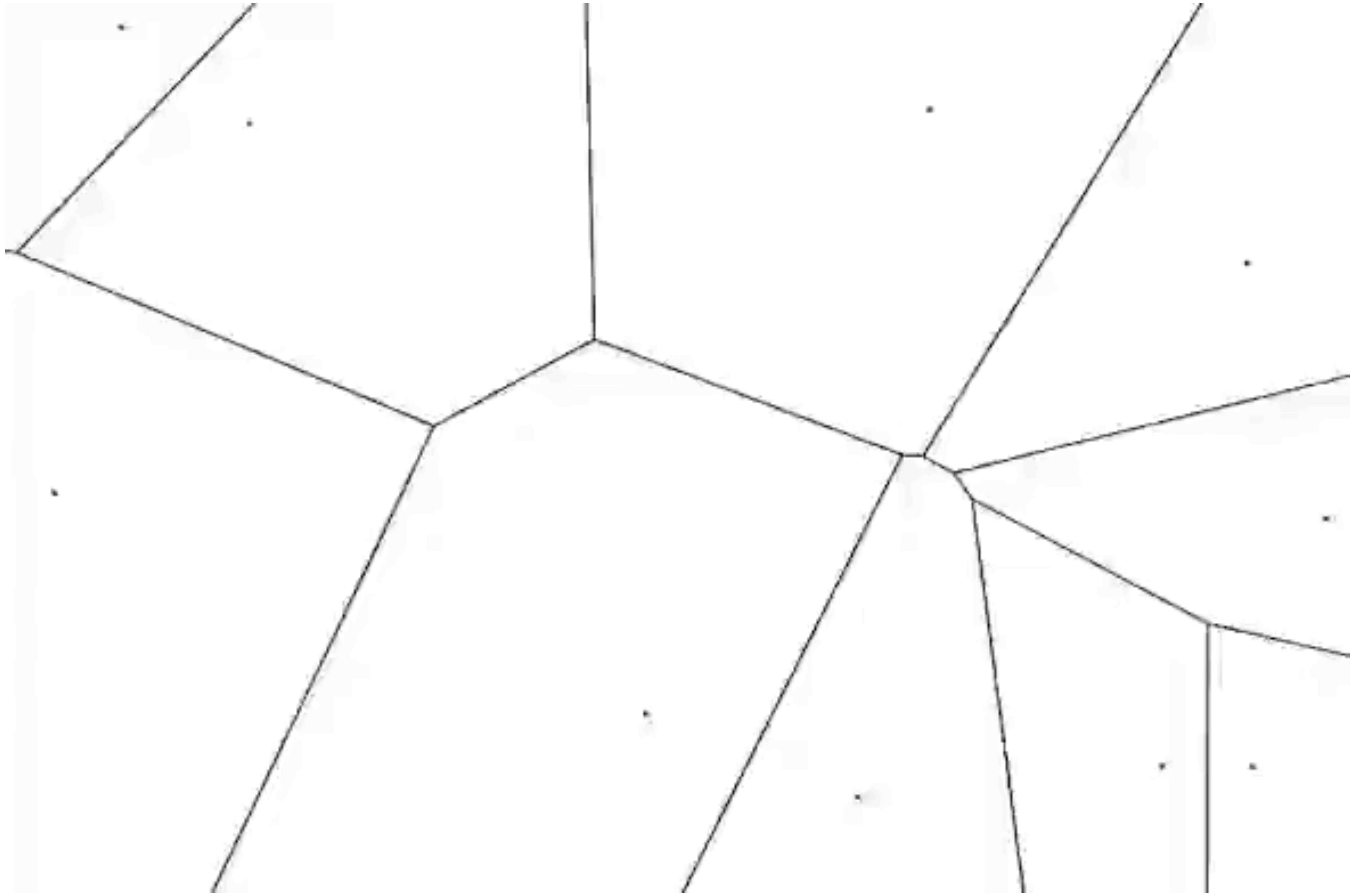
Expected power spectrum



Radial mean

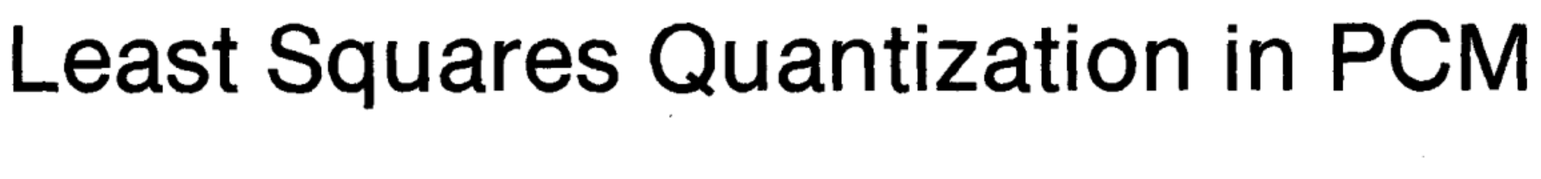


Lloyd relaxation for Poisson disc sampling



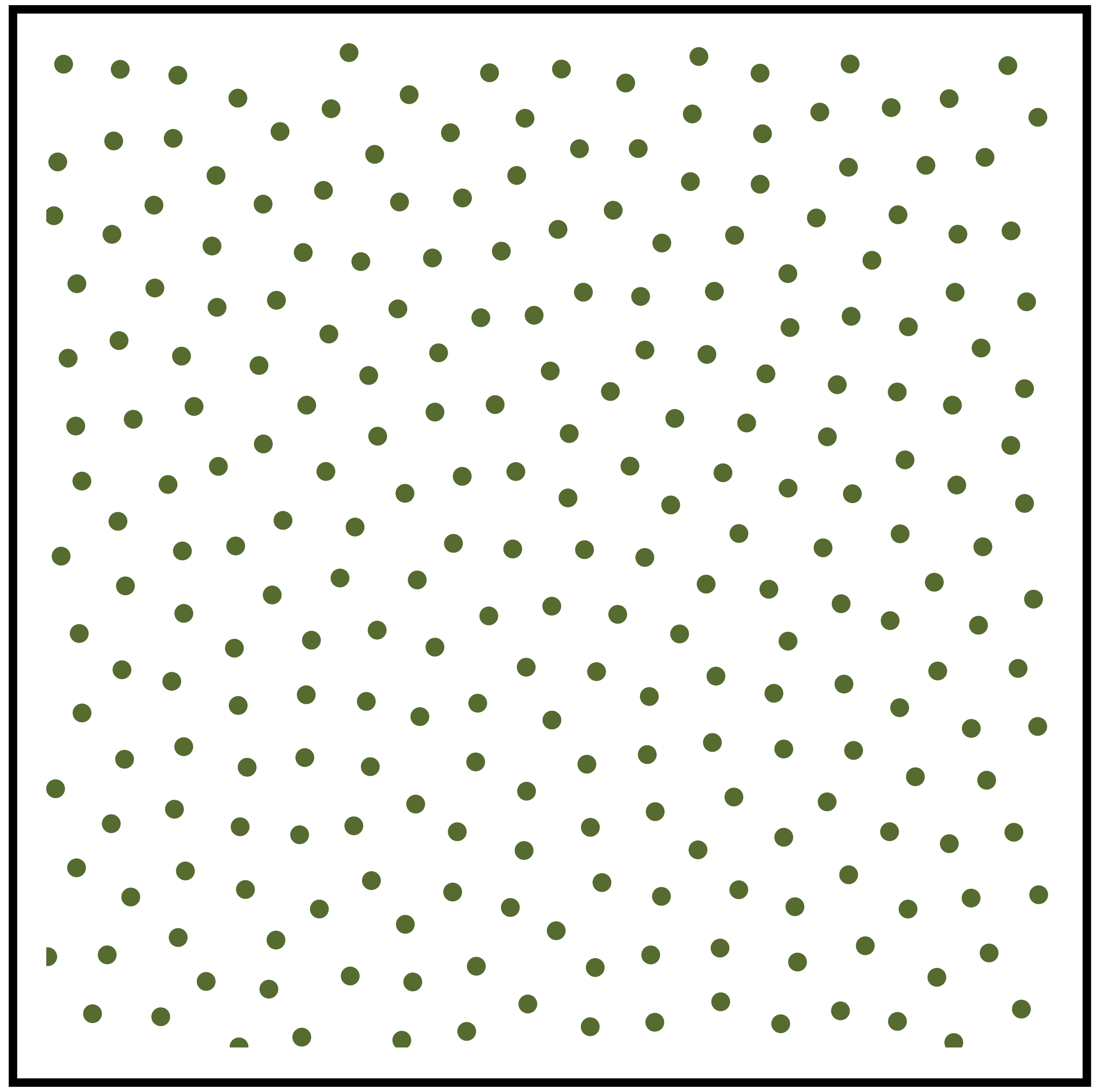
video from
http://www.codeplastic.com/2017/12/30/voronoi-relaxation-lloyds-algorithm-in-processing/

developed at ~1957, published at 1982

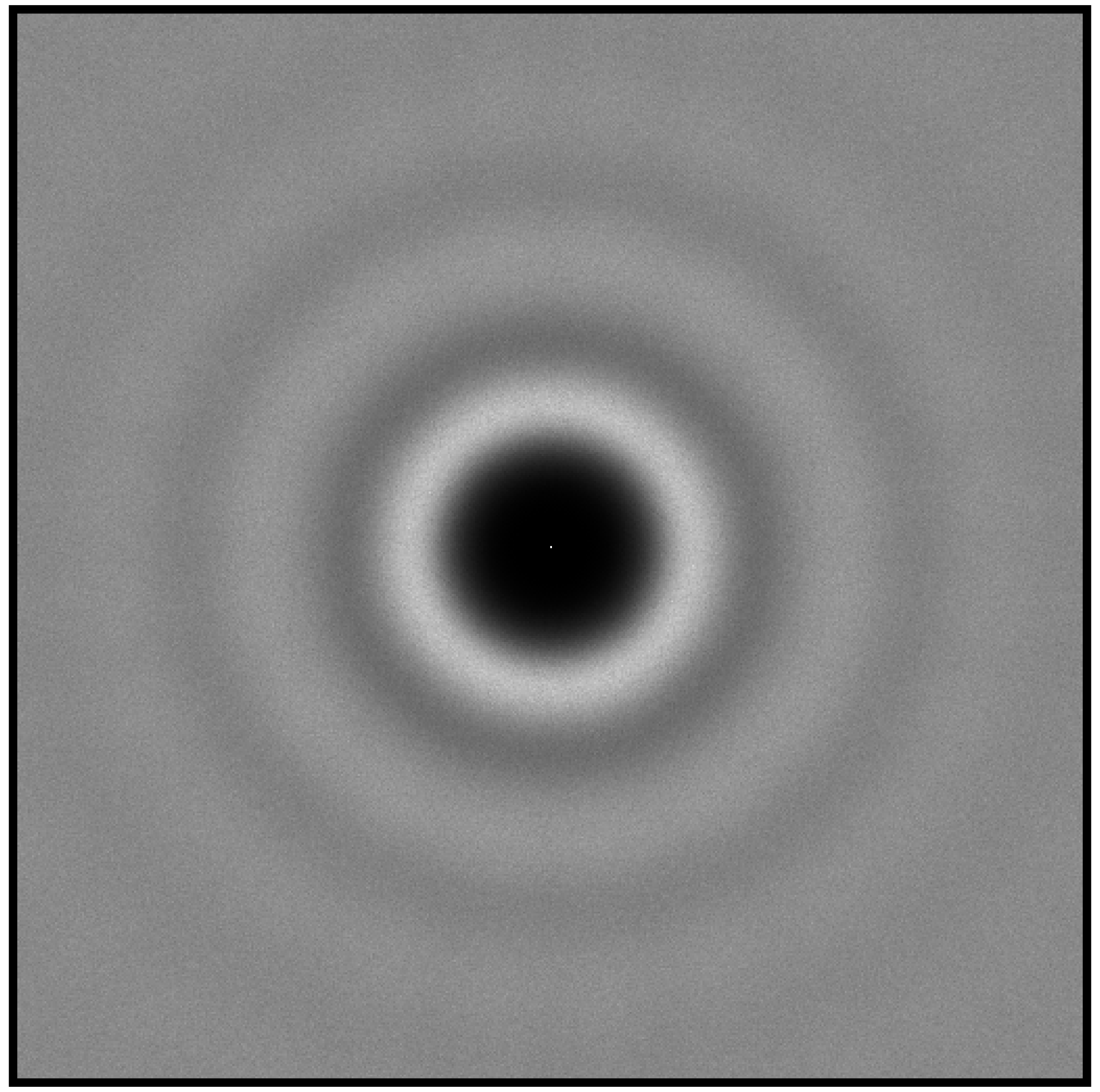


Power spectrum of CCVT sampling
[Balzer et al. 2009]

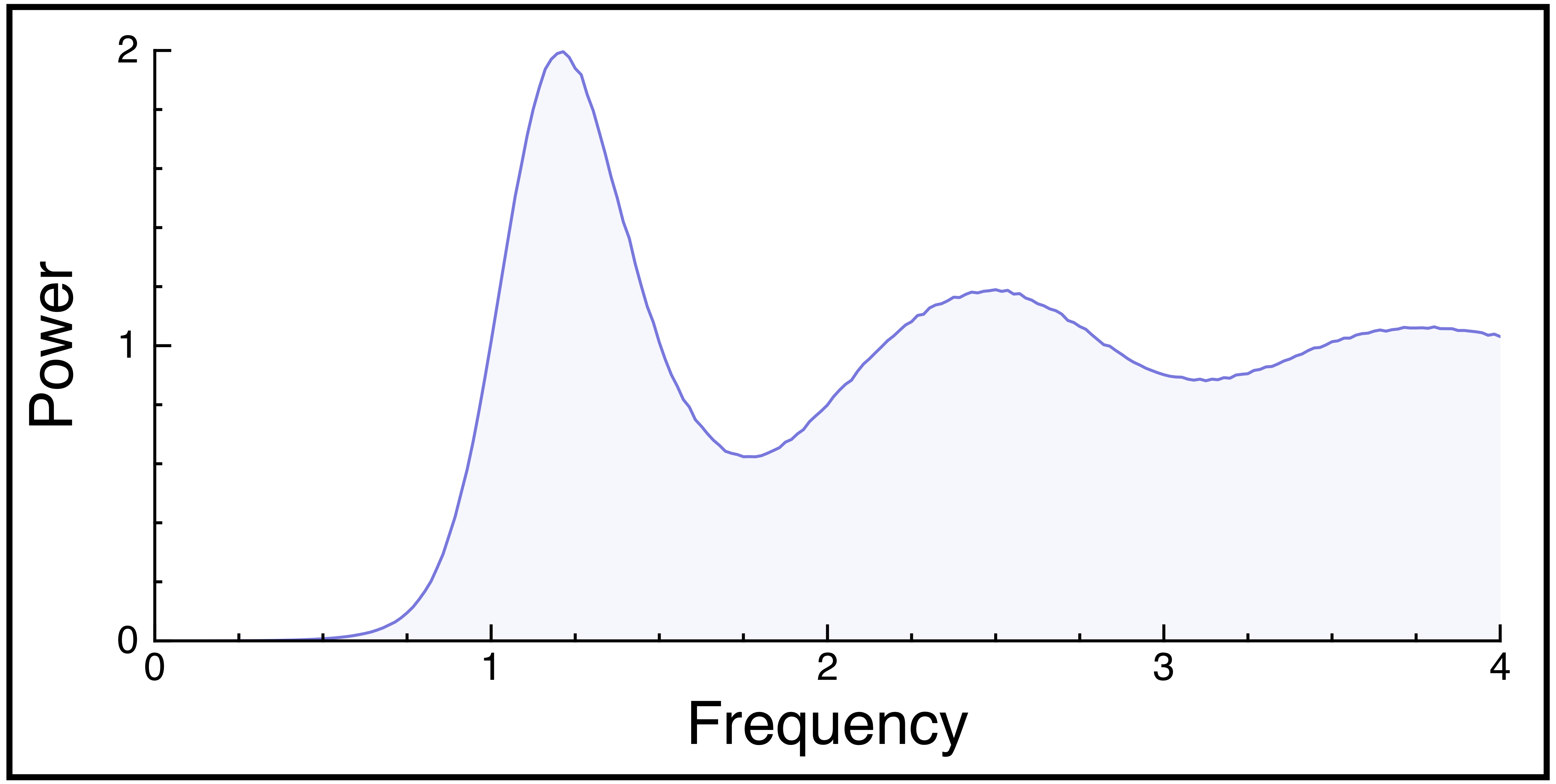
Samples



Expected power spectrum



Radial mean



Theoretical convergence rate (in 2D)

- all sampling sequences work best for low frequency/smooth signals

Samplers	Worst Case	Best Case
Random	$\mathcal{O}(N^{-1})$	$\mathcal{O}(N^{-1})$
Jitter	$\mathcal{O}(N^{-1.5})$	$\mathcal{O}(N^{-2})$
Poisson Disk	$\mathcal{O}(N^{-1})$	$\mathcal{O}(N^{-1})$
CCVT	$\mathcal{O}(N^{-1.5})$	$\mathcal{O}(N^{-3})$

quiz: what is the downside of CCVT compared to jittered sampling?

Curse of dimensionality

- in high-dimensional space with high frequency between dimensions, all methods fail

best possible worst case convergence rate (with C1 continuity)

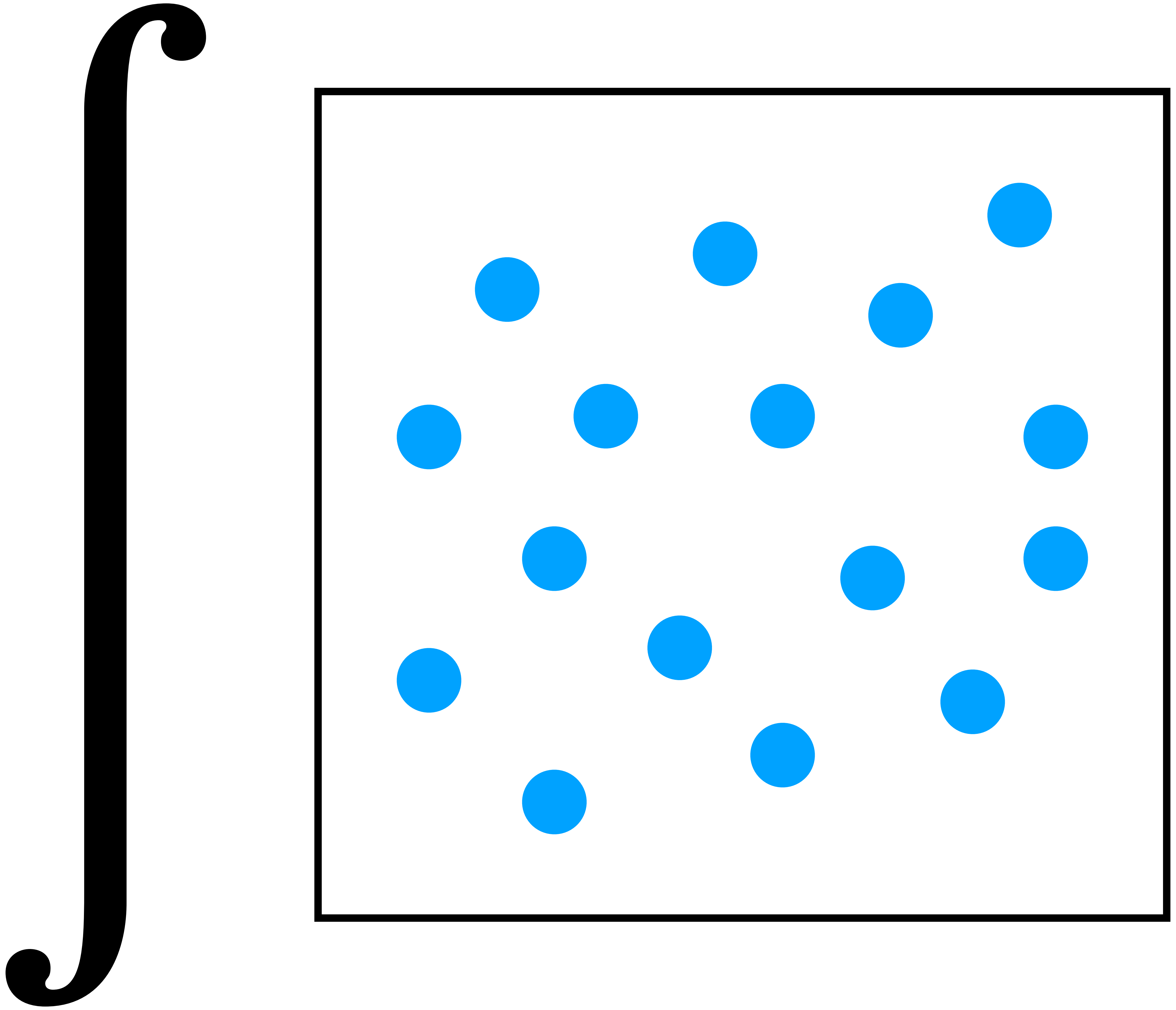
$$O(n^{-\frac{2}{d}-1})$$

Stochastic Quadrature Formulas

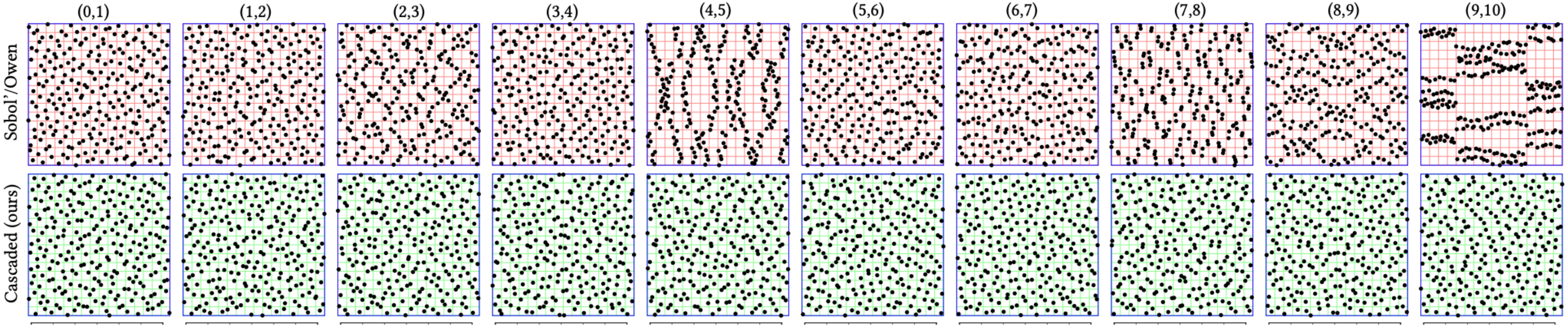
By Seymour Haber

Big picture: numerical integration is about placing samples to measure integrals

don't get stuck by things like unbiasedness!



Next: low-discrepancy sampling



https://perso.liris.cnrs.fr/david.coeurjolly/publication/cascaded2021/