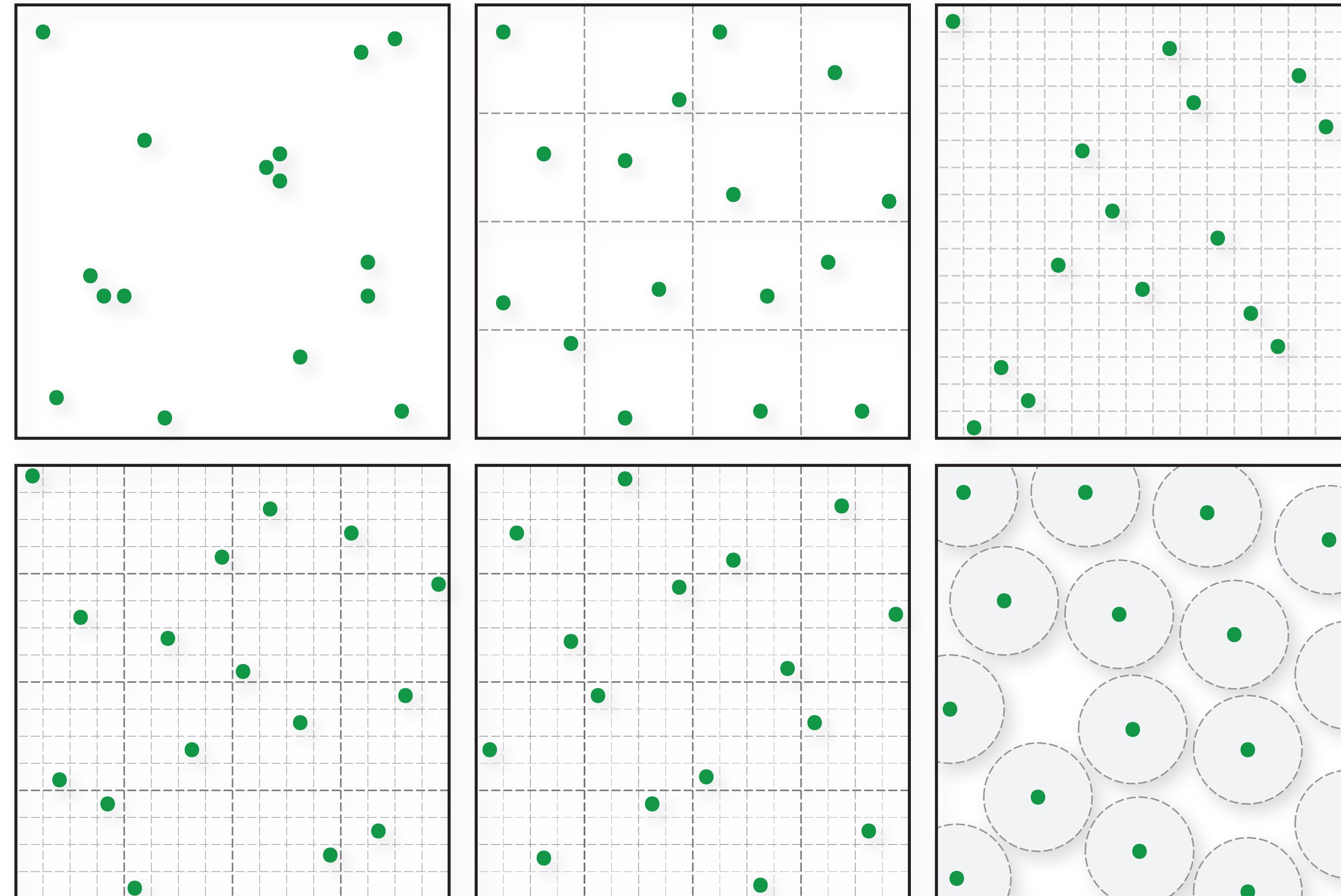


POPULAR SAMPLING PATTERNS

Fourier Analysis of Numerical Integration in Monte Carlo Rendering



Dartmouth

Wojciech Jarosz
wjarosz@dartmouth.edu

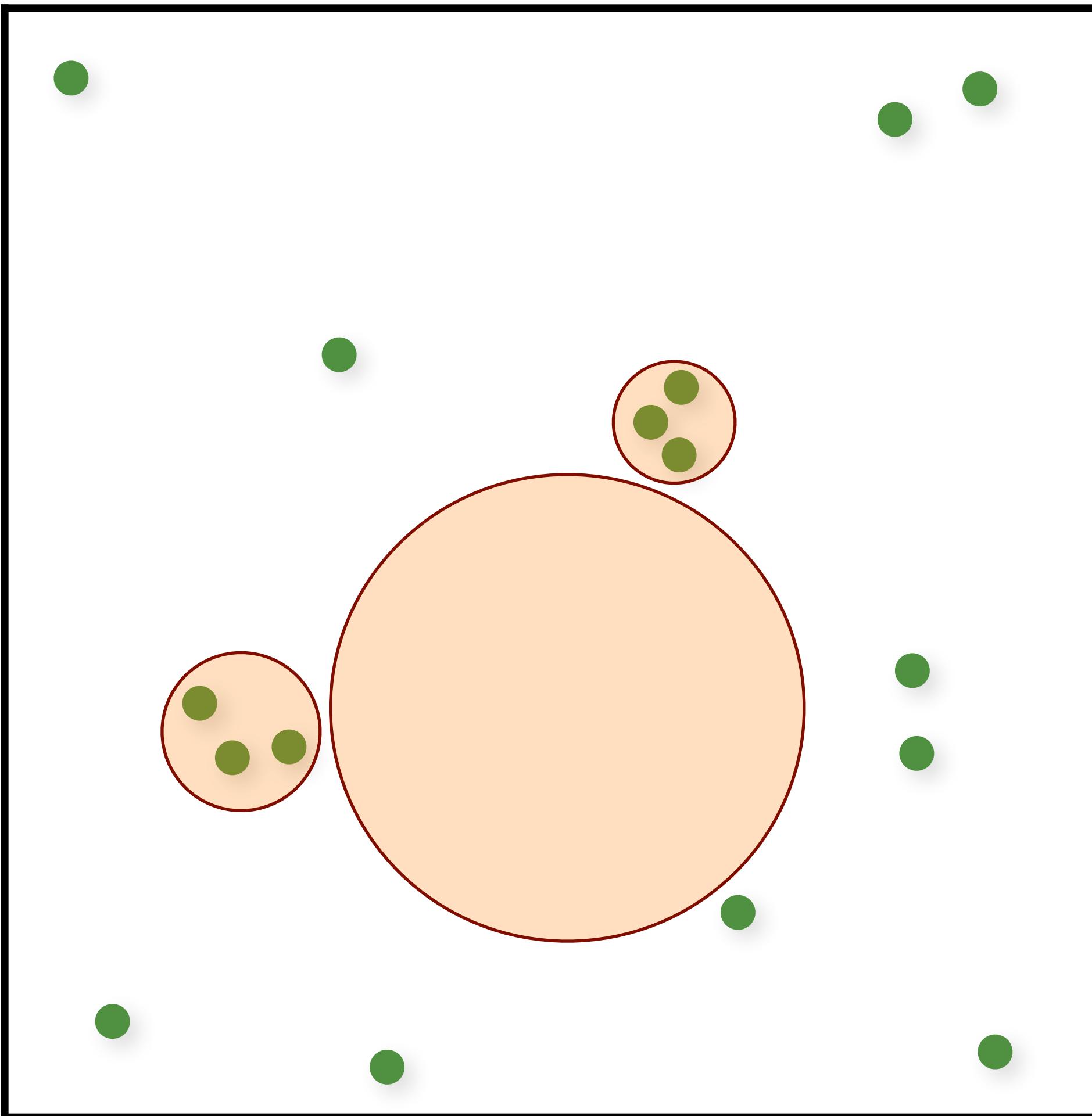
✓ CL

Need some segue/intro slide

Independent Random Sampling

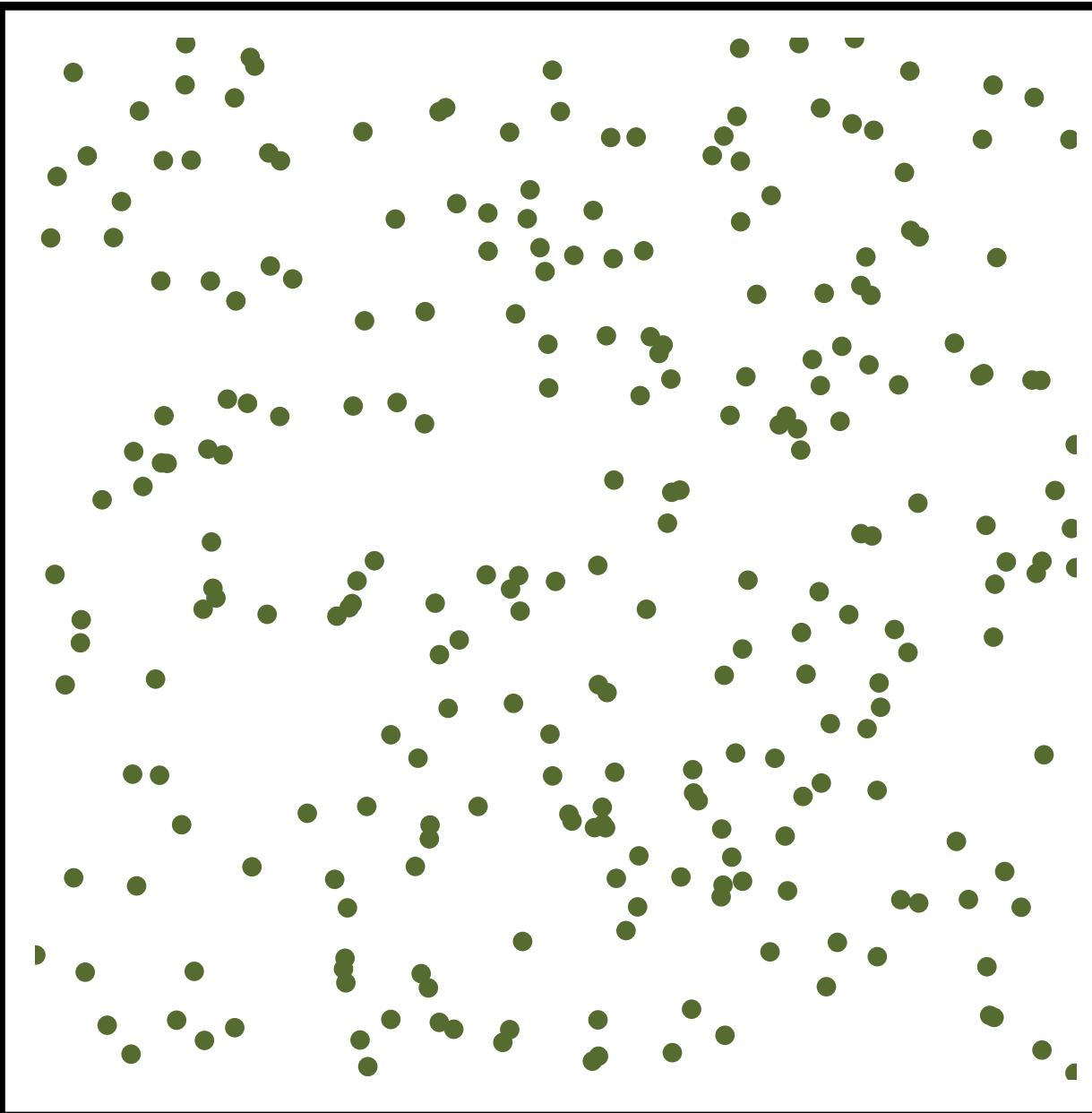
```
for (uint i = 0; i < numSamples; i++)  
{  
    samples(i).x = randf();  
    samples(i).y = randf();  
}
```

Independent Random Sampling

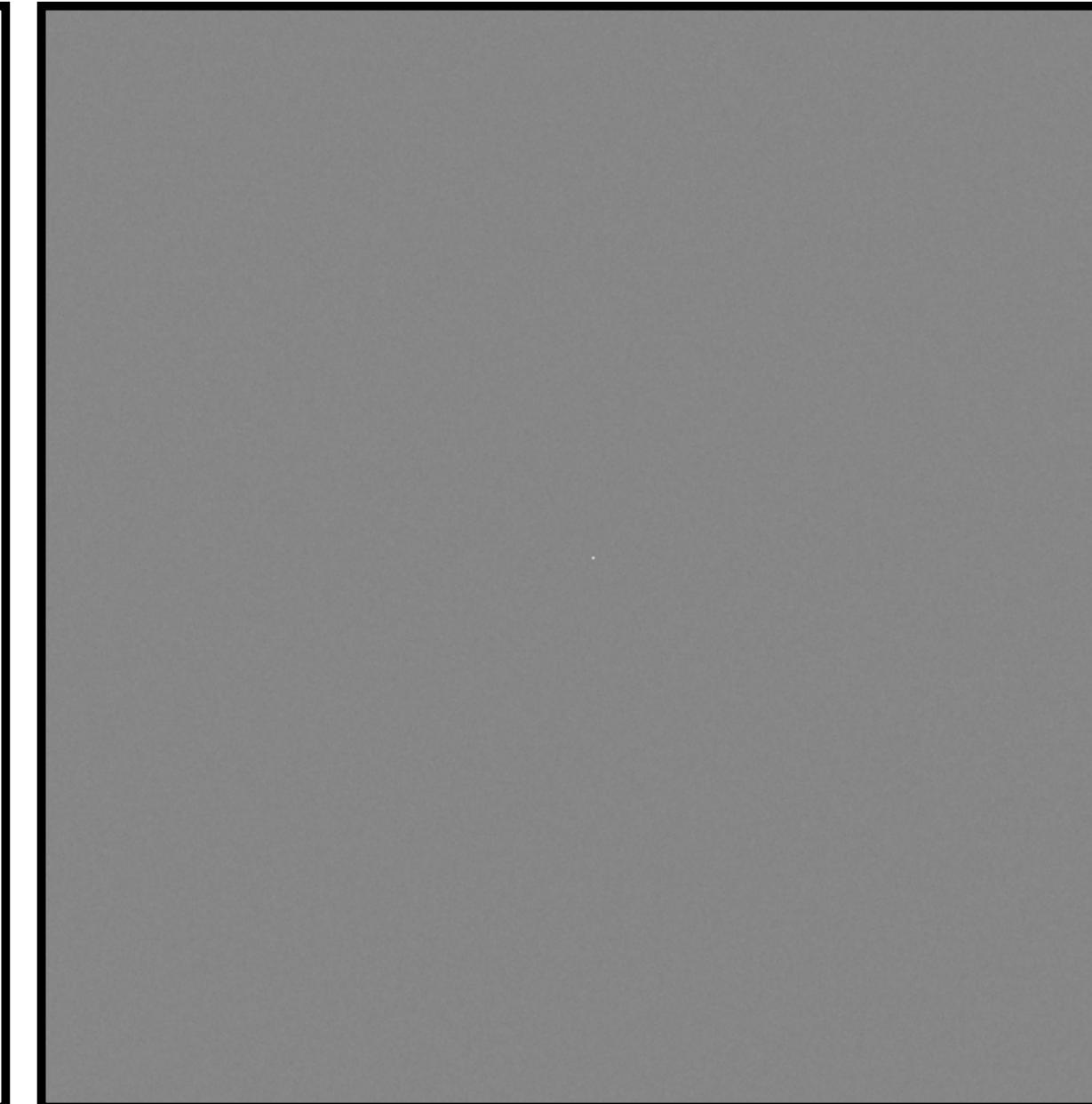


Independent Random Sampling

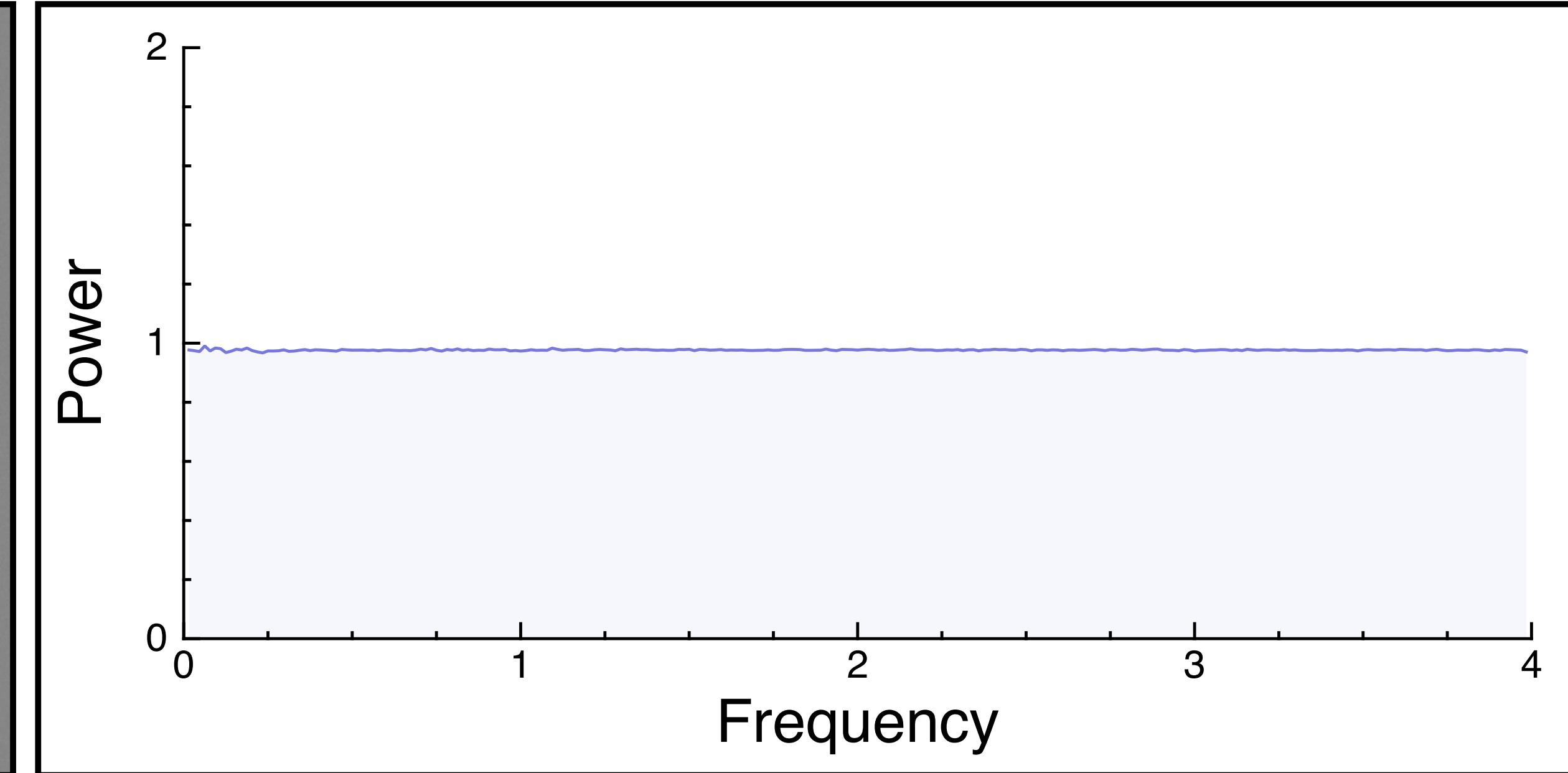
Samples



Power spectrum



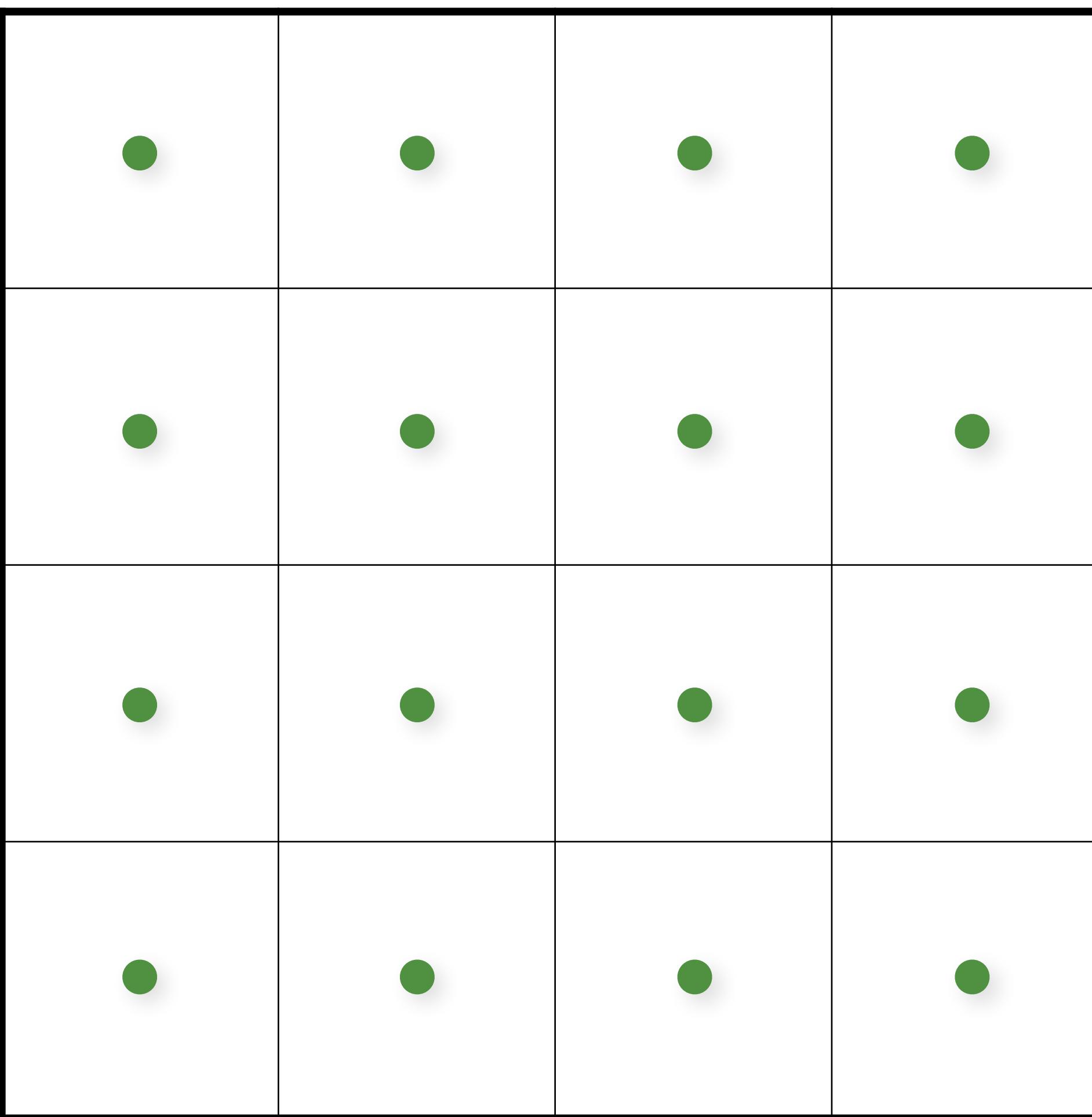
Radial mean



Regular Sampling Pseudo-code

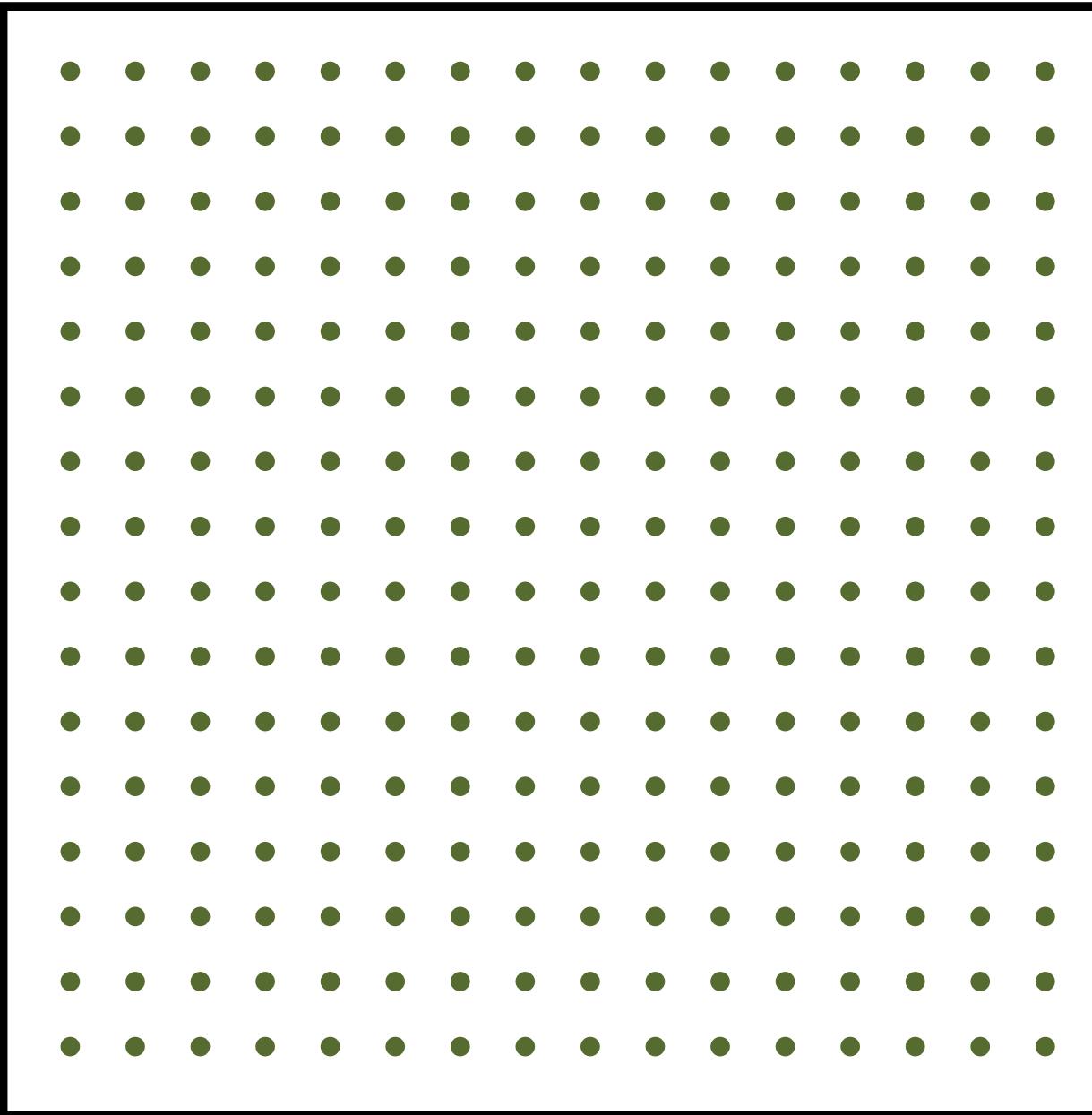
```
for (uint i = 0; i < numSamplesX; i++)
{
    for (uint j = 0; j < numSamplesY; j++)
    {
        samples(i,j).x = (i + 0.5)/numSamplesX;
        samples(i,j).y = (j + 0.5)/numSamplesY;
    }
}
```

Regular Sampling

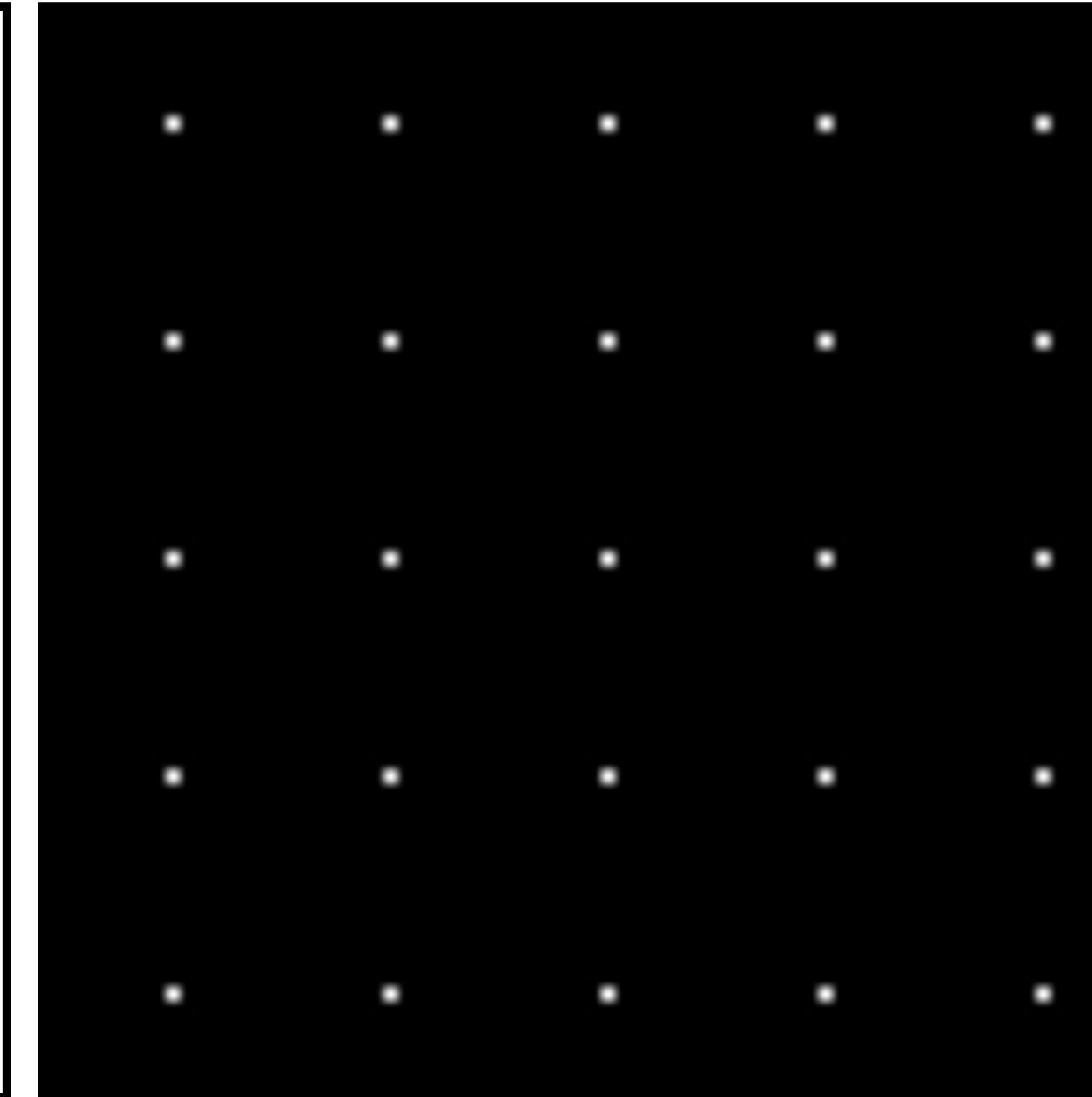


Independent Random Sampling

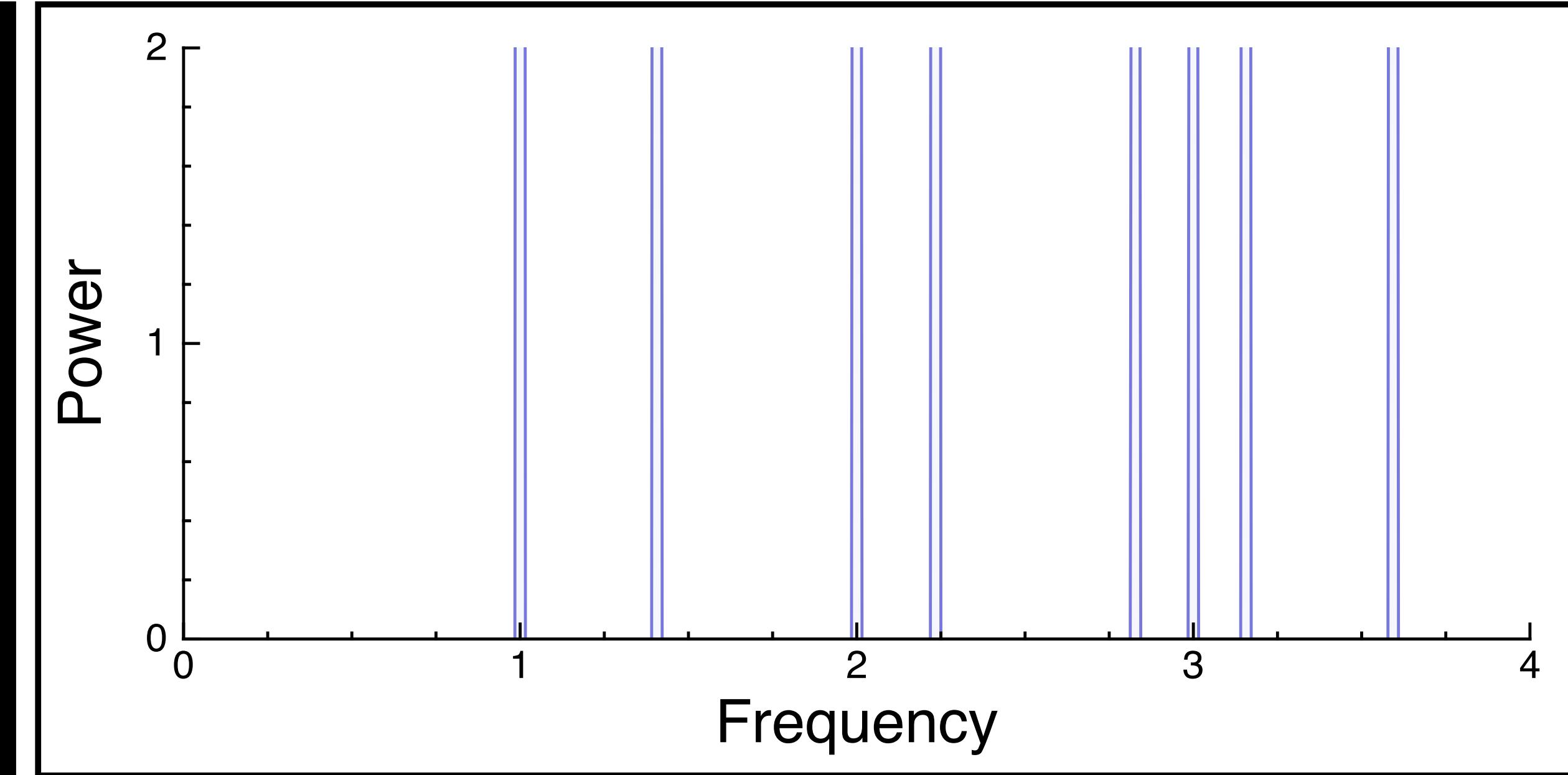
Samples



Power spectrum



Radial mean



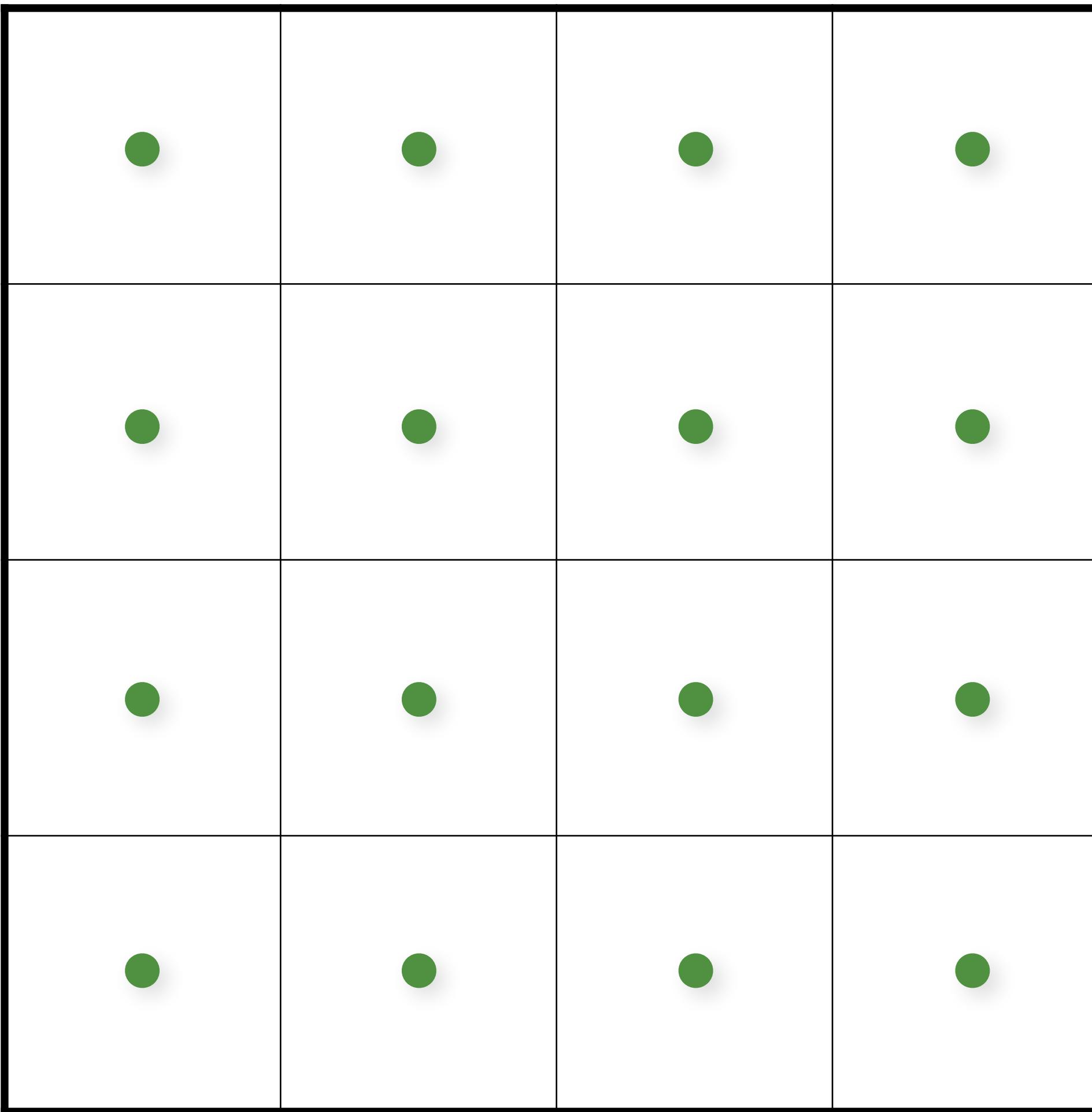
Regular Sampling Pseudo-code

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

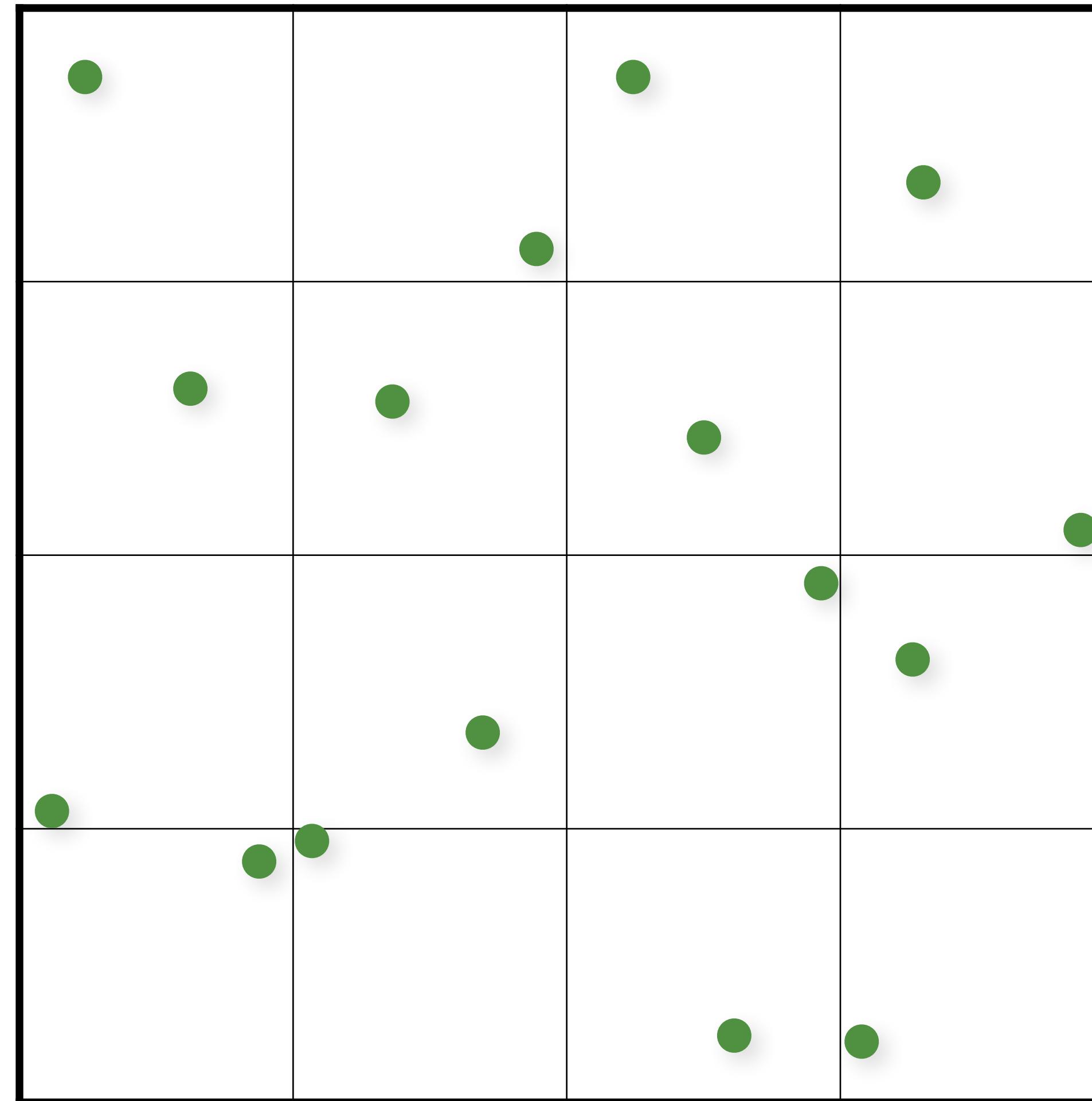
Jittered Sampling Pseudo-code

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

Regular Sampling

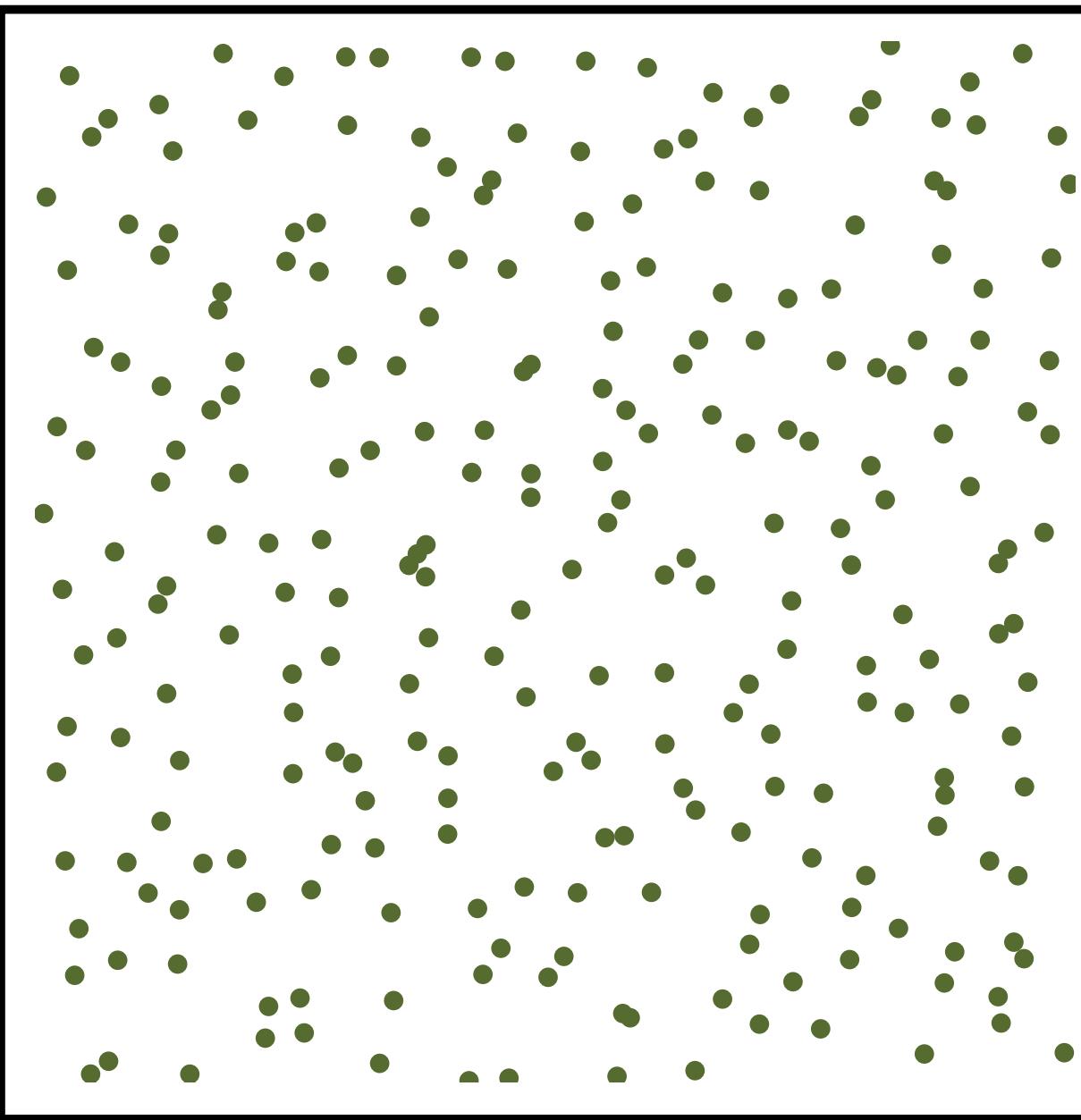


Stratified/Jittered Sampling

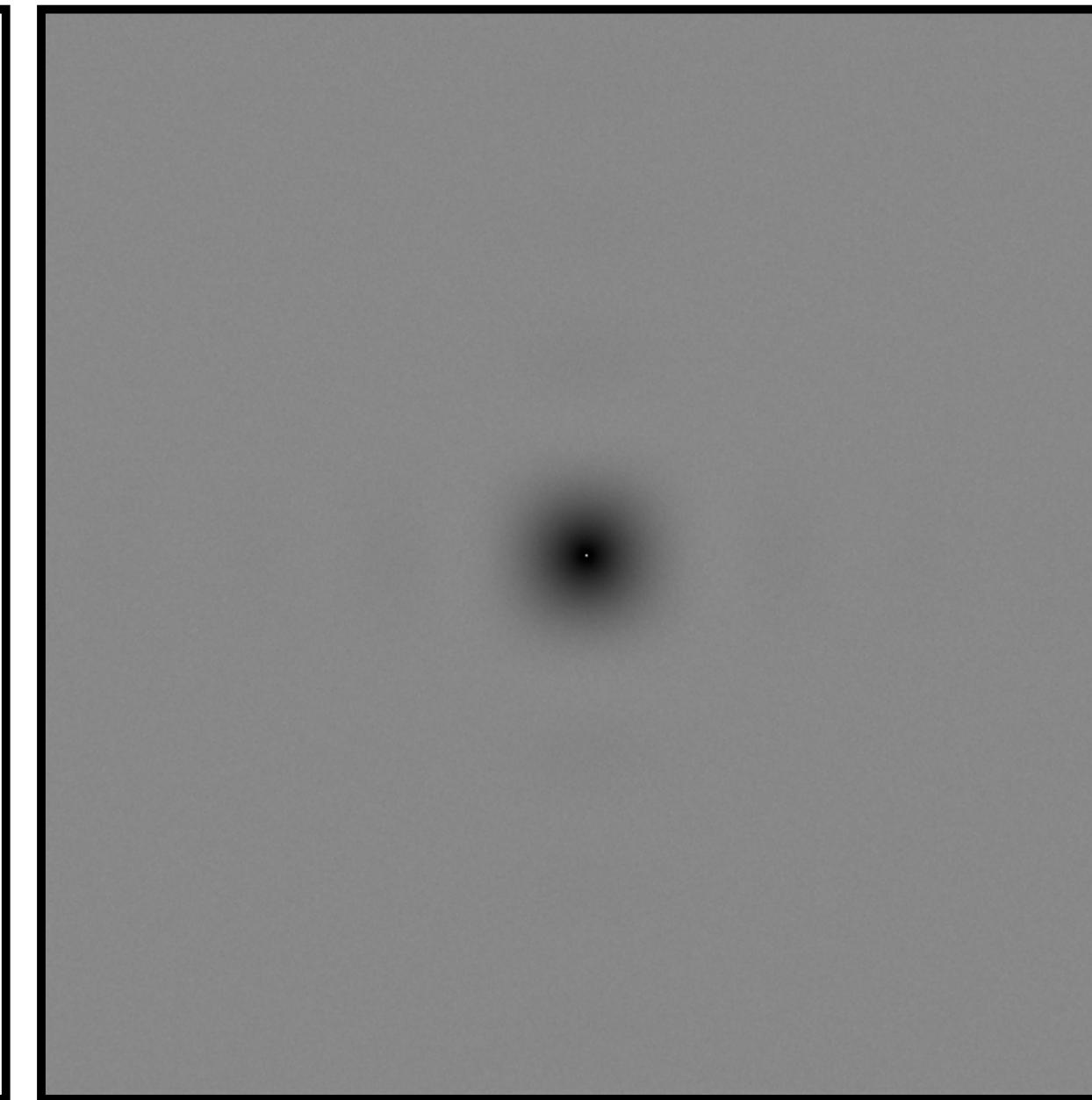


Stratified/Jittered Sampling

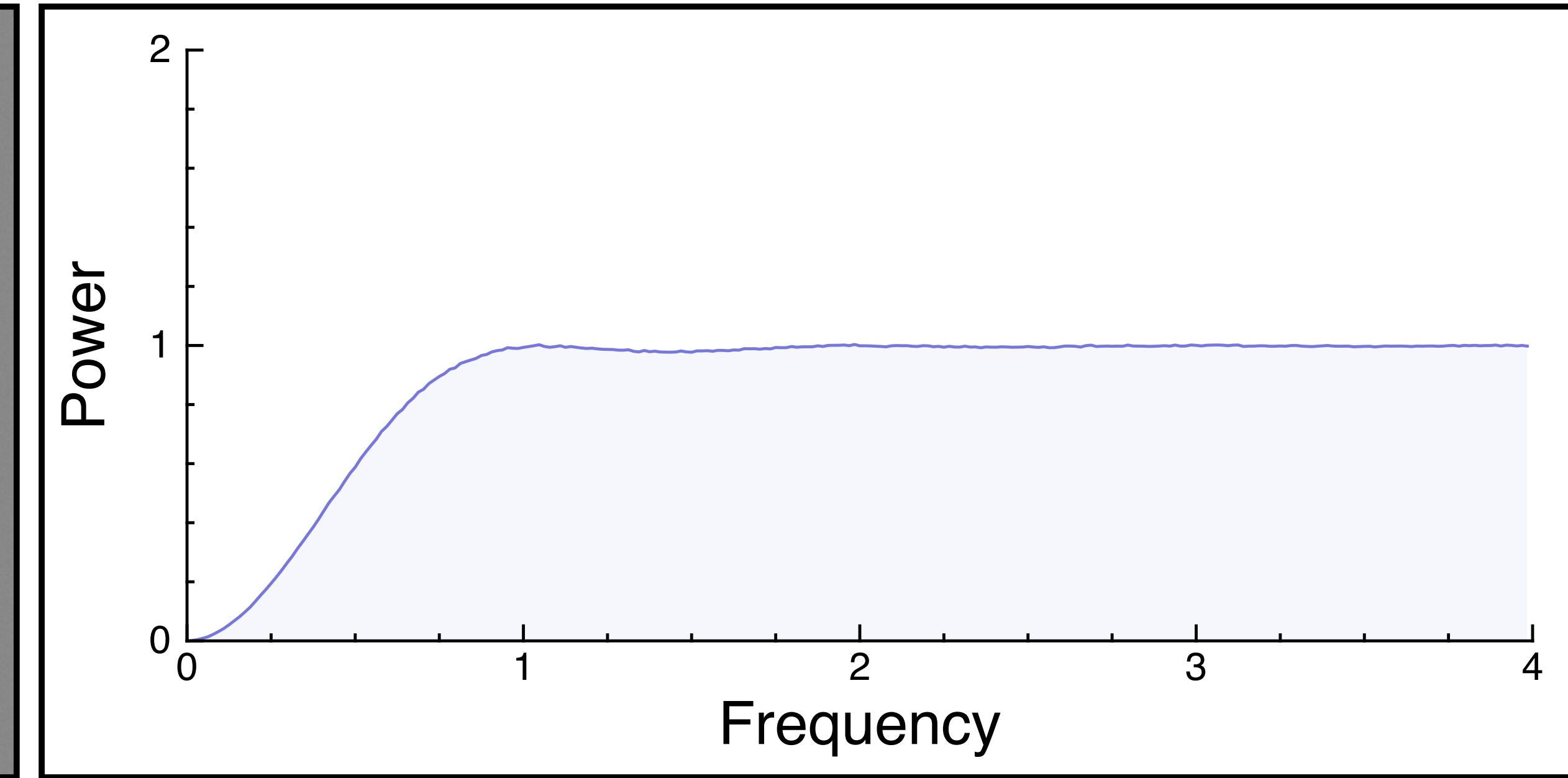
Samples



Power spectrum

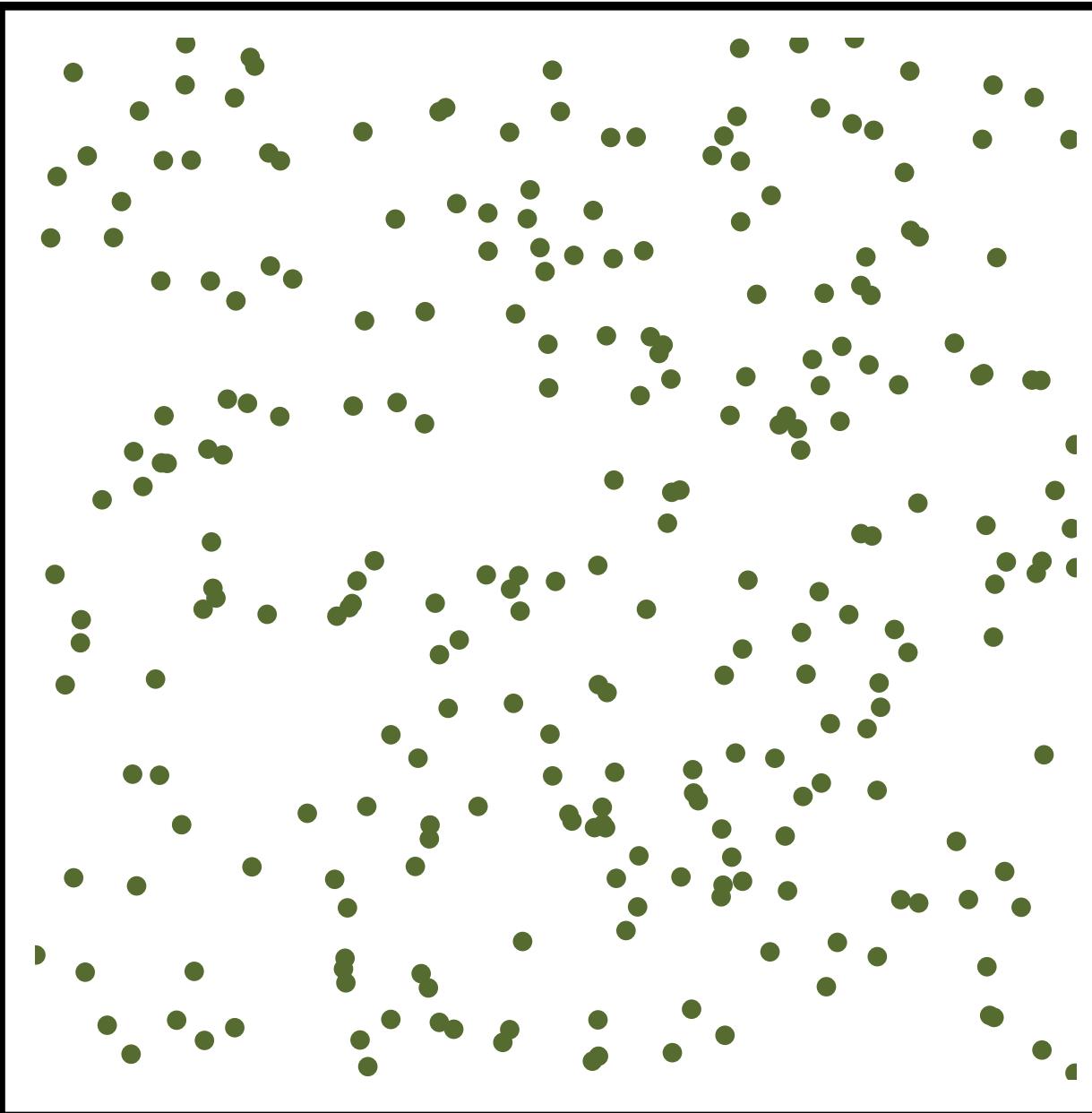


Radial mean

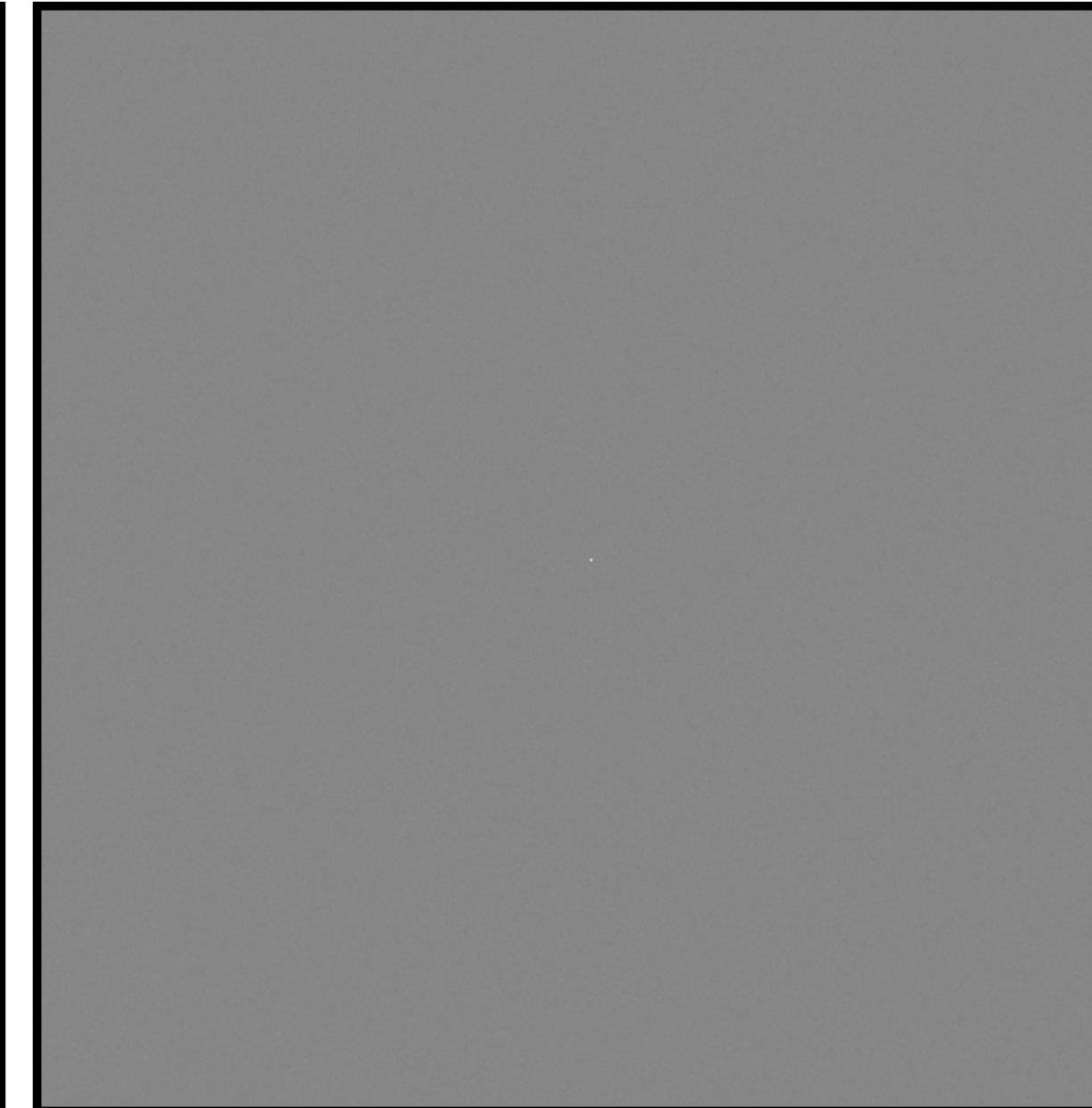


Independent Random Sampling

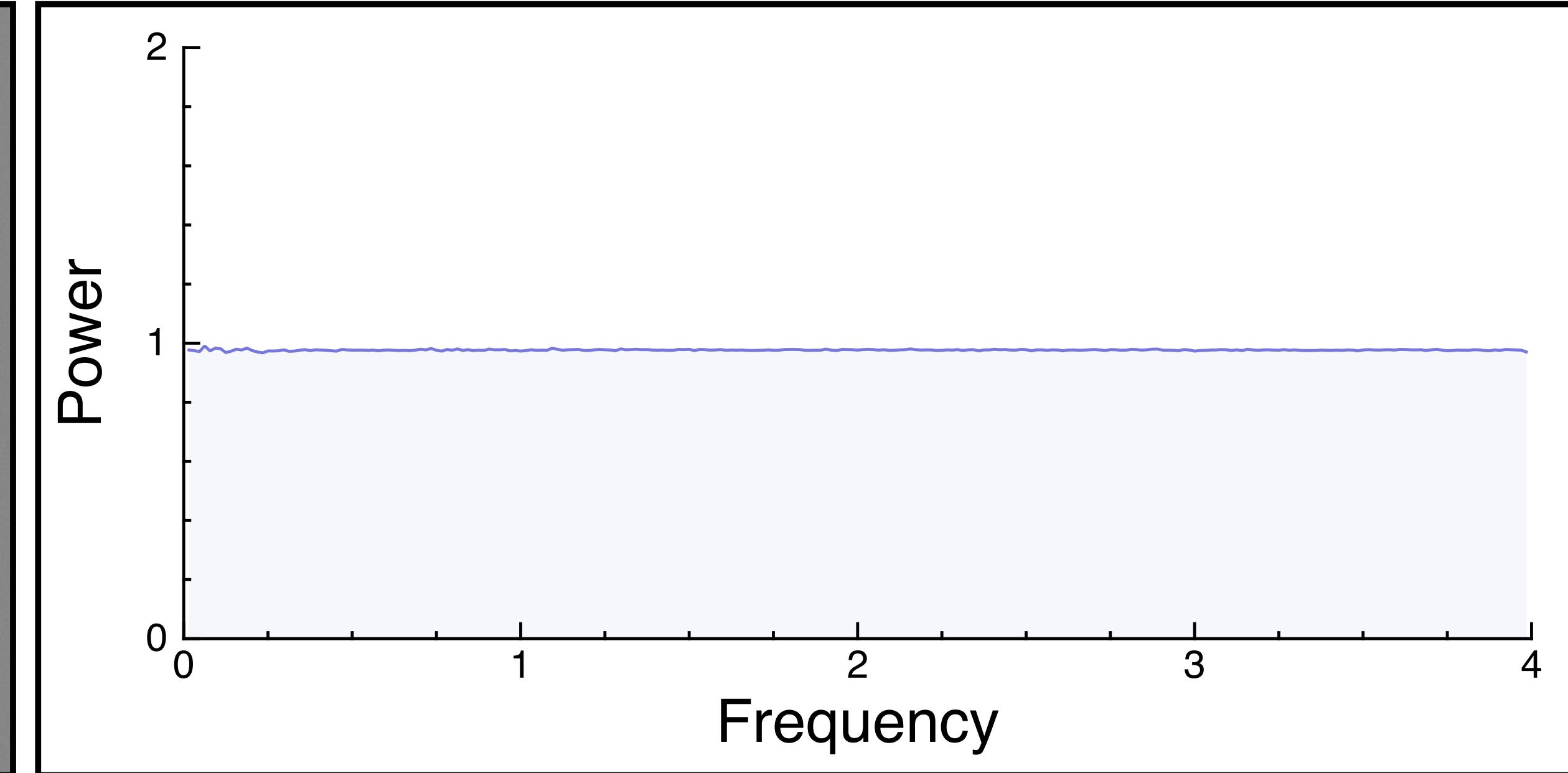
Samples



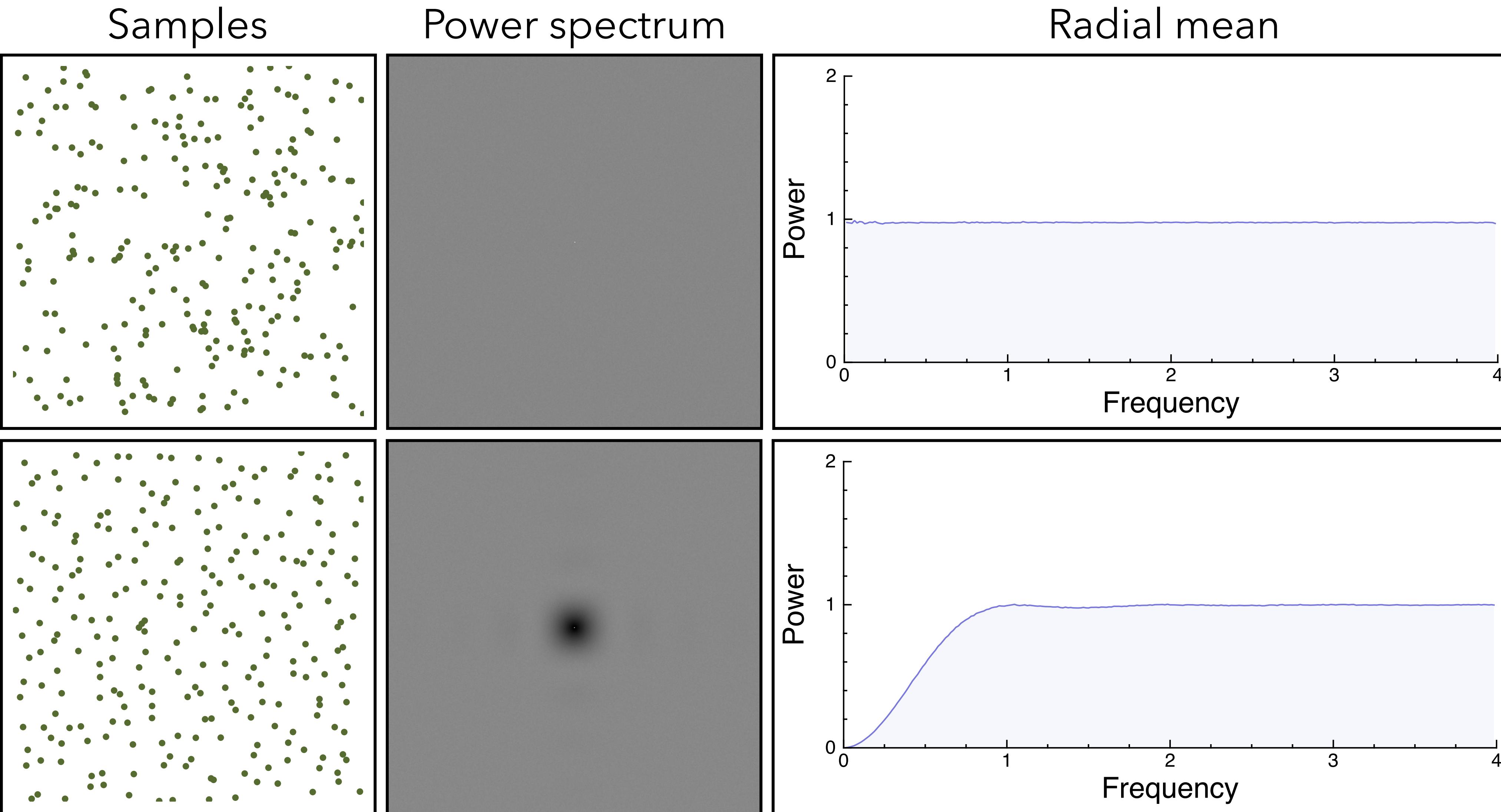
Power spectrum



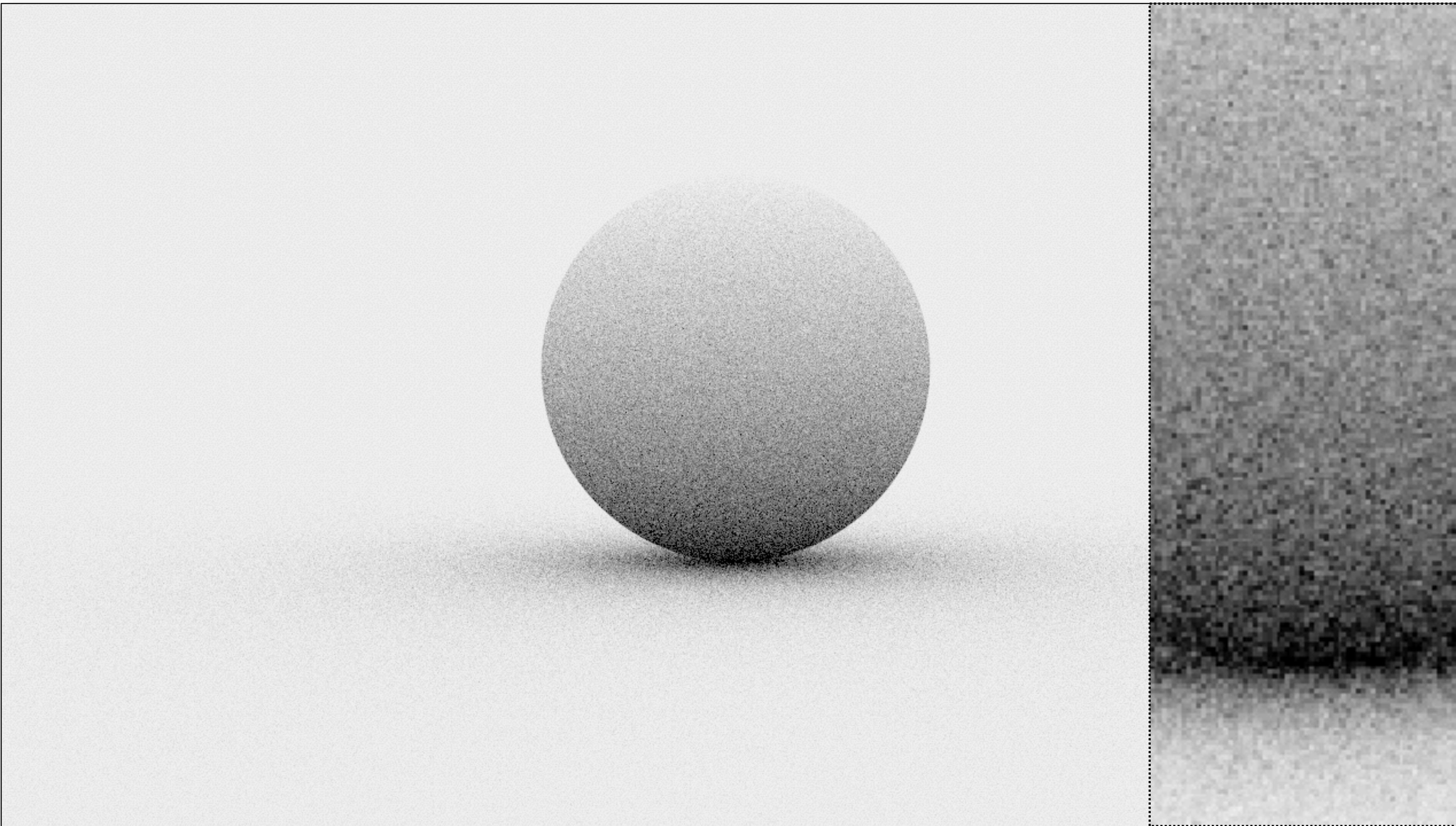
Radial mean



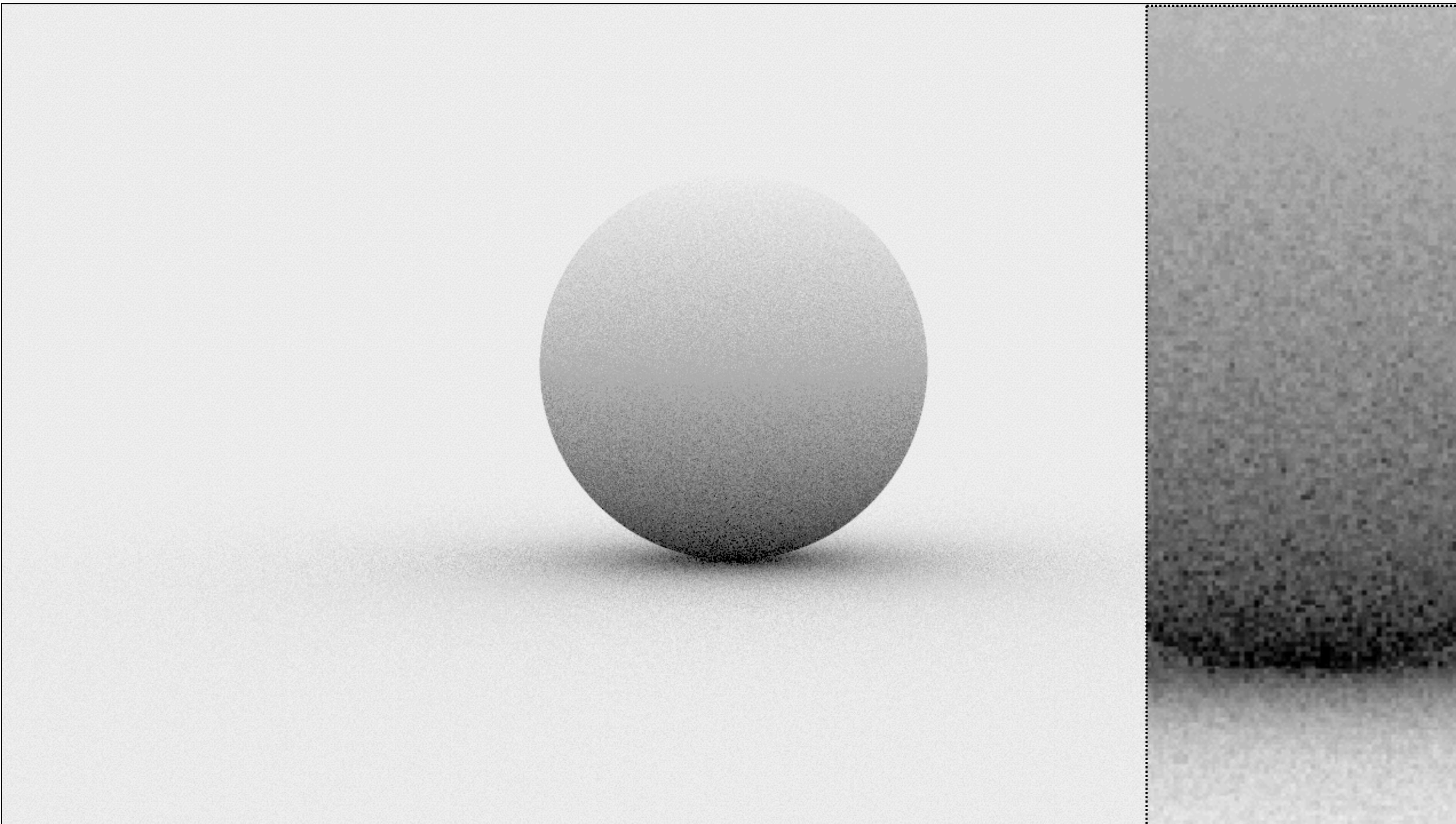
Random vs. Jittered Sampling



Monte Carlo (16 random samples)



Monte Carlo (16 stratified samples)



Stratifying in Higher Dimensions

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!

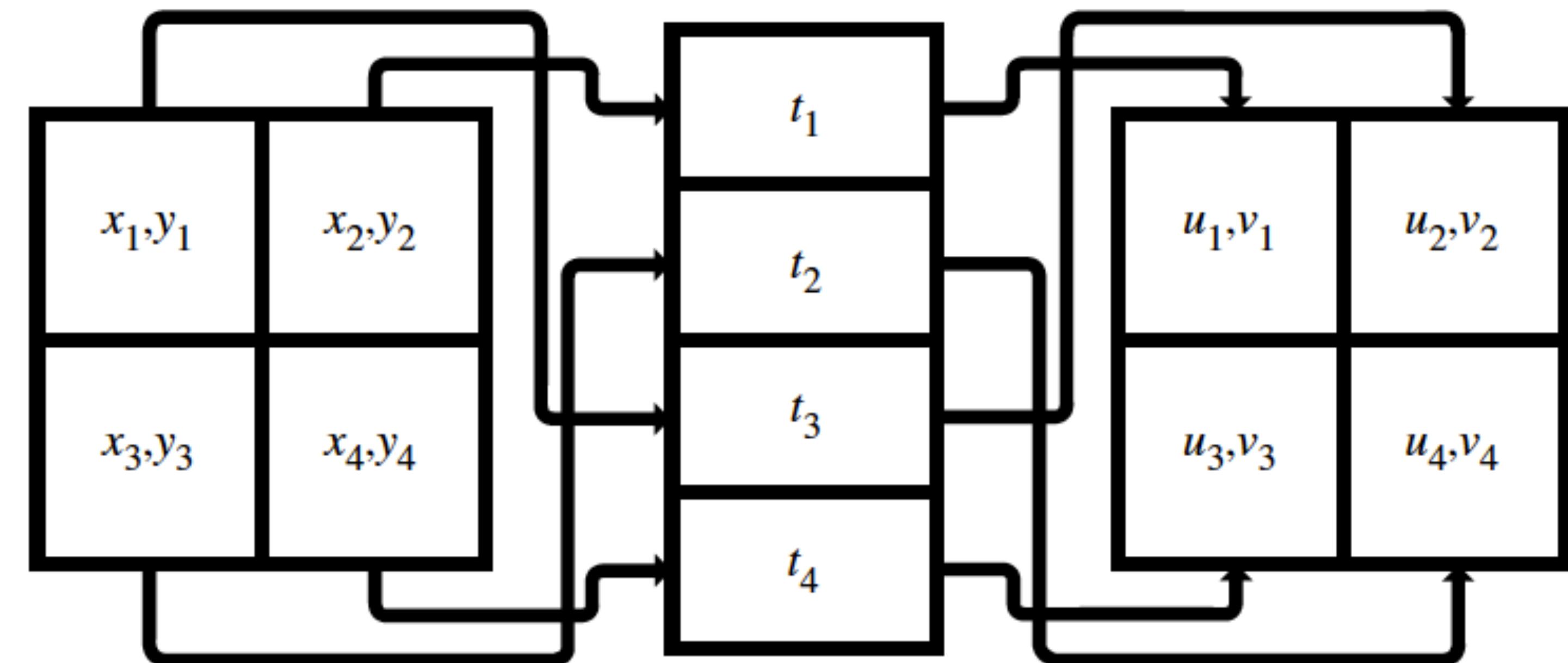
Inconvenient for large d

- cannot select sample count with fine granularity

Uncorrelated Jitter [Cook et al. 84]

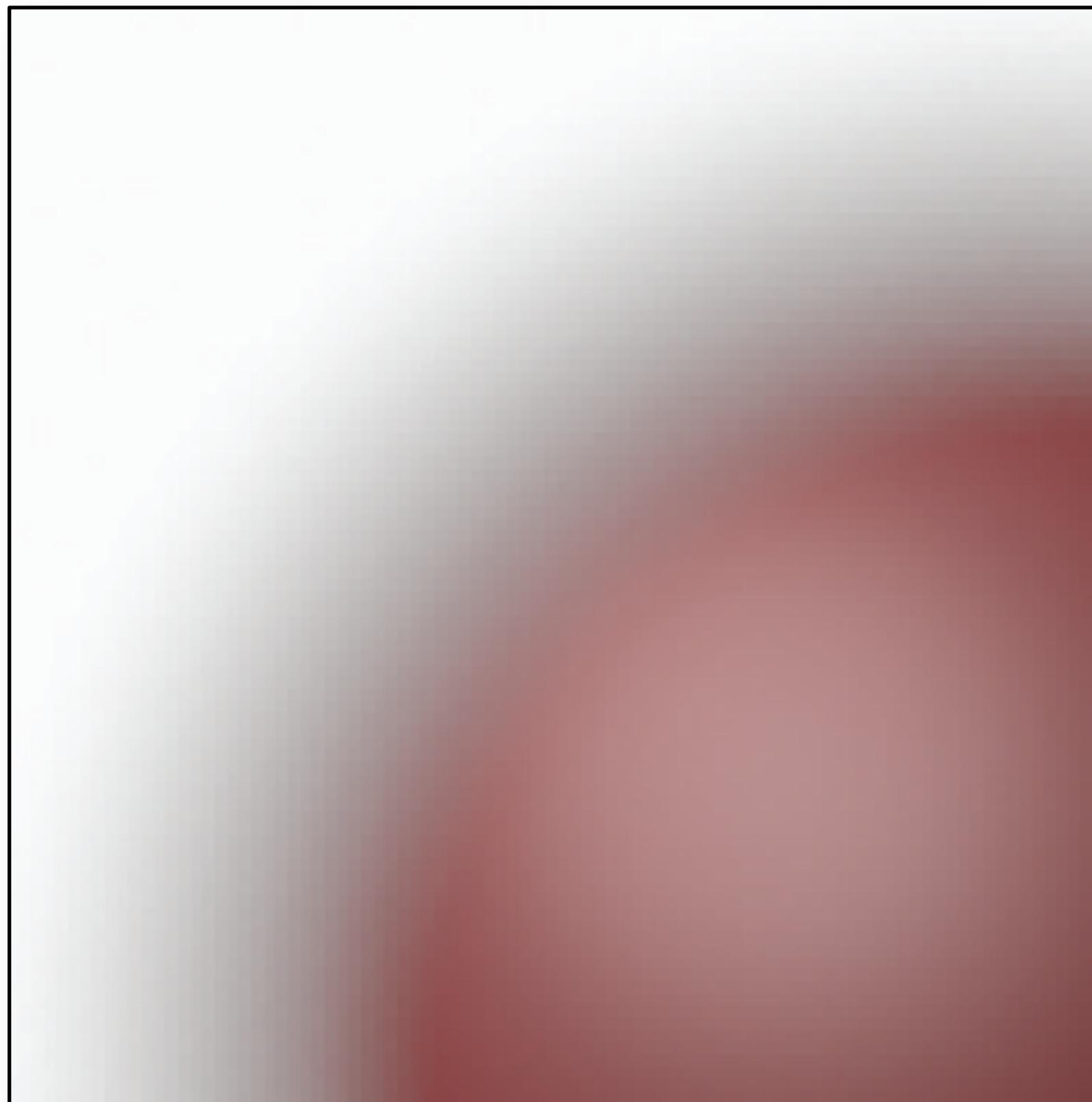
Compute stratified samples in lower-dimensions

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

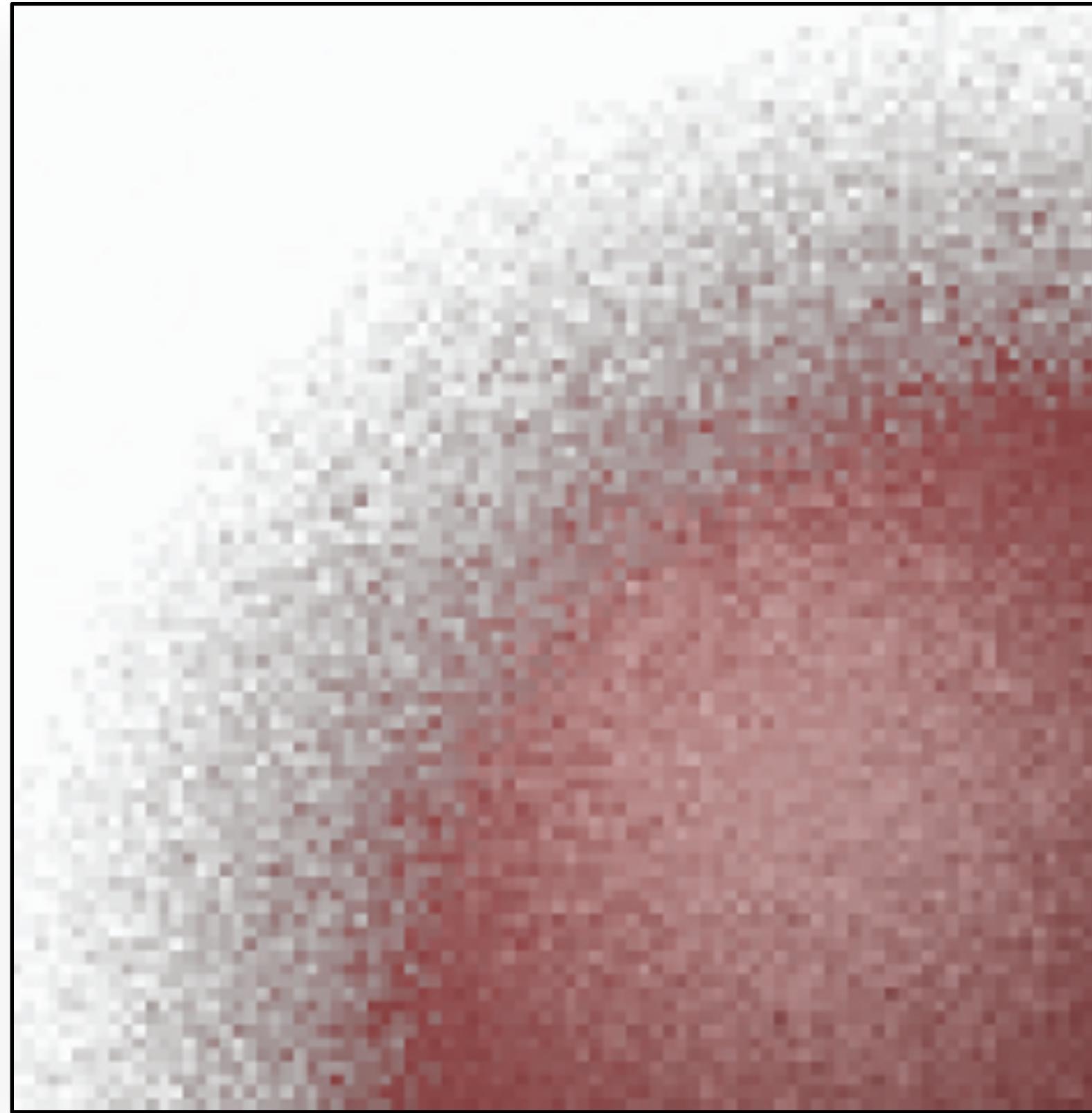


Depth of Field (4D)

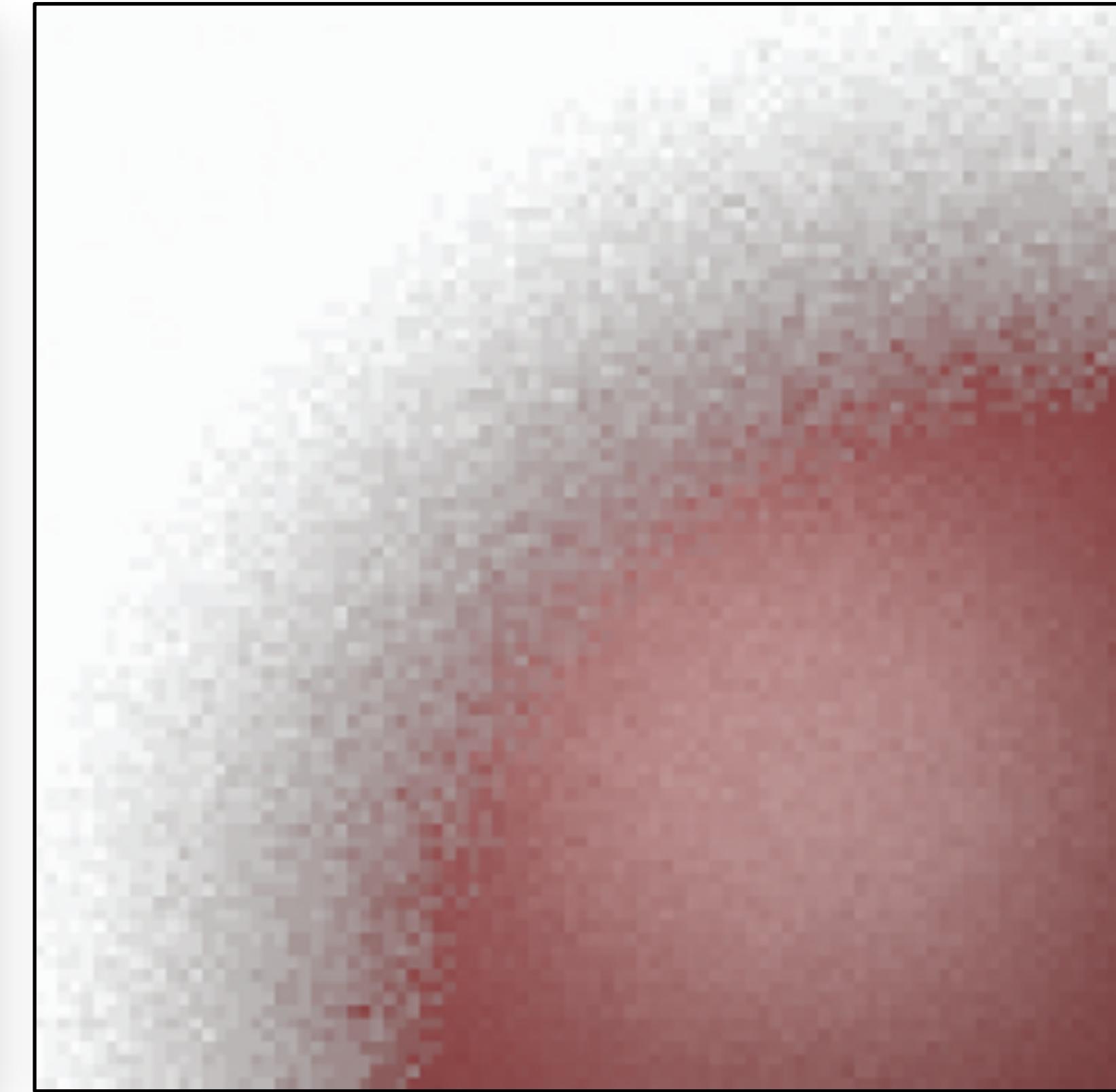
Reference



Random Sampling



Uncorrelated Jitter

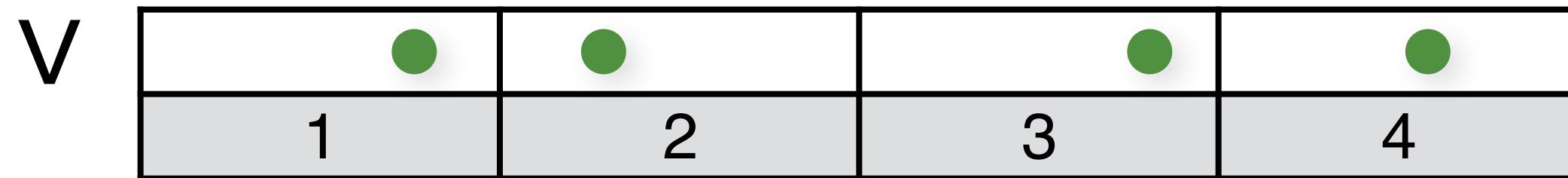
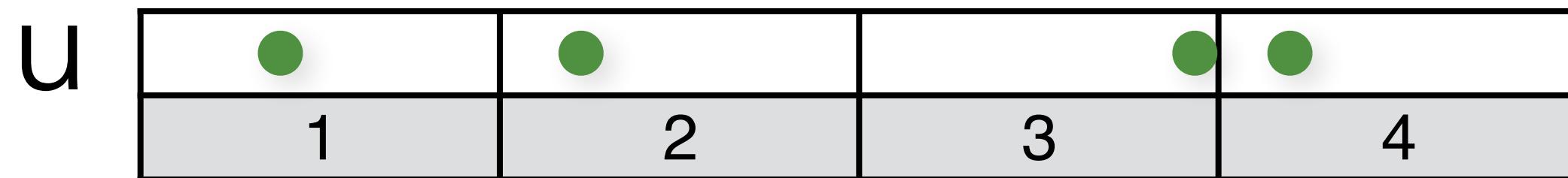
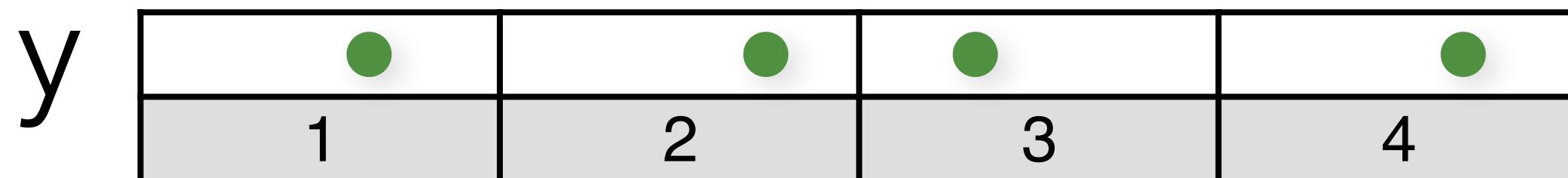
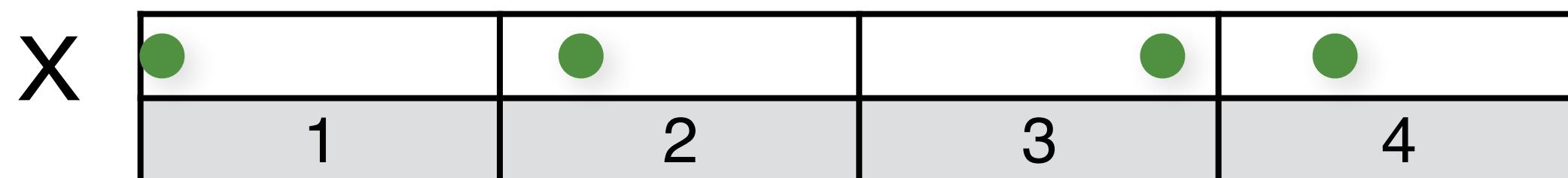


Uncorrelated Jitter -> Latin Hypercube

Stratify samples in each dimension separately

- for 5D: 5 separate 1D jittered point sets

- combine dimensions
in random order

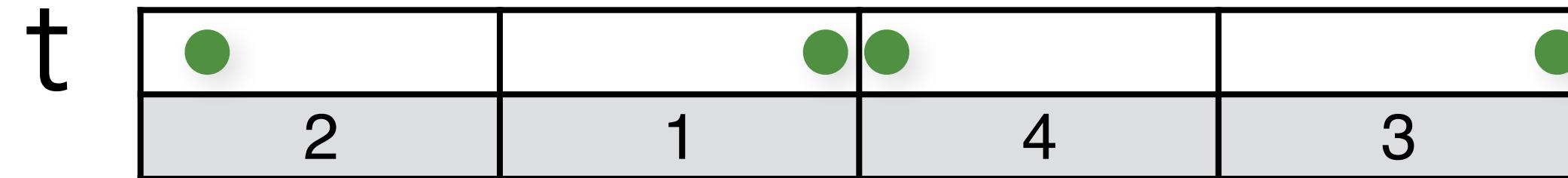
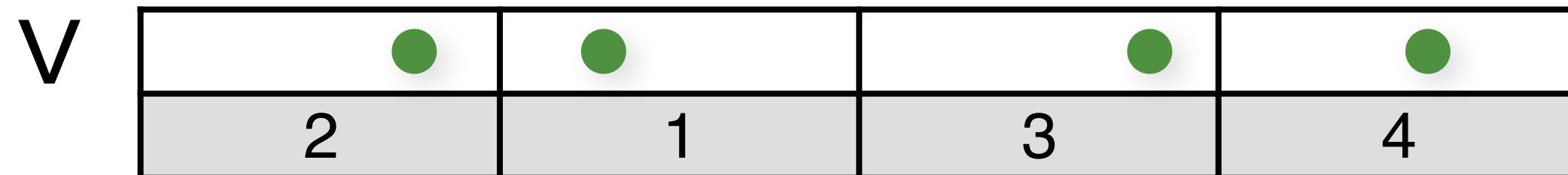
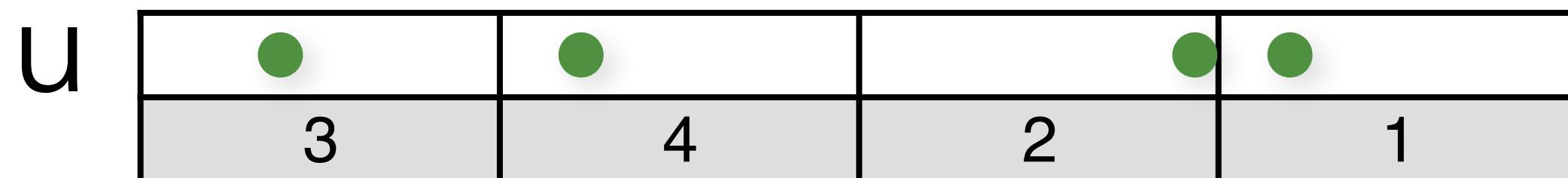
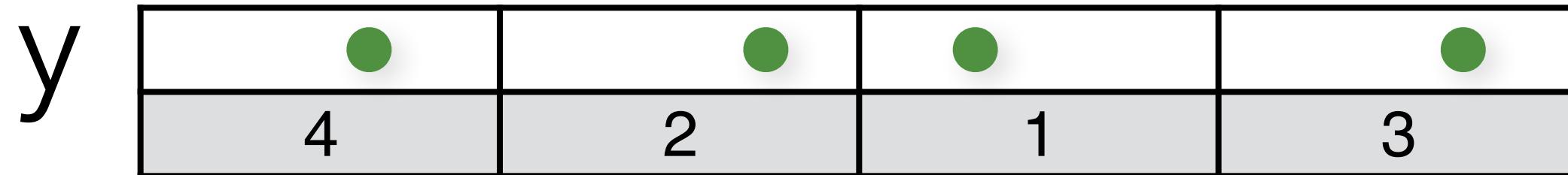
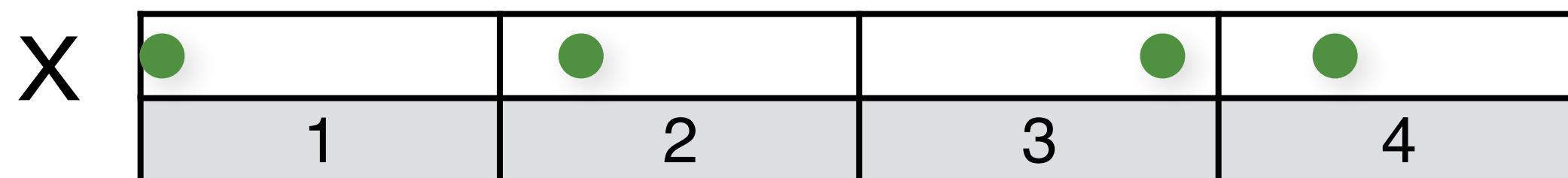


Uncorrelated Jitter -> Latin Hypercube

Stratify samples in each dimension separately

- for 5D: 5 separate 1D jittered point sets

- combine dimensions
in random order



Latin Hypercube Sampling

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

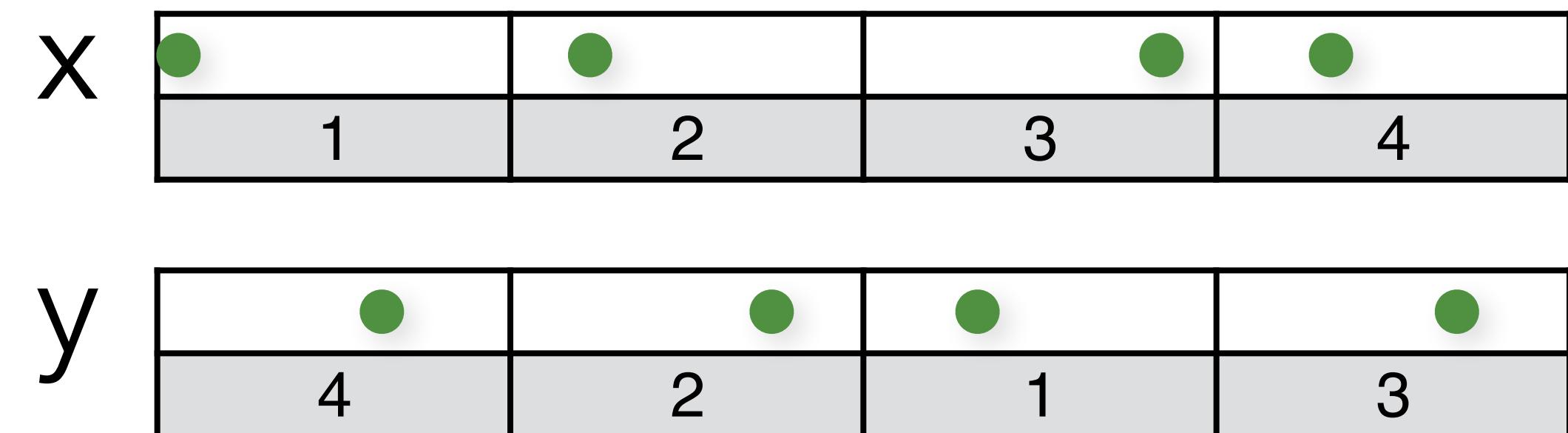
// shuffle each dimension independently
for (uint d = 0; d < numDimensions; d++)
    for (uint i = numSamples - 1; i >= 1; i--)
        swap(samples(d,i), samples(d, randi(0,i))));
```

N-Rooks = 2D Latin Hypercube

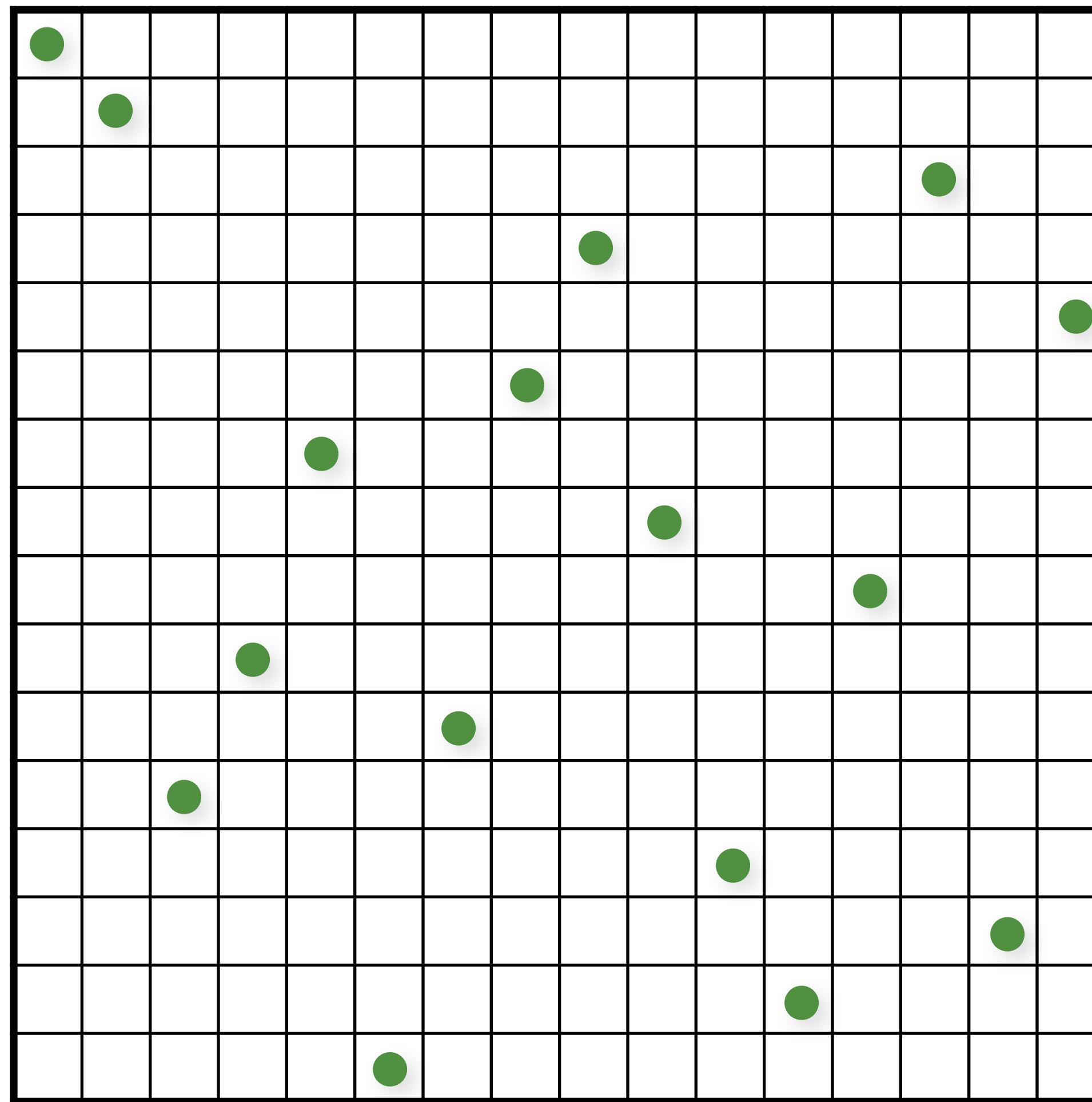
Stratify samples in each dimension separately

- for 2D: 2 separate 1D jittered point sets

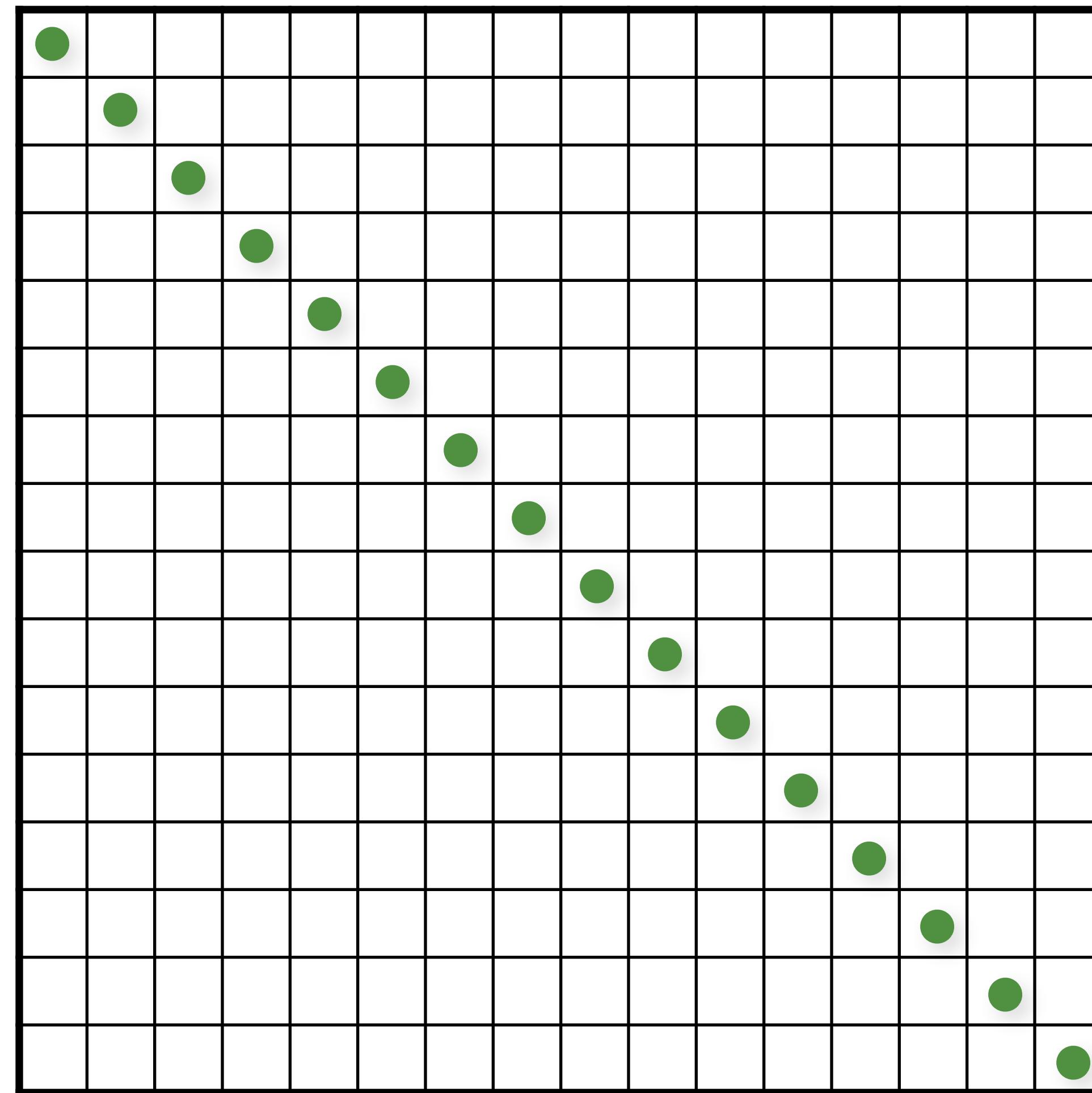
- combine dimensions
in random order



Latin Hypercube (N-Rooks) Sampling

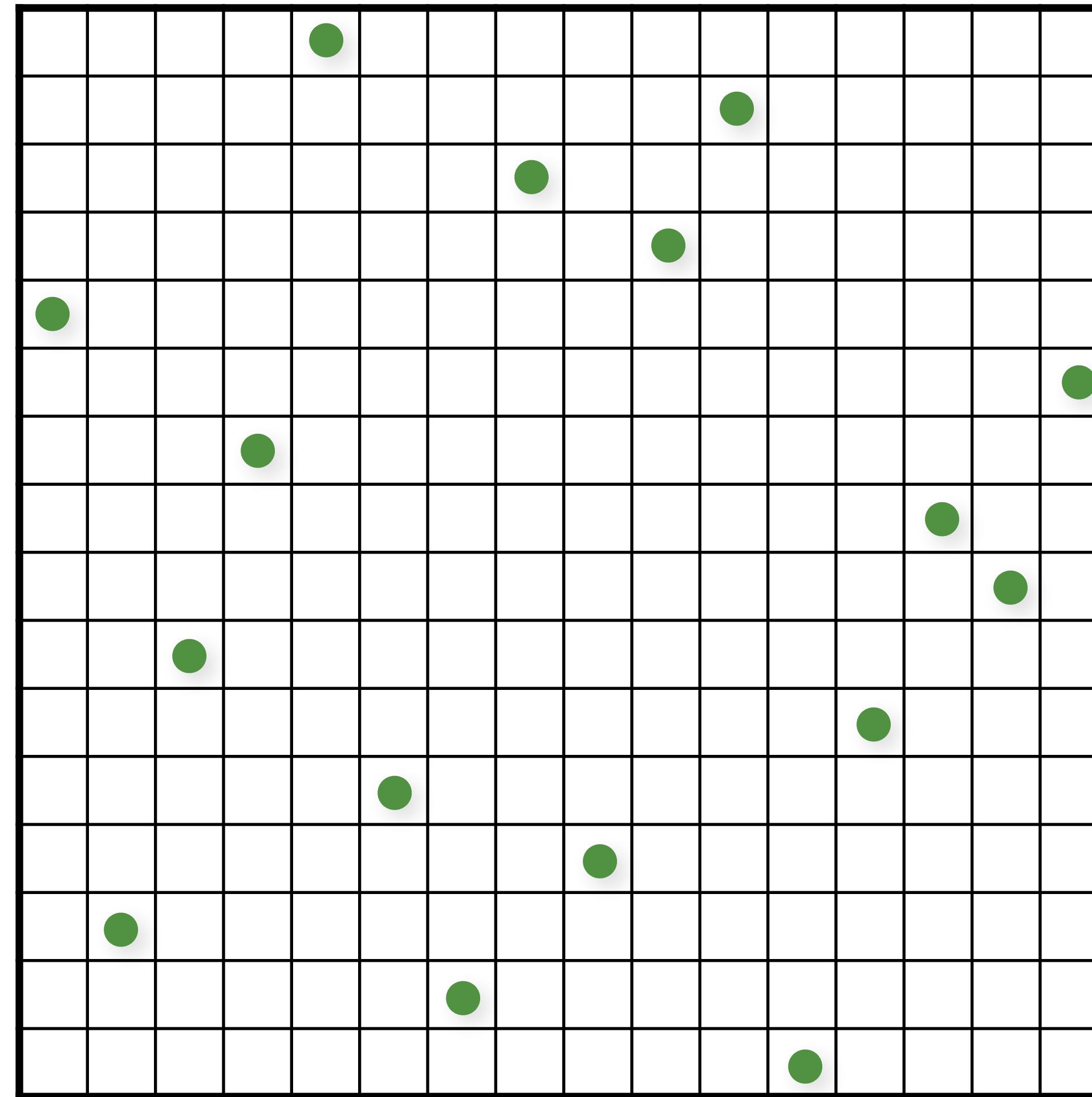


Latin Hypercube (N-Rooks) Sampling



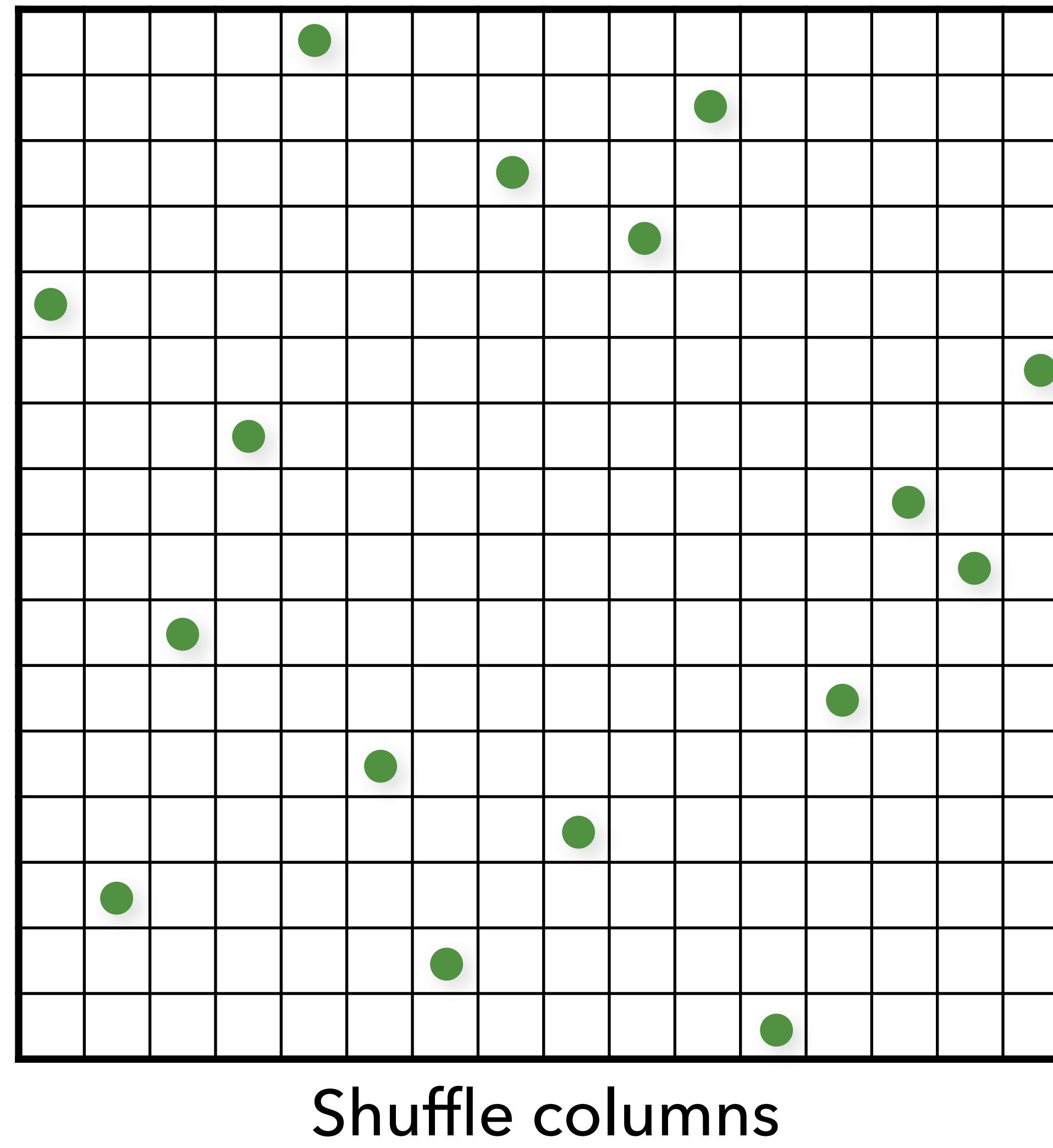
Shuffles

Latin Hypercube (N-Rooks) Sampling

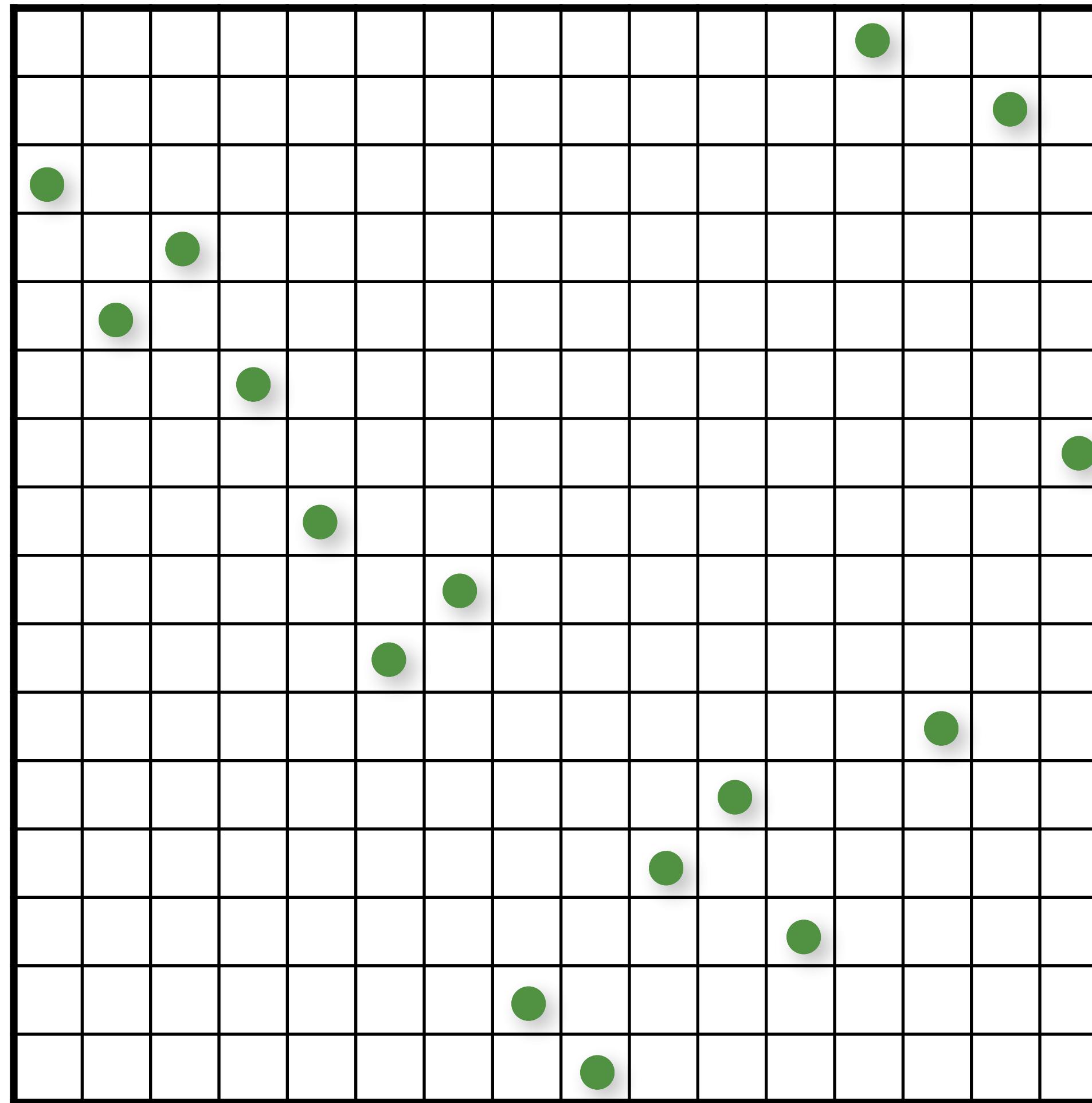


Shuffleleowns

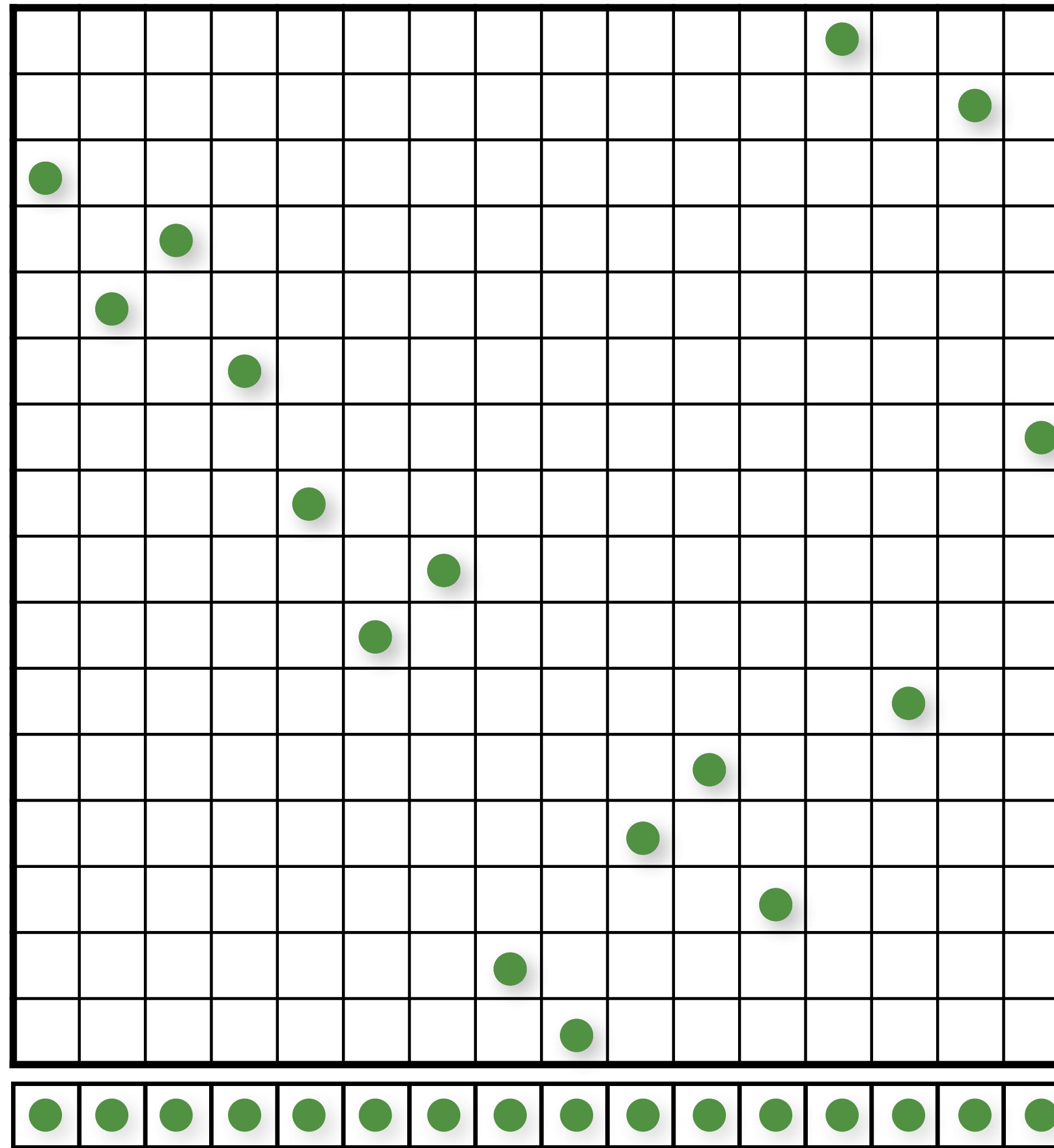
Latin Hypercube (N-Rooks) Sampling



Latin Hypercube (N-Rooks) Sampling

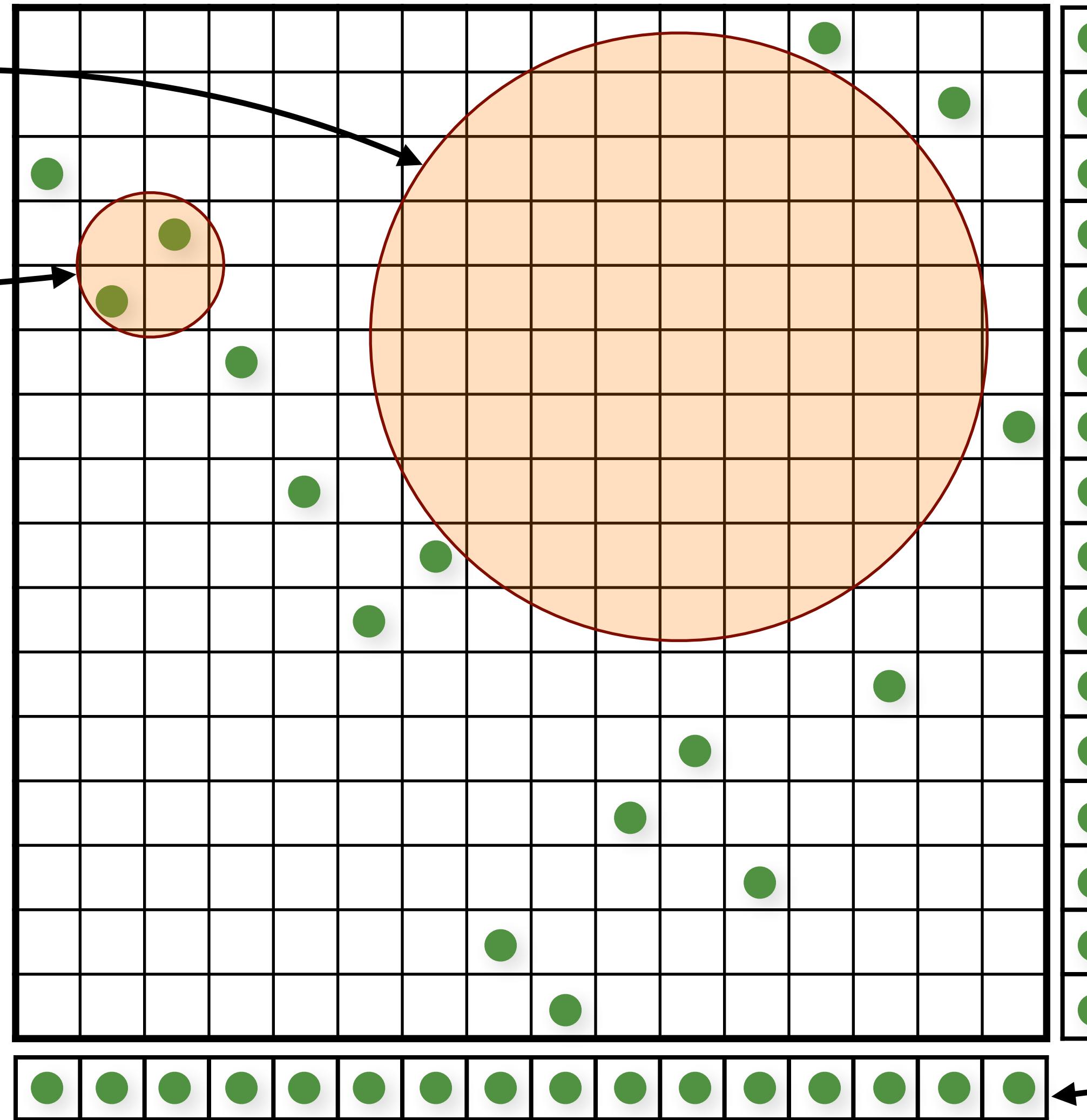


Latin Hypercube (N-Rooks) Sampling



Latin Hypercube (N-Rooks) Sampling

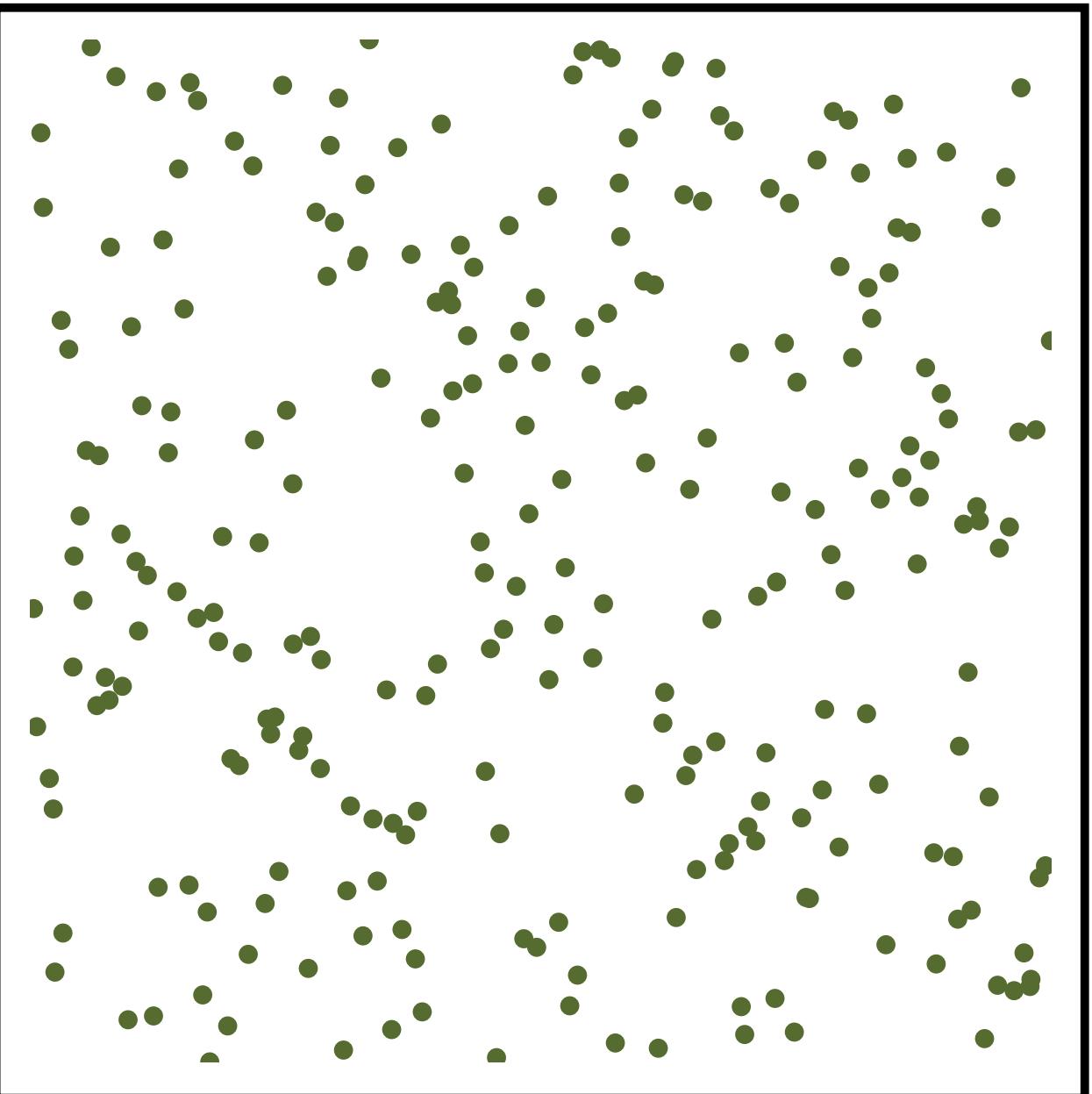
Unevenly distributed
in n-dimensions



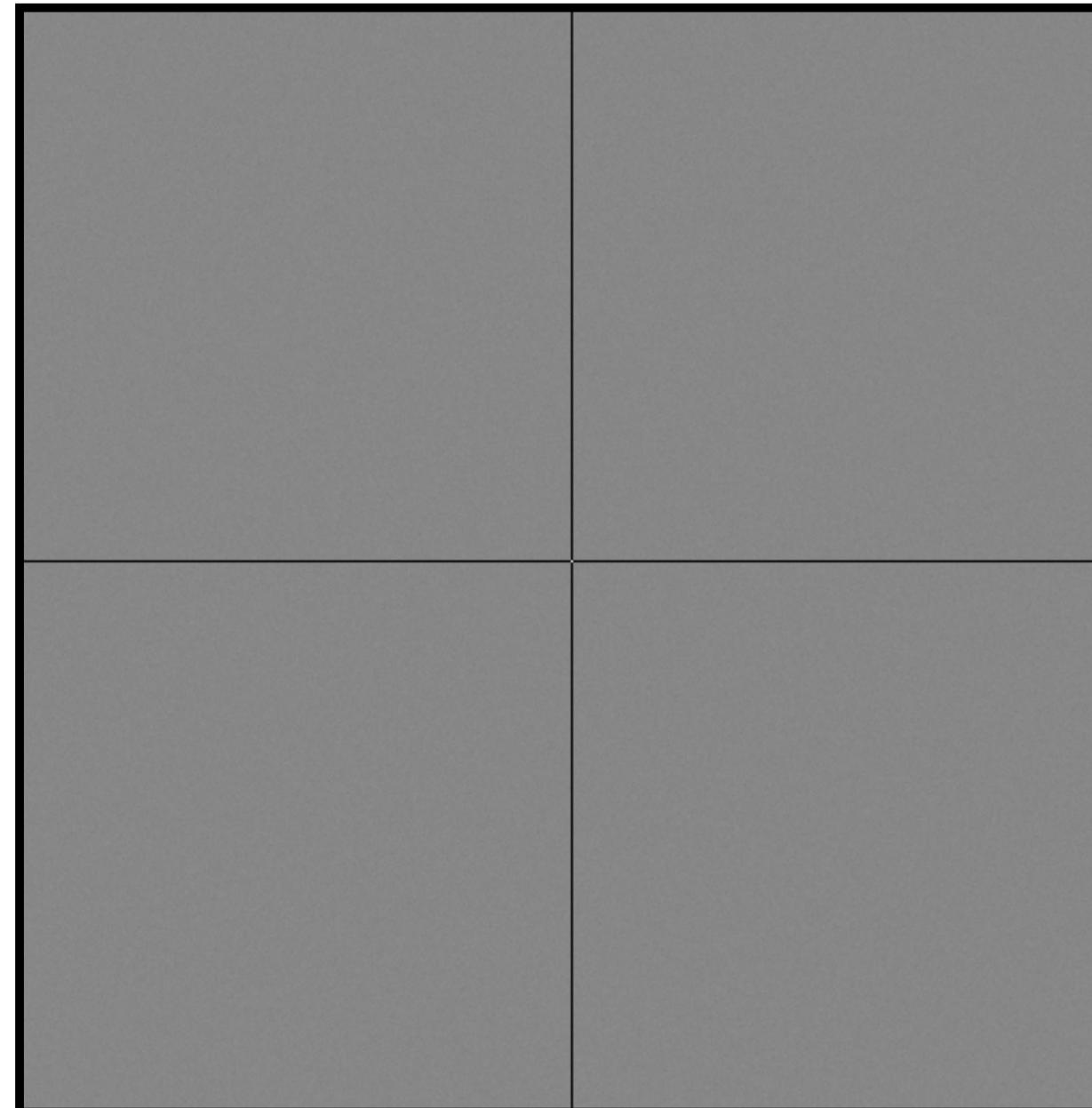
Evenly distributed in each
individual dimension

N-Rooks Sampling

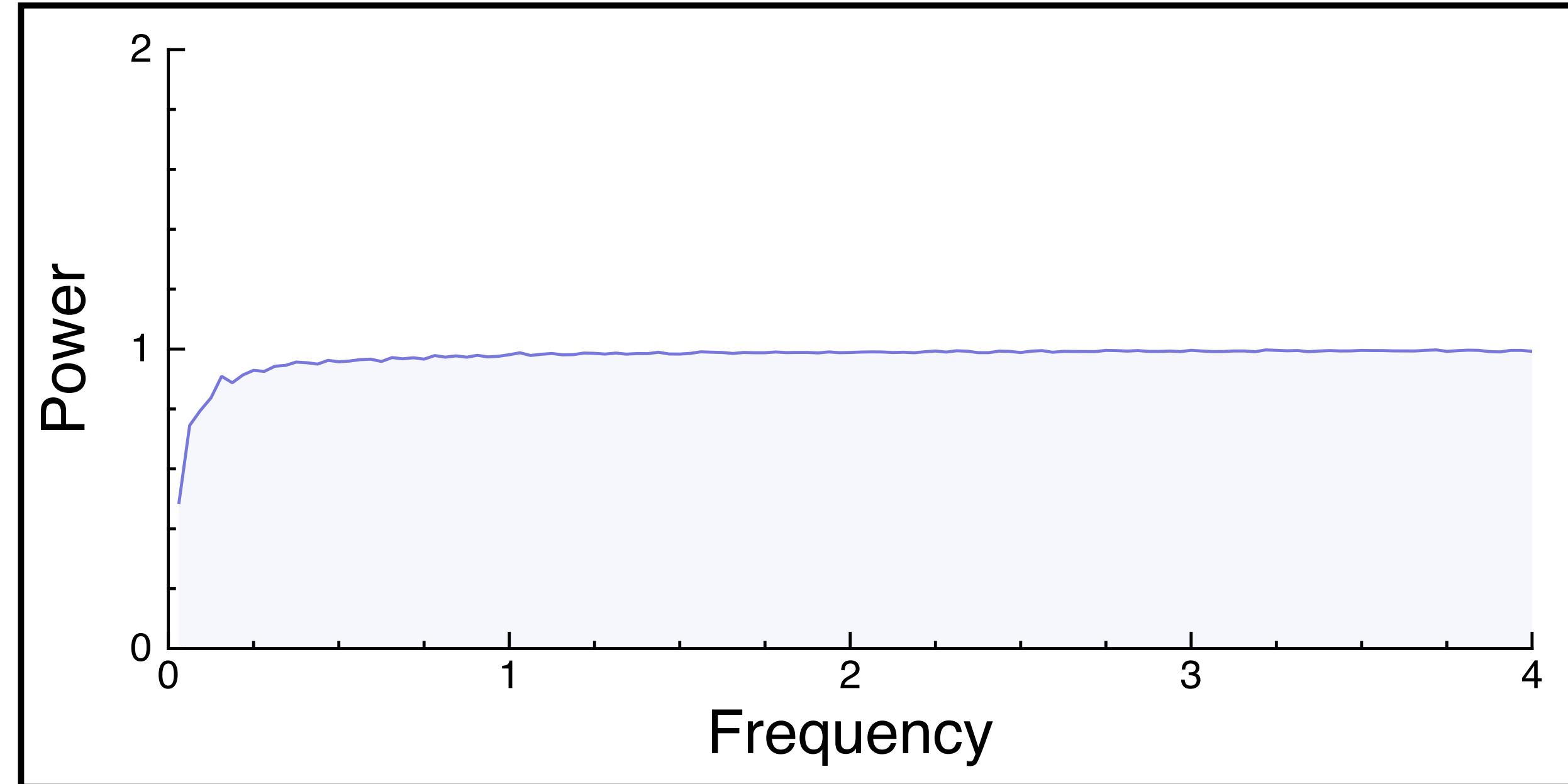
Samples



Power spectrum



Radial mean

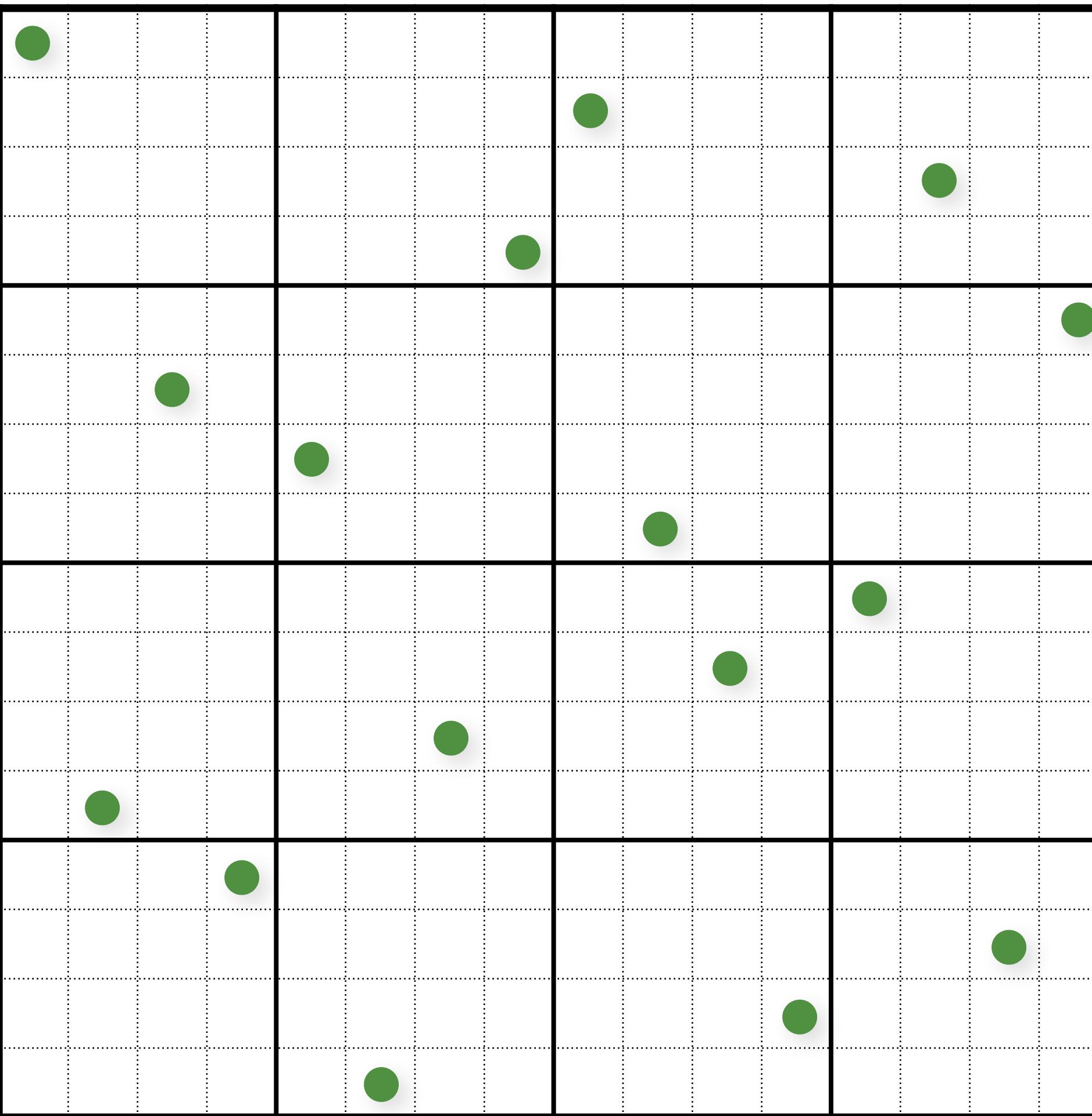


Multi-Jittered Sampling

Kenneth Chiu, Peter Shirley, and Changyaw Wang.
“Multi-jittered sampling.” In *Graphics Gems IV*, pp.
370–374. Academic Press, May 1994.

- combine N-Rooks and Jittered stratification constraints

Multi-Jittered Sampling



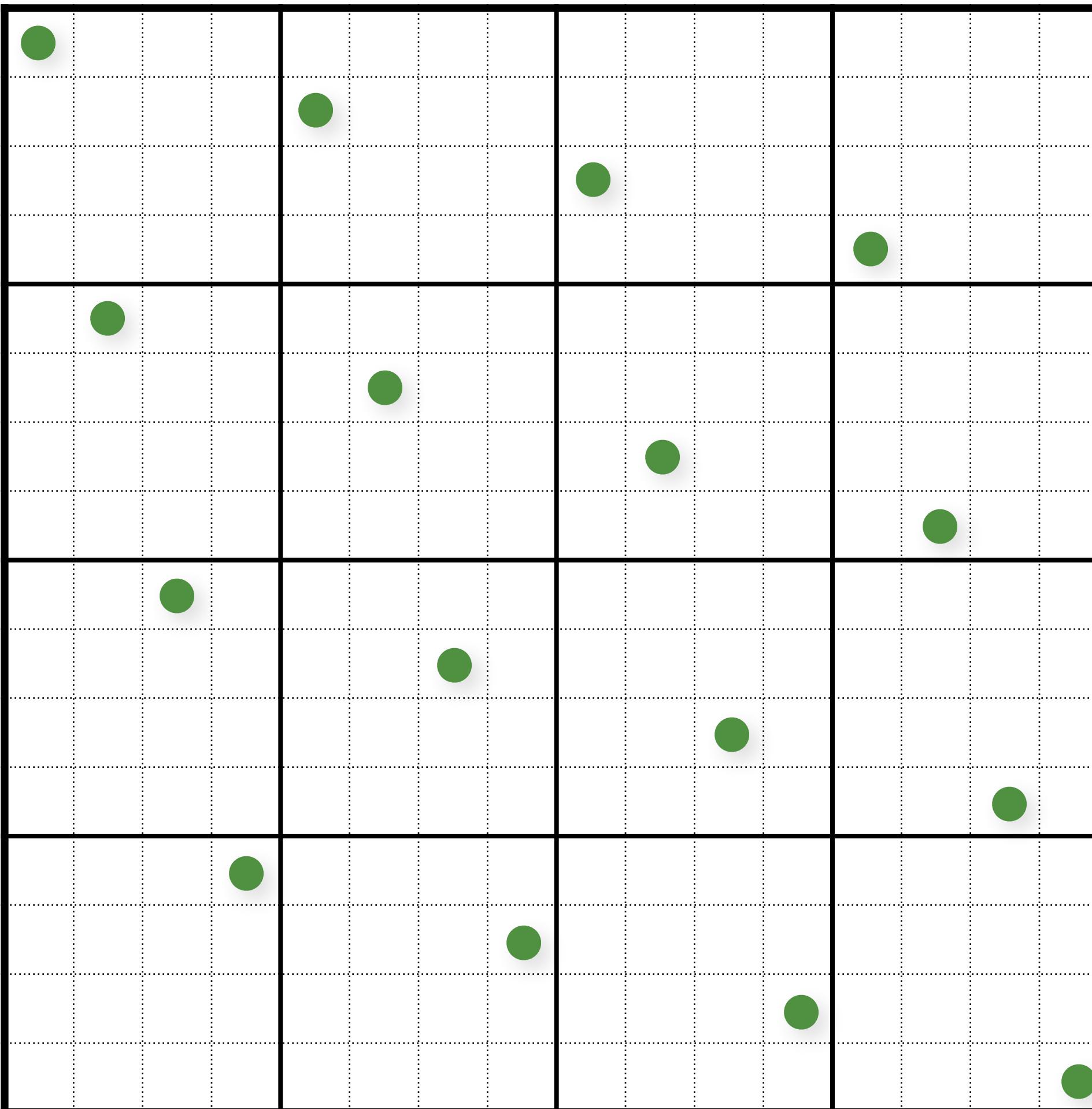
Multi-Jittered Sampling

```
// initialize
float cellsize = 1.0 / (resX*resY);
for (uint i = 0; i < resX; i++)
    for (uint j = 0; j < resY; j++)
{
    samples(i,j).x = i/resX + (j+randf()) / (resX*resY);
    samples(i,j).y = j/resY + (i+randf()) / (resX*resY);
}

// shuffle x coordinates within each column of cells
for (uint i = 0; i < resX; i++)
    for (uint j = resY-1; j >= 1; j--)
        swap(samples(i, j).x, samples(i, randi(0, j)).x);

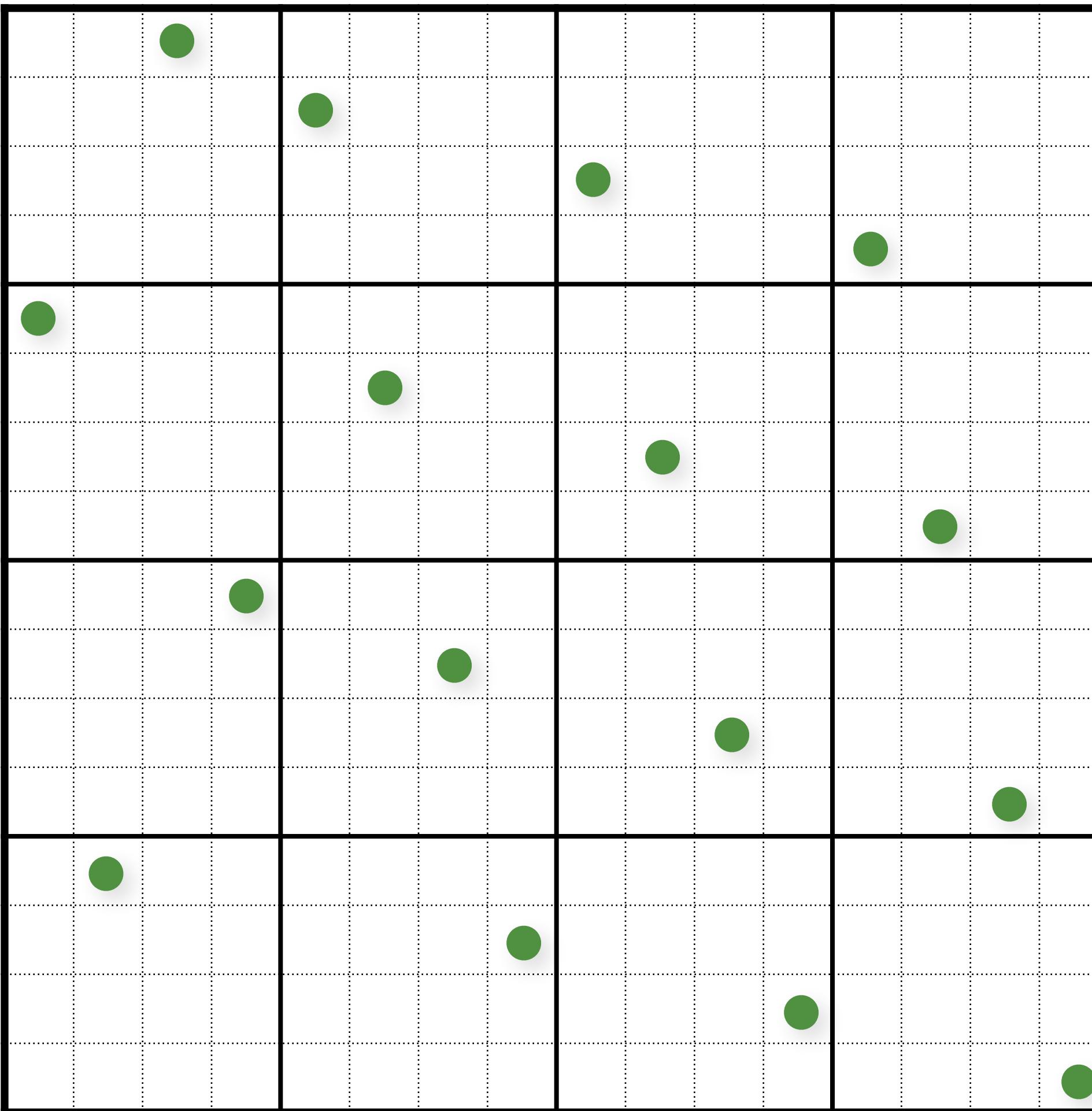
// shuffle y coordinates within each row of cells
for (unsigned j = 0; j < resY; j++)
    for (unsigned i = resX-1; i >= 1; i--)
        swap(samples(i, j).y, samples(randi(0, i), j).y);
```

Multi-Jittered Sampling



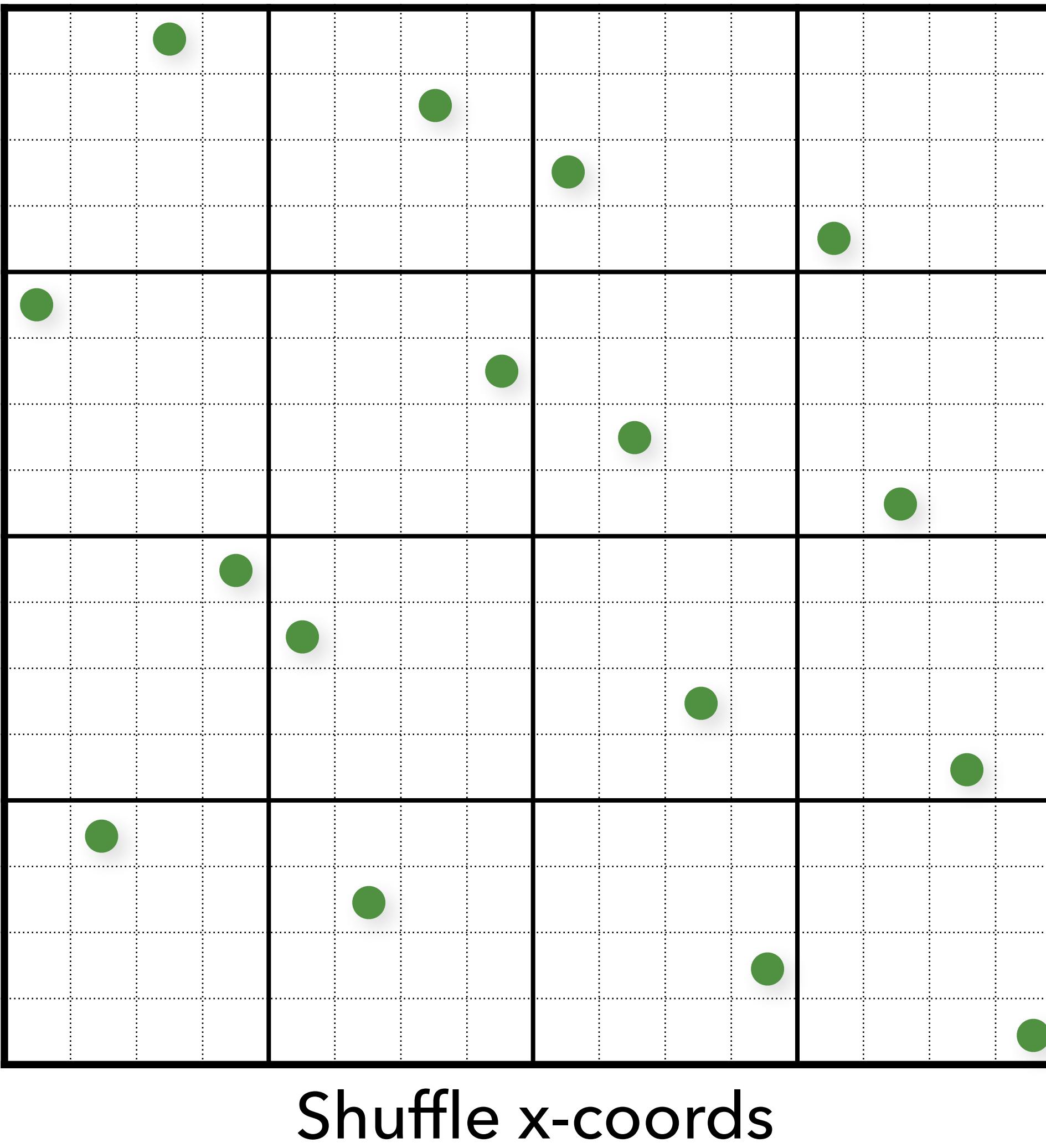
Shuffled coordinates

Multi-Jittered Sampling

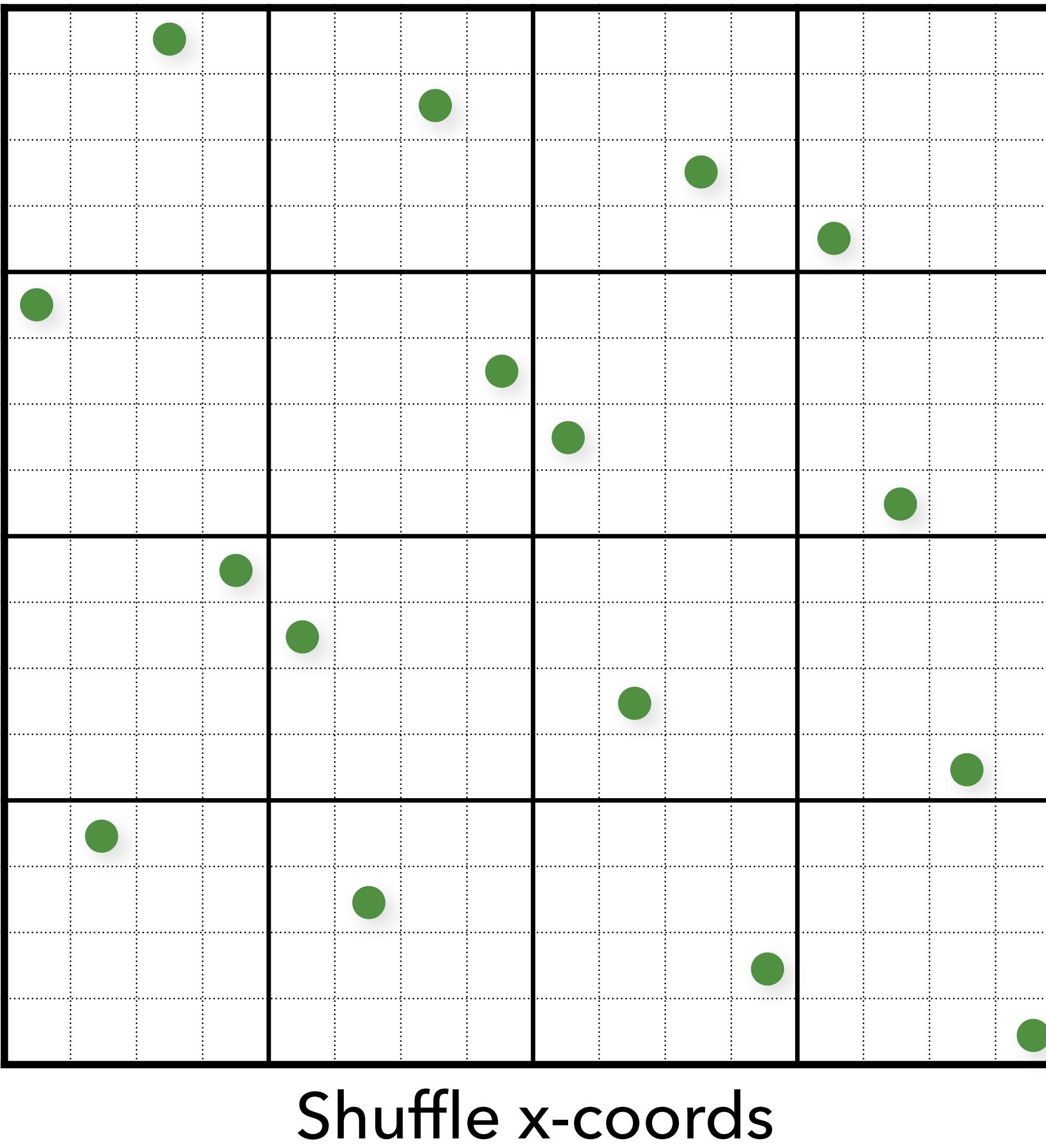


Shuffle x-coords

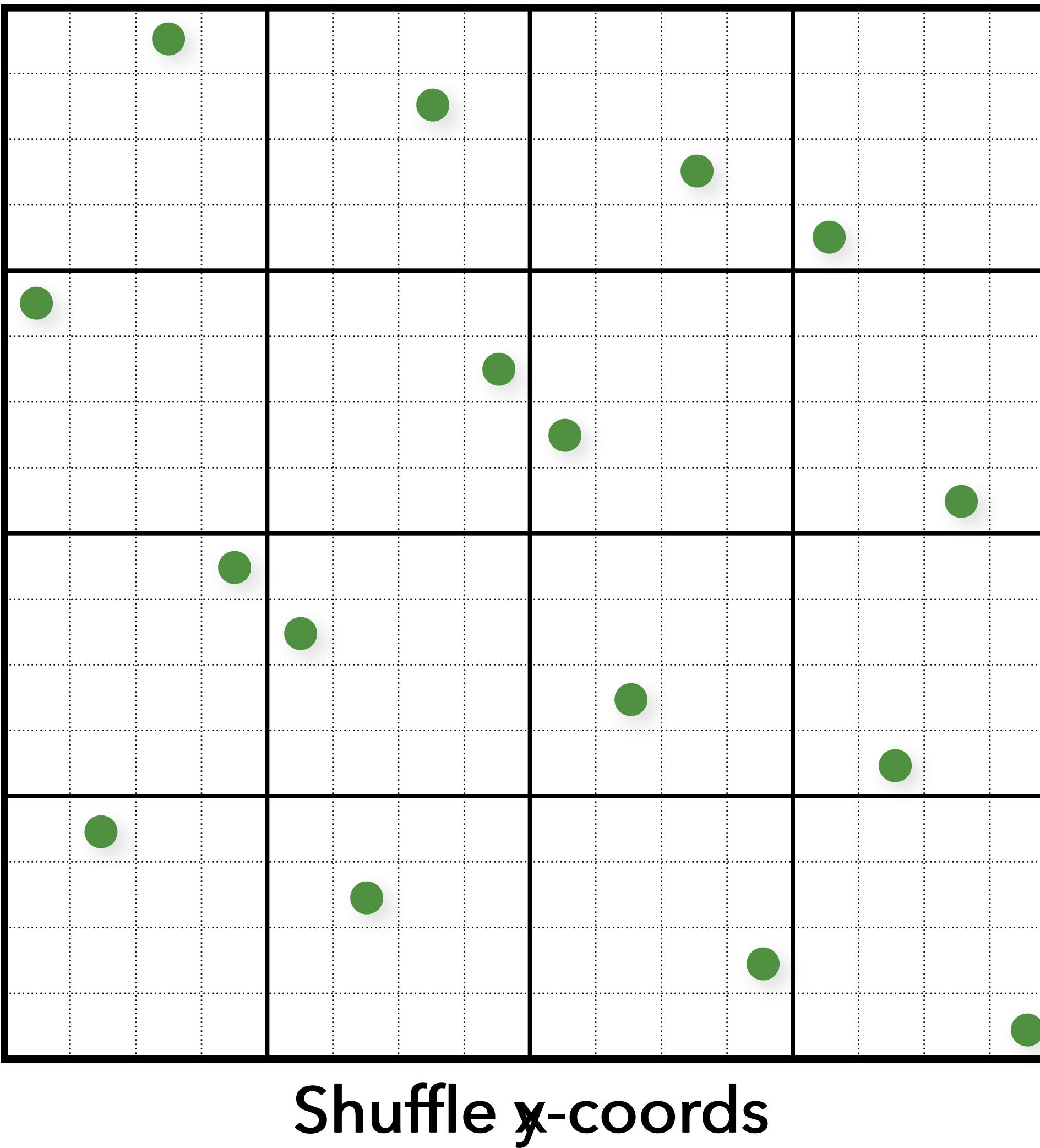
Multi-Jittered Sampling



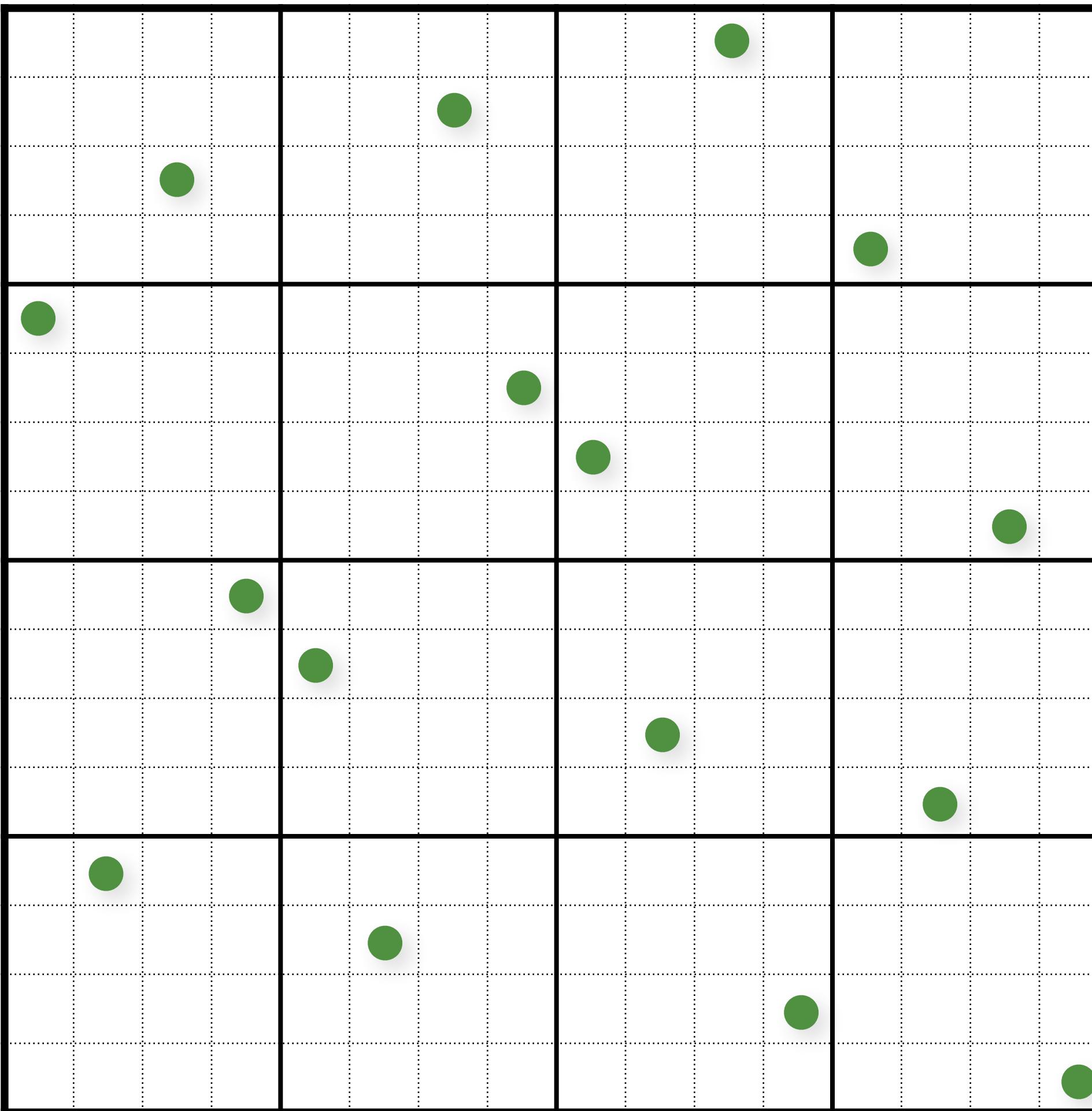
Multi-Jittered Sampling



Multi-Jittered Sampling

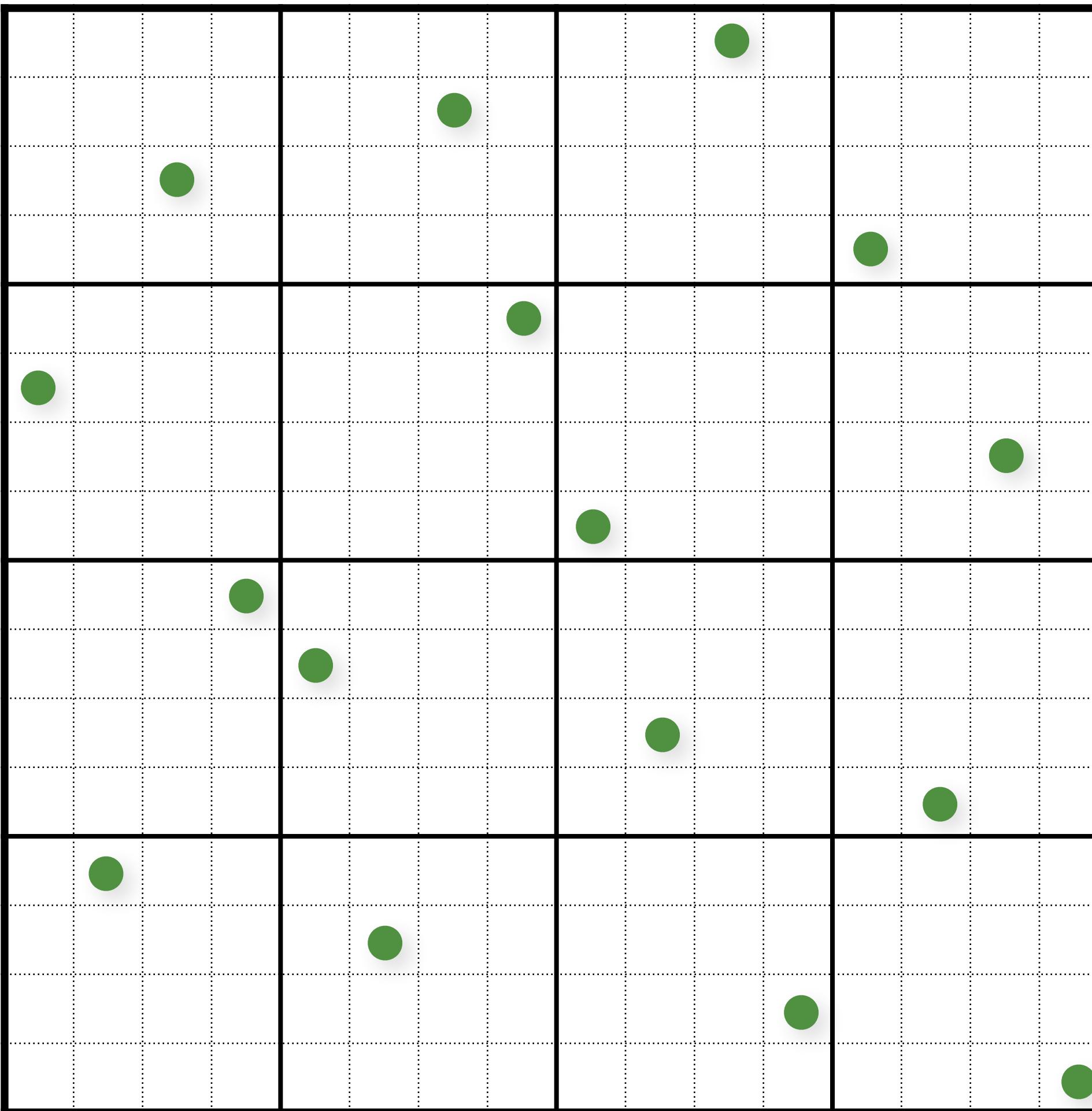


Multi-Jittered Sampling



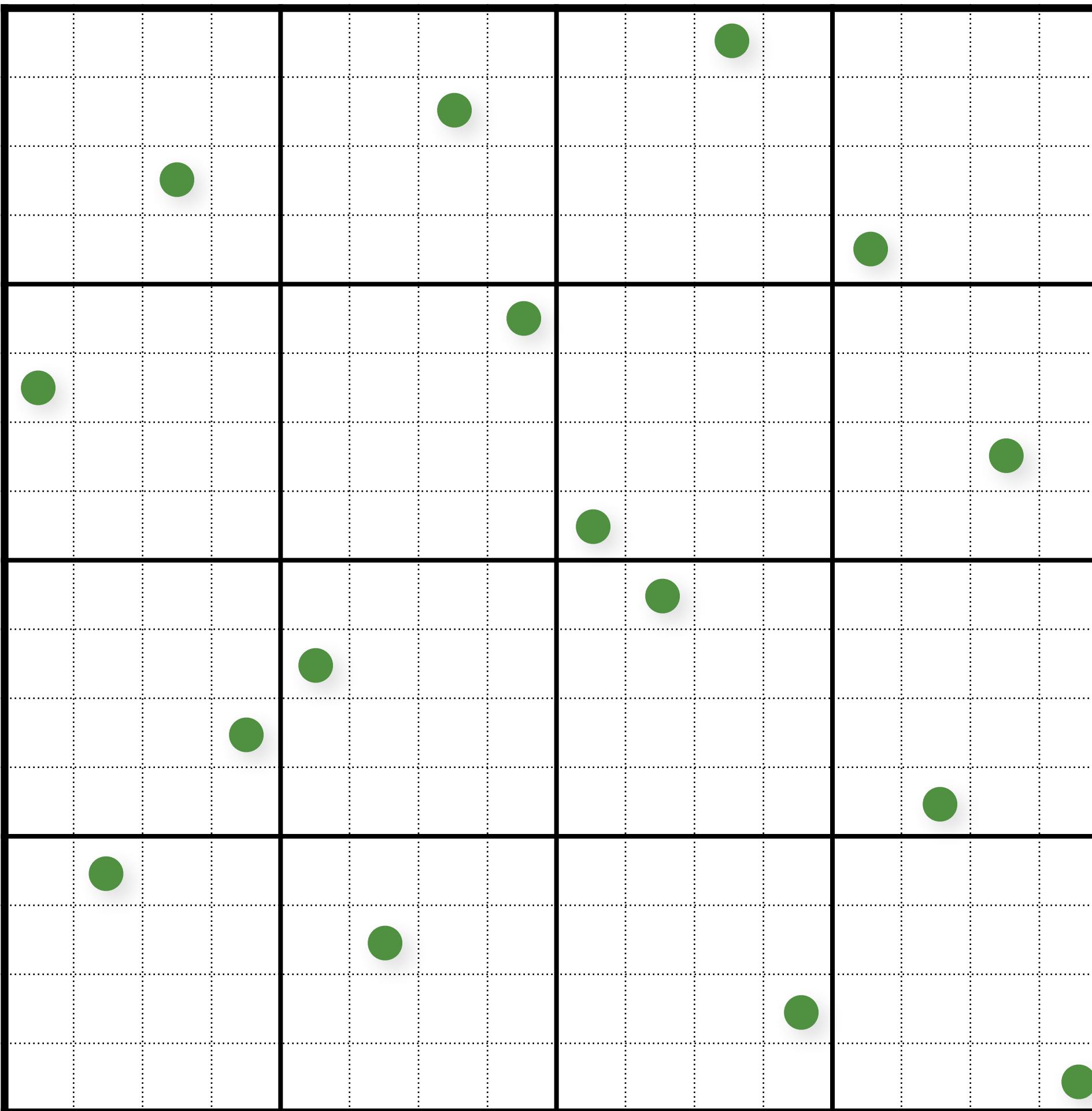
Shuffle y-coords

Multi-Jittered Sampling



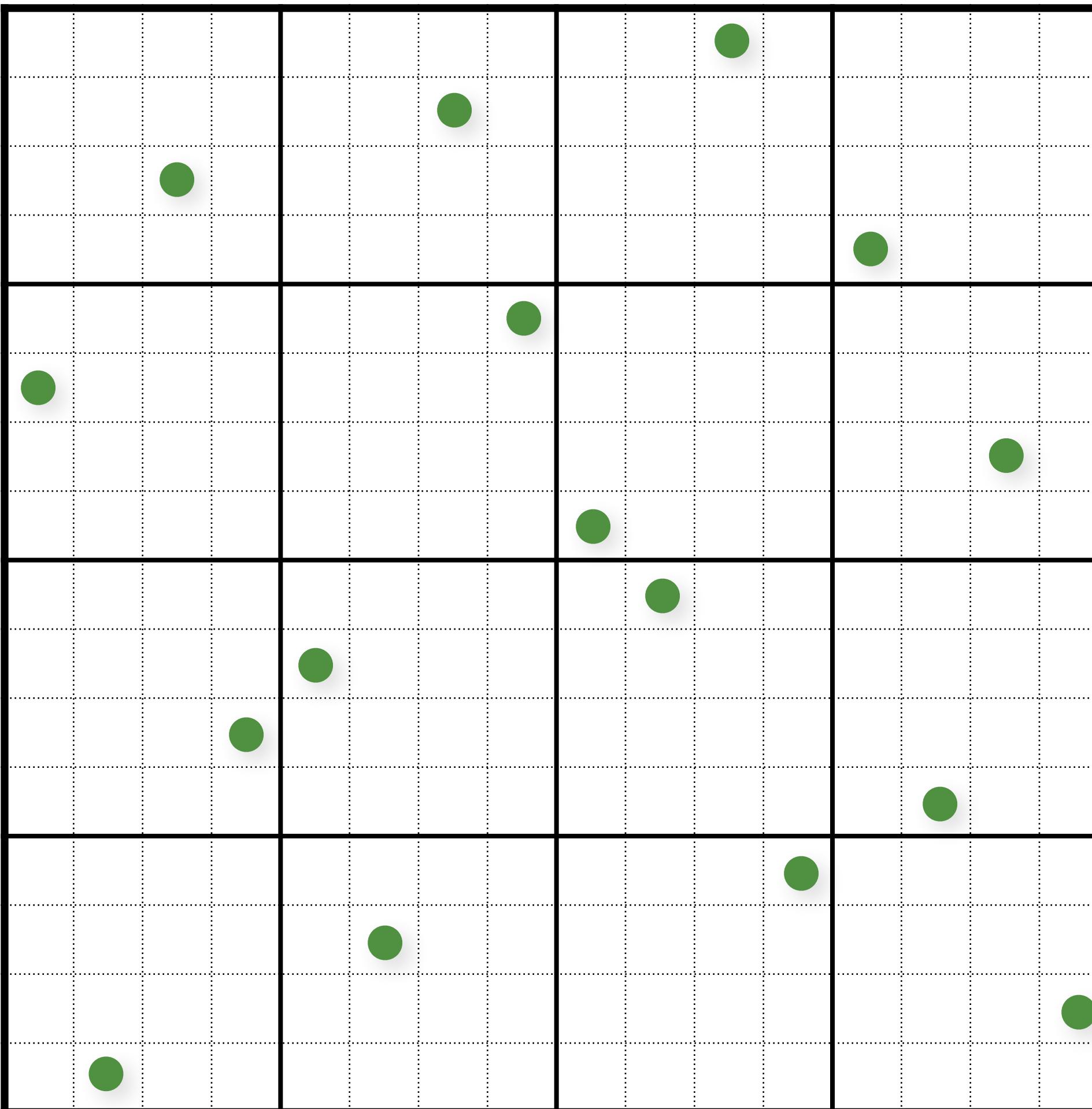
Shuffle y-coords

Multi-Jittered Sampling



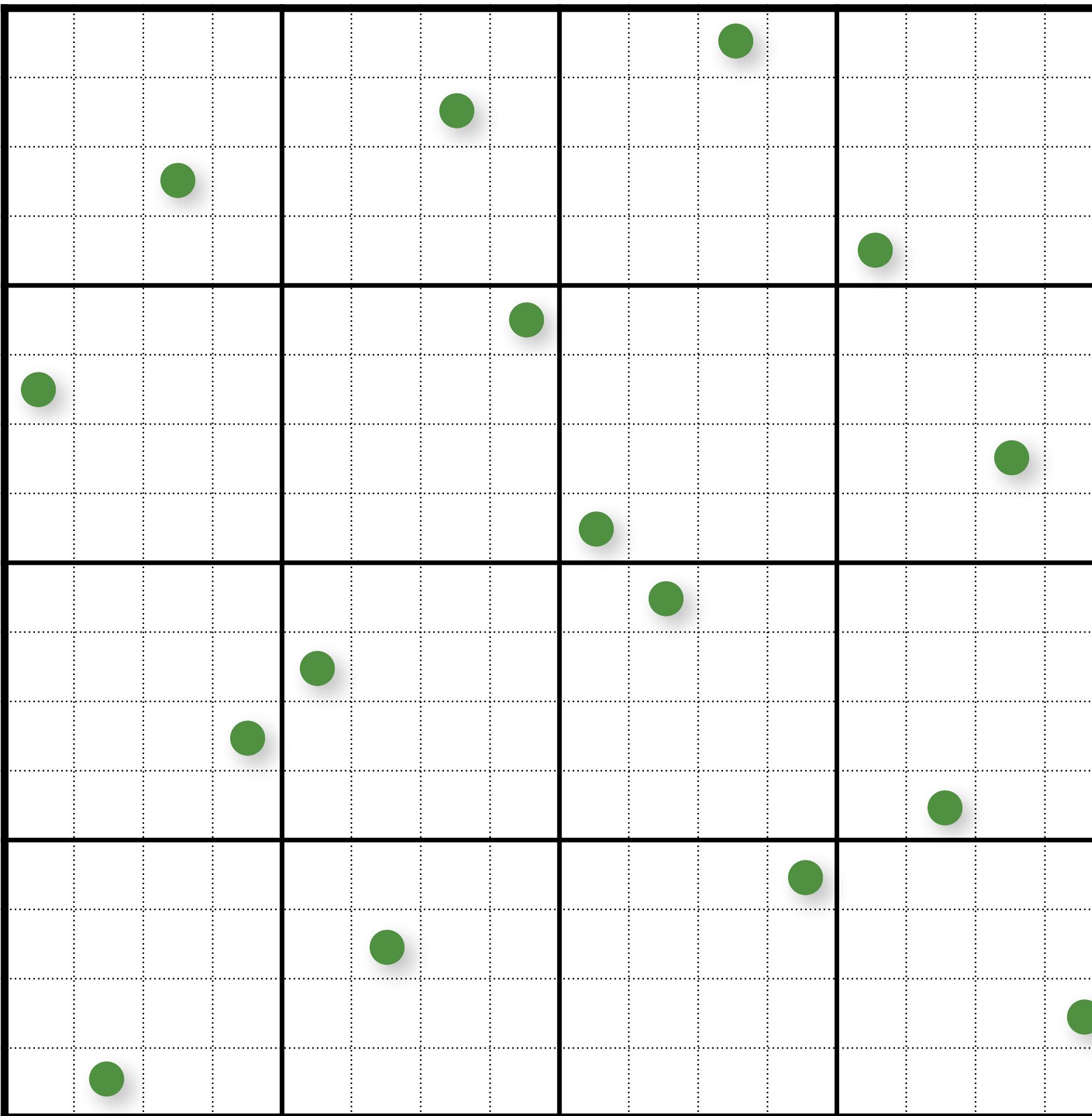
Shuffle y-coords

Multi-Jittered Sampling

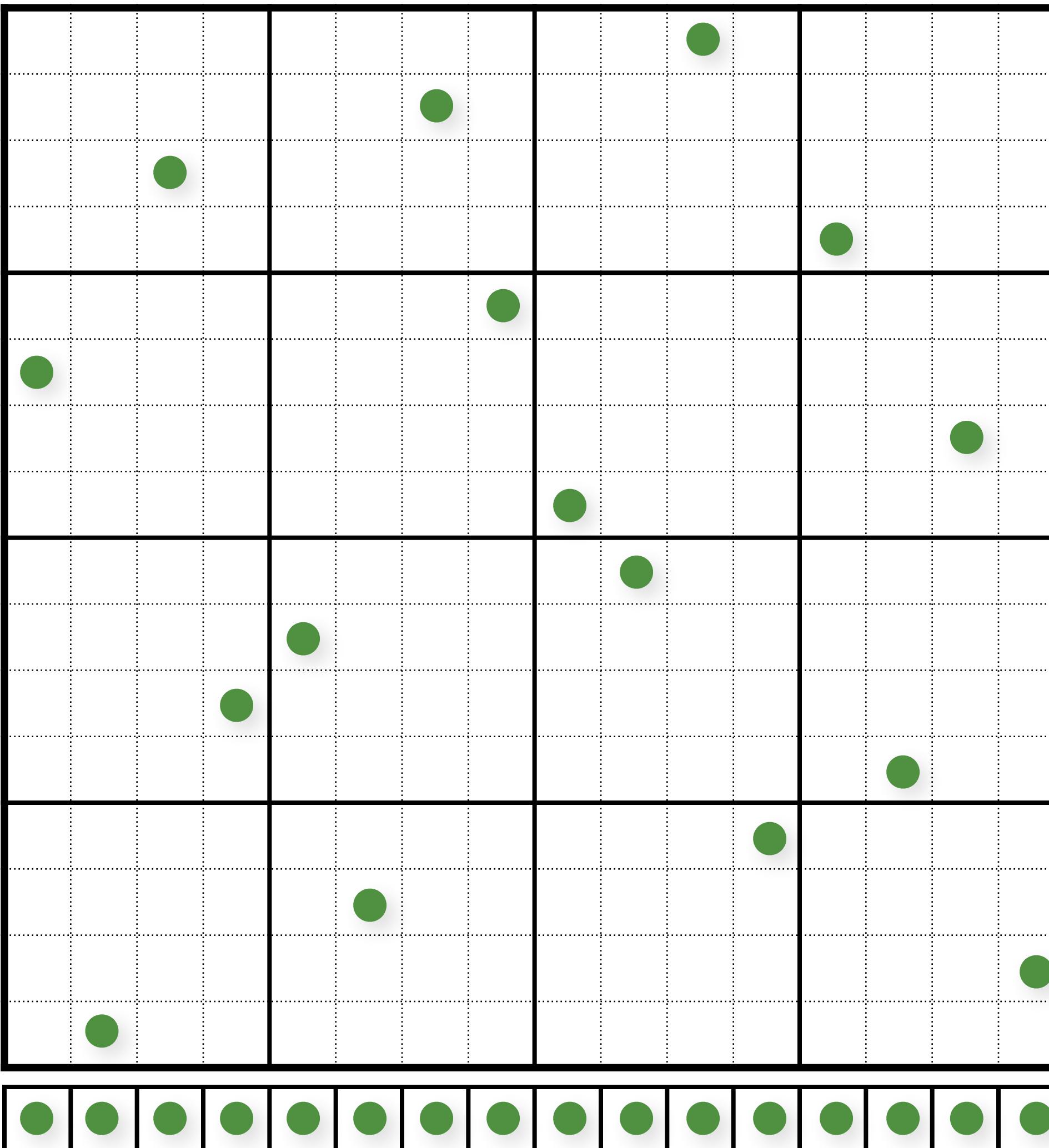


Shuffle y-coords

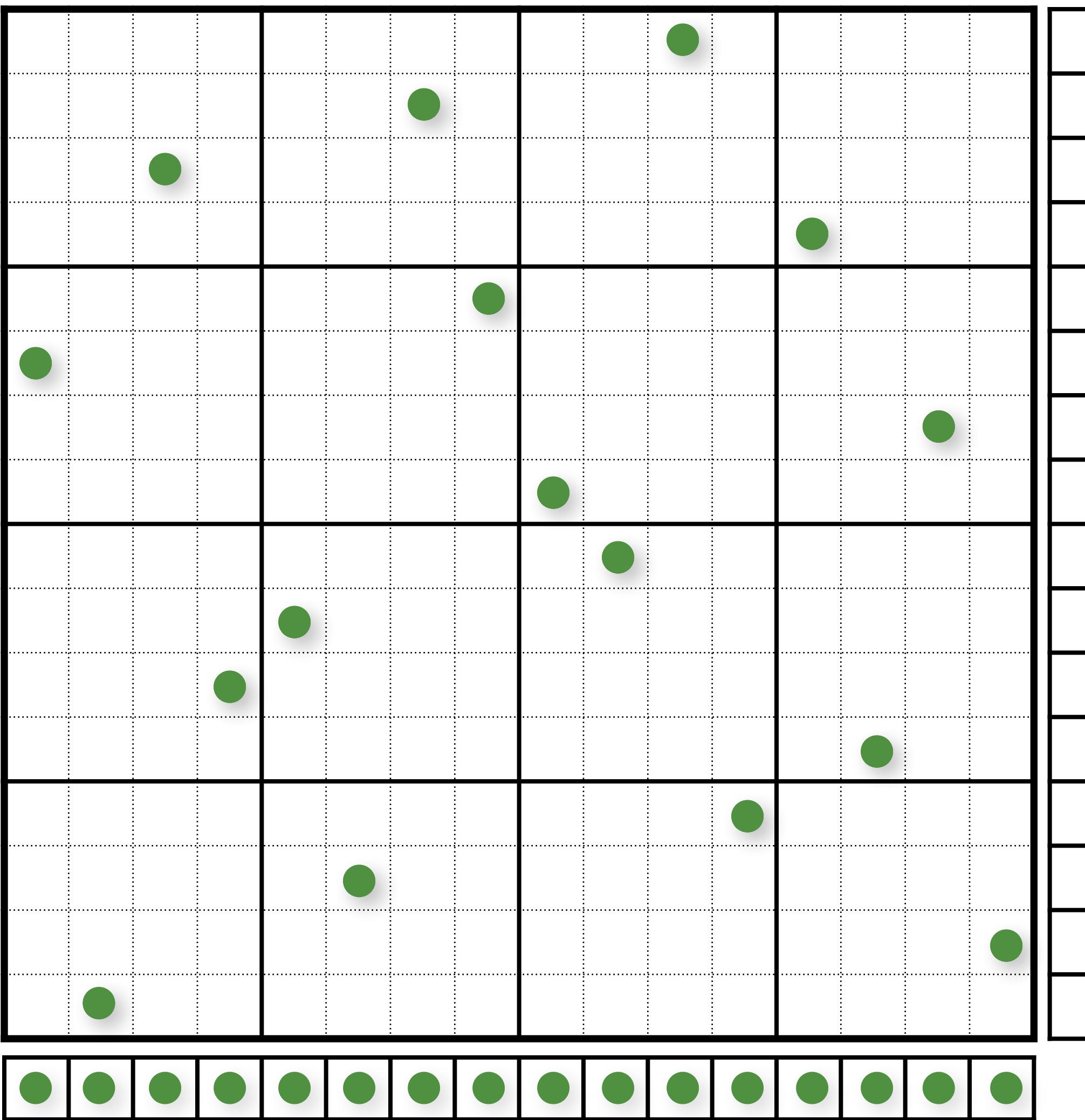
Multi-Jittered Sampling (Projections)



Multi-Jittered Sampling (Projections)

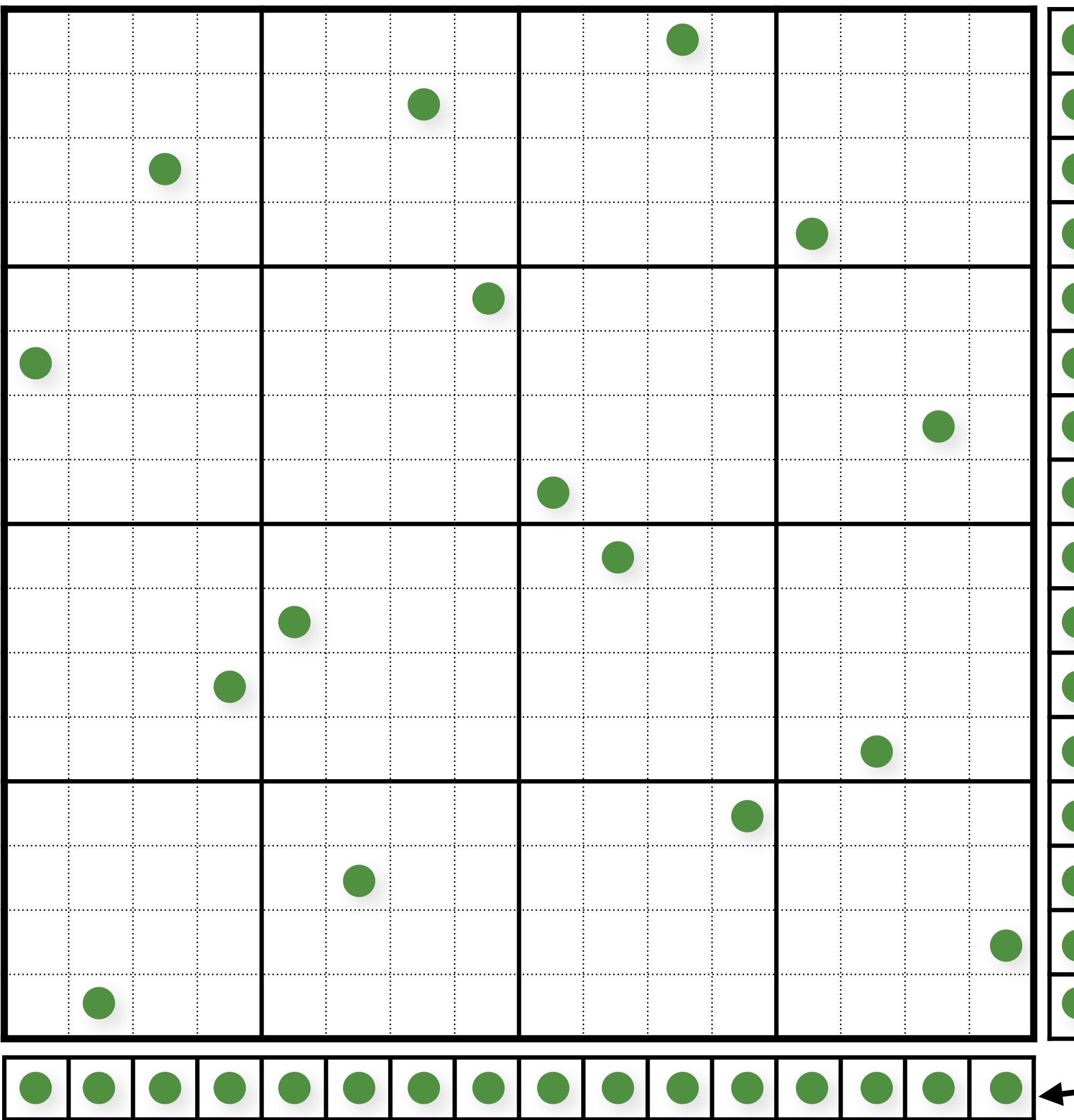


Multi-Jittered Sampling (Projections)



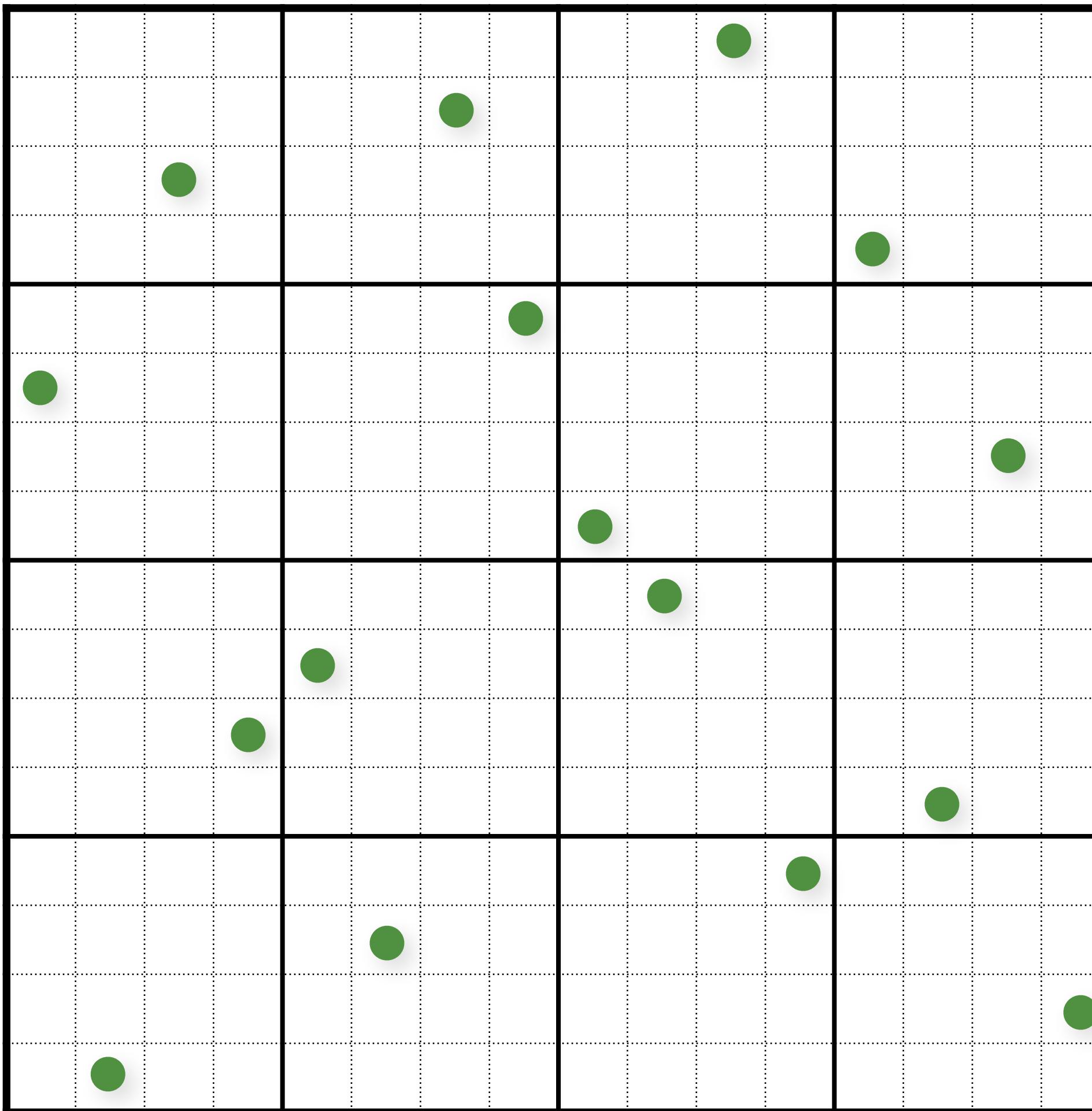
Multi-Jittered Sampling (Projections)

Evenly distributed
in n-dimensions!



Evenly distributed in each
individual dimension

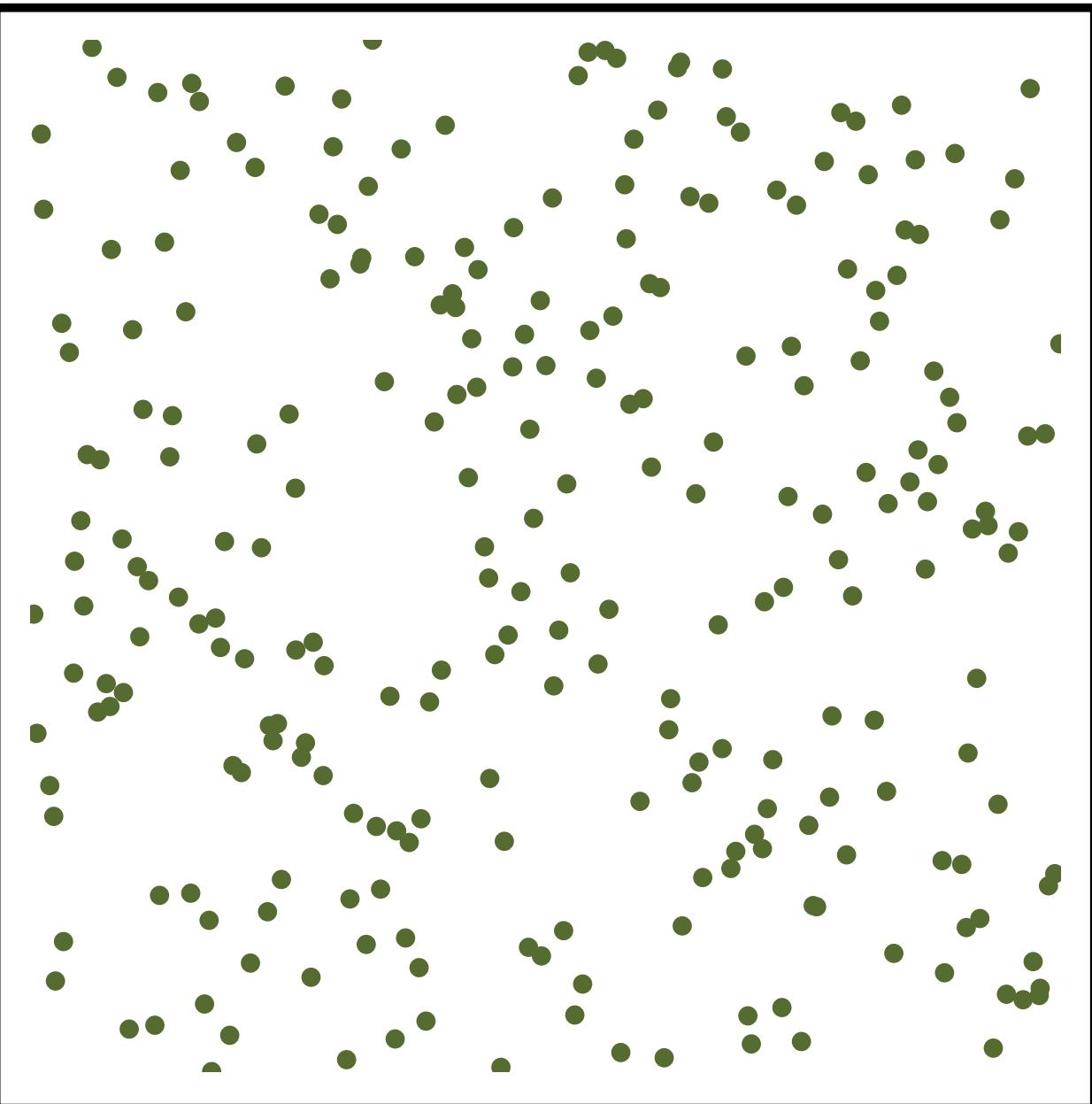
Multi-Jittered Sampling (Sudoku)



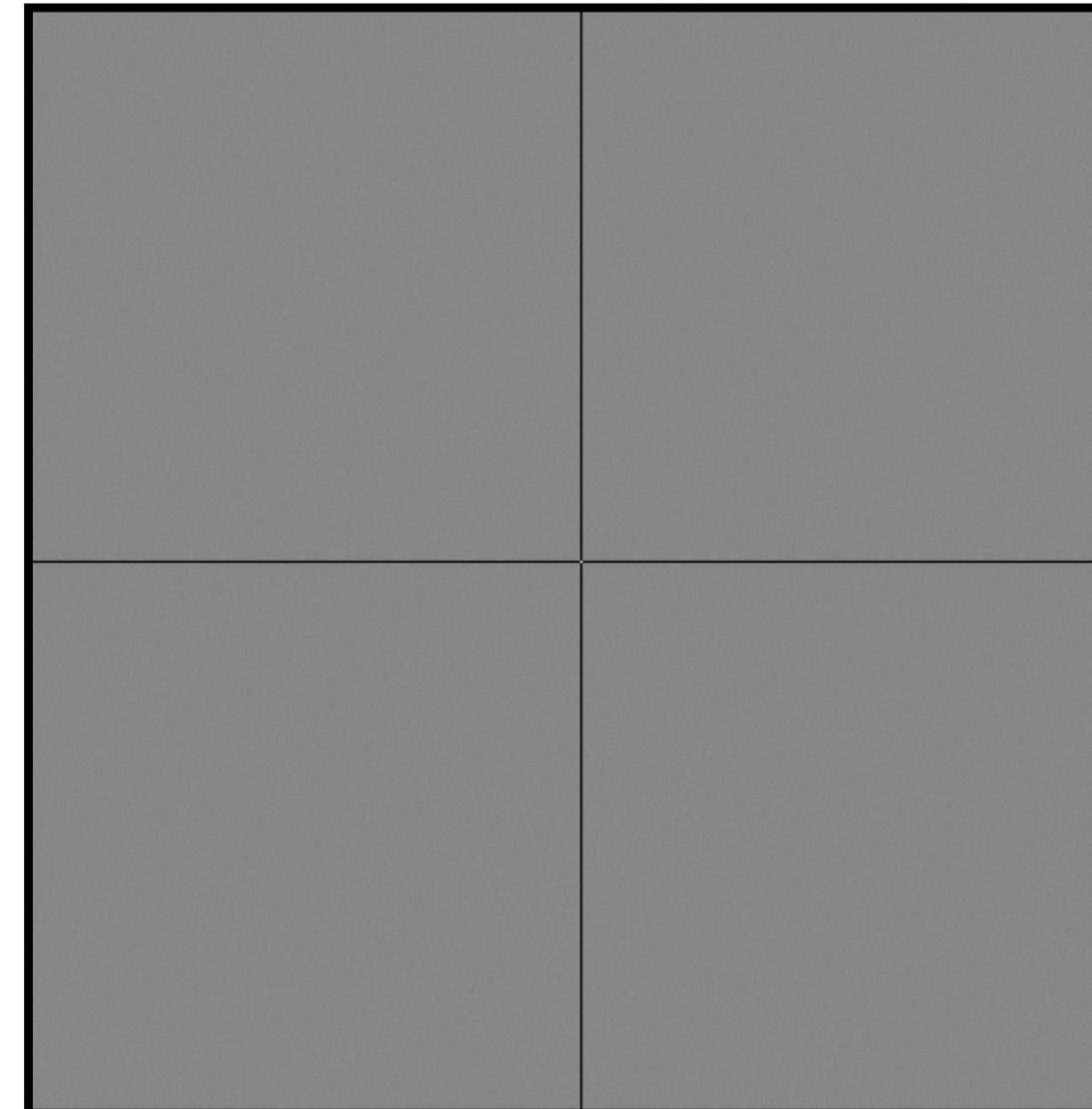
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
9	10	11	12	1	2	3	4	13	14	15	16	5	6	7	8
5	6	7	8	13	14	15	16	1	2	3	4	9	10	11	12
13	14	15	16	9	10	11	12	5	6	7	8	1	2	3	4
3	1	4	2	7	5	8	6	11	9	14	10	15	12	16	13
11	9	14	10	3	1	4	2	15	12	16	13	7	5	8	6
7	5	8	6	15	12	16	13	3	1	4	2	11	9	14	10
15	12	16	13	11	9	14	10	7	5	8	6	3	1	4	2
2	4	1	3	6	8	5	7	10	15	9	11	12	16	13	14
10	15	9	11	2	4	1	3	12	16	13	14	6	8	5	7
6	8	5	7	12	16	13	14	2	4	1	3	10	15	9	11
12	16	13	14	10	15	9	11	6	8	5	7	2	4	1	3
4	3	2	1	8	7	6	5	14	11	10	9	16	13	12	15
14	11	10	9	4	3	2	1	16	13	12	15	8	7	6	5
8	7	6	5	16	13	12	15	4	3	2	1	14	11	10	9
16	13	12	15	14	11	10	9	8	7	6	5	4	3	2	1

N-Rooks Sampling

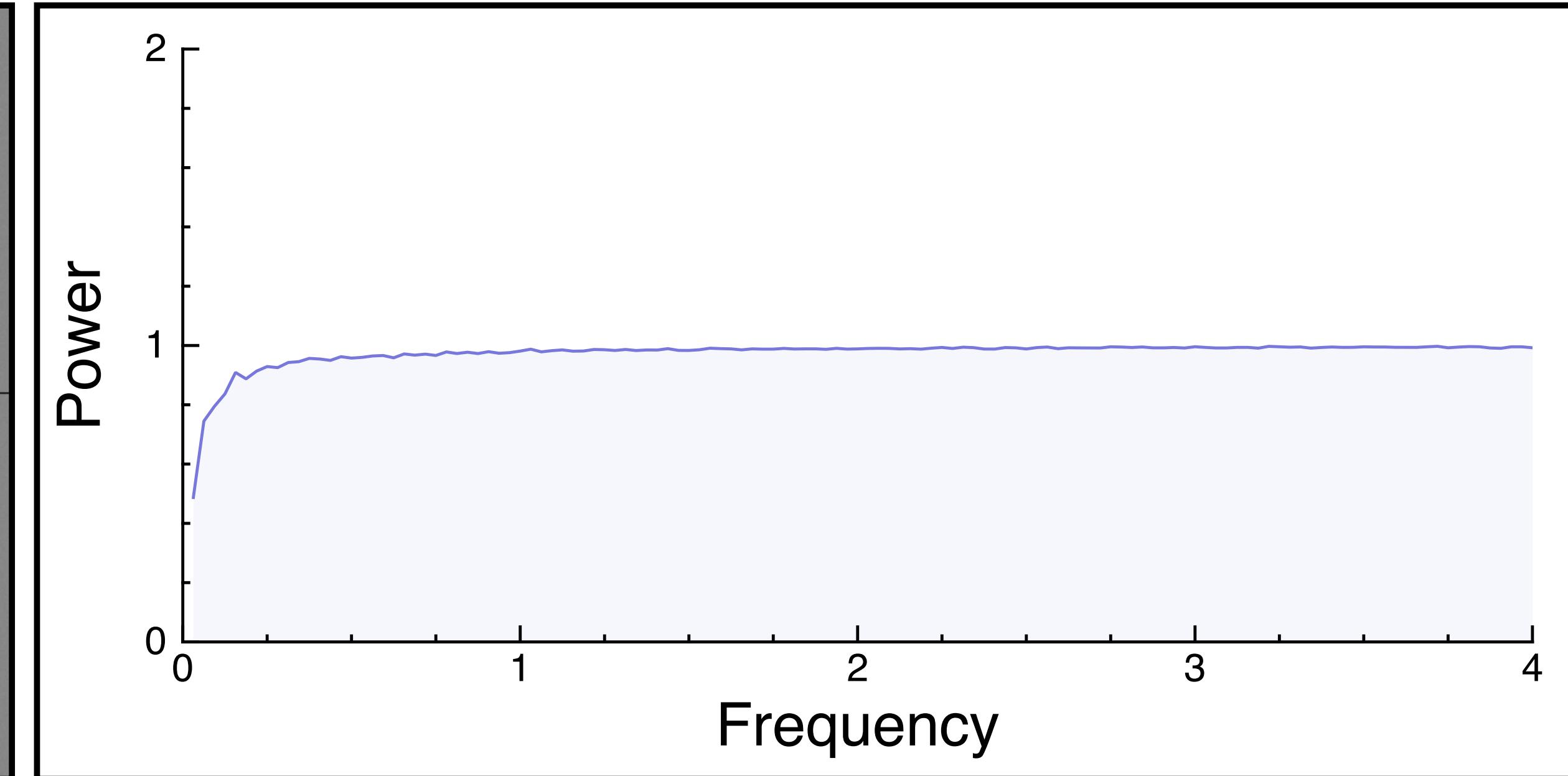
Samples



Power spectrum

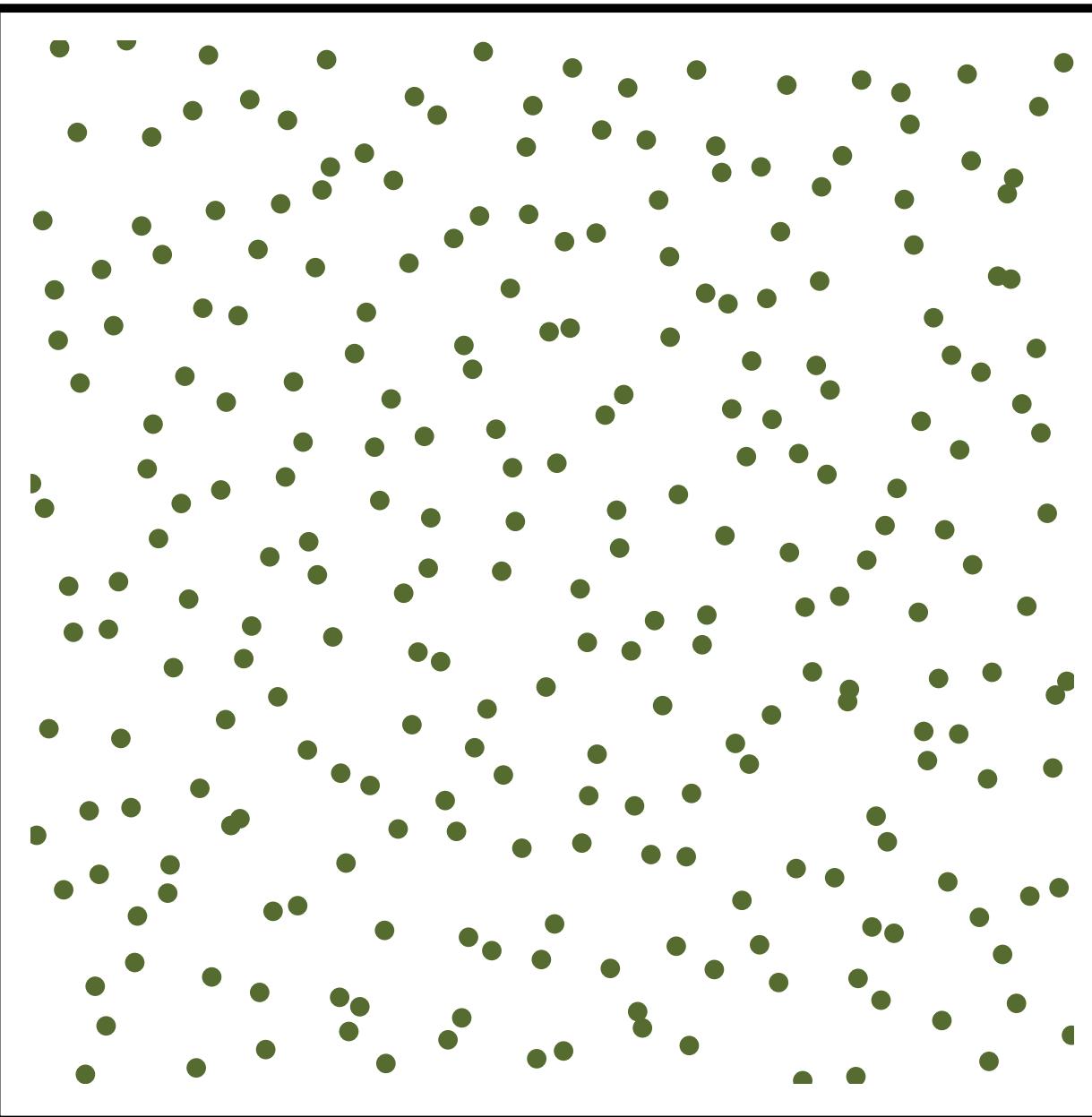


Radial mean

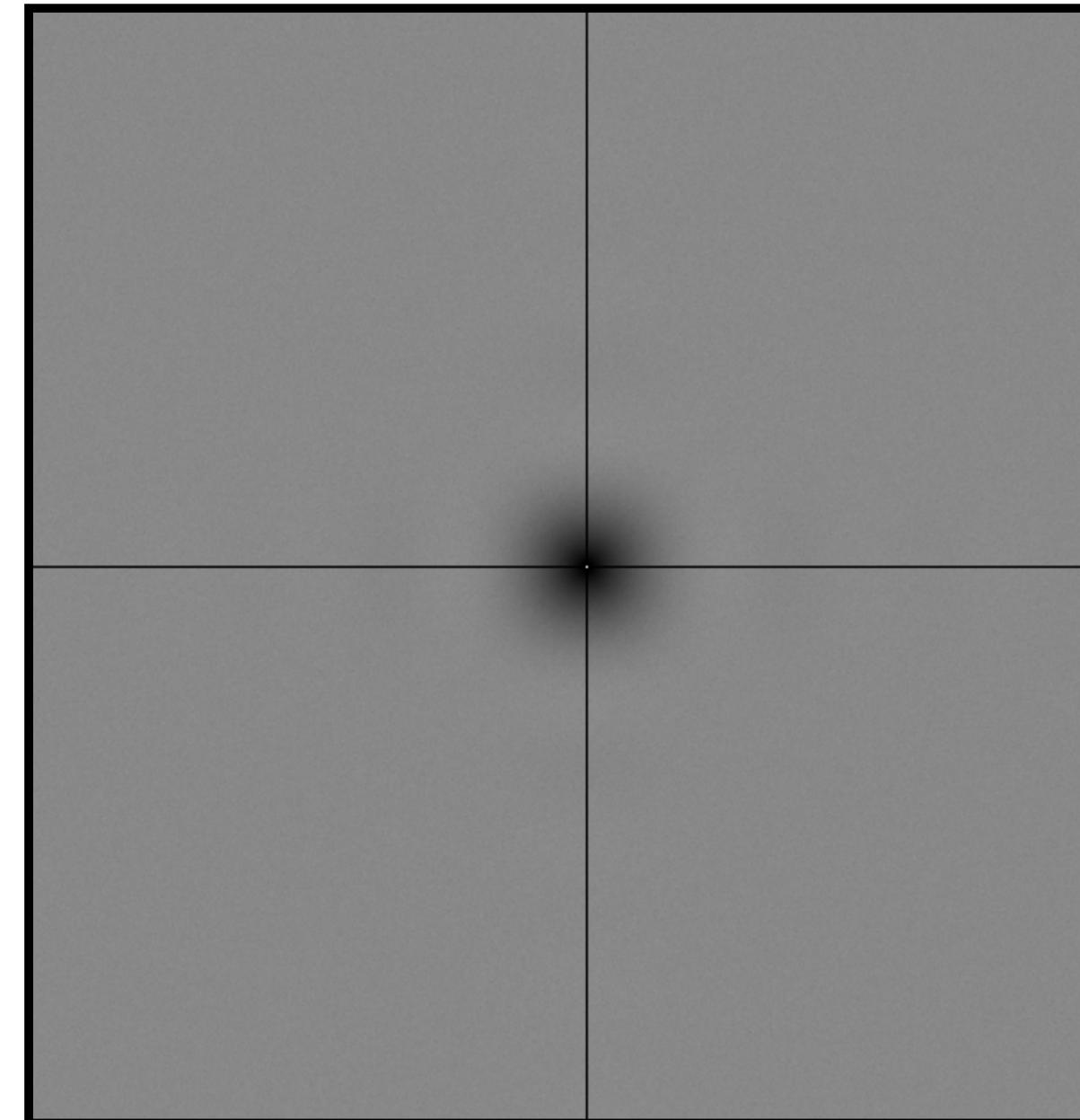


Multi-Jittered Sampling

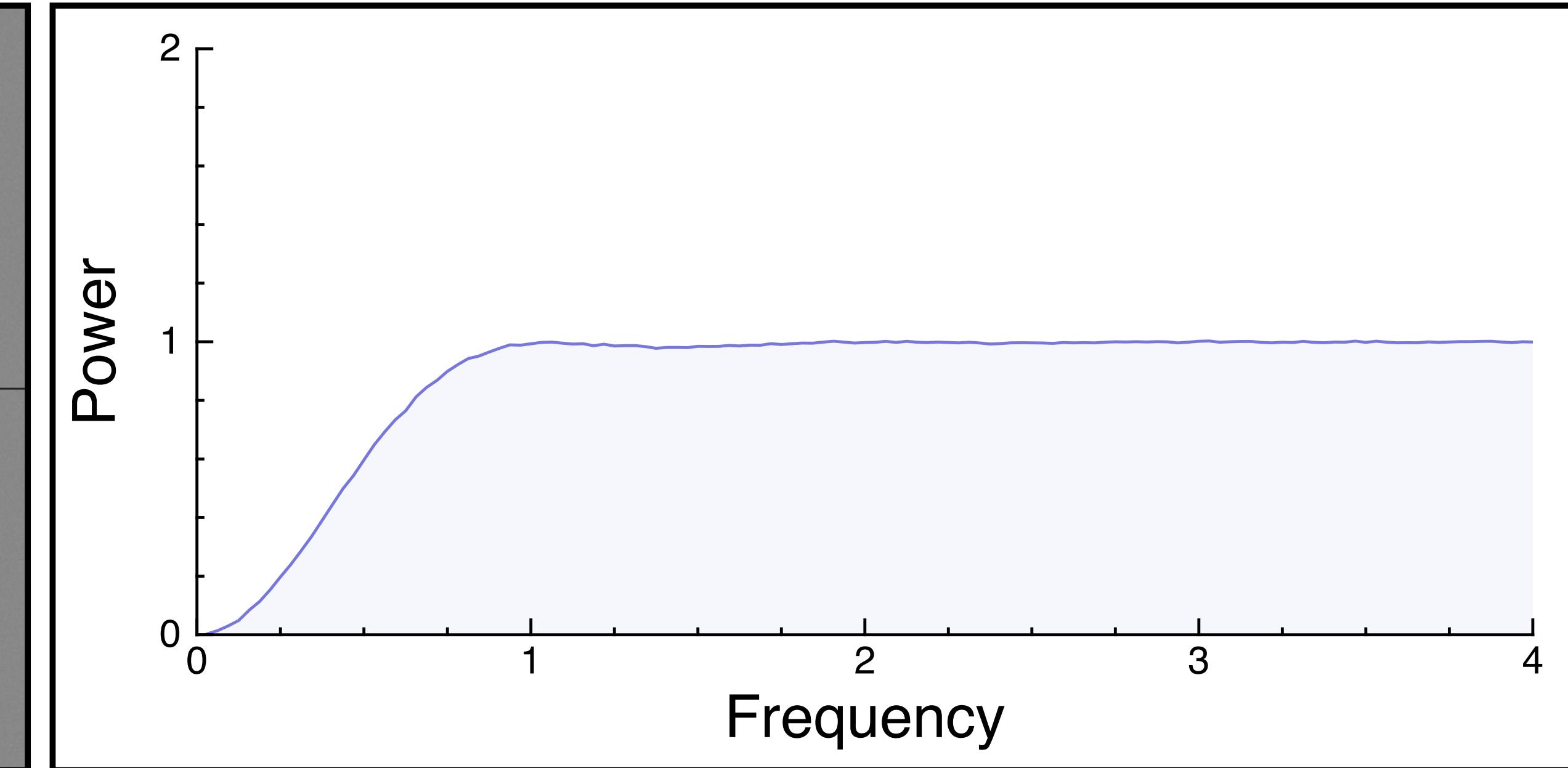
Samples



Power spectrum



Radial mean



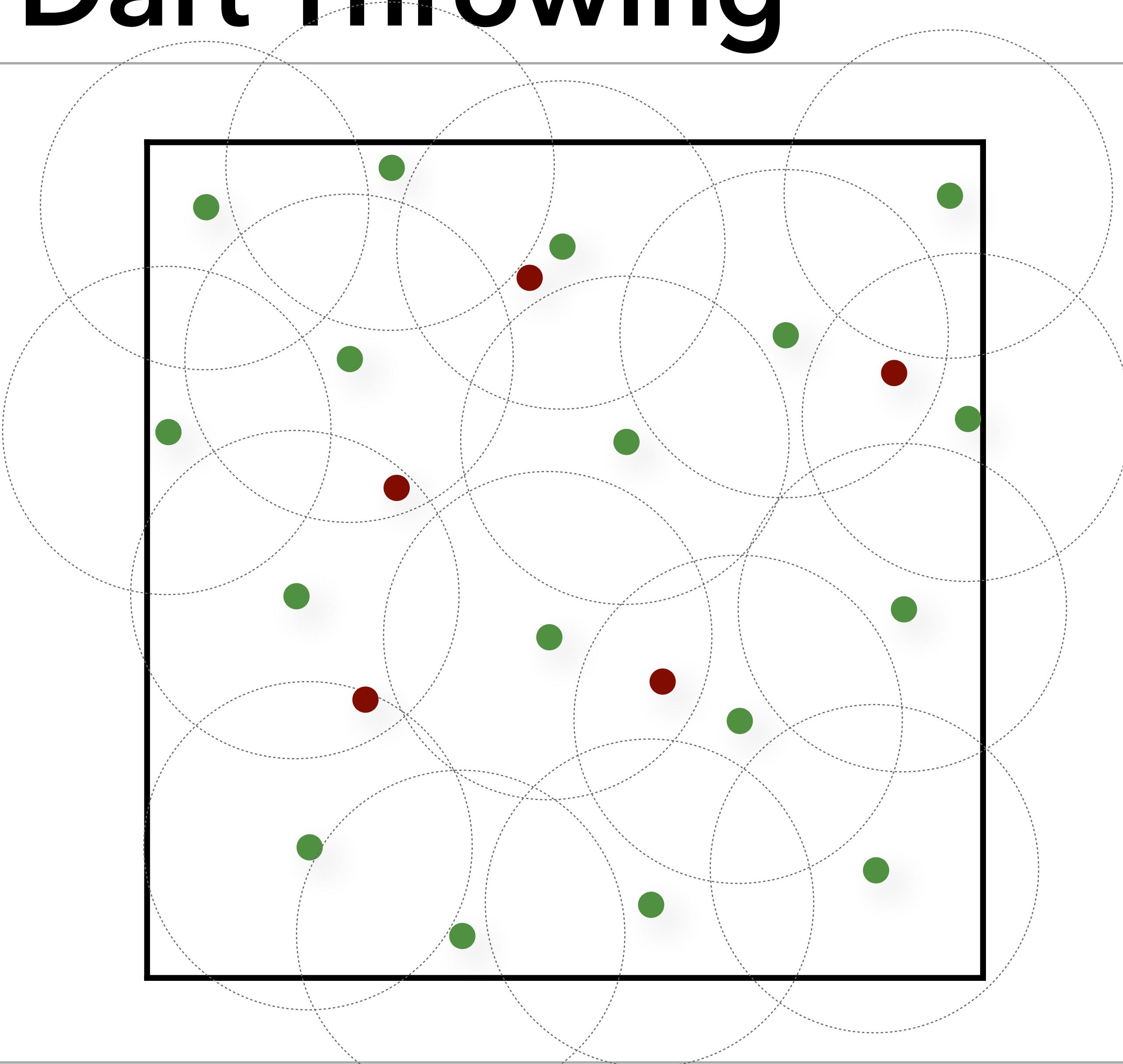
Poisson-Disk/Blue-Noise Sampling

Enforce a minimum distance between points

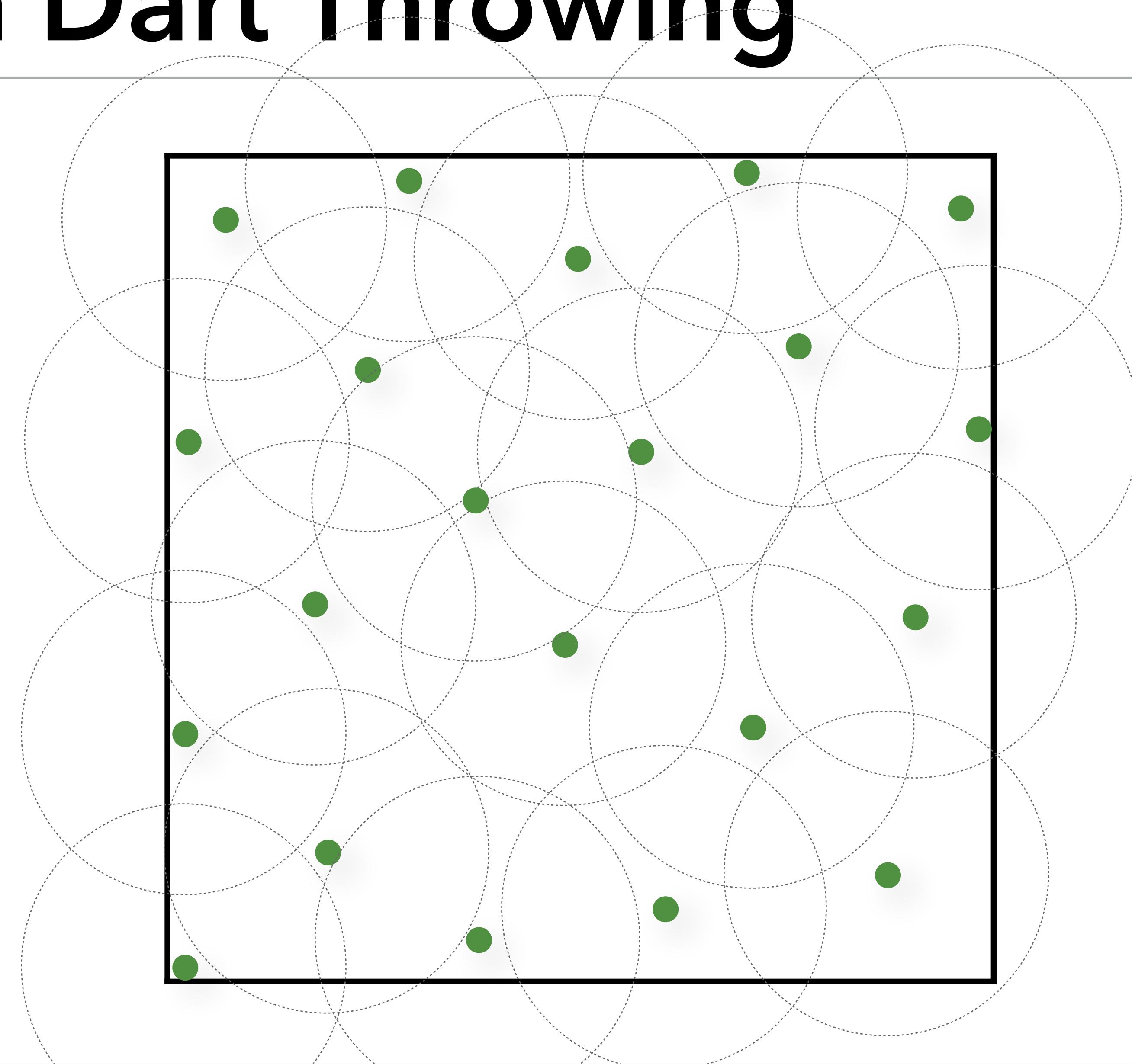
Poisson-Disk Sampling:

- Robert L. Cook. "Stochastic sampling in computer graphics." ACM Transactions on Graphics. 1986.
- Ares Lagae and Philip Dutré. "A comparison of methods for generating Poisson disk distributions." Computer Graphics Forum, 2008.

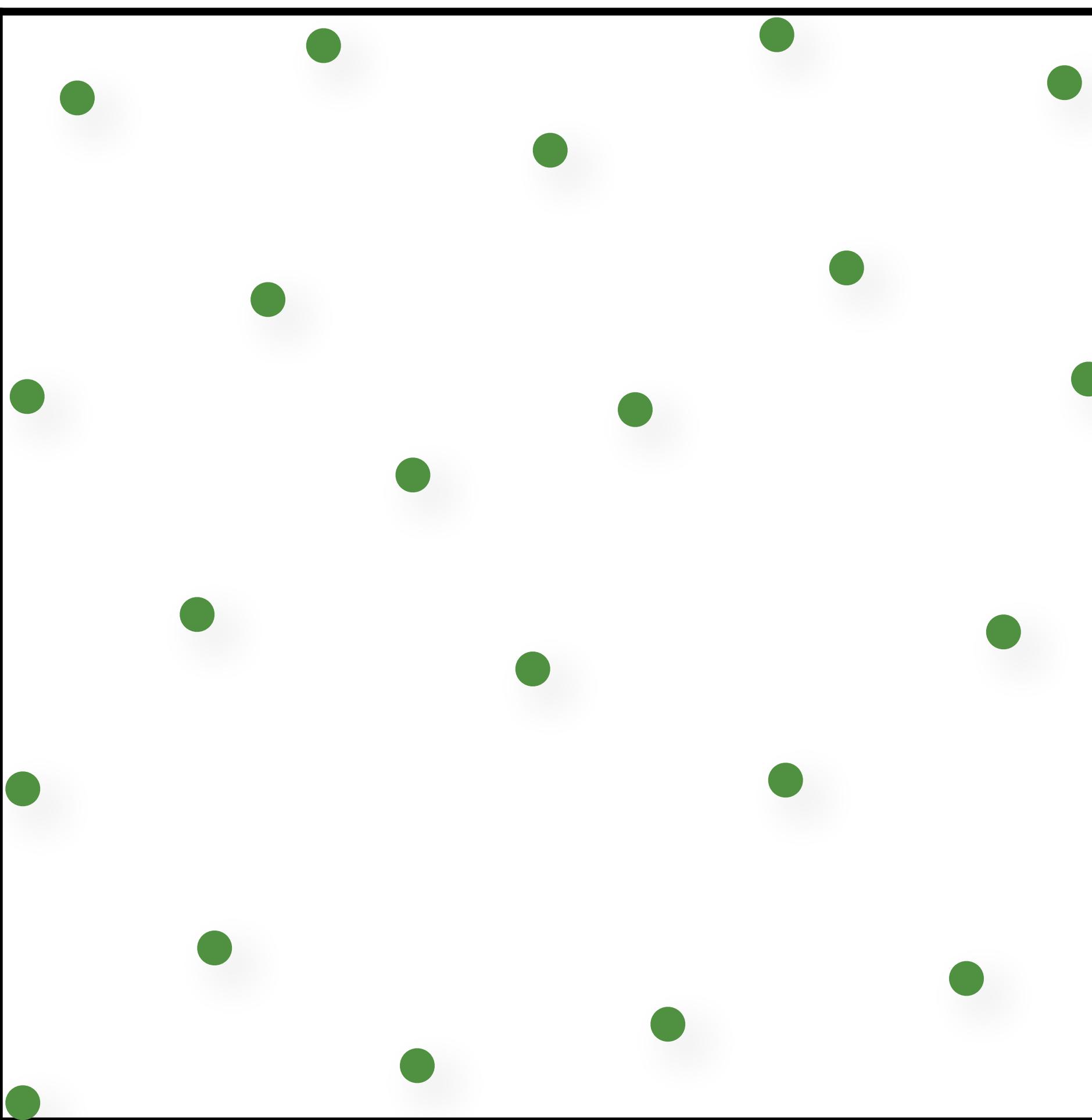
Random Dart Throwing



Random Dart Throwing

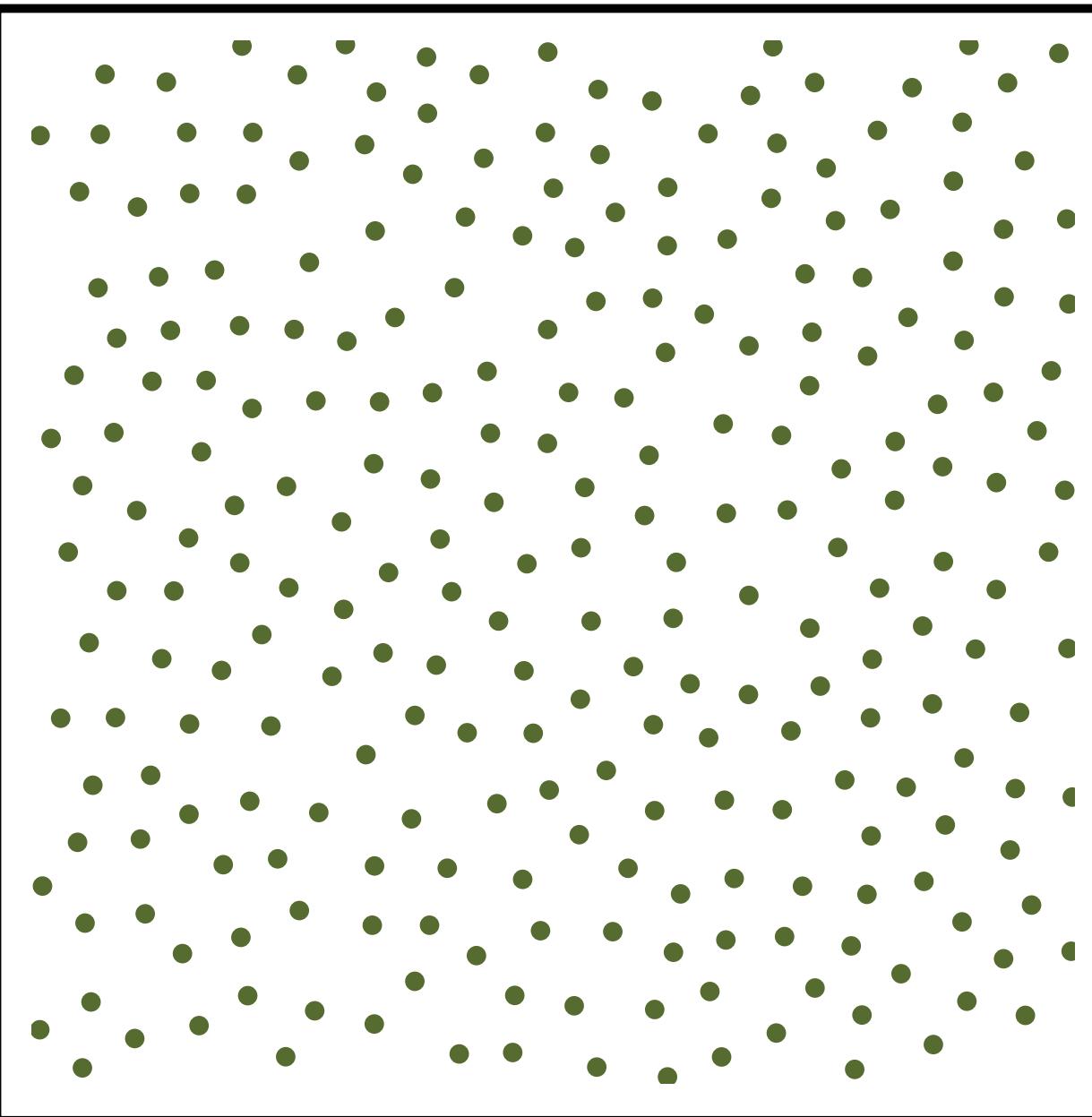


Random Dart Throwing

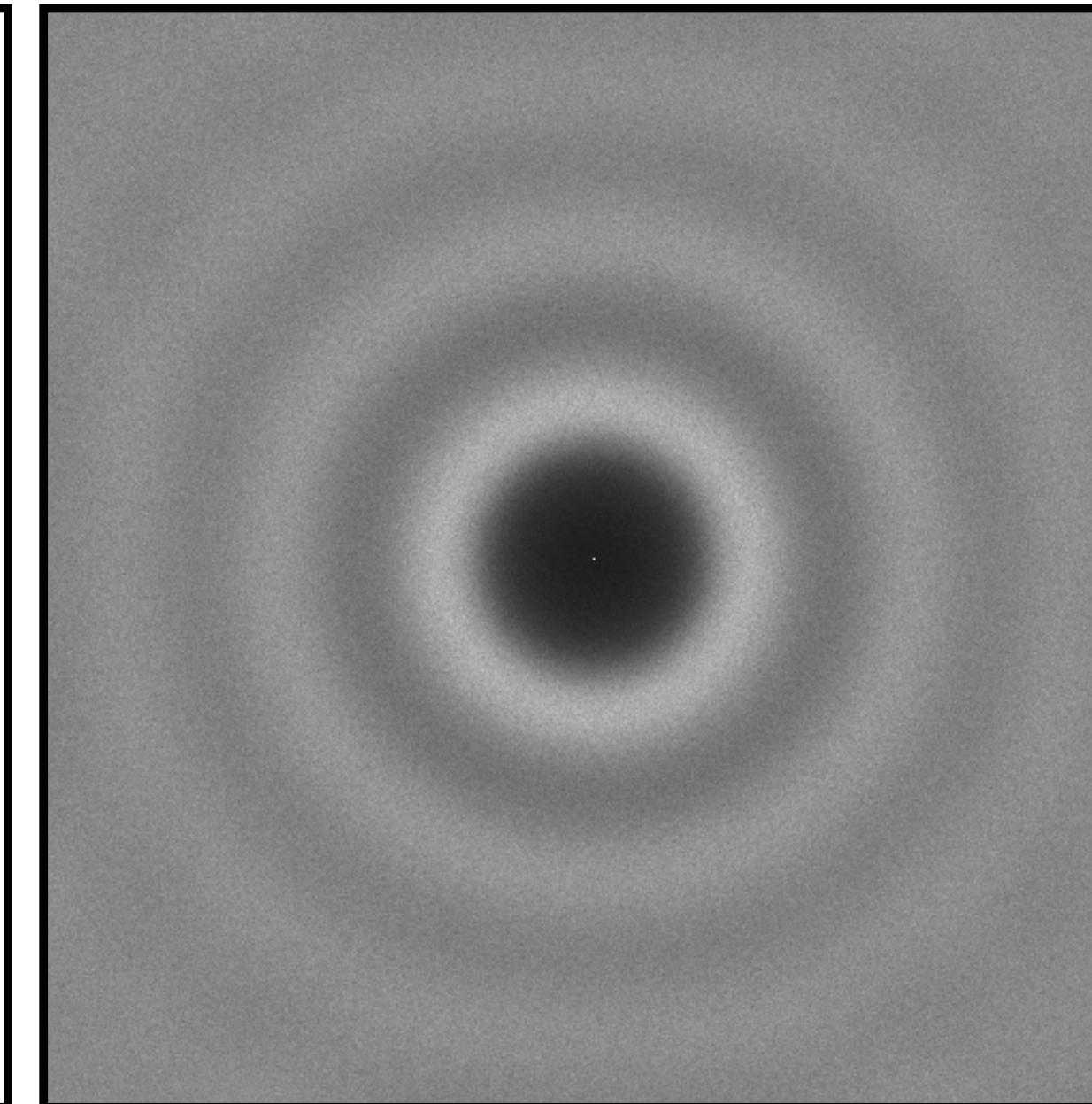


Poisson Disk Sampling

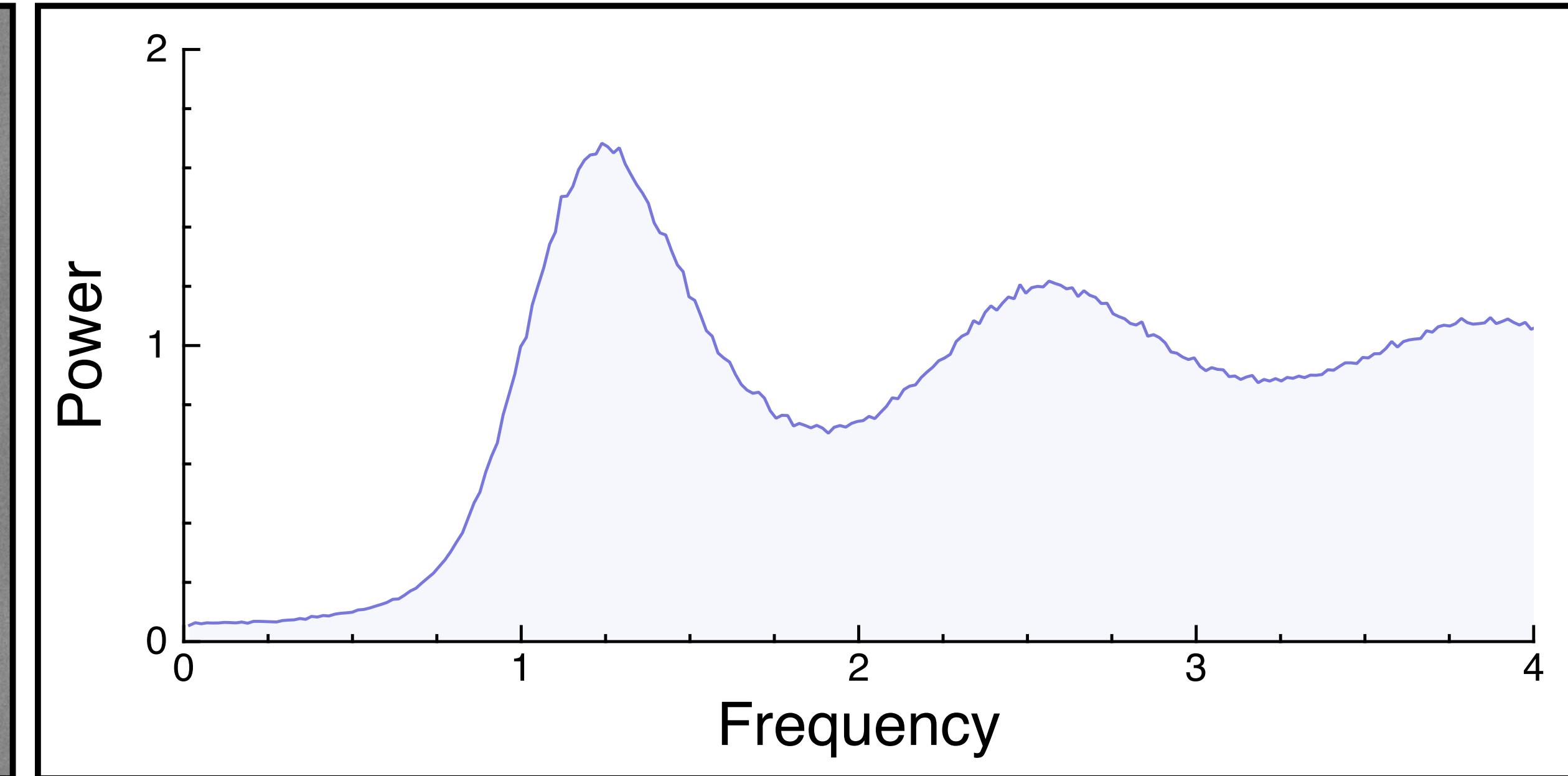
Samples



Power spectrum



Radial mean

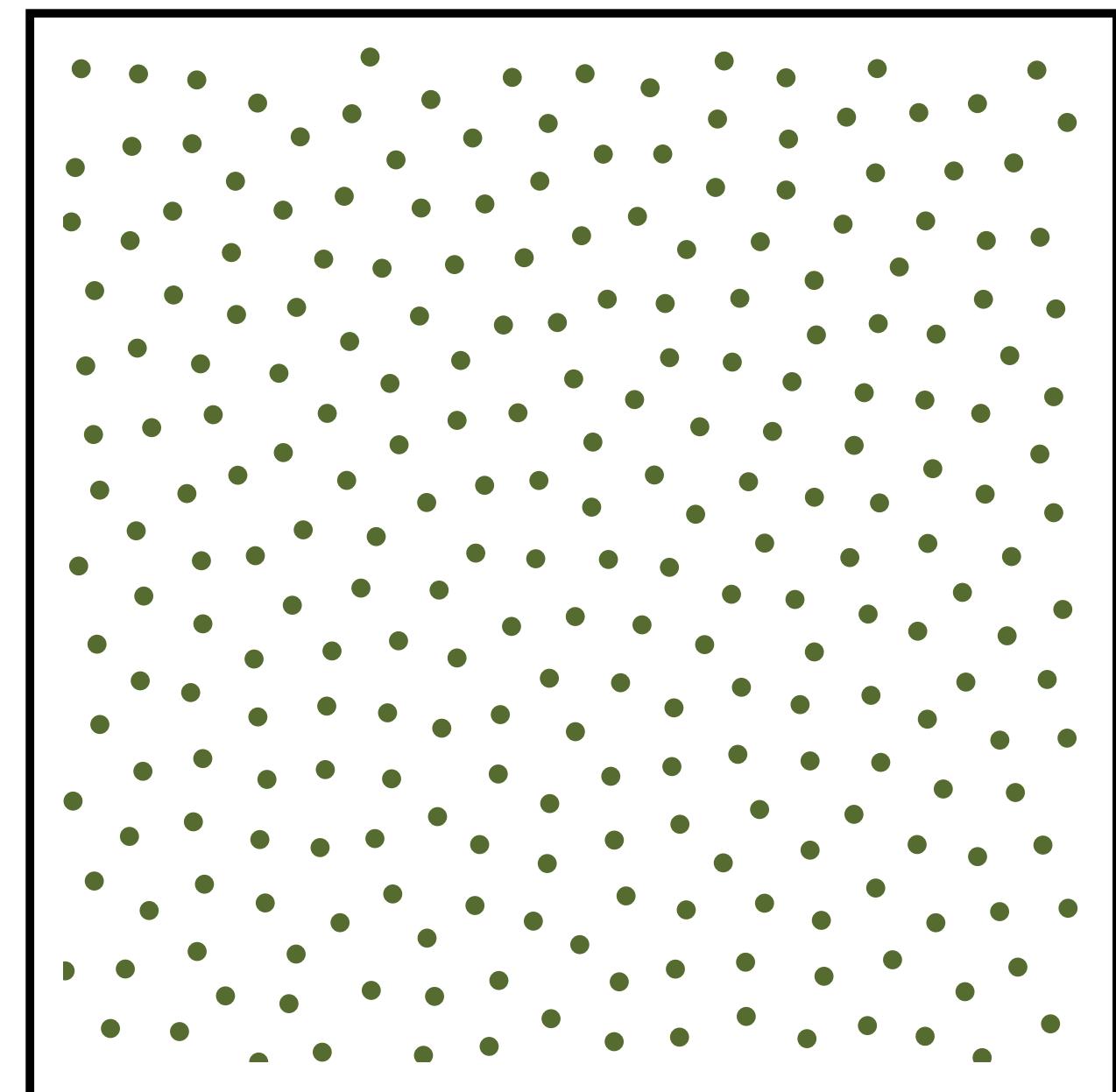


Blue-Noise Sampling (Relaxation-based)

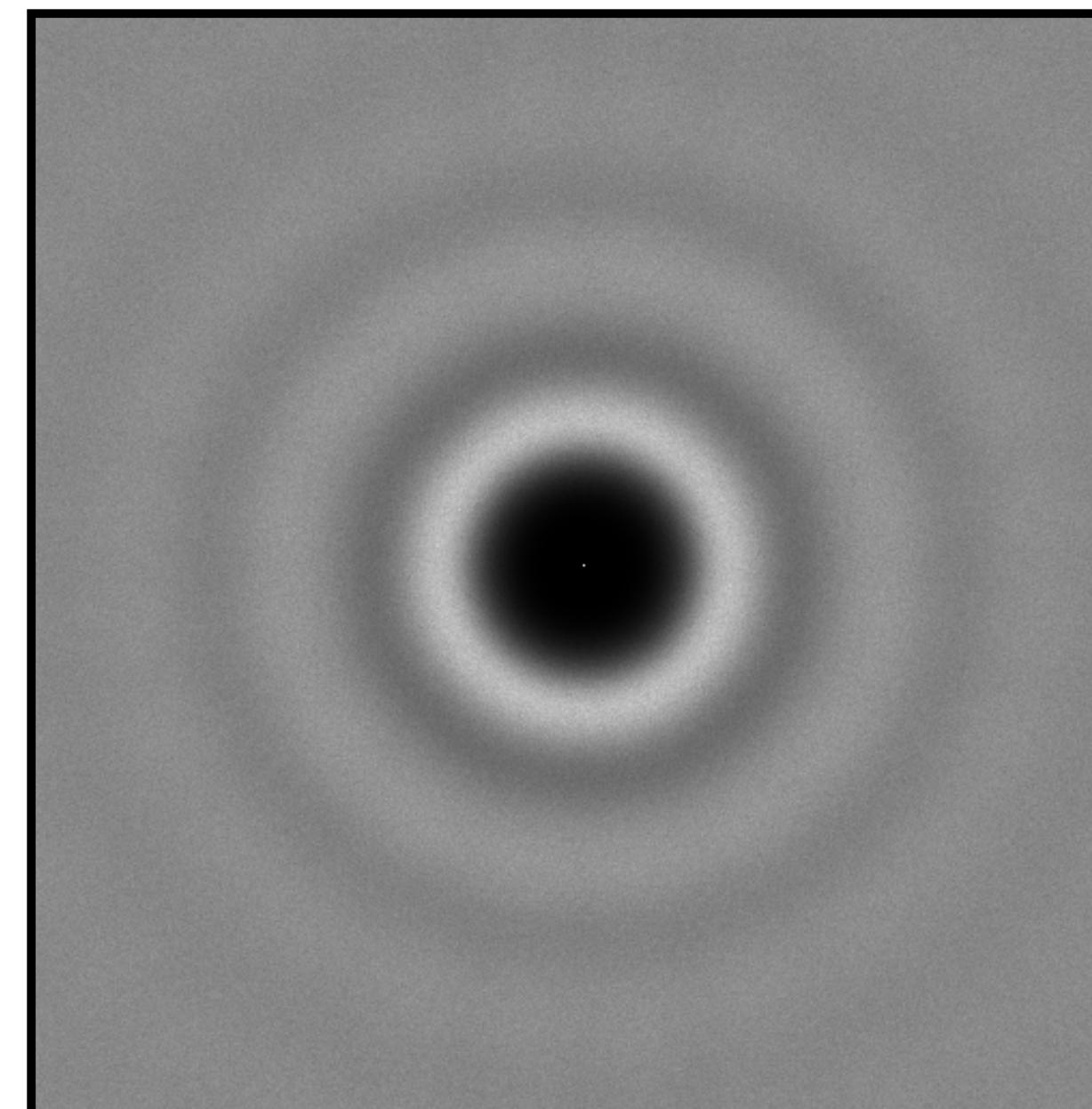
1. Initialize sample positions (e.g. random)
2. Use an iterative relaxation to move samples away from each other.

CCVT Sampling

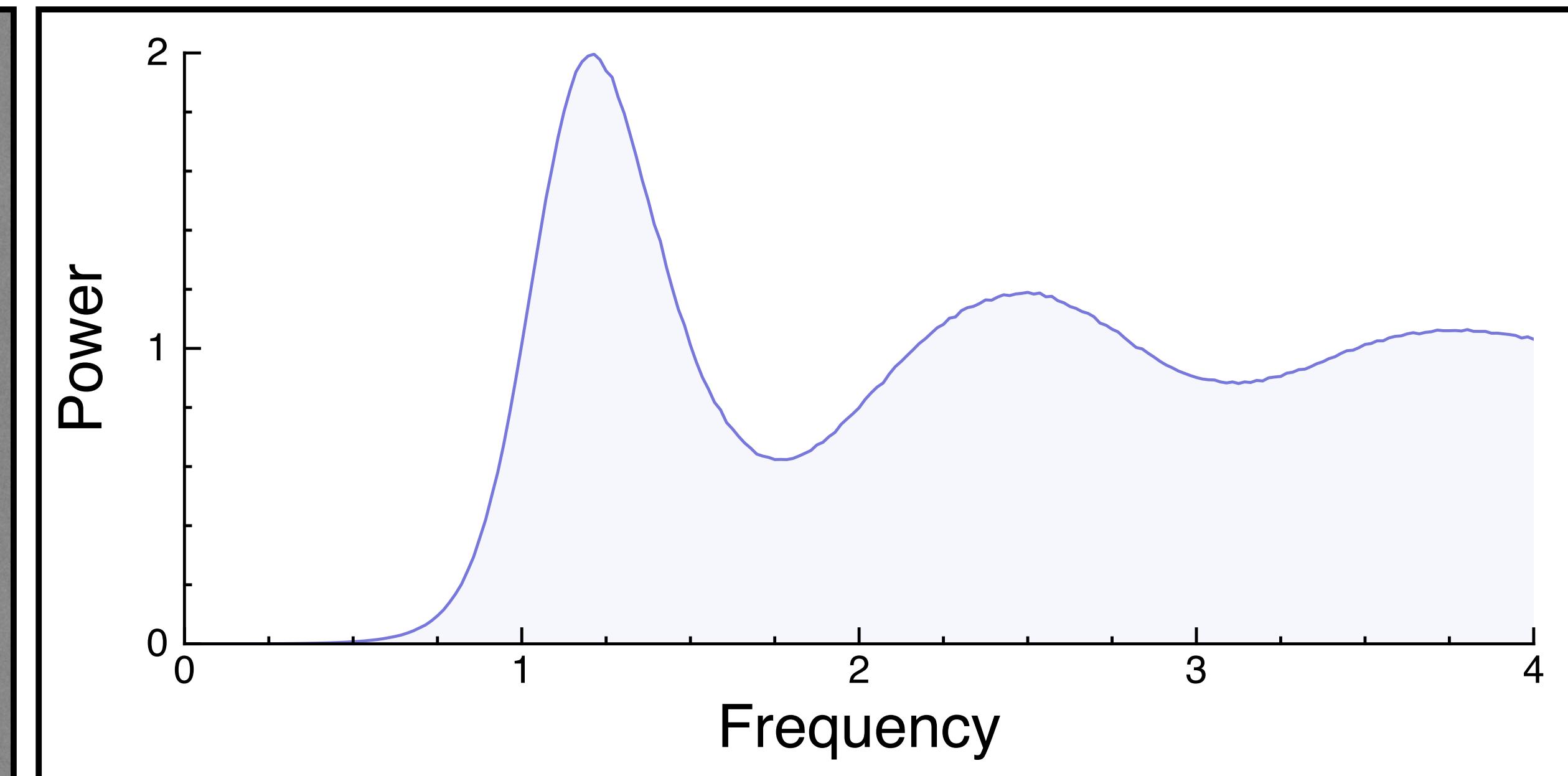
Samples



Power spectrum

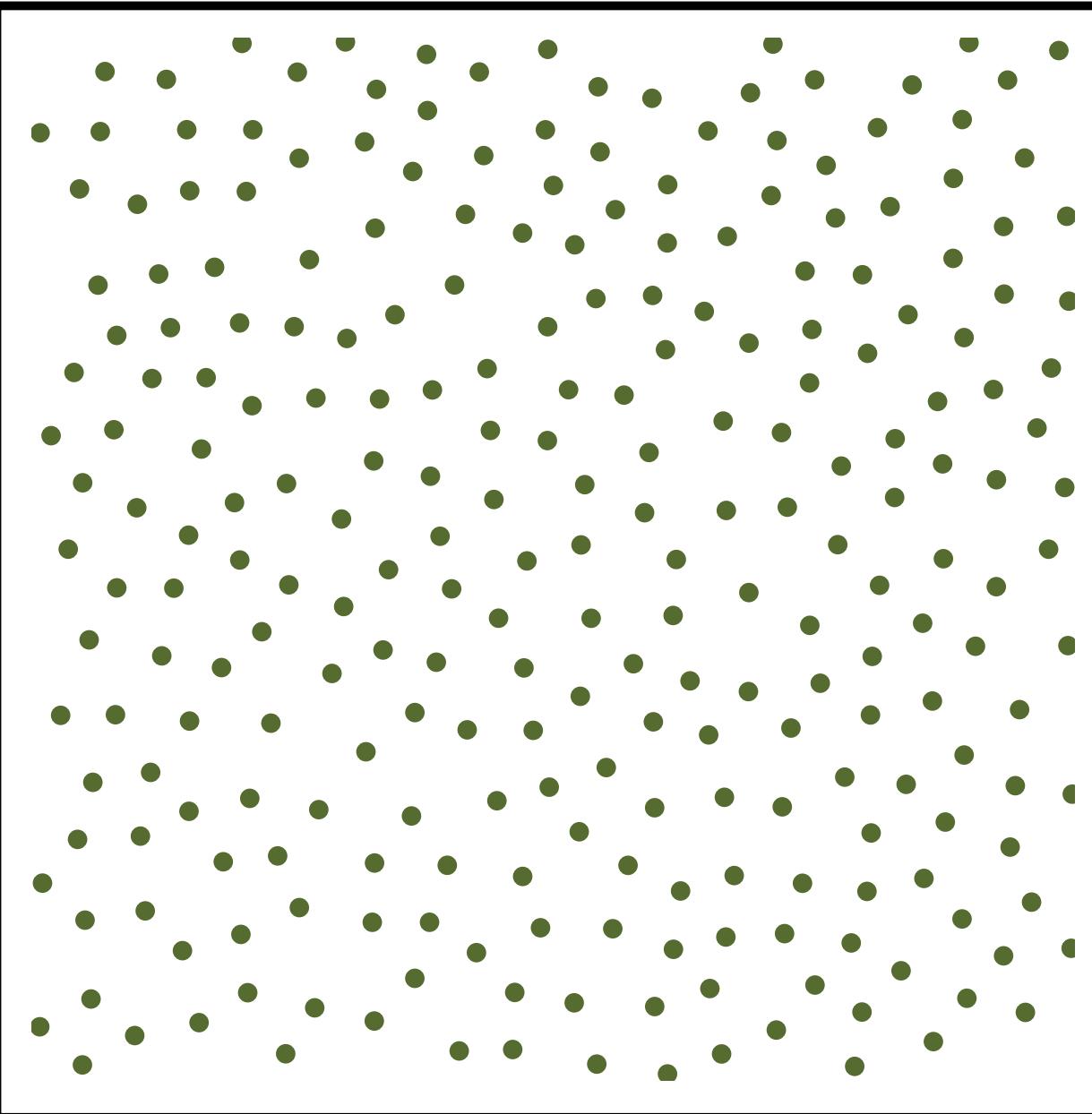


Radial mean

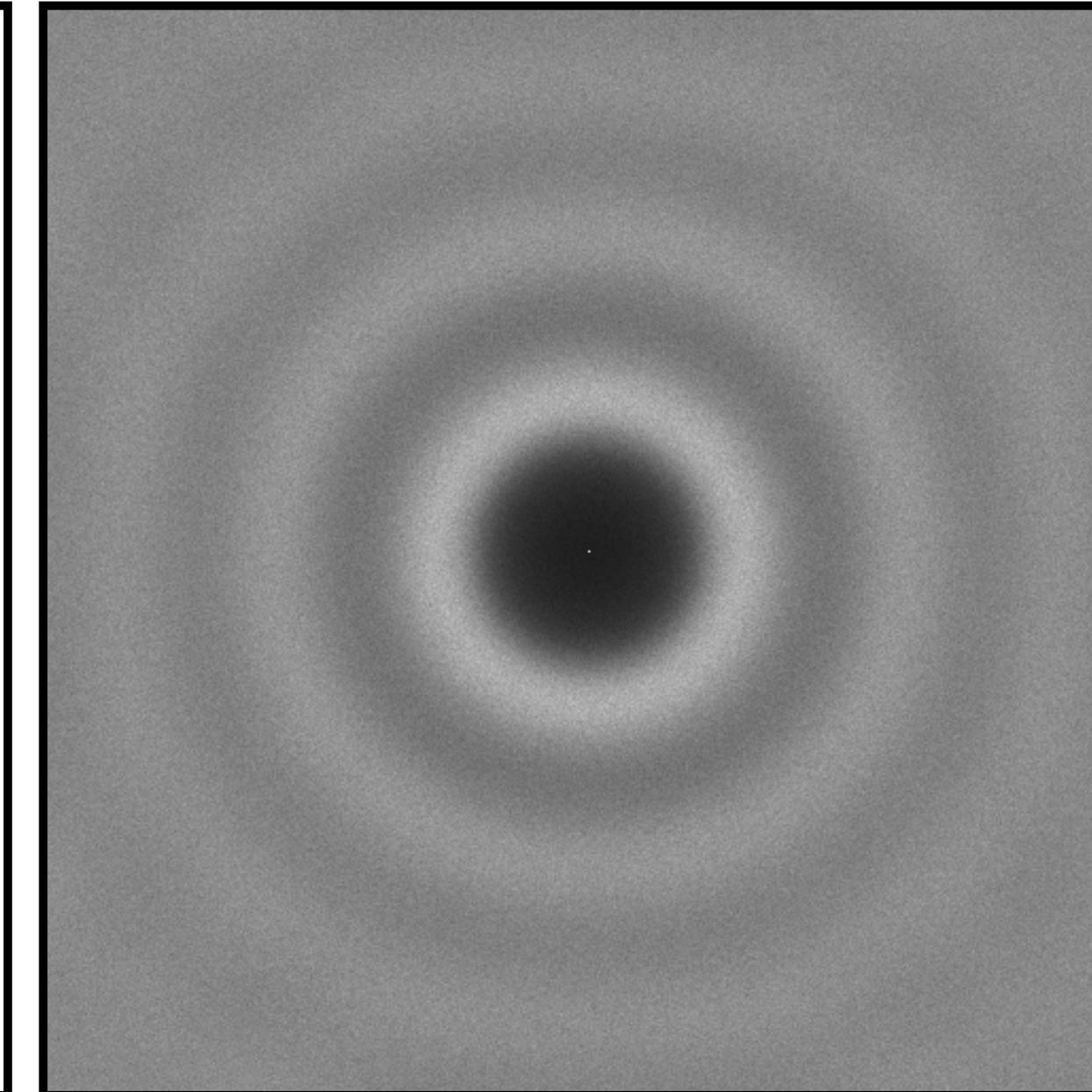


Poisson Disk Sampling

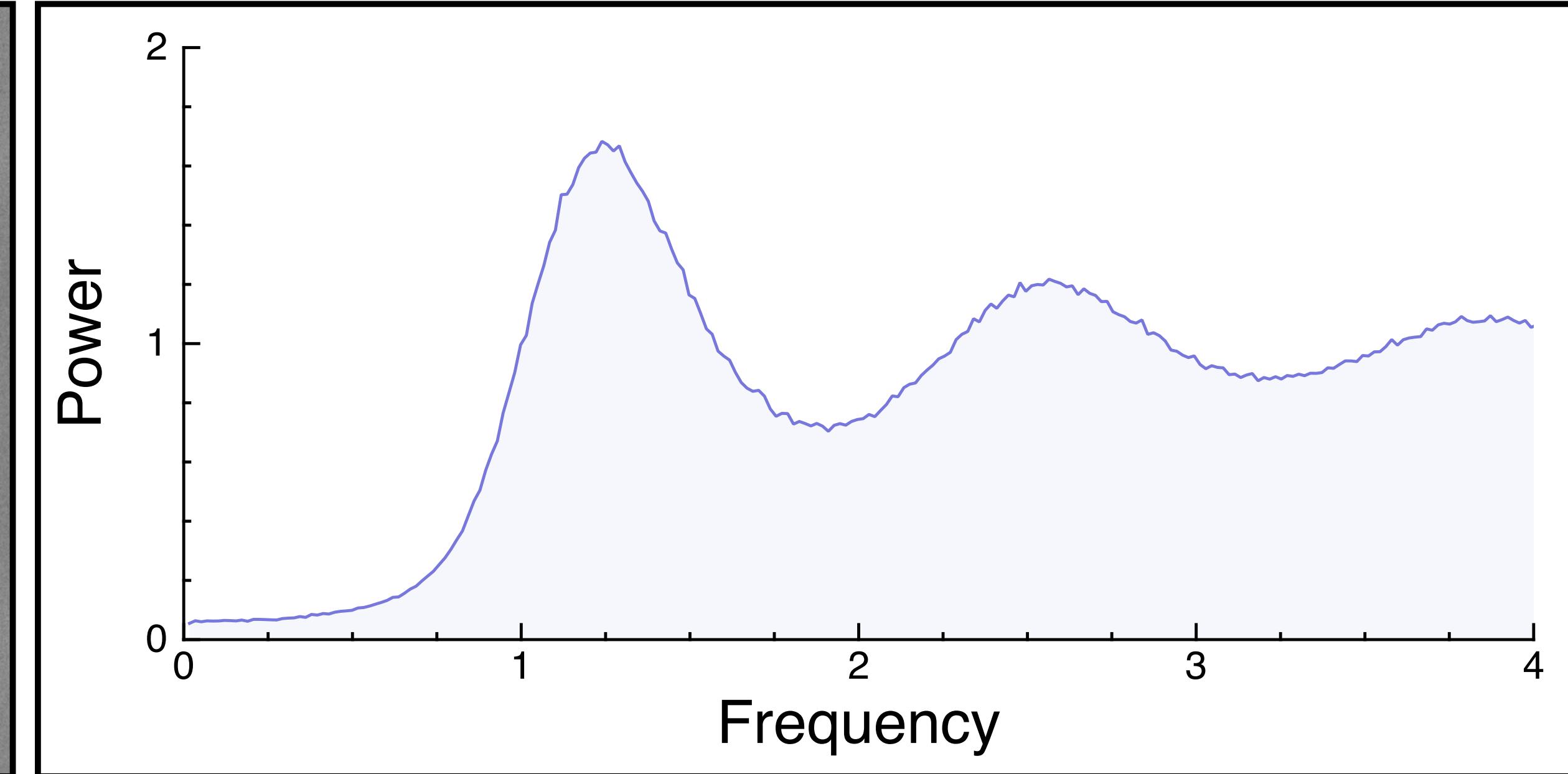
Samples



Power spectrum



Radial mean



Low-Discrepancy Sampling

Discrepancy is a measure of how “evenly spaced” samples are

Low-Discrepancy point sequences are deterministic sets of points specially crafted to be evenly distributed (have small discrepancy).

Entire field of study called Quasi-Monte Carlo (QMC)

The Radical Inverse

A positive integer value n can be expressed in a base b with a sequence of digits $d_m \dots d_2 d_1$

The radical inverse function Φ_b in base b converts a nonnegative integer n to a floating-point value in $[0, 1)$ by reflecting these digits about the decimal point:

$$\Phi_b(n) = 0.d_1 d_2 \dots d_m$$

Subsequent points “fall into biggest holes”

The Radical Inverse

```
float radicalInverse(int n, int base, float inv)
{
    float v = 0.0f;
    for (float p = inv; n != 0; p *= inv, n /= base)
        v += (n % base) * p;
    return v;
}
```

```
float radicalInverse(int n, int base)
{
    return radicalInverse(n, base, 1.0f / base);
}
```

More efficient version available for base 2

The Van der Corput Sequence

Radical Inverse in base two

n	Base 2	Φ_b
1	1	.1 = 1/2
2	10	.01 = 1/4
3	11	.11 = 3/4
4	100	.001 = 1/8
5	101	.101 = 5/8
6	110	.011 = 3/8
7	111	.111 = 7/8
...		



The Radical Inverse (Base 2)

```
float vanDerCorputRIU(uint n)
{
    n = (n << 16) | (n >> 16);
    n = ((n & 0x00ff00ff) << 8) | ((n & 0xff00ff00) >> 8);
    n = ((n & 0x0f0f0f0f) << 4) | ((n & 0xf0f0f0f0) >> 4);
    n = ((n & 0x33333333) << 2) | ((n & 0xcccccccc) >> 2);
    n = ((n & 0x55555555) << 1) | ((n & 0xaaaaaaaa) >> 1);
    return n / float (0x100000000LL);
}
```

The Halton Sequence

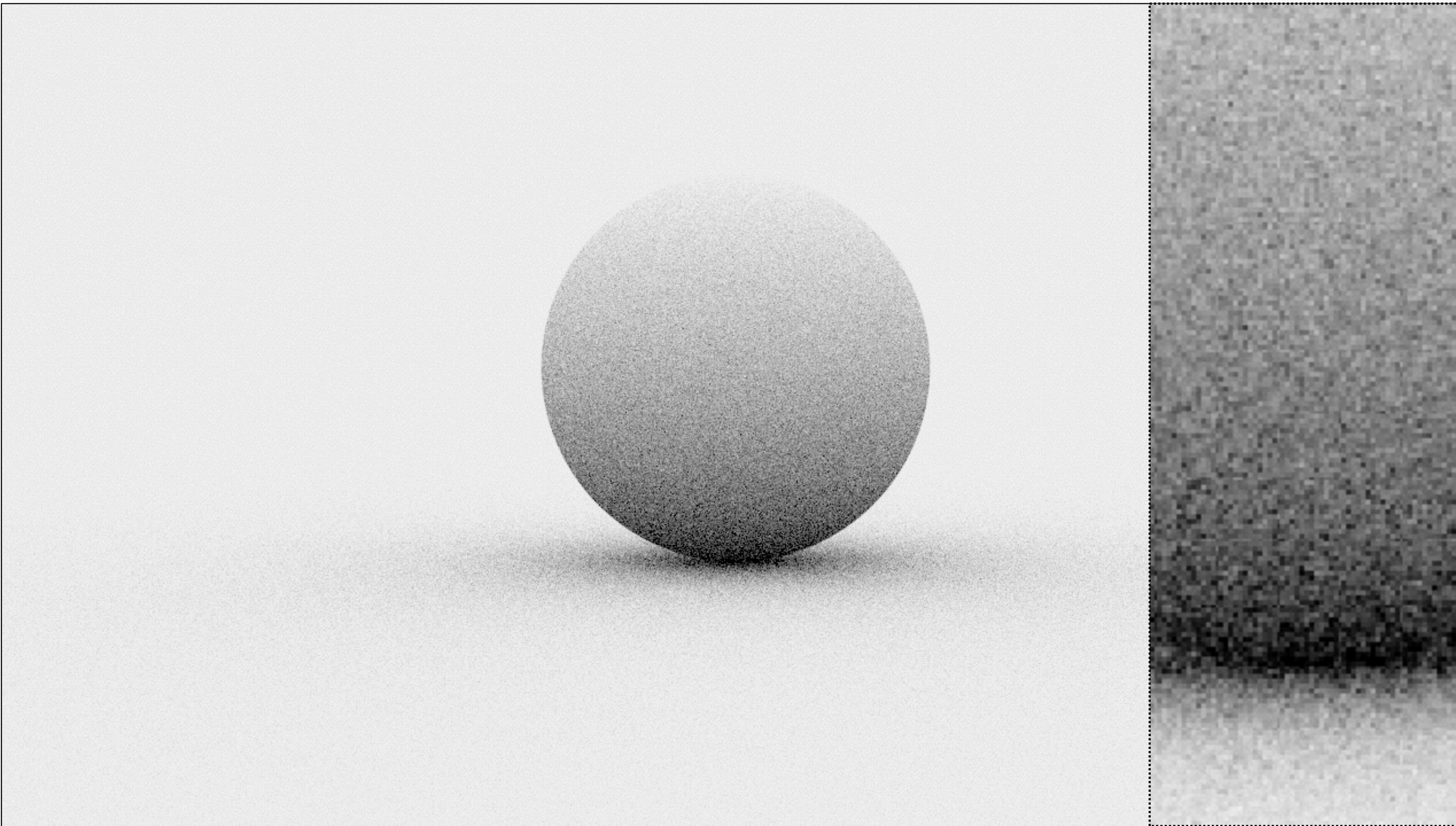
An n-dimensional Halton sequence uses the radical inverse with a different base b for each dimension.

$$x_i = (\Phi_2(i), \Phi_3(i), \Phi_5(i), \dots, \Phi_{p_n}(i))$$

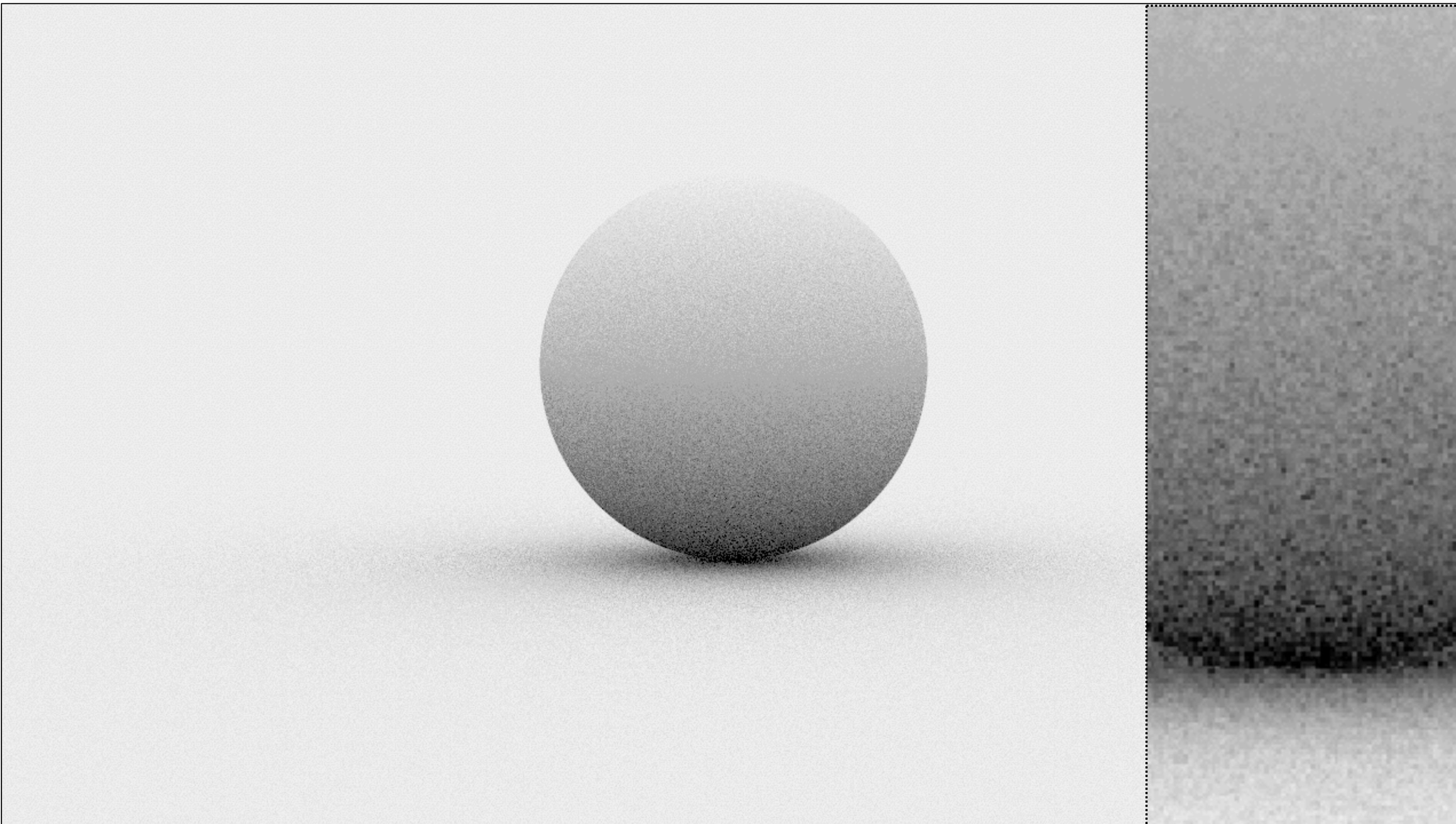
The bases should all be relatively prime.

Incremental/progressive generation of samples

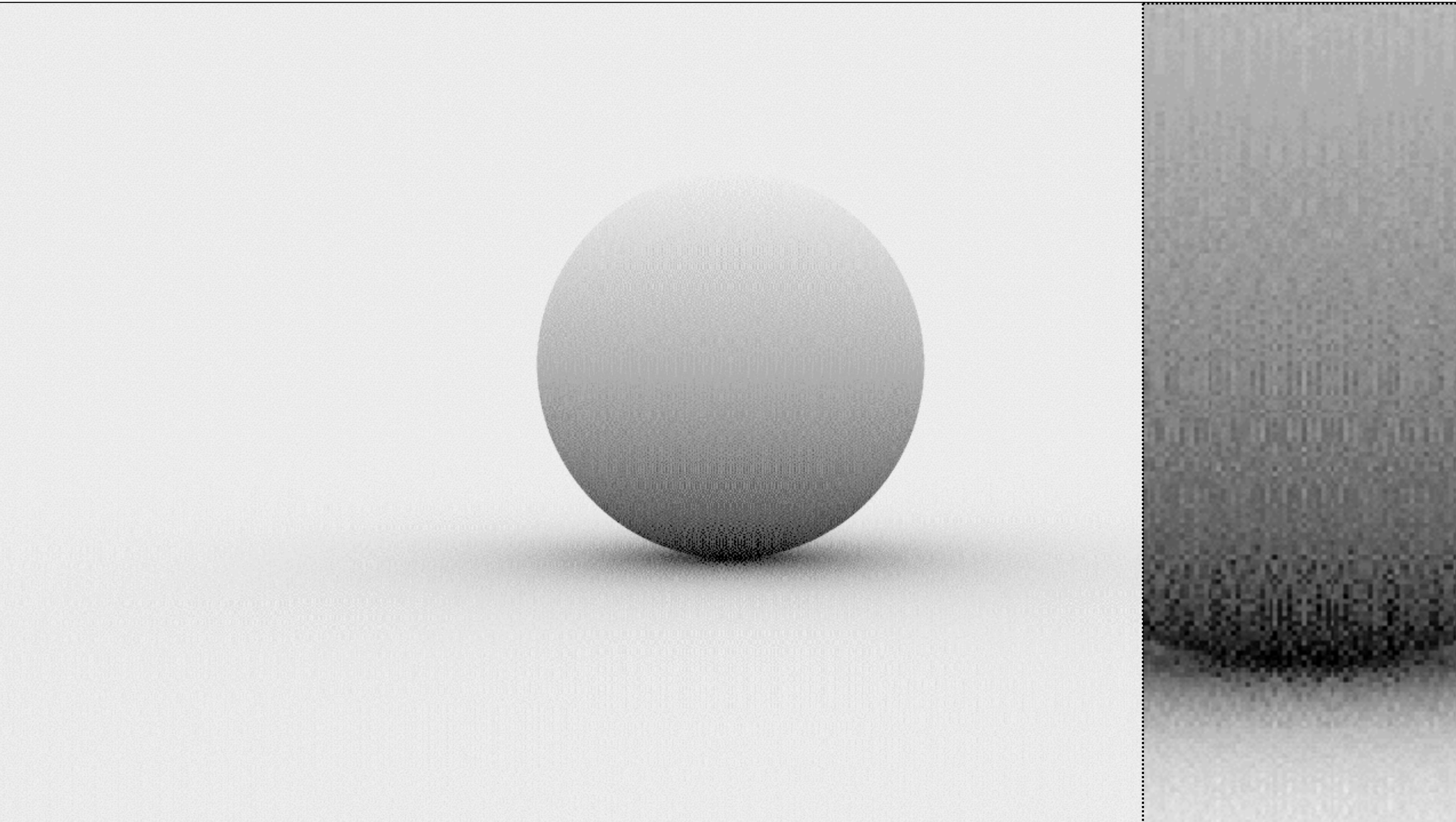
Monte Carlo (16 random samples)



Monte Carlo (16 stratified samples)



Quasi-Monte Carlo (16 Halton samples)



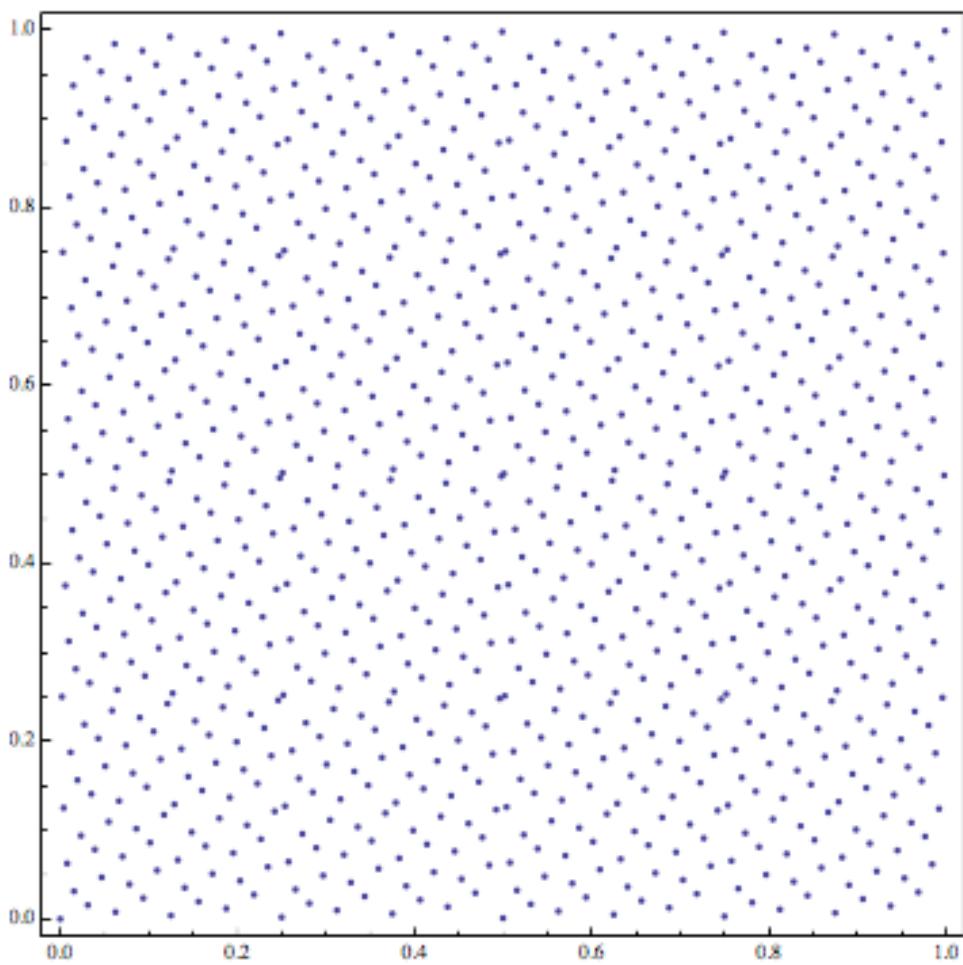
The Hammersley Sequence

Same as Halton, but uses i/N for first dimension:

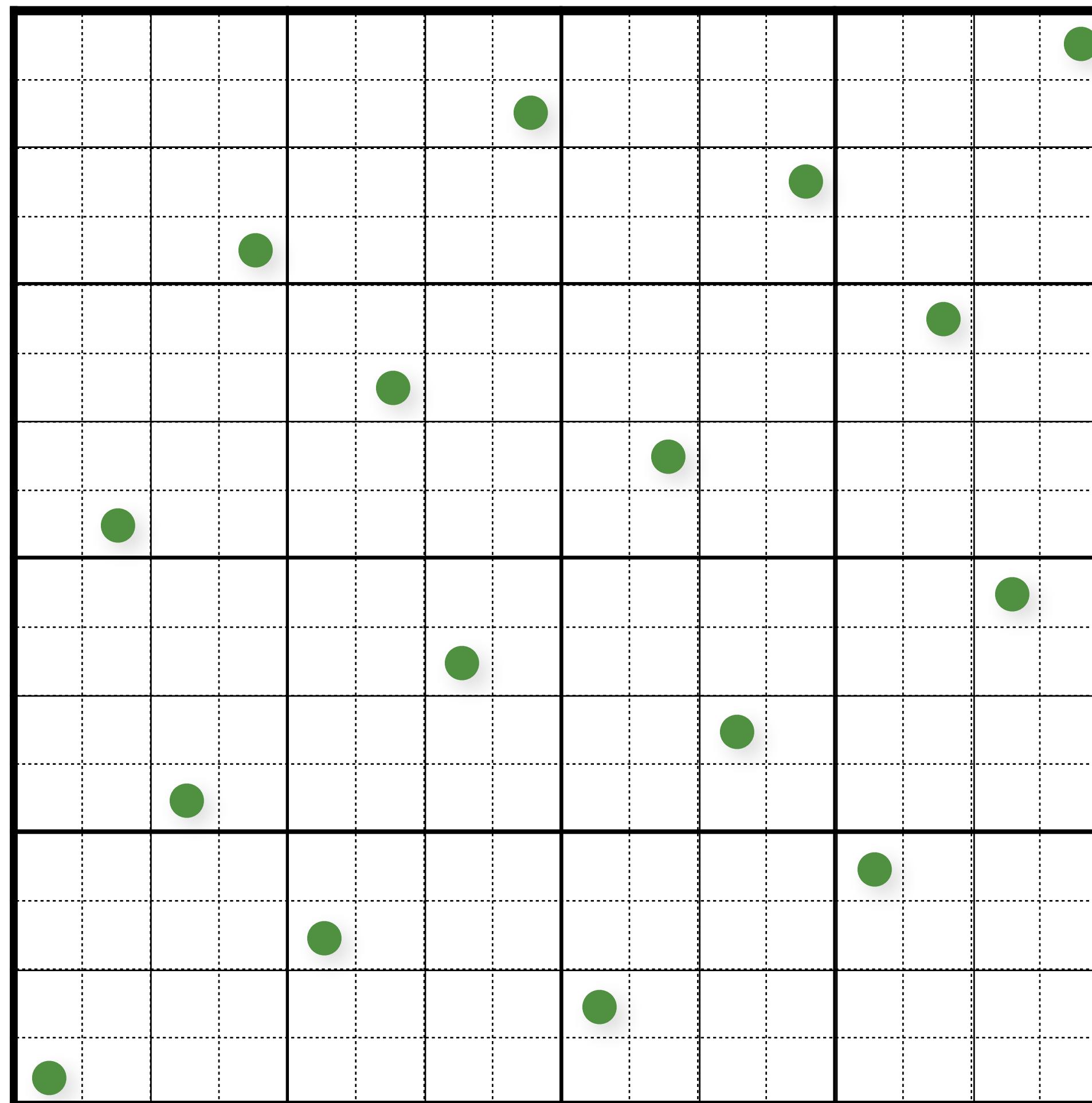
$$x_i = \left(\frac{i}{N}, \Phi_2(i), \Phi_3(i), \dots, \Phi_{p_n}(i) \right)$$

Provides slightly lower discrepancy

Not incremental, need to know total number of samples, N , in advance

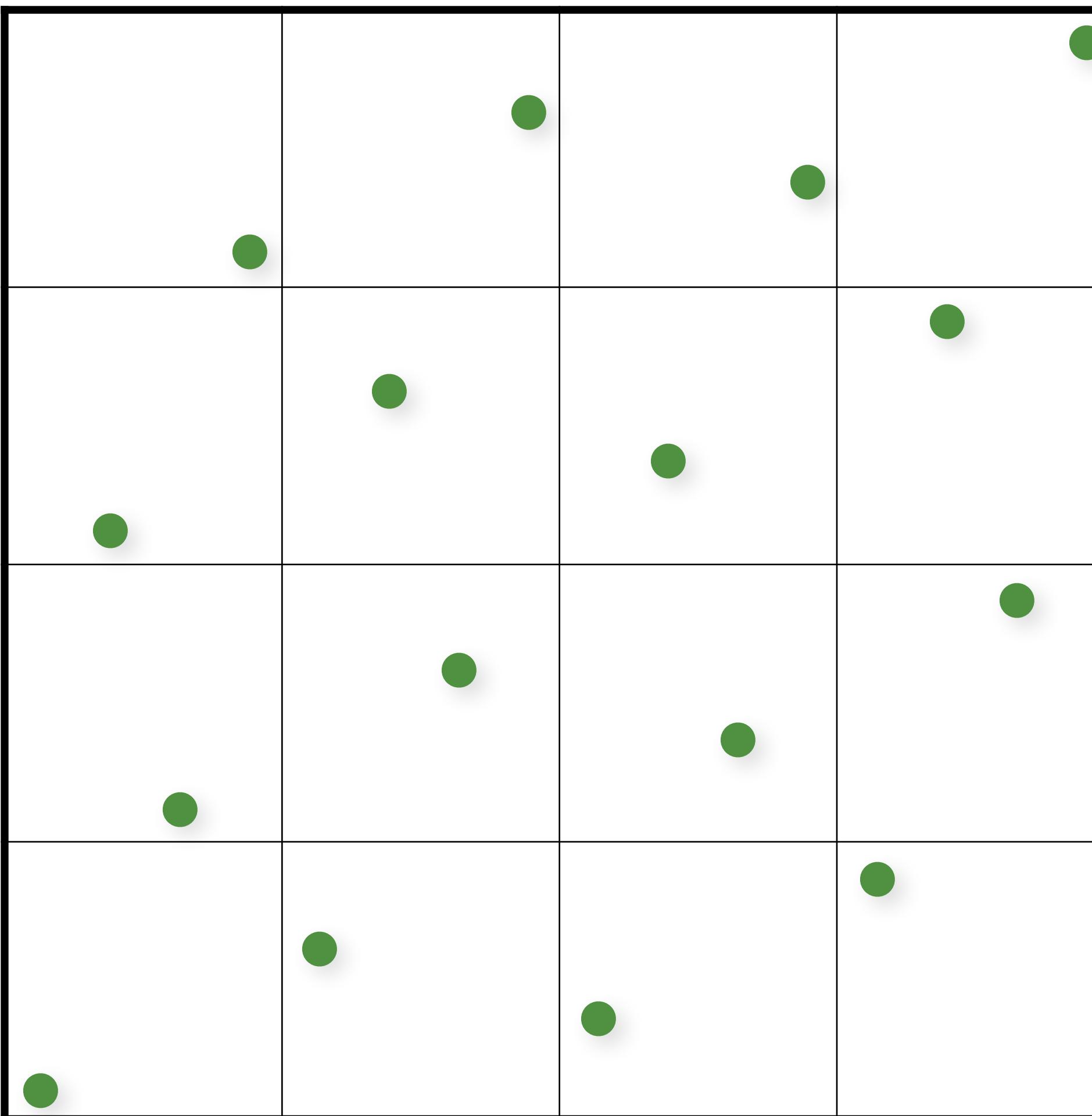


The Hammersley Sequence



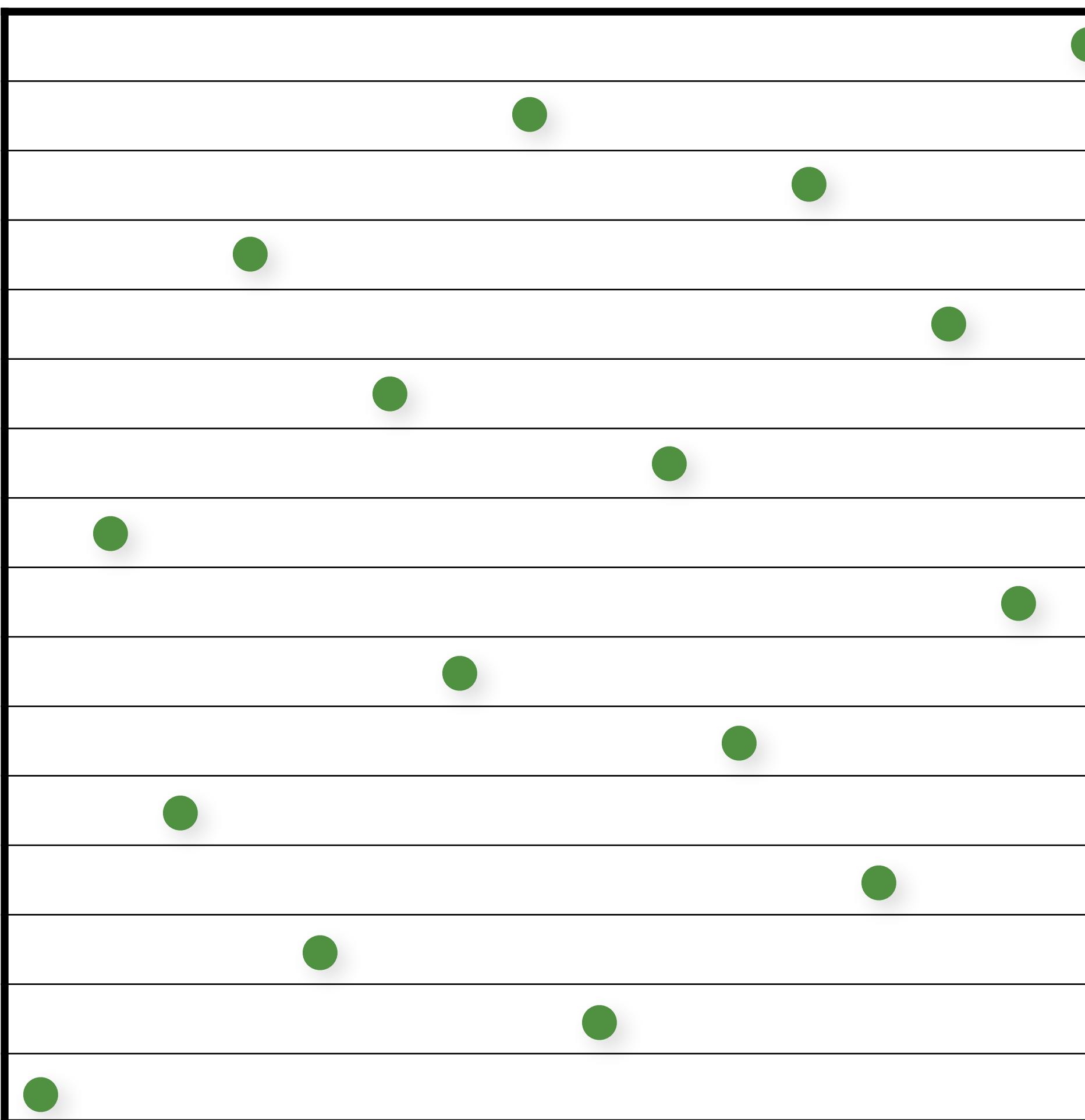
1 sample in each “elementary interval”

The Hammersley Sequence



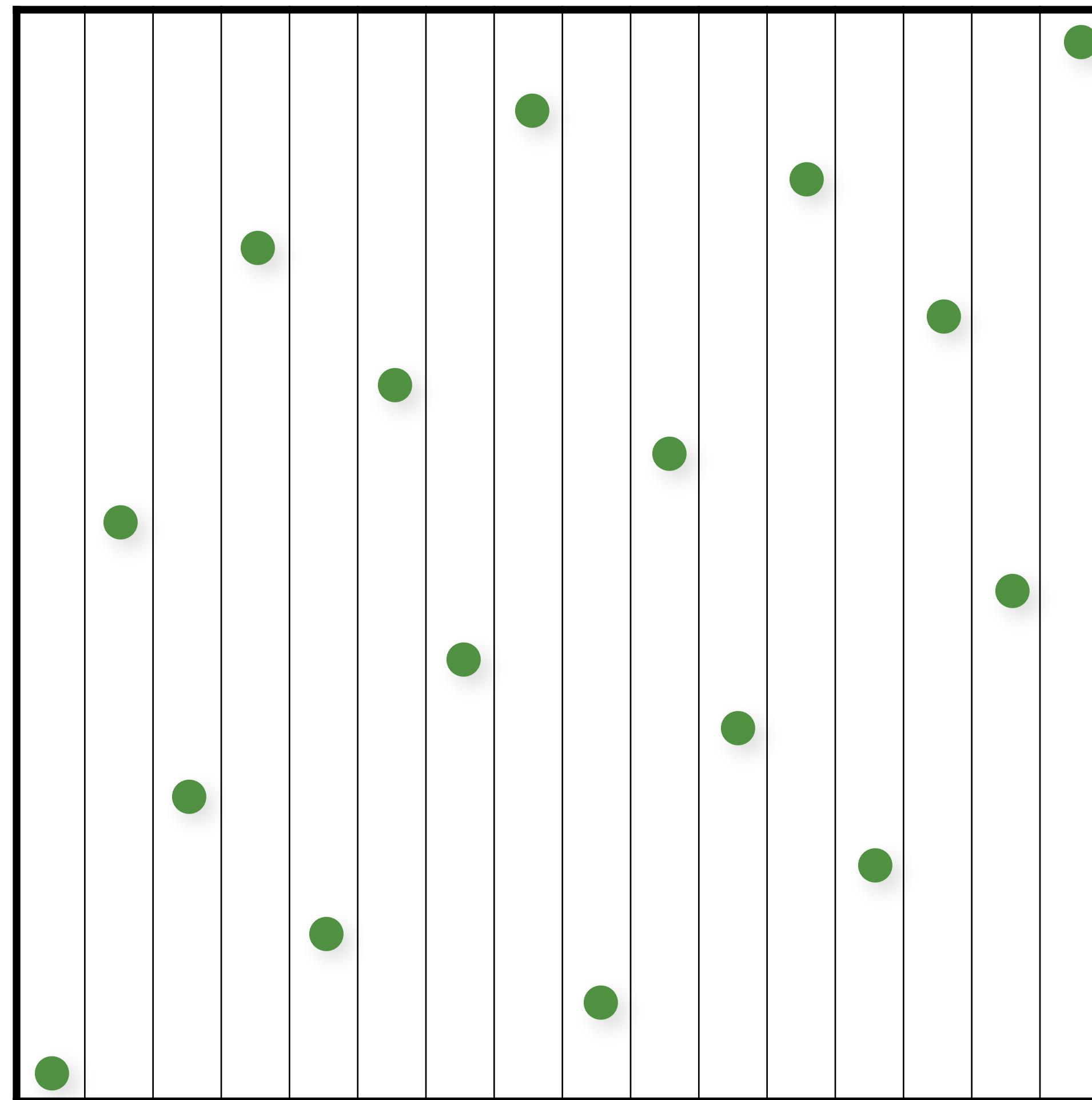
1 sample in each “elementary interval”

The Hammersley Sequence

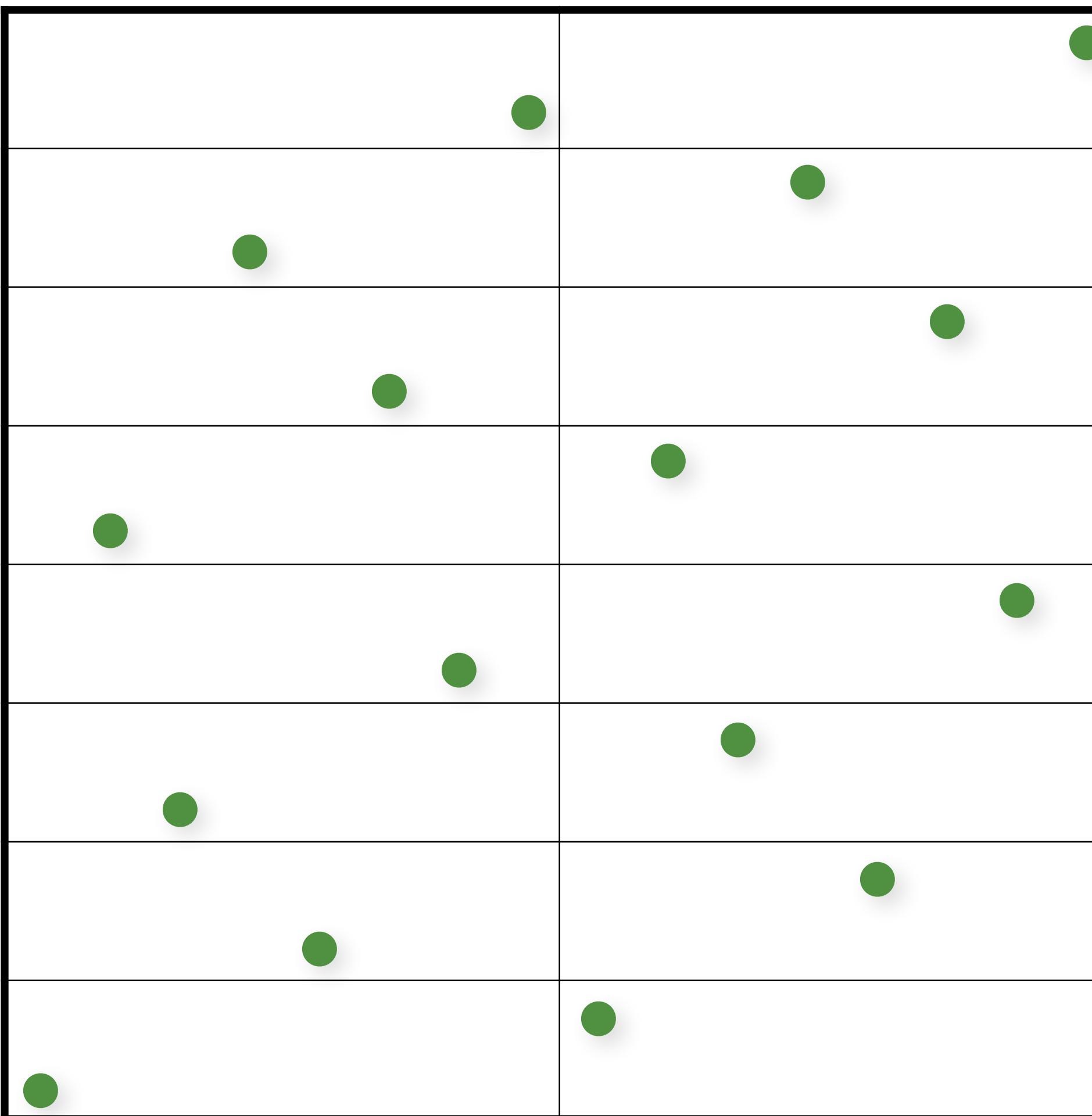


1 sample in each “elementary interval”

The Hammersley Sequence

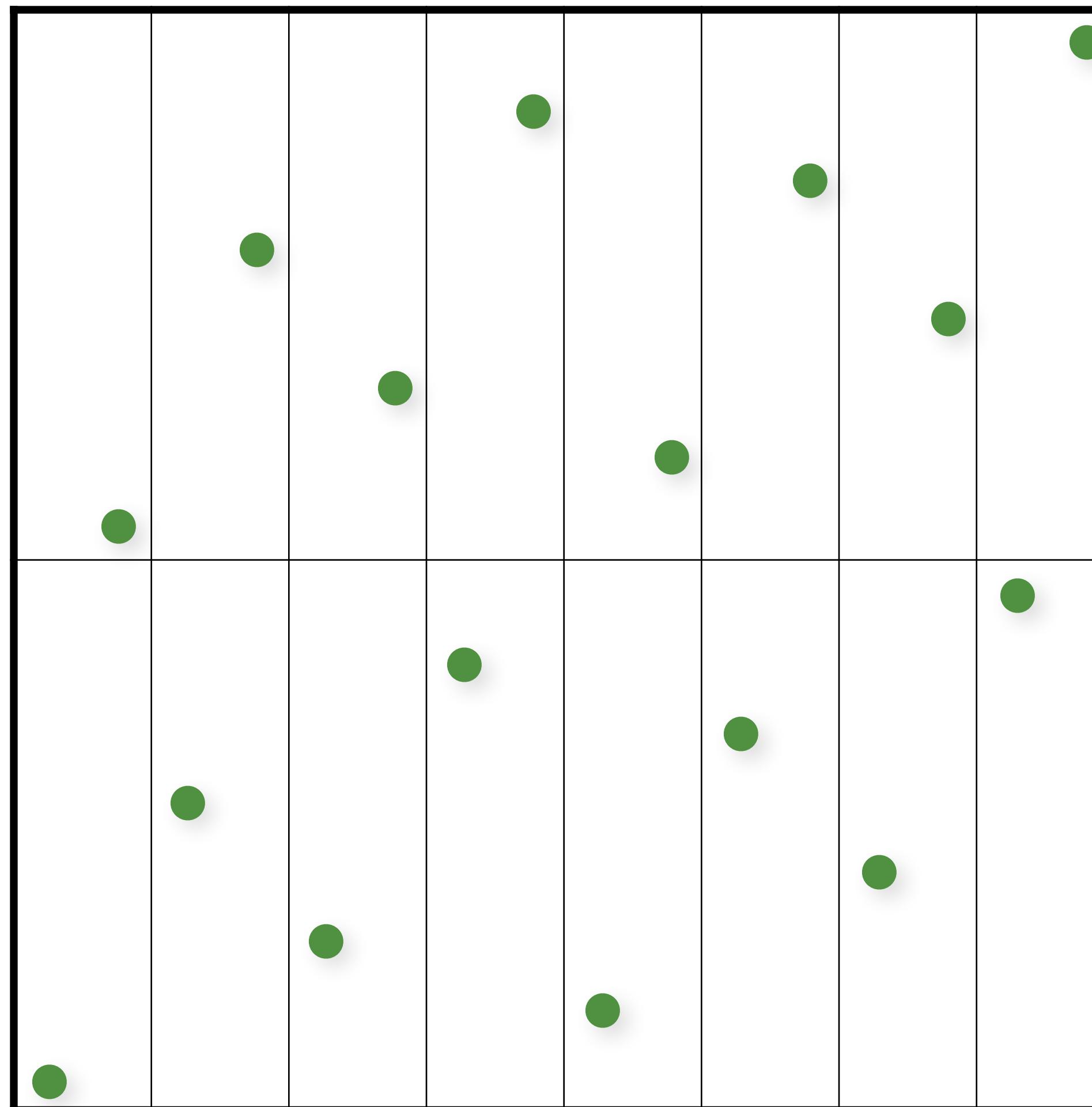


The Hammersley Sequence



1 sample in each “elementary interval”

The Hammersley Sequence



1 sample in each “elementary interval”

Many more...

Sobol

Faure

Larcher-Pillichshammer

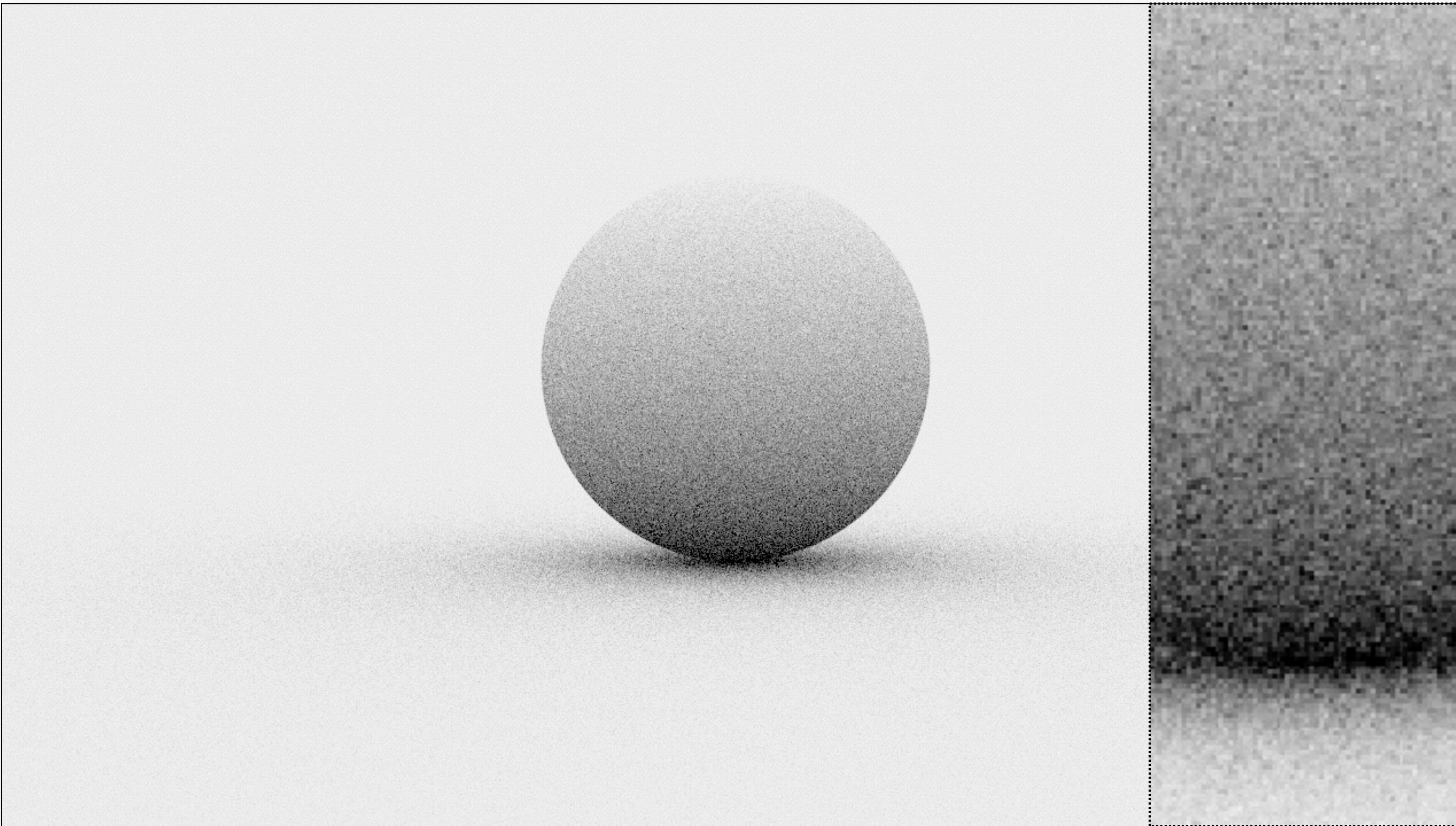
Folded Radical Inverse

(t,s)-sequences & (t,m,s)-nets

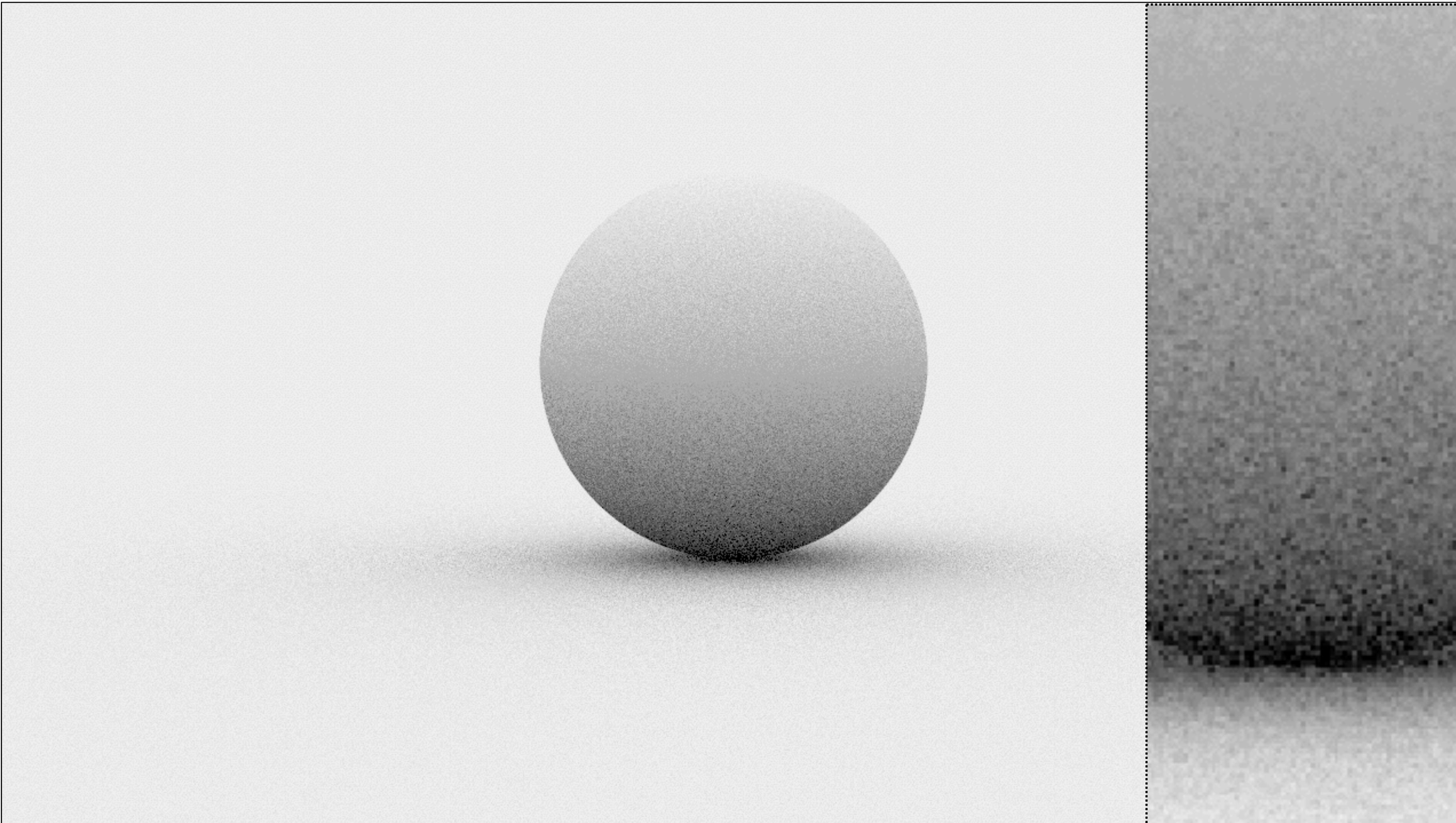
Scrambling/randomization

much more...

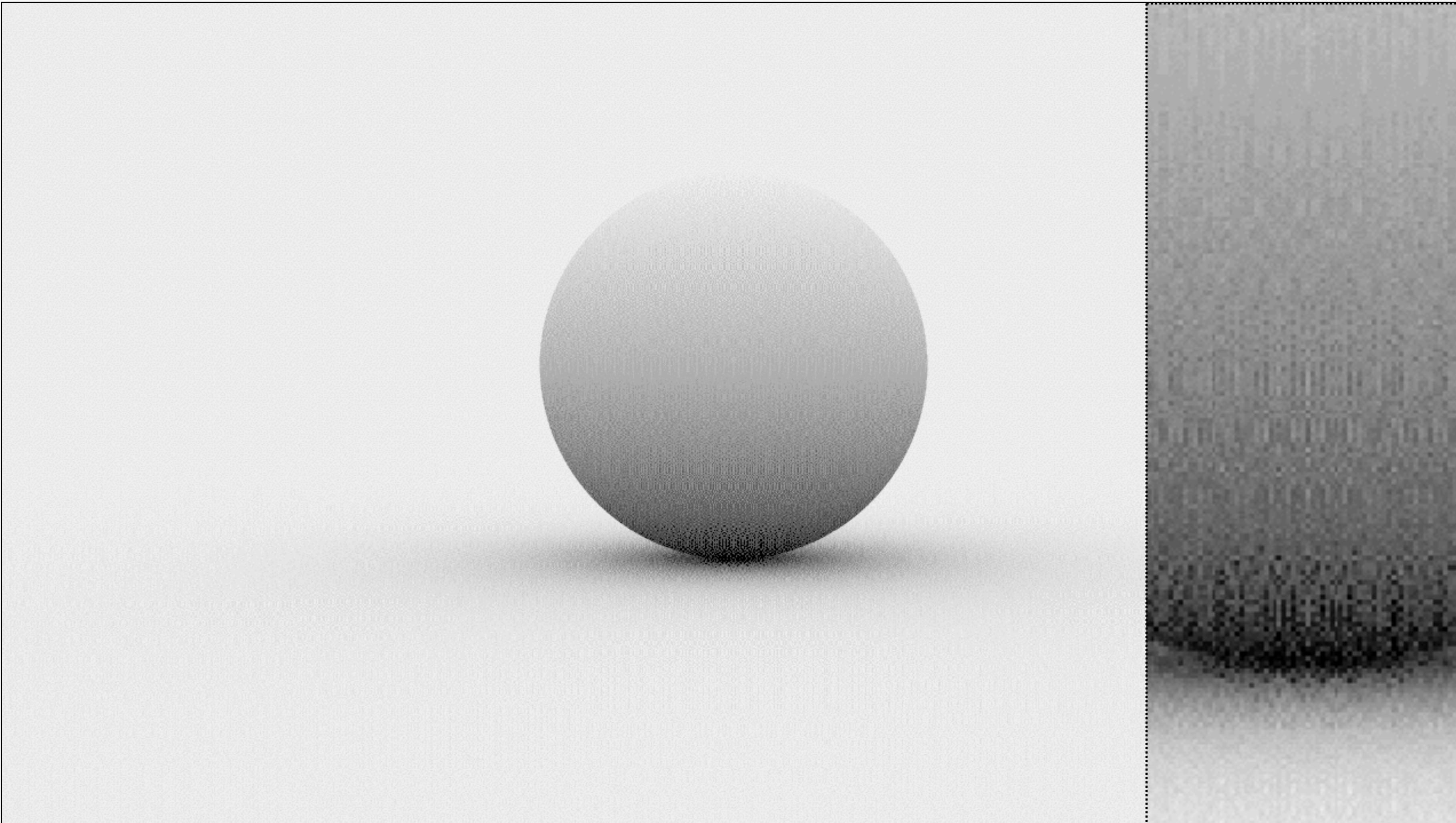
Monte Carlo (16 random samples)



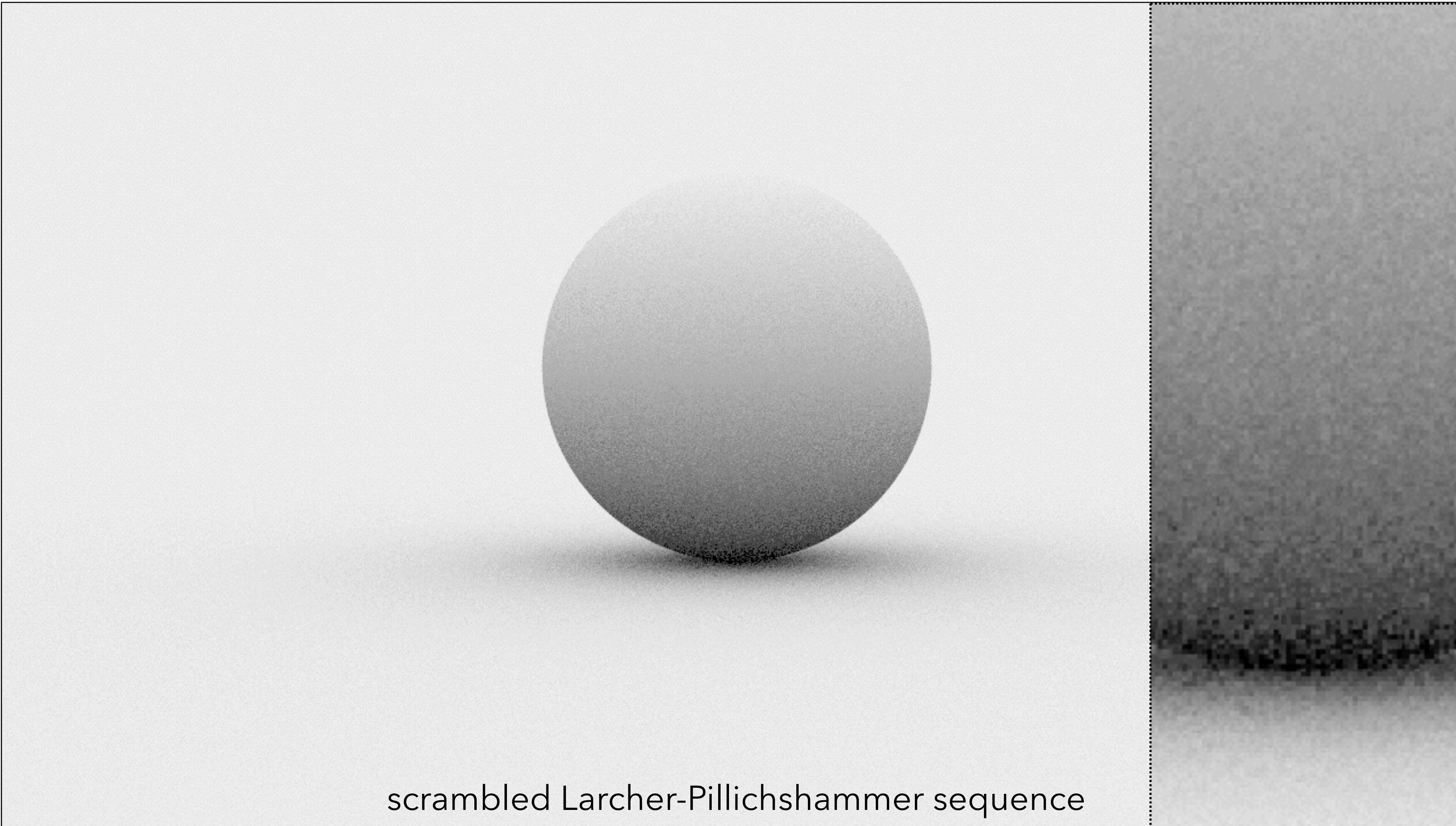
Monte Carlo (16 stratified samples)



Quasi-Monte Carlo (16 Halton samples)



Scrambled Quasi-Monte Carlo



scrambled Larcher-Pillichshammer sequence