My favorite Samples

Alexander Keller

Schedule

Course web page at https://sites.google.com/view/myfavoritesamples

- 9:00 My favorite Samples
  - Alexander Keller, NVIDIA
- 9:40 Progressive Multi-Jittered Sequences
  - Per Christensen, Pixar
- 10:15 Warp and Effect
  - Matt Pharr, NVIDIA
- break
- 11:05 Low-Discrepancy Blue Noise Sampling
  - Abdalla Ahmed, King Abdulla University and Victor Ostromoukhov, Université Claude Bernard Lyon 1
- 11:40 Blue-Noise Dithered Sampling
  - Iliyan Georgiev, Autodesk

My favorite Samples

For modeling

- discrete density approximation

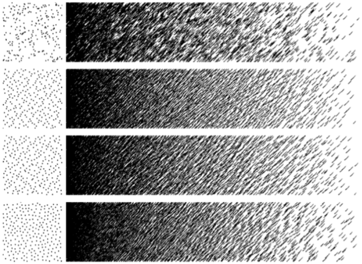


Figure 3: *Comparison of different input distributions with a point distribution and a grayscale ramp. From top to bottom: Poisson distribution, Halton sequence, Sobol sequence and hierarchical Poisson disk sequence.*

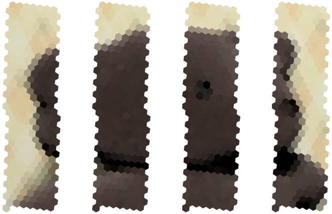
Figure 7: *Rendering of the Lena image using hatching and cross hatching. Primary strokes are aligned perpendicular to the gradient in regions of strong gradients and at a 45° angle in areas where the gradient is small.*

► Fast primitive distribution for illustration

My favorite Samples

For approximation

- displays and textures represented by rank-1 lattices

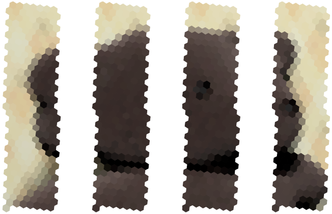


- Image Synthesis by Rank-1 Lattices
- Efficient Search for Two-Dimensional Rank-1 Lattices with Applications in Graphics

My favorite Samples

For approximation

- displays and textures represented by rank-1 lattices



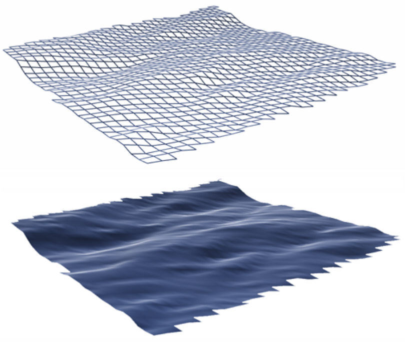
► Image Synthesis by Rank-1 Lattices

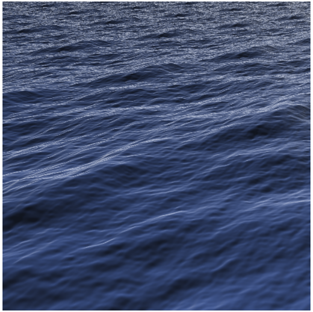
► Efficient Search for Two-Dimensional Rank-1 Lattices with Applications in Graphics

My favorite Samples

For simulation

- Fourier transform on rank-1 lattices





► Simulation on Rank-1 Lattices

My favorite Samples

For integration

- Monte Carlo methods

$$\int_{[0,1)^s} f(x)dx$$

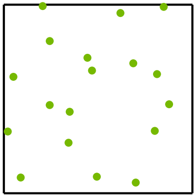
My favorite Samples

For integration

- Monte Carlo methods

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



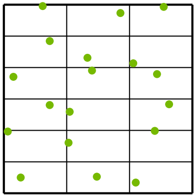
My favorite Samples

For integration

- Monte Carlo methods

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



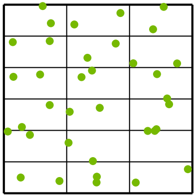
My favorite Samples

For integration

- Monte Carlo methods

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



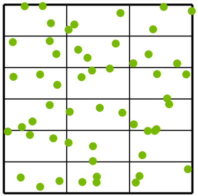
My favorite Samples

For integration

- Monte Carlo methods

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



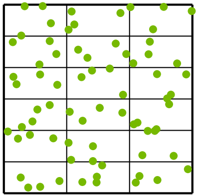
My favorite Samples

For integration

- Monte Carlo methods

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



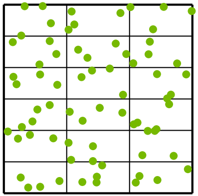
My favorite Samples

For integration

- Monte Carlo methods

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



- quasi-Monte Carlo methods

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

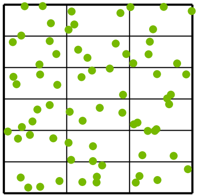
My favorite Samples

For integration

- Monte Carlo methods

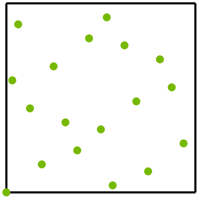
$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



- quasi-Monte Carlo methods

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$



  - much more uniform correlated samples x_i
  - realized by low-discrepancy sequences, which are progressive Latin-hypercube samples

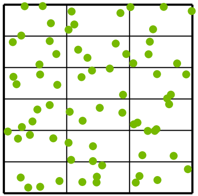
My favorite Samples

For integration

- Monte Carlo methods

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

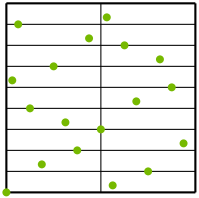
  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



- quasi-Monte Carlo methods

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

  - much more uniform correlated samples x_i
  - realized by low-discrepancy sequences, which are progressive Latin-hypercube samples



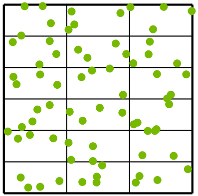
My favorite Samples

For integration

- Monte Carlo methods

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

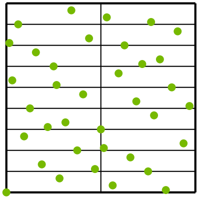
  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



- quasi-Monte Carlo methods

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

  - much more uniform correlated samples x_i
  - realized by low-discrepancy sequences, which are progressive Latin-hypercube samples



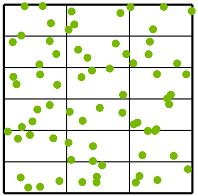
My favorite Samples

For integration

- Monte Carlo methods

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

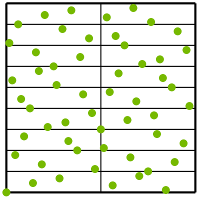
  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



- quasi-Monte Carlo methods

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

  - much more uniform correlated samples x_i
  - realized by low-discrepancy sequences, which are progressive Latin-hypercube samples



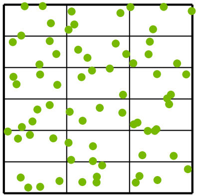
My favorite Samples

For integration

- Monte Carlo methods

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

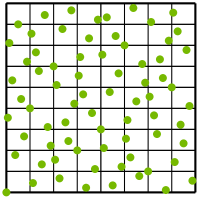
  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



- quasi-Monte Carlo methods

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

  - much more uniform correlated samples x_i
  - realized by low-discrepancy sequences, which are progressive Latin-hypercube samples



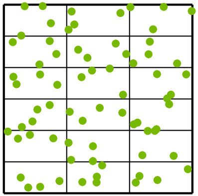
My favorite Samples

For integro-approximation

- Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(y,x_i)$$

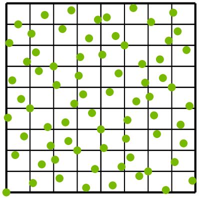
  - uniform, independent, unpredictable random samples x_i
  - simulated by pseudo-random numbers



- quasi-Monte Carlo methods

$$g(y) = \int_{[0,1)^s} f(y,x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(y,x_i)$$

  - much more uniform correlated samples x_i
  - realized by low-discrepancy sequences, which are progressive Latin-hypercube samples



Random or Deterministic?

What matters

- deterministic
  - may improve speed of convergence
  - reproducible and simple to parallelize

Random or Deterministic?

What matters

- deterministic
  - may improve speed of convergence
  - reproducible and simple to parallelize
- unbiased
  - zero difference between expectation and mathematical object
  - not sufficient for convergence

Random or Deterministic?

What matters

- deterministic
  - may improve speed of convergence
  - reproducible and simple to parallelize
- biased
  - allows for ameliorating the problem of insufficient techniques
  - can tremendously increase efficiency

Random or Deterministic?

What matters

- deterministic
  - may improve speed of convergence
  - reproducible and simple to parallelize
- biased
  - allows for ameliorating the problem of insufficient techniques
  - can tremendously increase efficiency
- consistent
  - error vanishes with increasing set of samples
  - no persistent artifacts introduced by algorithm

► Quasi-Monte Carlo image synthesis in a nutshell
► The Iray light transport simulation and rendering system

Random or Deterministic?

What matters

- deterministic
  - may improve speed of convergence
  - reproducible and simple to parallelize
- biased
  - allows for ameliorating the problem of insufficient techniques
  - can tremendously increase efficiency
- consistent
  - error vanishes with increasing set of samples
  - no persistent artifacts introduced by algorithm

▶ Quasi-Monte Carlo image synthesis in a nutshell
▶ The Iray light transport simulation and rendering system

Numerical Integration and Integro-Approximation

Sampling

- transform your problem onto the s-dimensional unit cube $[0,1)^s$
- generate uniformly distributed points in $[0,1)^s$
  - pseudo random numbers
  - points with blue noise characteristic (on the unit torus)
- compute your averages

► Non-uniform random variate generation

► Massively parallel construction of radix tree forests for the efficient sampling of discrete probability distributions

► Neural importance sampling

Numerical Integration and Integro-Approximation

Sampling

- transform your problem onto the s-dimensional unit cube $[0,1)^s$
- generate uniformly distributed points in $[0,1)^s$
  - pseudo random numbers
  - points with blue noise characteristic (on the unit torus)
  - radical inverse based points
  - rank-1 lattice
- compute your averages

► Non-uniform random variate generation

► Massively parallel construction of radix tree forests for the efficient sampling of discrete probability distributions

► Neural importance sampling

Numerical Integration and Integro-Approximation

Sampling

- transform your problem onto the s-dimensional unit cube $[0,1)^s$

- generate uniformly distributed points in $[0,1)^s$
  - pseudo random numbers
  - points with blue noise characteristic (on the unit torus)
  - radical inverse based points and randomizations
  - rank-1 lattice and randomizations

- compute your averages

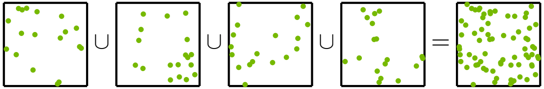
▶ Non-uniform random variate generation
▶ Massively parallel construction of radix tree forests for the efficient sampling of discrete probability distributions
▶ Neural importance sampling

Quasi-Monte Carlo Points

Quasi-Monte Carlo Points

Uniform sampling in Monte Carlo and quasi-Monte Carlo methods

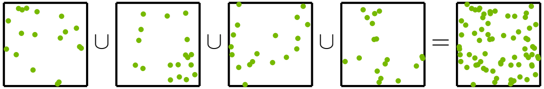
- random



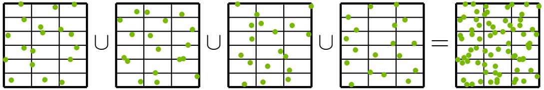
Quasi-Monte Carlo Points

Uniform sampling in Monte Carlo and quasi-Monte Carlo methods

- random



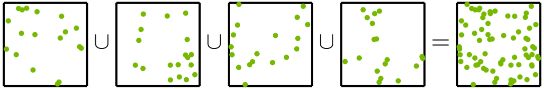
- stratified random



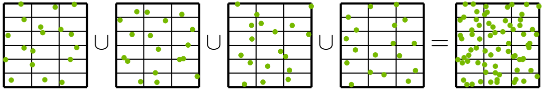
Quasi-Monte Carlo Points

Uniform sampling in Monte Carlo and quasi-Monte Carlo methods

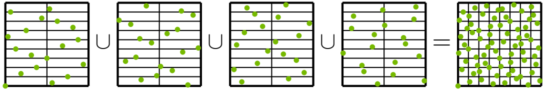
- random



- stratified random



- deterministic low discrepancy



Quasi-Monte Carlo Points

Radical inversion

- van der Corput sequence in base *b*

$$\begin{aligned}\Phi_b : \mathbb{N}_0 &\to \mathbb{Q} \cap [0,1) \\ i = \sum_{l=0}^{\infty} a_l(i) b^l &\mapsto \Phi_b(i) := \sum_{l=0}^{\infty} a_l(i) b^{-l-1}\end{aligned}$$

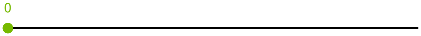
Quasi-Monte Carlo Points

Radical inversion

- van der Corput sequence in base *b*

$$\Phi_b : \mathbb{N}_0 \quad \to \quad \mathbb{Q} \cap [0,1)$$

$$i = \sum_{l=0}^{\infty} a_l(i) b^l \quad \mapsto \quad \Phi_b(i) := \sum_{l=0}^{\infty} a_l(i) b^{-l-1}, \text{ e.g. } \Phi_2(i) \equiv$$



Quasi-Monte Carlo Points

Radical inversion

- van der Corput sequence in base *b*

$$\begin{aligned} \Phi_b : \mathbb{N}_0 &\to \mathbb{Q} \cap [0,1) \\ i = \sum_{l=0}^{\infty} a_l(i) b^l &\mapsto \Phi_b(i) := \sum_{l=0}^{\infty} a_l(i) b^{-l-1}, \text{ e.g. } \Phi_2(i) \equiv \end{aligned}$$

Quasi-Monte Carlo Points

Radical inversion

- van der Corput sequence in base b

$$\begin{aligned}\Phi_b : \mathbb{N}_0 &\to \mathbb{Q} \cap [0,1) \\ i = \sum_{l=0}^{\infty} a_l(i) b^l &\mapsto \Phi_b(i) := \sum_{l=0}^{\infty} a_l(i) b^{-l-1}, \text{ e.g. } \Phi_2(i) \equiv\end{aligned}$$

Quasi-Monte Carlo Points

Radical inversion

- van der Corput sequence in base b

$$\begin{aligned}\Phi_b : \mathbb{N}_0 &\to \mathbb{Q} \cap [0,1) \\ i = \sum_{l=0}^{\infty} a_l(i) b^l &\mapsto \Phi_b(i) := \sum_{l=0}^{\infty} a_l(i) b^{-l-1}, \text{ e.g. } \Phi_2(i) \equiv\end{aligned}$$

Quasi-Monte Carlo Points

Radical inversion

- van der Corput sequence in base b

$$
\begin{aligned}
\Phi_b : \mathbb{N}_0 &\to \mathbb{Q} \cap [0,1) \\
i = \sum_{l=0}^{\infty} a_l(i) b^l &\mapsto \Phi_b(i) := \sum_{l=0}^{\infty} a_l(i) b^{-l-1}, \text{ e.g. } \Phi_2(i) \equiv
\end{aligned}
$$

- properties
  - subsequent points that “fall into biggest holes”
  - not completely uniform distributed (CUD)

Quasi-Monte Carlo Points

Radical inversion

- van der Corput sequence in base b

$$\Phi_b : \mathbb{N}_0 \quad \to \quad \mathbb{Q} \cap [0,1)$$

$$i = \sum_{l=0}^{\infty} a_l(i) b^l \quad \mapsto \quad \Phi_b(i) := \sum_{l=0}^{\infty} a_l(i) b^{-l-1}, \text{ e.g. } \Phi_2(i) \equiv$$

- properties
  - subsequent points that “fall into biggest holes”
  - not completely uniform distributed (CUD)

Quasi-Monte Carlo Points

Radical inversion

- van der Corput sequence in base b

$$\begin{aligned} \Phi_b : \mathbb{N}_0 &\rightarrow \mathbb{Q} \cap [0,1) \\ i = \sum_{l=0}^{\infty} a_l(i) b^l &\mapsto \Phi_b(i) := \sum_{l=0}^{\infty} a_l(i) b^{-l-1}, \text{ e.g. } \Phi_2(i) \equiv \end{aligned}$$

- properties
  - subsequent points that “fall into biggest holes”
  - not completely uniform distributed (CUD)
  - contiguous blocks of stratified points x_i for $kb^m \leq i < (k+1)b^m - 1$
    - for each block the $\Phi_b(i)$ are equidistant
    - for each block the integers $\lfloor b^m \Phi_b(i) \rfloor$ are a **permutation** of $\{0, \ldots, b^m - 1\}$

Quasi-Monte Carlo Points

Radical inversion

- van der Corput sequence in base b

$$\Phi_b : \mathbb{N}_0 \to \mathbb{Q} \cap [0,1)$$

$$i = \sum_{l=0}^{\infty} a_l(i) b^l \mapsto \Phi_b(i) := \sum_{l=0}^{\infty} a_l(i) b^{-l-1}, \text{ e.g. } \Phi_2(i) \equiv$$

- properties
  - subsequent points that “fall into biggest holes”
  - not completely uniform distributed (CUD)
  - contiguous blocks of stratified points x_i for $kb^m \leq i < (k+1)b^m - 1$
    - for each block the $\Phi_b(i)$ are equidistant
    - for each block the integers $\lfloor b^m \Phi_b(i) \rfloor$ are a **permutation** of $\{0, \ldots, b^m - 1\}$

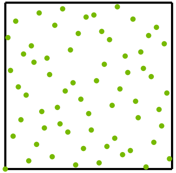
Quasi-Monte Carlo Points

Halton sequence and Hammersley points

- let the b_j be co-prime, for example the j-th prime number

  Halton sequence

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



$$(\Phi_2(i), \Phi_3(i))_{i=0}^{63}$$

  - contiguous blocks of stratified points x_i for $k \prod_{j=1}^{s} b_j^{m_j} \le i < (k+1) \prod_{j=1}^{s} b_j^{m_j} - 1$

Quasi-Monte Carlo Points

Halton sequence and Hammersley points

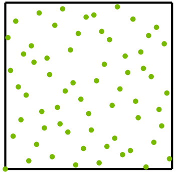
- let the b_j be co-prime, for example the j-th prime number

Halton sequence

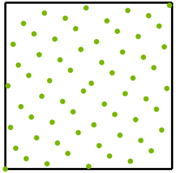
$$x_i := \left(\Phi_{b_1}(i), \ldots, \Phi_{b_s}(i)\right)$$

Hammersley point sets

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



$$\left(\Phi_2(i), \Phi_3(i)\right)_{i=0}^{63}$$



$$\left(\frac{i}{64}, \Phi_2(i)\right)_{i=0}^{63}$$

- contiguous blocks of stratified points x_i for $k \prod_{j=1}^{s} b_j^{m_j} \leq i < (k+1) \prod_{j=1}^{s} b_j^{m_j} - 1$

Quasi-Monte Carlo Points

Halton sequence and Hammersley points

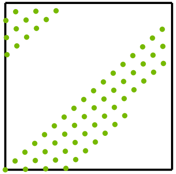
- let the b_j be co-prime, for example the j-th prime number

Halton sequence

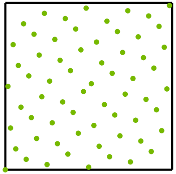
$$x_i := \left(\Phi_{b_1}(i), \ldots, \Phi_{b_s}(i)\right)$$

Hammersley point sets

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



$$\left(\Phi_{17}(i), \Phi_{19}(i)\right)_{i=0}^{63}$$



$$\left(\frac{i}{64}, \Phi_2(i)\right)_{i=0}^{63}$$

  - correlations in low dimensional projections

Quasi-Monte Carlo Points

Halton sequence and Hammersley points

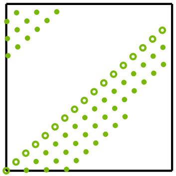
- let the b_j be co-prime, for example the j-th prime number

Halton sequence

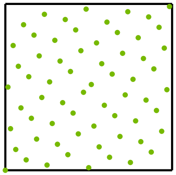
$$x_i := \left(\Phi_{b_1}(i), \dots, \Phi_{b_s}(i)\right)$$

Hammersley point sets

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



$$\left(\Phi_{17}(i), \Phi_{19}(i)\right)_{i=0}^{63}$$



$$\left(\frac{i}{64}, \Phi_2(i)\right)_{i=0}^{63}$$

- correlations in low dimensional projections

Quasi-Monte Carlo Points

Scrambling

- algorithm: start with $H = I^s$ and for each axis j
  1. slice H into b_j equally sized volumes $H_1, H_2, \ldots, H_{b_j}$ along the axis
  2. permute these volumes
  3. for each H_h recursively repeat the procedure with $H = H_h$

Quasi-Monte Carlo Points

Scrambling

- algorithm: start with $H = I^s$ and for each axis j
  1. slice H into b_j equally sized volumes $H_1, H_2, \ldots, H_{b_j}$ along the axis
  2. permute these volumes
  3. for each H_h recursively repeat the procedure with $H = H_h$
- stratification invariant under scrambling
- many variants, simplifications, and generalizations
  - example: unit square $[0,1)^2$

Quasi-Monte Carlo Points

Scrambling

- algorithm: start with $H = I^s$ and for each axis j
  1. slice H into b_j equally sized volumes $H_1, H_2, \ldots, H_{b_j}$ along the axis
  2. permute these volumes
  3. for each H_h recursively repeat the procedure with $H = H_h$
- stratification invariant under scrambling
- many variants, simplifications, and generalizations
  - example: `xor`-scrambling bit 1 of x

Quasi-Monte Carlo Points

Scrambling

- algorithm: start with $H = I^s$ and for each axis j
  1. slice H into b_j equally sized volumes $H_1, H_2, \ldots, H_{b_j}$ along the axis
  2. permute these volumes
  3. for each H_h recursively repeat the procedure with $H = H_h$

- stratification invariant under scrambling

- many variants, simplifications, and generalizations
  - example: `xor`-scrambling bit 2 of x

Quasi-Monte Carlo Points

Scrambling

- algorithm: start with $H = I^s$ and for each axis j
  1. slice H into b_j equally sized volumes $H_1, H_2, \ldots, H_{b_j}$ along the axis
  2. permute these volumes
  3. for each H_h recursively repeat the procedure with $H = H_h$
- stratification invariant under scrambling
- many variants, simplifications, and generalizations
  - example: `xor`-scrambling bit 3 of x

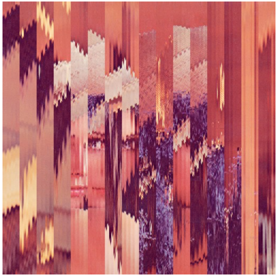
Quasi-Monte Carlo Points

Scrambling

- algorithm: start with $H = I^s$ and for each axis j
  1. slice H into b_j equally sized volumes $H_1, H_2, \ldots, H_{b_j}$ along the axis
  2. permute these volumes
  3. for each H_h recursively repeat the procedure with $H = H_h$

- stratification invariant under scrambling

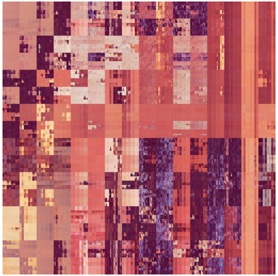
- many variants, simplifications, and generalizations
  - example: `xor`-scrambling all bits of x



Quasi-Monte Carlo Points

Scrambling

- algorithm: start with $H = I^s$ and for each axis j
  1. slice H into b_j equally sized volumes $H_1, H_2, \ldots, H_{b_j}$ along the axis
  2. permute these volumes
  3. for each H_h recursively repeat the procedure with $H = H_h$
- stratification invariant under scrambling
- many variants, simplifications, and generalizations
  - example: `xor`-scrambling all bits of x and y



Quasi-Monte Carlo Points

Scrambled radical inversion

- example: deterministic permutations σ_b by Faure

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

Quasi-Monte Carlo Points

Scrambled radical inversion

- example: deterministic permutations σ_b by Faure

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

  - 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

Quasi-Monte Carlo Points

Scrambled radical inversion

- example: deterministic permutations σ_b by Faure

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

  - 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

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

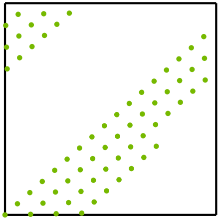
Quasi-Monte Carlo Points

Scrambled radical inversion

- example: deterministic permutations σ_b by Faure

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

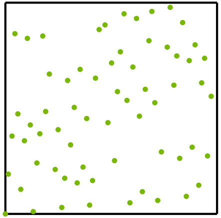
  - b is even: Take $2\sigma_{\frac{b}{2}}$ and append $2\sigma_{\frac{b}{2}}+1$
  - b is odd: Take σ_{b-1}, increment each value $\geq \frac{b-1}{2}$ and insert $\frac{b-1}{2}$ in the middle



$\rightarrow$

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

$\rightarrow$



Quasi-Monte Carlo Points

Efficient generation of the Faure-scrambled radical inverse

```
double RadicalInverse(const int Base, int i)
{
  double Digit, Radical, Inverse = 0.0;
  Digit = Radical = 1.0 / (double) Base;

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

  return Inverse;
}
```

Quasi-Monte Carlo Points

Efficient generation of the Faure-scrambled radical inverse

```
double IntegerRadicalInverse(int Base, int i)
{
  int numPoints, inverse;
  numPoints = 1;

  for(inverse = 0; i > 0; i /= Base)
  {
    inverse = inverse * Base + (i % Base);
    numPoints = numPoints * Base;
  }

  return (double) inverse / (double) numPoints;
}
```

Quasi-Monte Carlo Points

Efficient generation of the Faure-scrambled radical inverse

- compact branchless code using one look-up table for multiple digits
  - example: $\sigma_5 = (0,3,2,1,4)$

$$\sigma_5 \times \sigma_5 = \begin{pmatrix} (0,0) & (0,3) & (0,2) & (0,1) & (0,4) \\ (3,0) & (3,3) & (3,2) & (3,1) & (3,4) \\ (2,0) & (2,3) & (2,2) & (2,1) & (2,4) \\ (1,0) & (1,3) & (1,2) & (1,1) & (1,4) \\ (4,0) & (4,3) & (4,2) & (4,1) & (4,4) \end{pmatrix}$$

Quasi-Monte Carlo Points

Efficient generation of the Faure-scrambled radical inverse

- compact branchless code using one look-up table for multiple digits
  - example: $\sigma_5 = (0,3,2,1,4)$

$$\sigma_5 \times \sigma_5 = \begin{pmatrix} (0,0) & (0,3) & (0,2) & (0,1) & (0,4) \\ (3,0) & (3,3) & (3,2) & (3,1) & (3,4) \\ (2,0) & (2,3) & (2,2) & (2,1) & (2,4) \\ (1,0) & (1,3) & (1,2) & (1,1) & (1,4) \\ (4,0) & (4,3) & (4,2) & (4,1) & (4,4) \end{pmatrix} \cong \begin{pmatrix} 0 & 3 & 2 & 1 & 4 \\ 25 & 28 & 27 & 26 & 29 \\ 10 & 13 & 12 & 11 & 14 \\ 5 & 8 & 7 & 6 & 9 \\ 20 & 23 & 22 & 21 & 24 \end{pmatrix}$$

Quasi-Monte Carlo Points

Efficient generation of the Faure-scrambled radical inverse

- compact branchless code using one look-up table for multiple digits
  - example: $\sigma_5 = (0,3,2,1,4)$, for $b = 5$ and 3 digits, i.e. $\sigma_5 \times \sigma_5 \times \sigma_5$

```
static const unsigned short perm5[] = { 0, 75, 50, 25, 100, 15, 90, 65, 40, 115, 10, 85, 60, 35, 110, 5, 80, 55,
    30, 105, 20, 95, 70, 45, 120, 3, 78, 53, 28, 103, 18, 93, 68, 43, 118, 13, 88, 63, 38, 113, 8, 83, 58, 33, 108,
    23, 98, 73, 48, 123, 2, 77, 52, 27, 102, 17, 92, 67, 42, 117, 12, 87, 62, 37, 112, 7, 82, 57, 32, 107, 22, 97,
    72, 47, 122, 1, 76, 51, 26, 101, 16, 91, 66, 41, 116, 11, 86, 61, 36, 111, 6, 81, 56, 31, 106, 21, 96, 71, 46,
    121, 4, 79, 54, 29, 104, 19, 94, 69, 44, 119, 14, 89, 64, 39, 114, 9, 84, 59, 34, 109, 24, 99, 74, 49, 124 };

inline float halton5(const unsigned index)
{
    return (perm5[index % 125u] * 1953125u +
            perm5[(index / 125u) % 125u] * 15625u +
            perm5[(index / 15625u) % 125u] * 125u +
            perm5[(index / 1953125u) % 125u]) * (0x1.fffffep-1 / 244140625u); // For results < 1.
}
```

Quasi-Monte Carlo Points

(t,s)-sequences and (t,m,s)-nets in base b

- elementary interval

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

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

Quasi-Monte Carlo Points

(t,s)-sequences and (t,m,s)-nets in base b

- elementary interval

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

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

- 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.

Quasi-Monte Carlo Points

(t,s)-sequences and (t,m,s)-nets in base b

- elementary interval

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

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

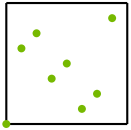
- For two integers $0 \leq t \leq m$, a finite point set of b^m points in s dimensions is called a (t,m,s)-net in base b, if every elementary interval of volume $\lambda_s(E) = b^{t-m}$ contains exactly b^t points.

- For $t \geq 0$, 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 $x_{kb^m}, \ldots, x_{(k+1)b^m-1} \in I^s$ form a (t,m,s)-net.

Quasi-Monte Carlo Points

(t,s)-sequences are sequences of (t,m,s)-nets in base b

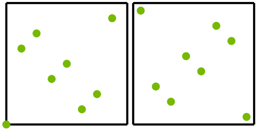
- example: stratification properties of the Sobol' $(0,2)$-sequence in base 2
  - the sequence of $(0,3,2)$-nets



Quasi-Monte Carlo Points

(t, s)-sequences are sequences of (t, m, s)-nets in base b

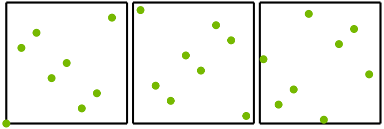
- example: stratification properties of the Sobol' $(0, 2)$-sequence in base 2
  - the sequence of $(0, 3, 2)$-nets



Quasi-Monte Carlo Points

(t,s)-sequences are sequences of (t,m,s)-nets in base b

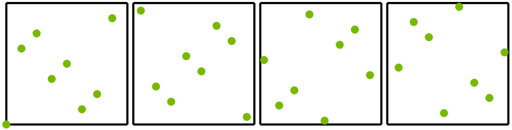
- example: stratification properties of the Sobol' $(0,2)$-sequence in base 2
  - the sequence of $(0,3,2)$-nets



Quasi-Monte Carlo Points

(t,s)-sequences are sequences of (t,m,s)-nets in base b

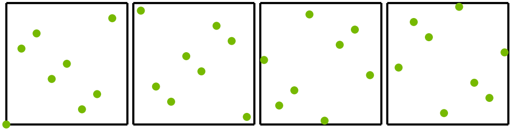
- example: stratification properties of the Sobol' $(0,2)$-sequence in base 2
  - the sequence of $(0,3,2)$-nets



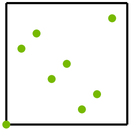
Quasi-Monte Carlo Points

(t,s)-sequences are sequences of (t,m,s)-nets in base b

- example: stratification properties of the Sobol' $(0,2)$-sequence in base 2
  - the sequence of $(0,3,2)$-nets



  - the sequence of $(0,4,2)$-nets

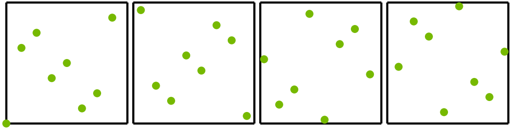


Quasi-Monte Carlo Points

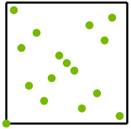
(t,s)-sequences are sequences of (t,m,s)-nets in base b

- example: stratification properties of the Sobol' $(0,2)$-sequence in base 2

  – the sequence of $(0,3,2)$-nets



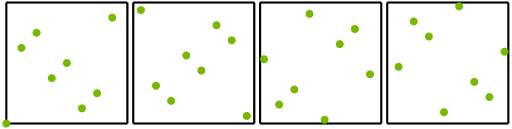
  – the sequence of $(0,4,2)$-nets



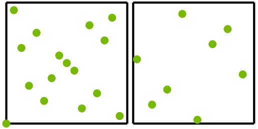
Quasi-Monte Carlo Points

(t,s)-sequences are sequences of (t,m,s)-nets in base b

- example: stratification properties of the Sobol' $(0,2)$-sequence in base 2
  - the sequence of $(0,3,2)$-nets



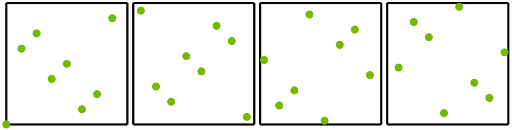
  - the sequence of $(0,4,2)$-nets



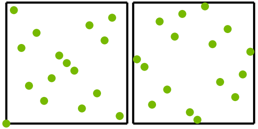
Quasi-Monte Carlo Points

(t, s)-sequences are sequences of (t, m, s)-nets in base b

- example: stratification properties of the Sobol' $(0, 2)$-sequence in base 2
  - the sequence of $(0, 3, 2)$-nets



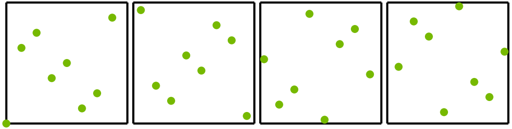
  - the sequence of $(0, 4, 2)$-nets



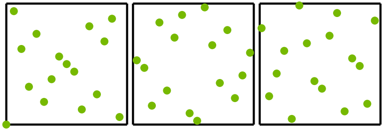
Quasi-Monte Carlo Points

(t,s)-sequences are sequences of (t,m,s)-nets in base b

- example: stratification properties of the Sobol' $(0,2)$-sequence in base 2
  - the sequence of $(0,3,2)$-nets



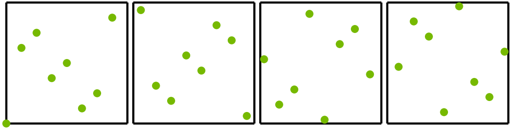
  - the sequence of $(0,4,2)$-nets



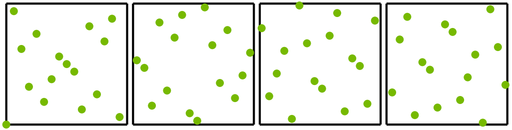
Quasi-Monte Carlo Points

(t,s)-sequences are sequences of (t,m,s)-nets in base b

- example: stratification properties of the Sobol' $(0,2)$-sequence in base 2
  - the sequence of $(0,3,2)$-nets



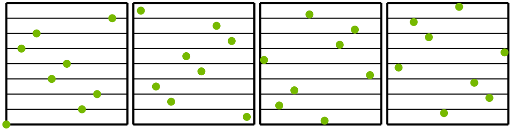
  - the sequence of $(0,4,2)$-nets



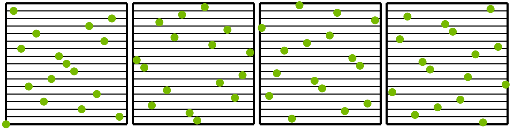
Quasi-Monte Carlo Points

(t,s)-sequences are sequences of (t,m,s)-nets in base b

- example: stratification properties of the Sobol' $(0,2)$-sequence in base 2
  - the sequence of $(0,3,2)$-nets



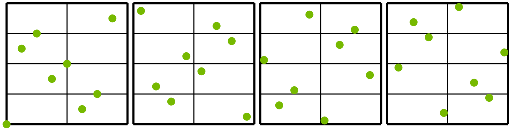
  - the sequence of $(0,4,2)$-nets



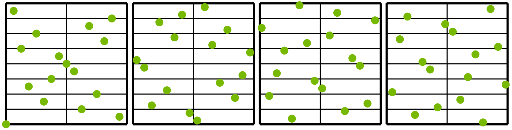
Quasi-Monte Carlo Points

(t,s)-sequences are sequences of (t,m,s)-nets in base b

- example: stratification properties of the Sobol' $(0,2)$-sequence in base 2
  - the sequence of $(0,3,2)$-nets



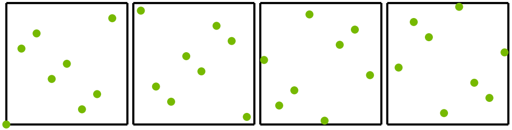
  - the sequence of $(0,4,2)$-nets



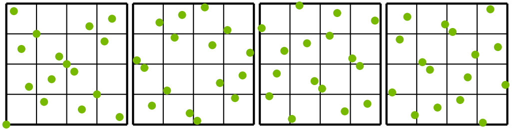
Quasi-Monte Carlo Points

(t,s)-sequences are sequences of (t,m,s)-nets in base b

- example: stratification properties of the Sobol' $(0,2)$-sequence in base 2
  - the sequence of $(0,3,2)$-nets



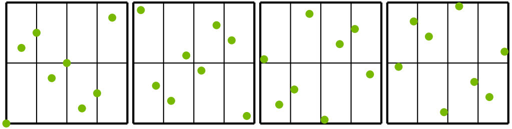
  - the sequence of $(0,4,2)$-nets



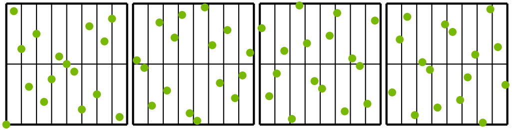
Quasi-Monte Carlo Points

(t,s)-sequences are sequences of (t,m,s)-nets in base b

- example: stratification properties of the Sobol' $(0,2)$-sequence in base 2
  - the sequence of $(0,3,2)$-nets



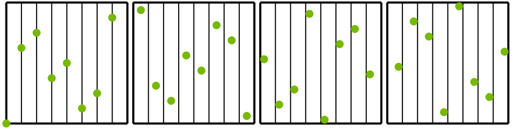
  - the sequence of $(0,4,2)$-nets



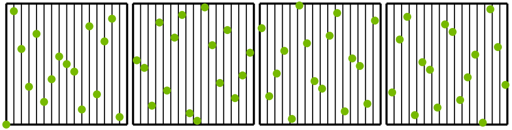
Quasi-Monte Carlo Points

(t,s)-sequences are sequences of (t,m,s)-nets in base b

- example: stratification properties of the Sobol' $(0,2)$-sequence in base 2
  - the sequence of $(0,3,2)$-nets



  - the sequence of $(0,4,2)$-nets



  - all components of the Sobol' sequence are $(0,1)$-sequences in base 2 $\Rightarrow$ deterministic LHS

Quasi-Monte Carlo Points

Digital (t,s)-sequences in base b

- use one $m \times m$ generator matrix C_j for each component

$$x_i^{(j)} = \begin{pmatrix} b^{-1} \\ \vdots \\ b^{-m} \end{pmatrix}^T \underbrace{C_j \begin{pmatrix} a_0(i) \\ \vdots \\ a_{m-1}(i) \end{pmatrix}}_{\text{multiplication in } F_b}$$

▶ Sobol sequence generator matrices

▶ Fast Sobol' sequence generator (including pixel enumeration), inverse matrices, and Faure scrambled Halton sampler

Quasi-Monte Carlo Points

Digital (t, s)-sequences in base b

- use one $m \times m$ generator matrix C_j for each component

$$x_i^{(j)} = \begin{pmatrix} b^{-1} \\ \vdots \\ b^{-m} \end{pmatrix}^T \underbrace{C_j \begin{pmatrix} a_0(i) \\ \vdots \\ a_{m-1}(i) \end{pmatrix}}_{\text{multiplication in } F_b}$$

```
double x_base_2(uint i, uint r = 0)
{
  for (uint k = 0; i; i >>= 1, ++k)
    if (i & 1)
      r ^= C[k]; // SIMD addition of column

  return (double) r / (double) (1 << m);
}
```

- optimized implementation similar to scrambled radical inverse as before

▶ Sobol sequence generator matrices

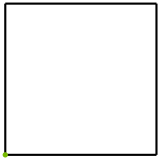
▶ Fast Sobol' sequence generator (including pixel enumeration), inverse matrices, and Faure scrambled Halton sampler

Quasi-Monte Carlo Points

Rank-1 lattices

- given generator vector $(g_0, \ldots, g_{s-1}) \in \mathbb{N}^s$

$$x_i := \frac{i}{n}(g_0, \ldots, g_{s-1}) \bmod [0,1)^s$$

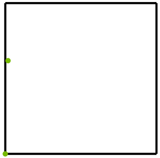


Quasi-Monte Carlo Points

Rank-1 lattices

- given generator vector $(g_0, \ldots, g_{s-1}) \in \mathbb{N}^s$

$$x_i := \frac{i}{n}(g_0, \ldots, g_{s-1}) \bmod [0,1)^s$$

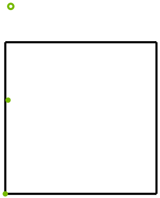


Quasi-Monte Carlo Points

Rank-1 lattices

- given generator vector $(g_0, \ldots, g_{s-1}) \in \mathbb{N}^s$

$$x_i := \frac{i}{n}(g_0, \ldots, g_{s-1}) \bmod [0,1)^s$$

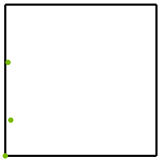


Quasi-Monte Carlo Points

Rank-1 lattices

- given generator vector $(g_0, \ldots, g_{s-1}) \in \mathbb{N}^s$

$$x_i := \frac{i}{n}(g_0, \ldots, g_{s-1}) \bmod [0,1)^s$$

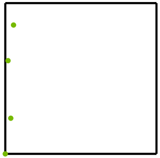


Quasi-Monte Carlo Points

Rank-1 lattices

- given generator vector $(g_0, \ldots, g_{s-1}) \in \mathbb{N}^s$

$$x_i := \frac{i}{n}(g_0, \ldots, g_{s-1}) \bmod [0,1)^s$$

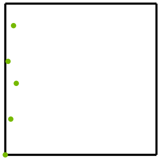


Quasi-Monte Carlo Points

Rank-1 lattices

- given generator vector $(g_0, \ldots, g_{s-1}) \in \mathbb{N}^s$

$$x_i := \frac{i}{n}(g_0, \ldots, g_{s-1}) \bmod [0,1)^s$$



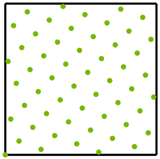
NVIDIA

Quasi-Monte Carlo Points

Rank-1 lattices

- given generator vector $(g_0, \ldots, g_{s-1}) \in \mathbb{N}^s$

$$x_i := \frac{i}{n}(g_0, \ldots, g_{s-1}) \bmod [0,1)^s$$

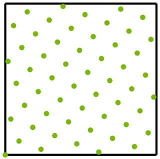


Quasi-Monte Carlo Points

Rank-1 lattices

- given generator vector $(g_0, \ldots, g_{s-1}) \in \mathbb{N}^s$

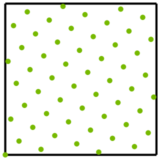
$$x_i := \frac{i}{n}(g_0, \ldots, g_{s-1}) \bmod [0,1)^s$$



- generator vectors
  - Korobov form $(1, a, a^2, a^3, \ldots)$
  - rare constructions
    - example: Fibonacci lattice with $n = F_k$ and $(g_0, g_1) = (1, F_{k-1})$

Quasi-Monte Carlo Points

Rank-1 lattices



- given generator vector $(g_0, \ldots, g_{s-1}) \in \mathbb{N}^s$

$$x_i := \frac{i}{n}(g_0, \ldots, g_{s-1}) \bmod [0,1)^s$$

- generator vectors
  - Korobov form $(1, a, a^2, a^3, \ldots)$
  - rare constructions
    - example: Fibonacci lattice with $n = F_k$ and $(g_0, g_1) = (1, F_{k-1})$
  - usually tabulated coefficients a or g_j
    - search by certain criteria, e.g. maximized minimum distance, projections, ...
    - component by component construction (CBC)

Quasi-Monte Carlo Points

Rank-1 lattice sequences

- replace $\frac{i}{n}$ by radical inverse

$$x_i = \phi_b(i) \cdot (g_0, \ldots, g_{s-1}) \bmod [0,1)^s$$

Quasi-Monte Carlo Points

Rank-1 lattice sequences

- replace $\frac{i}{n}$ by radical inverse

$$x_i = \phi_b(i) \cdot (g_0, \ldots, g_{s-1}) \bmod [0,1)^s, \text{ where } g_j =_b \cdots g_{j,3}g_{j,2}g_{j,1}g_{j,0} \text{ are infinite sequences of digits}$$

Quasi-Monte Carlo Points

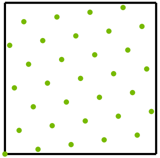
Rank-1 lattice sequences

- replace $\frac{i}{n}$ by radical inverse

$$x_i = \phi_b(i) \cdot (g_0, \ldots, g_{s-1}) \bmod [0,1)^s, \text{ where } g_j =_b \cdots g_{j,3} g_{j,2} g_{j,1} g_{j,0} \text{ are infinite sequences of digits}$$

- $\vec{x}_{kb^m}, \ldots, \vec{x}_{(k+1)b^m-1}$ form a shifted lattice
  - shift Δ in the $k+1$st block for $n = b^m$

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



Quasi-Monte Carlo Points

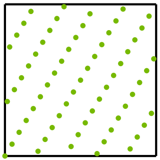
Rank-1 lattice sequences

- replace $\frac{i}{n}$ by radical inverse

$$x_i = \phi_b(i) \cdot (g_0, \ldots, g_{s-1}) \bmod [0,1)^s, \text{ where } g_j =_b \cdots g_{j,3} g_{j,2} g_{j,1} g_{j,0} \text{ are infinite sequences of digits}$$

- $\vec{x}_{kb^m}, \ldots, \vec{x}_{(k+1)b^m-1}$ form a shifted lattice
  - shift Δ in the $k+1$st block for $n = b^m$

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



Quasi-Monte Carlo Points

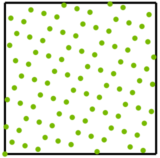
Rank-1 lattice sequences

- replace $\frac{i}{n}$ by radical inverse

$$x_i = \phi_b(i) \cdot (g_0, \ldots, g_{s-1}) \bmod [0,1)^s, \text{ where } g_j =_b \cdots g_{j,3} g_{j,2} g_{j,1} g_{j,0} \text{ are infinite sequences of digits}$$

- $\vec{x}_{kb^m}, \ldots, \vec{x}_{(k+1)b^m-1}$ form a shifted lattice
  - shift Δ in the $k+1$st block for $n = b^m$

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



Quasi-Monte Carlo Points

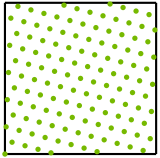
Rank-1 lattice sequences

- replace $\frac{i}{n}$ by radical inverse

$$x_i = \phi_b(i) \cdot (g_0, \ldots, g_{s-1}) \bmod [0,1)^s, \text{ where } g_j =_b \cdots g_{j,3} g_{j,2} g_{j,1} g_{j,0} \text{ are infinite sequences of digits}$$

- $\vec{x}_{kb^m}, \ldots, \vec{x}_{(k+1)b^m-1}$ form a shifted lattice
  - shift Δ in the $k+1$st block for $n = b^m$

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



- similar to (t,s)-sequences
  - for b and g_j relatively prime, $\phi_b(i) g_j \bmod [0,1)$ are $(0,1)$-sequences

▶ Lattice rule generating vectors

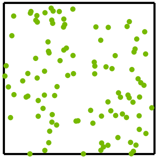
▶ Construction of a rank-1 lattice sequence based on primitive polynomials

Light transport simulation using a rank-1 lattice sequence based on primitive polynomials

Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \dots, x_{n-1}\} \in [0,1)^s$

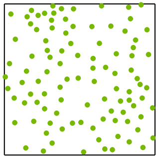
- maximum minimum distance $d_{\min}(P_n) := \min_{0 \le i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T



Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

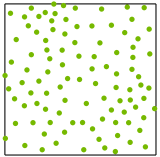
- maximum minimum distance $d_{\min}(P_n) := \min_{0 \leq i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T



Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

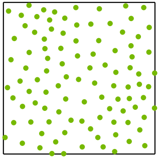
- maximum minimum distance $d_{\min}(P_n) := \min_{0 \le i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T



Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

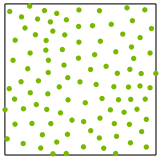
- maximum minimum distance $d_{\min}(P_n) := \min_{0 \le i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T



Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

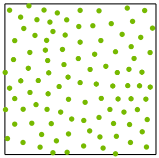
- maximum minimum distance $d_{\min}(P_n) := \min_{0 \le i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T



Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

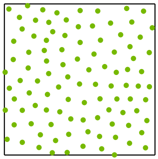
- maximum minimum distance $d_{\min}(P_n) := \min_{0 \leq i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T



Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

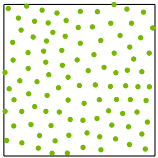
- maximum minimum distance $d_{\min}(P_n) := \min_{0 \le i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T



Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

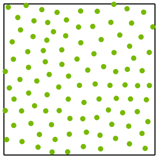
- maximum minimum distance $d_{\min}(P_n) := \min_{0 \le i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T



Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

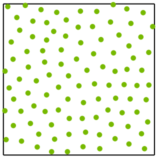
- maximum minimum distance $d_{\min}(P_n) := \min_{0 \le i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T



Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

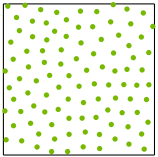
- maximum minimum distance $d_{\min}(P_n) := \min_{0 \le i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T



Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

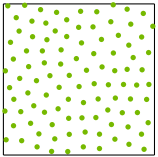
- maximum minimum distance $d_{\min}(P_n) := \min_{0 \le i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T



Quasi-Monte Carlo Points

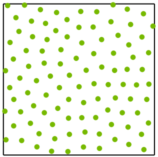
Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

- maximum minimum distance $d_{\min}(P_n) := \min_{0 \le i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T



Quasi-Monte Carlo Points

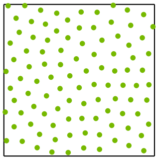
Uniformity of a point set $P_n := \{x_0, \dots, x_{n-1}\} \in [0,1)^s$



- maximum minimum distance $d_{\min}(P_n) := \min_{0 \le i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T

Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

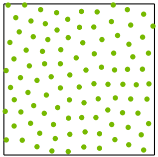


- maximum minimum distance $d_{\min}(P_n) := \min_{0 \le i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T

Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

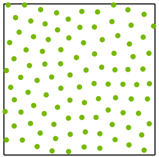
- maximum minimum distance $d_{\min}(P_n) := \min_{0 \le i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T



Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

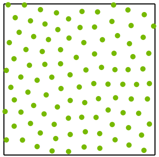
- maximum minimum distance $d_{\min}(P_n) := \min_{0 \leq i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T



Quasi-Monte Carlo Points

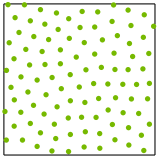
Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

- maximum minimum distance $d_{\min}(P_n) := \min_{0 \le i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T



Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

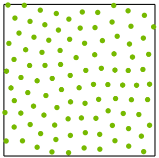


- maximum minimum distance $d_{\min}(P_n) := \min_{0 \le i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T

Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

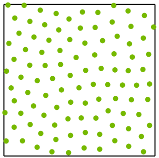
- maximum minimum distance $d_{\min}(P_n) := \min_{0 \leq i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T



Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

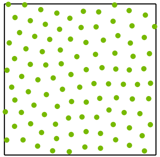
- maximum minimum distance $d_{\min}(P_n) := \min_{0 \le i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T



Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

- maximum minimum distance $d_{\min}(P_n) := \min_{0 \le i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T



Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

- maximum minimum distance $d_{\min}(P_n) := \min_{0 \le i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T
- low discrepancy

$$D^*(P_n) := \sup_{A = \prod_{j=1}^{s} [0, a_j) \subseteq [0,1)^s} \left| \int_{[0,1)^s} \chi_A(x) dx - \frac{1}{n} \sum_{i=0}^{n-1} \chi_A(x_i) \right| \in \mathcal{O}\left(\frac{\log^s n}{n} \right)$$

Quasi-Monte Carlo Points

Uniformity of a point set $P_n := \{x_0, \ldots, x_{n-1}\} \in [0,1)^s$

- maximum minimum distance $d_{\min}(P_n) := \min_{0 \le i < n} \min_{i < j < n} \|x_j - x_i\|_T$ on torus T

- low discrepancy

$$D^*(P_n) := \sup_{A = \prod_{j=1}^{s} [0, a_j) \subseteq [0,1)^s} \left| \int_{[0,1)^s} \chi_A(x) dx - \frac{1}{n} \sum_{i=0}^{n-1} \chi_A(x_i) \right| \in \mathcal{O}\left(\frac{\log^s n}{n} \right)$$

- Let $(X, \mathscr{B}, \mu)$ be an arbitrary probability space and let $\mathscr{M}$ be a nonempty subset of $\mathscr{B}$. A point set P_n of n elements of X is called $(\mathscr{M}, \mu)$-uniform if

$$\sum_{i=0}^{n-1} \chi_M(\vec{x}_i) = \mu(M) \cdot n \qquad \text{for all } M \in \mathscr{M},$$

where $\chi_M(\vec{x}_i) = 1$ if $\vec{x}_i \in M$, zero otherwise.

Quasi-Monte Carlo Points

Error bounds depend on function classes

- Lipschitz continuous functions

$$\left| \int_{[0,1]^s} f(x)dx - \frac{1}{n}\sum_{i=0}^{n-1} f(x_i) \right| \leq L \cdot r(n,g)$$

  - maximum minimum distance $r(n,g)$ of rank-1 lattice

Quasi-Monte Carlo Points

Error bounds depend on function classes

- Lipschitz continuous functions

$$\left| \int_{[0,1]^s} f(x)dx - \frac{1}{n}\sum_{i=0}^{n-1} f(x_i) \right| \leq L \cdot r(n,g)$$

  – maximum minimum distance $r(n,g)$ of rank-1 lattice

- Koksma-Hlawka inequality for functions of bounded variation

$$\left| \int_{[0,1)^s} f(x)dx - \frac{1}{n}\sum_{i=0}^{n-1} f(x_i) \right| \leq V(f)D^*(P_n)$$

  – variation often unbounded in practical settings

Quasi-Monte Carlo Points

Error bounds depend on function classes

- Lipschitz continuous functions

$$\left| \int_{[0,1]^s} f(x)dx - \frac{1}{n}\sum_{i=0}^{n-1} f(x_i) \right| \leq L \cdot r(n,g)$$

  - maximum minimum distance $r(n,g)$ of rank-1 lattice

- Koksma-Hlawka inequality for functions of bounded variation

$$\left| \int_{[0,1)^s} f(x)dx - \frac{1}{n}\sum_{i=0}^{n-1} f(x_i) \right| \leq V(f)D^*(P_n)$$

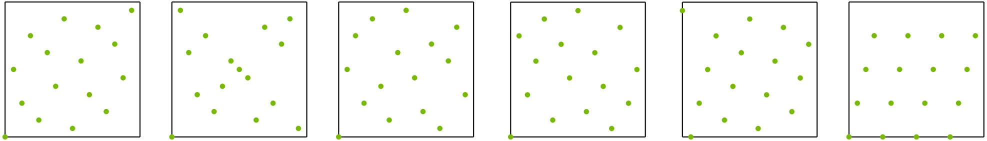
  - variation often unbounded in practical settings

- functions with sufficiently fast vanishing Fourier coefficients
  - another bound for rank-1 lattices

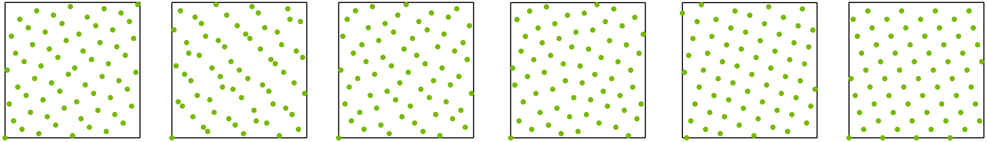
Quasi-Monte Carlo Points

More uniform than random points can be

Hammersley	Sobol‘	L.-P’hammer	opt. $(0, m, 2)$	Perm.-nets	Rank-1 lattice



$d_{min} = 0.0884$	$d_{min} = 0.0884$	$d_{min} = 0.1768$	$d_{min} = 0.2253$	$d_{min} = 0.2253$	$d_{min} = 0.25$



$d_{min} = 0.0221$	$d_{min} = 0.0442$	$d_{min} = 0.0884$	$d_{min} = 0.1127$	$d_{min} = 0.1138$	$d_{min} = 0.1259$

Quasi-Monte Carlo Points

Searching for (t,m,s)-nets in base $b = 2$

- verifying the $t = 0$ property for a point with integer coordinates $(i,j) \in [0,2^m)^2$

```
for (k = 1; k < m; k++)
{
  // combine k bits of i and m-k bits of j to form index
  idx = (i >> (m - k)) + (j & (0xFFFFFFFF << k));

  if(elementaryInterval[k][idx]++) // already one point there?
    break;  // t > 0 !
}
```

▶ (t,m,s)-Nets and Maximized Minimum Distance

▶ (t,m,s)-Nets and Maximized Minimum Distance, Part II

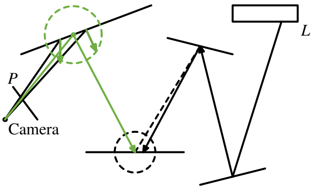
Questions over Questions

Low- or High-Dimensional?

Light transport simulation

- ways to formulate the radiance L_r reflected in a surface point x

$$
\begin{aligned}
& L_r(x, \omega_r) \\
& \quad = \int_{\mathscr{S}_-^2(x)} L_i(x, \omega) f_r(\omega_r, x, \omega) \cos\theta_x \, d\omega
\end{aligned}
$$

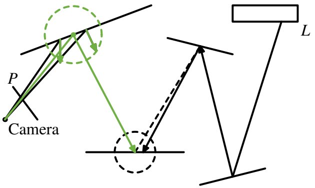


Low- or High-Dimensional?

Light transport simulation

- ways to formulate the radiance L_r reflected in a surface point x

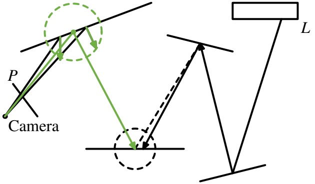
$$
\begin{aligned}
L_r(x, \omega_r) \\
&= \int_{\mathscr{S}_-^2(x)} L_i(x, \omega) f_r(\omega_r, x, \omega) \cos\theta_x \, d\omega \\
&= \lim_{r(x)\to 0} \int_{\mathscr{S}_-^2(x)} \frac{\int_{B(x)} w(x, x') L_i(x', \omega) dx'}{\int_{B(x)} w(x, x') dx'} f_r(\omega_r, x, \omega) \cos\theta_x \, d\omega
\end{aligned}
$$



Low- or High-Dimensional?

Light transport simulation

- ways to formulate the radiance L_r reflected in a surface point x

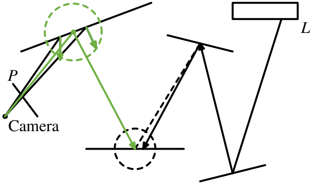


$$
\begin{aligned}
L_r(x,\omega_r) &= \int_{\mathcal{S}^2_-(x)} L_i(x,\omega) f_r(\omega_r,x,\omega)\cos\theta_x \, d\omega \\
&= \lim_{r(x)\to 0} \int_{\mathcal{S}^2_-(x)} \frac{\int_{B(x)} w(x,x') L_i(x',\omega)\, dx'}{\int_{B(x)} w(x,x')\, dx'} f_r(\omega_r,x,\omega)\cos\theta_x \, d\omega \\
&= \int_{\partial V} V(x,y) L_i(x,\omega) f_r(\omega_r,x,\omega)\cos\theta_x \frac{\cos\theta_y}{|x-y|^2} \, dy
\end{aligned}
$$

Low- or High-Dimensional?

Light transport simulation

- ways to formulate the radiance L_r reflected in a surface point x

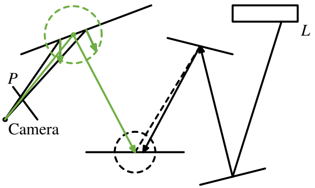


$$
\begin{aligned}
& L_r(x,\omega_r) \\
= & \int_{\mathscr{S}^2_-(x)} L_i(x,\omega) f_r(\omega_r,x,\omega) \cos\theta_x \, d\omega \\
= & \lim_{r(x)\to 0} \int_{\mathscr{S}^2_-(x)} \frac{\int_{B(x)} w(x,x') L_i(x',\omega) dx'}{\int_{B(x)} w(x,x') dx'} f_r(\omega_r,x,\omega) \cos\theta_x \, d\omega \\
= & \int_{\partial V} V(x,y) L_i(x,\omega) f_r(\omega_r,x,\omega) \cos\theta_x \frac{\cos\theta_y}{|x-y|^2} dy \\
= & \lim_{r(x)\to 0} \int_{\partial V} \int_{\mathscr{S}^2_-(y)} \frac{\chi_B(x-h(y,\omega))}{\pi r(x)^2} L_i(h(y,\omega),\omega) f_r(\omega_r,h(y,\omega),\omega) \cos\theta_y \, d\omega \, dy
\end{aligned}
$$

Low- or High-Dimensional?

Light transport simulation

- ways to formulate the radiance L_r reflected in a surface point x



$$
\begin{aligned}
L_r(x,\omega_r) &= \int_{\mathscr{S}_-^2(x)} L_i(x,\omega) f_r(\omega_r,x,\omega)\cos\theta_x\, d\omega \\
&= \lim_{r(x)\to 0} \int_{\mathscr{S}_-^2(x)} \frac{\int_{B(x)} w(x,x') L_i(x',\omega)\, dx'}{\int_{B(x)} w(x,x')\, dx'} f_r(\omega_r,x,\omega)\cos\theta_x\, d\omega \\
&= \int_{\partial V} V(x,y) L_i(x,\omega) f_r(\omega_r,x,\omega)\cos\theta_x \frac{\cos\theta_y}{|x-y|^2}\, dy \\
&= \lim_{r(x)\to 0} \int_{\partial V} \int_{\mathscr{S}_-^2(y)} \frac{\chi_B(x-h(y,\omega))}{\pi r(x)^2} L_i(h(y,\omega),\omega) f_r(\omega_r,h(y,\omega),\omega)\cos\theta_y\, d\omega\, dy
\end{aligned}
$$

- actually an integro-approximation problem: Integrals depend on x and reflection direction ω_r

Low- or High-Dimensional?

Light transport simulation

- radiance L is light sources L_e plus transported radiance $T_f L$

$$L = L_e + T_f L$$

Low- or High-Dimensional?

Light transport simulation

- radiance L is light sources L_e plus transported radiance $T_f L$

$$L = L_e + T_f L = \sum_{i=0}^{\infty} T^i L_e$$

Low- or High-Dimensional?

Light transport simulation

- radiance L is light sources L_e plus transported radiance $T_f L$

$$L = L_e + T_f L = \sum_{i=0}^{\infty} T^i L_e$$

- reinforcement learning to compute low-dimensional approximation

$$L'_c = (1-\alpha) L_c + \alpha (L_e + T_f L_c)$$

Low- or High-Dimensional?

Light transport simulation

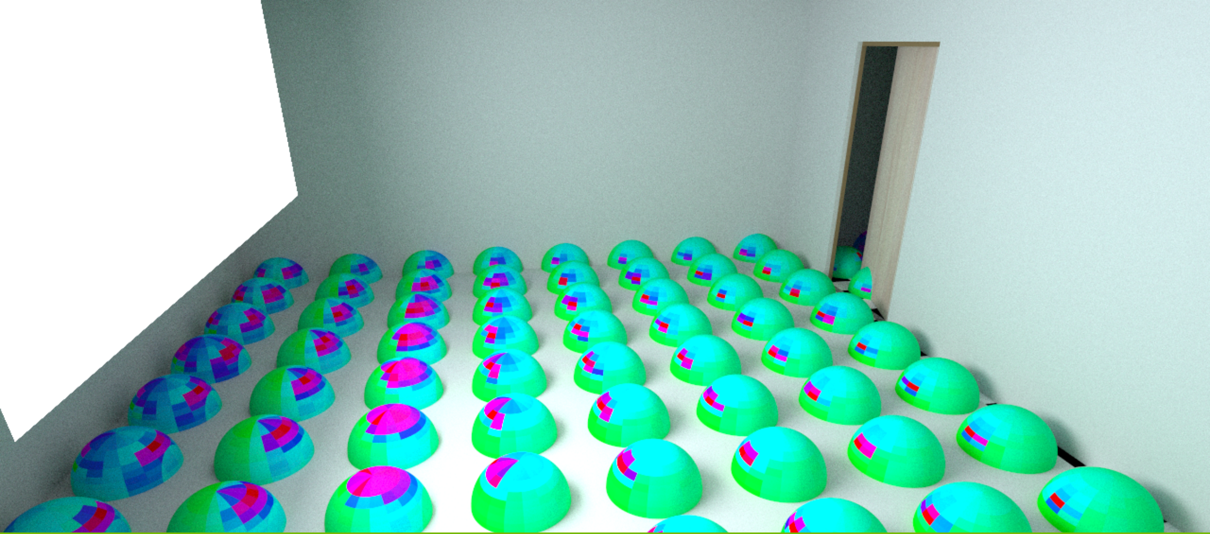
- radiance L is light sources L_e plus transported radiance $T_f L$

$$L = L_e + T_f L = \sum_{i=0}^{\infty} T^i L_e$$

- reinforcement learning to compute low-dimensional approximation

$$L'_c = (1-\alpha)L_c + \alpha(L_e + T_f L_c)$$

  to guide high-dimensional paths
  - itself using approximation instead of tracing paths with higher variance



approximate solution Q stored on discretized hemispheres across scene surface

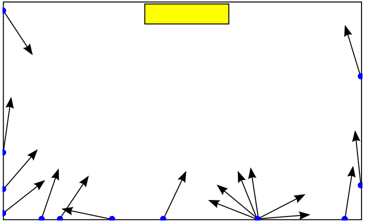
2048 paths traced with BRDF importance sampling in a scene with challenging visibility

Path tracing with online reinforcement learning at the same number of paths

Low- or High-Dimensional?

Simultaneous Simulation of Markov Chains

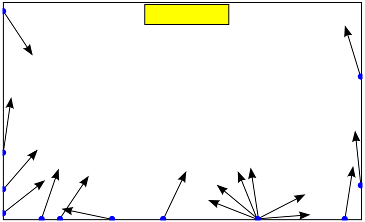
- reordering to benefit from uniformity

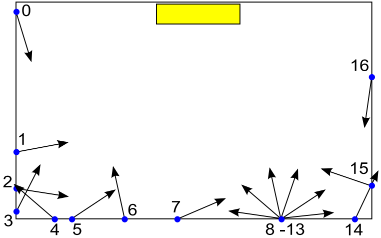


Low- or High-Dimensional?

Simultaneous Simulation of Markov Chains

- reordering to benefit from uniformity





- algorithm
  - simultaneously trace multiple paths bounce by bounce
  - enumerate points along route of proximity (e.g. Z-curve) to make sub-sequence property work

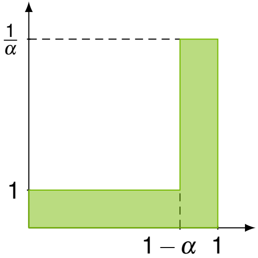
What do you want to see?

Anti-aliasing

- given $\alpha \in (0, 1]$, integrating

$$f(x) = \begin{cases} 1 & x < 1 - \alpha \\ \frac{1}{\alpha} & \text{else} \end{cases}$$

seems simple



What do you want to see?

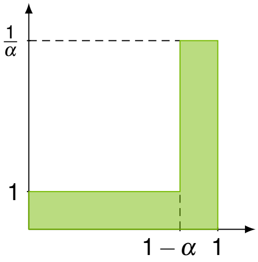
Anti-aliasing

- given $\alpha \in (0,1]$, integrating

$$f(x) = \begin{cases} 1 & x < 1-\alpha \\ \frac{1}{\alpha} & \text{else} \end{cases}$$

seems simple

$$\int_0^1 f(x)dx \quad = \quad (1-\alpha)\cdot 1 + \alpha\frac{1}{\alpha} = 2-\alpha$$



What do you want to see?

Anti-aliasing

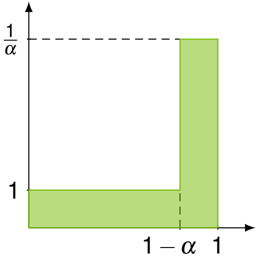
- given $\alpha \in (0,1]$, integrating

$$f(x) = \begin{cases} 1 & x < 1-\alpha \\ \frac{1}{\alpha} & \text{else} \end{cases}$$

seems simple

$$\int_0^1 f(x)dx \quad = \quad (1-\alpha)\cdot 1 + \alpha\frac{1}{\alpha} = 2-\alpha \approx \frac{1}{n}\sum_{i=0}^{n-1} f(x_i)$$

but numerical integration becomes increasingly difficult for $a \to 0$



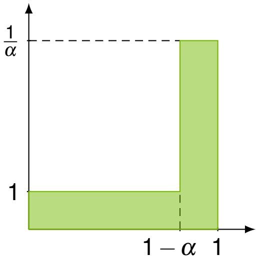
What do you want to see?

Anti-aliasing

- given $\alpha \in (0,1]$, integrating

$$f(x) = \begin{cases} 1 & x < 1-\alpha \\ \frac{1}{\alpha} & \text{else} \end{cases}$$

seems simple



$$\int_0^1 f(x)dx \quad = \quad (1-\alpha)\cdot 1 + \alpha\frac{1}{\alpha} = 2-\alpha \approx \frac{1}{n}\sum_{i=0}^{n-1} f(x_i)$$

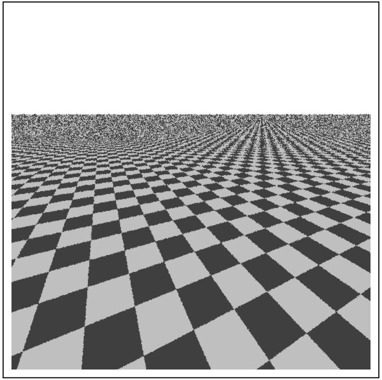
but numerical integration becomes increasingly difficult for $a \to 0$

- example: each sample $f(x_i)$ of brightness $\frac{1}{\alpha} = 10^{26}$ (e.g. the sun) requires at least $n \sim 10^{26}$ more samples to average out

What do you want to see?

Anti-aliasing

- 1 random sample per pixel

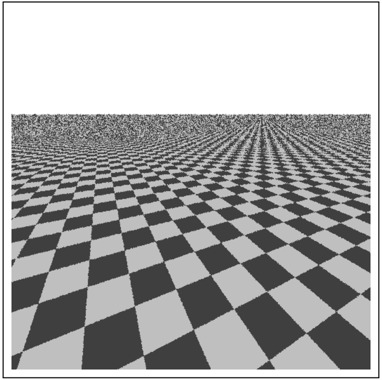


  - artifacts covered by noise

What do you want to see?

Anti-aliasing

- 1 random sample per pixel

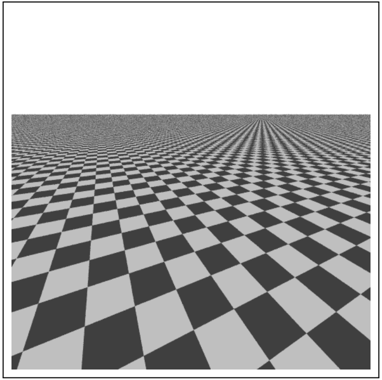


- artifacts covered by noise
- however, freckled edges

What do you want to see?

Anti-aliasing

- 16 random samples per pixel

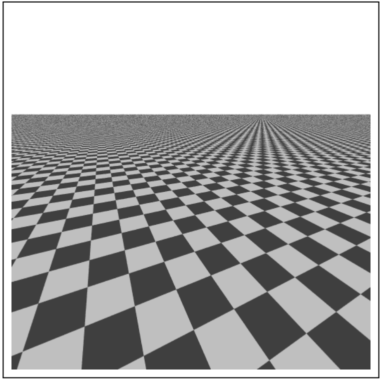


- slower
- reduced variance
- looks better

What do you want to see?

Anti-aliasing

- 4×4 stratified random samples per pixel

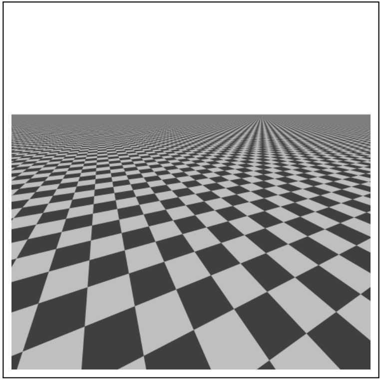


– often converges faster

What do you want to see?

Anti-aliasing

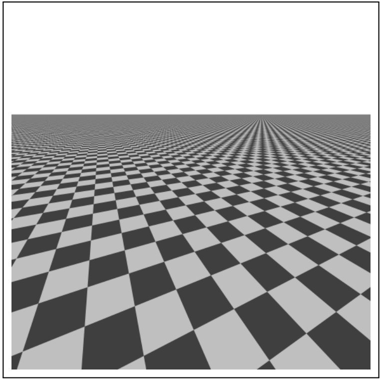
- 1024 $\times$ 1024 stratified random samples per pixel



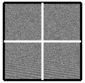
What do you want to see?

Anti-aliasing

- 1024 × 1024 stratified random samples per pixel, looking at 2 × 2 pixels



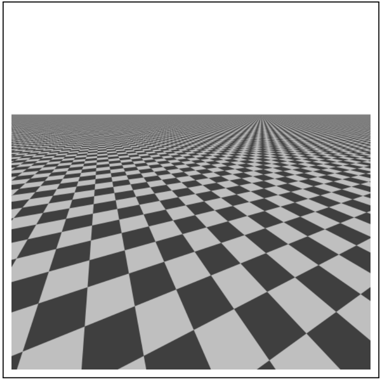
- at the horizon



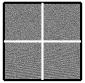
What do you want to see?

Anti-aliasing

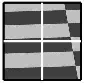
- 1024×1024 stratified random samples per pixel, looking at 2×2 pixels



- at the horizon



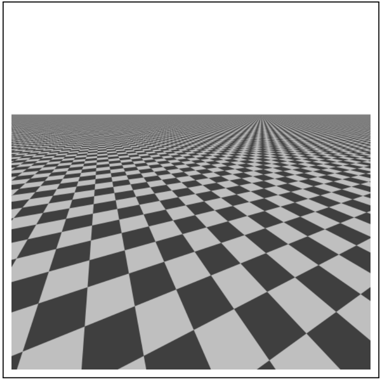
- in the middle



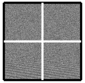
What do you want to see?

Anti-aliasing

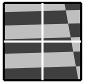
- 1024 $\times$ 1024 stratified random samples per pixel, looking at 2 $\times$ 2 pixels



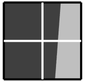
  - at the horizon



  - in the middle



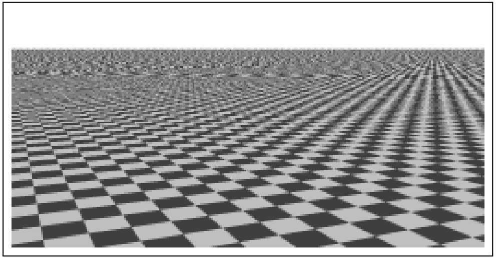
  - in the front

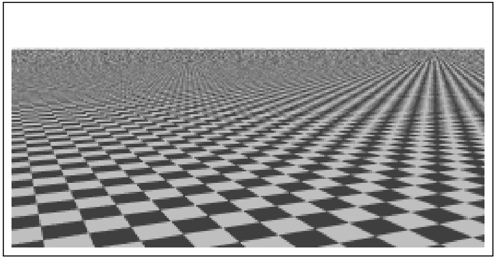


What do you want to see?

Anti-aliasing

- isotropic vs. anisotropic rank-1 lattices select by project normal



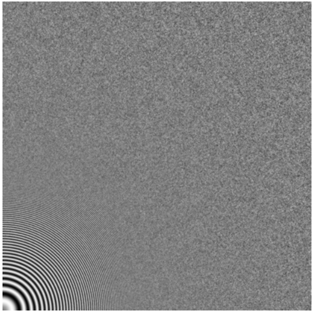


▶ Efficient Search for Two-Dimensional Rank-1 Lattices with Applications in Graphics

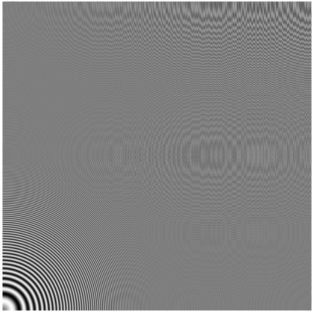
Images or Pixels?

Independence of pixels vs. independence of samples

- anti-aliasing a zone plate at 4 samples per pixel



jittered sampling



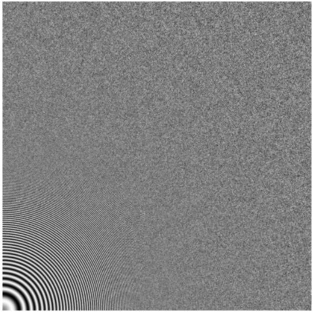
(t, s)-sequence

rank-1 lattice

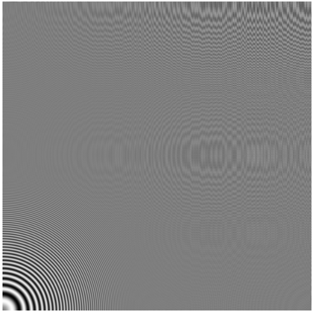
Images or Pixels?

Independence of pixels vs. independence of samples

- anti-aliasing a zone plate at 4 samples per pixel



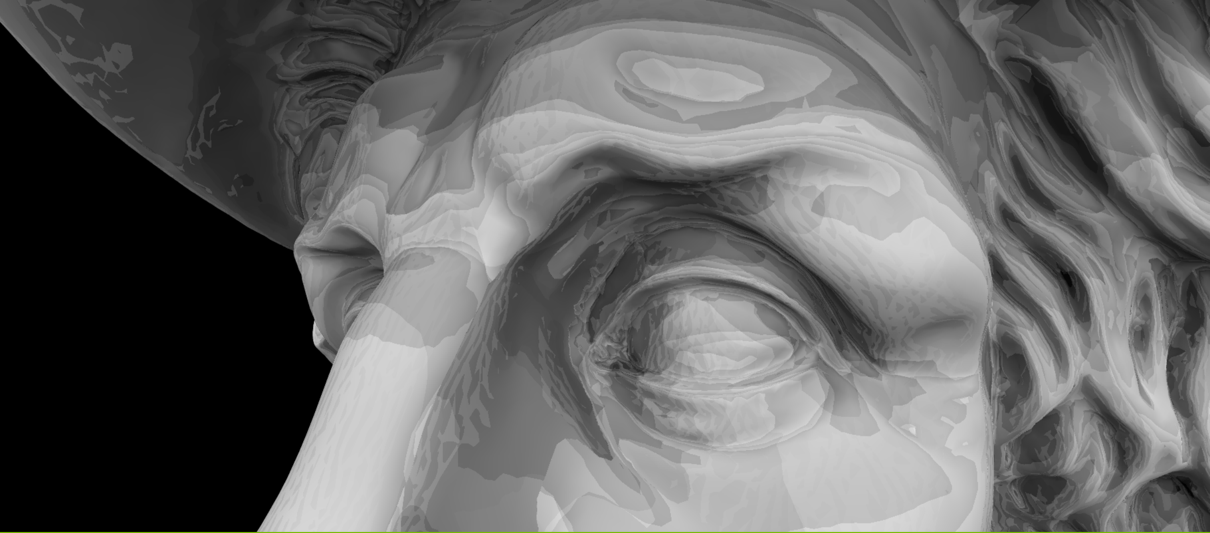
jittered sampling



(t, s)-sequence

rank-1 lattice

  - error bounds depend on a function class



Ambient occlusion at 16 rank-1 lattice samples per pixel

Ambient occlusion at 16 random samples per pixel

Ambient occlusion at 16 rank-1 lattice samples per pixel with Cranley-Patterson-rotation

My favorite Samples

My favorite Samples

Quasi-Monte Carlo points

- deterministic low discrepancy sequences
  - especially rank-1 lattice sequences

► proceedings of the MCQMC conference series
► Quasi-Monte Carlo image synthesis in a nutshell
► Myths of Computer Graphics
► The Iray light transport simulation and rendering system

My favorite Samples

Quasi-Monte Carlo points

- deterministic low discrepancy sequences
  - especially rank-1 lattice sequences

▶ proceedings of the MCQMC conference series

▶ Quasi-Monte Carlo image synthesis in a nutshell

▶ Myths of Computer Graphics

▶ The Iray light transport simulation and rendering system

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

Harald Niederreiter, Claremont, 1998.

My favorite Samples

Schedule

- 9:40 Progressive Multi-Jittered Sequences
  - Per Christensen, Pixar
- 10:15 Warp and Effect
  - Matt Pharr, NVIDIA
- break
- 11:05 Low-Discrepancy Blue Noise Sampling
  - Abdalla Ahmed, King Abdulla University and Victor Ostromoukhov, Université Claude Bernard Lyon 1
- 11:40 Blue-Noise Dithered Sampling
  - Iliyan Georgiev, Autodesk
- check `https://sites.google.com/view/myfavoritesamples`