Internbericht

Monte Carlo & Beyond
– Course Material –

A. Keller

320/02
Monte Carlo & Beyond
– Course Material –

A. Keller

320/02

Universität Kaiserslautern
AG Numerische Algorithmen
Postfach 30 49
67653 Kaiserslautern
Germany

Februar 2002

Herausgeber: AG Numerische Algorithmen
Leiter: Professor Dr. S. Heinrich
Monte Carlo and Beyond

Alexander Keller
keller@informatik.uni-kl.de

Numerical Algorithms Group
Dept. of Computer Science
University of Kaiserslautern

Monte Carlo and Beyond

- MC: Monte Carlo
  - random sampling

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

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

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

Applications in Computer Graphics

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

- QMC: Derandomized RenderMan
  - new graphics hardware

- RQMC: Ocean wave synthesis
  - discrete Fourier transform independent of dimension
  - error estimation for bidirectional path tracing
  - simpler algorithms

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

Reengineering the Classics of Computer Graphics

- Uncorrelated sampling
  - correlated sampling more efficient

- Uniformity is sufficient
  - low-discrepancy sampling more efficient

- Either stratification or Latin hypercube sampling
  - you can have both and even more...

- One dimensional stratified Monte Carlo integration
  - Cranley-Patterson rotations more efficient

- Antialiasing only by random sampling
  - deterministic low-discrepancy sampling more efficient

Monte Carlo and Beyond

- Day 1
  - computer graphics and visual effects
  - principles of rendering algorithms

- Day 2
  - Monte Carlo integration
  - quasi-Monte Carlo points

- Day 3
  - quasi-Monte Carlo integration
  - Monte Carlo extensions of quasi-Monte Carlo

- Day 4
  - application to computer graphics

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

- no real random on classical deterministic computers
- real random by measuring quantum registers
A Brief History of Interactive Computer Graphics

- Interactive computer graphics from 2d to 3d
- Applications of computer graphics
  - visualization
  - entertainment industry
  - image processing
  - modelling
  - image synthesis: Realism and Art
- Computer graphics is interdisciplinary

Early Interactive Computer Graphics

- ca. 1960: Sketchpad by Sutherland
- ca. 1970: Xerox Alto
- 1965: First CAD systems (GM, Lockheed)
- 1976: First computers with color raster display (Apple)

Graphical User Interfaces

- 1987: Apple introduces the Mac II
  ⇒ workbench metaphor
- 1993: Introduction of the Mosaic web-browser under X-Windows

Interactive Computer Graphics Today

- 1988: First public demonstration of virtual reality
- 1992: First cave
- What metaphor in space?
  ⇒ immersive user interfaces

Visualization

- Simulation of streamlines

Head mounted devices

Responsive workbench

Visualization

- Computational geometry
### Entertainment Industry

- Sony Playstation II
- Movies: The Matrix, Titanic, Fifth Element, Starship Troopers, ...

### Image Processing

- Image reconstruction, enhancement, and compression

#### Image Processing

- Morphing

#### Image-Based Rendering

#### Medical Image Processing

- Compositing
Free Form Surfaces
• 1975: The 'Trademark' of computer graphics: The Utah Teapot

Subdivision Surfaces
• Modelling and animation
• New, compact data structures, easier to handle

Rendering of Complex Models
• Modelling of natural systems by artificial life
• Procedural models

Photorealistic Image Synthesis
• New, faster numerical algorithms

Illustration

Art: Watercolor
Art: Oilpainting

Computer Graphics is Interdisciplinary
- Computer science
- Mathematics
- Physics
- Engineer sciences
- Biology
- Psychophysics
- Art
- Patent law

Oscars for Visual Effects
- 1977: 'Star Wars'.
- 1982: 'Tron': First movie with huge amounts of computer graphics (Disney).
- 1993: VFX-Oscar for 'Jurassic Park' (Industrial Light and Magic).
- 1998: VFX-Oscar for 'What Dreams May Come'.
- 2000: VFX-Oscar for 'Gladiator' (Mill Film London)


Visual Effects by the Computer
- Why?
  - It's cheaper.
  - You can do things that are impossible.
  - You can do things that are really impossible.

'Visual effects will become important in world domination.'
John Gaeta, ESC (former MANEX) Group CEO

It's cheaper...
You can do Things that are Impossible...

You can do Things that are really Impossible...

- Things that are physically impossible...
  - like the extremely fast camera movements in 'The Matrix'

- Procedure
  - green shot
  - lightprobe
  - background shot
  - geometry reconstruction
  - rendering

Green Shot

- Setup of the array of digital cameras

Green Shot

- Adjustment of height

Green Shot

- Adjustment of direction

Green Shot

- Movie cameras at both ends of the array
Green Shot
- Completion of the green box

Green Shot
- Keanu Reeves on the set...

Digital Green Shot

Digital Green Shot

Digital Green Shot

Digital Green Shot
<table>
<thead>
<tr>
<th>Digital Green Shot Finalized Material</th>
<th>Light probe: Real Light by High Dynamic Range Imaging</th>
</tr>
</thead>
</table>
| • Flow-Mo (flow motion contrary to freeze frame motion)  
  - 12000 frames per second  
  - cell animation | • Q: What do have the 'Terminator', 'The Matrix', and Christmas have in common?  
  • Christmas Balls - Low Resolution Spherical Map |
| • Compensation of systematic and measurement errors  
  • Inbetweening by morphing | |

<table>
<thead>
<tr>
<th>Rendering with Natural Light</th>
<th>Background Shot</th>
</tr>
</thead>
</table>
| • Simulation of light transport  
  - light sources: Light probe  
  - geometry:  
  - reconstructed by correspondences in single frames  
  - insertion of artificial objects  
  - reflectivity: Green and panorama shot as texture on geometry  
  - camera path through the positions given by the camera array | • High resolution panorama camera  
  • Digital virtual set |

| The Final Shot | |
|----------------|
Monte Carlo and Beyond

- Day 1
  - computer graphics and visual effects
  - principles of rendering algorithms

- Day 2
  - Monte Carlo integration
  - quasi-Monte Carlo points

- Day 3
  - quasi-Monte Carlo integration
  - Monte Carlo extensions of quasi-Monte Carlo

- Day 4
  - application to computer graphics

Interaction of Light and Matter

- Bidirectional scattering distribution function $f_s(\omega_i, x, \omega) : \Omega \times \partial V \times \Omega \to R_+^+$
  - may depend on wavelength
  - Helmholtz reciprocity principle $f_s(\omega_i, x, \omega) \equiv f_s(\omega, x, \omega_i)$

- Scattered radiance
  $$L_s(x, \omega) = \int_{\Omega} f_s(\omega_i, x, \omega) L_s(x, \omega_i) \delta(x - \omega_i) d\omega_i$$

- Integral operator shorthand
  $$L_n = T_1 L_1$$

Image Synthesis

- Flux responsivity $W : V \times \Omega \to R$

- Measurement
  $$\int_{\Omega} \int_{\Omega} W(x, \omega) L(x, \omega) d\omega dx = : \langle W, L \rangle$$

- Example: Pixelsensors $W_{m,n}$ of a pinhole camera
  - detects average radiance passing through a pixel

Scene Geometry

- Scene surface $\partial V := \bigcup_{i=1}^{N} S_i$
  - $S_i$: surface primitive, e.g. triangle
- Scene $V := \text{BoundingBox}(\partial V)$

- Set $\Omega$ of all unit directions $\omega$
  - surface of a unit sphere
- Surface normal $\hat{n} : \partial V \to \Omega$

- Ray $(x, \omega) \in V \times \Omega$
- Hitpoint $h : V \times \Omega \to \partial V \cup \{\infty\}$
  - first surface point hit, when shooting a ray from $x$ into direction $\omega$

Vacuum Radiance Transport

- Emitted radiance $L_e(x, \omega) : \partial V \times \Omega \to R_+^+$
  - usually in RGB color space
  - in vacuum $L(x, \omega) = L(h(x, -\omega), \omega)$

- Looking for $L(x, \omega) : V \times \Omega \to R_+^+$
  - sufficient to consider radiance for $x \in \partial V$
  $$L(x, \omega) = L_e(x, \omega) + L_r(x, \omega)$$

- Raytracing algorithm
  $$L = L_e + T_1 L_e + T_1^2 L_e + \cdots$$

- Neumann series, convergent if $\|T_1\| < 1$
  $$L = L_e + T_1 L_e + T_1^2 L_e + \cdots$$

- Radiance integral equation
  $$L(x, \omega) = L_e(x, \omega) + \int_{\Omega} f_s(\omega_i, x, \omega) L(h(x, \omega_i), -\omega_i) \cos \theta_i d\omega_i$$

- Global illumination problem in vacuum
  - Given the
    - scene surface $\partial V$
    - scattering properties $f_s$
    - radiance emission $L_e$
    - a sensor $W$
  - compute
    $$\langle W, (I - T_1)^{-1} L_e \rangle$$
  - the global illumination problem is reduced to an integration problem
Principles of Rendering Algorithms

- **Pipeline**: Transformation, opt. cull, shade, clip, rasterize with Z-buffer
  - GeForceII rasterization hardware
    * nVidia, California

- **Pipeline**: Split, cull, dice, shade micro-polygons, cull, cast rays with Z-buffer
  - RenderMan (REYES) ray caster
    * PIXAR, California

- **Pipeline**: Trace ray by culling, shade, recurse \( \Rightarrow \) no streaming
  - Entropy (BMRT, Torro) ray tracer with analytic anti-aliasing
    * Extuna, California
    * mental ray
    * mental images, Berlin

The Pinhole Camera: Camera Obscura

- Central projection onto image plane

Ray Tracing

- **Image**: Matrix of pixels
  - Pixel = Picture Element


A simple Ray Tracing Program: Sampling

```c
#include "Graphics.h"

int main(int Parameters, char *"Parameter")
{
    Image* Picture;
    Color Sample;
    Color SumOfSamples;
    Initialize(Parameters, Parameter);
    Picture = new Image(SizeX, SizeY);
    for (int x = 0; x < SizeX; ++x)
        for (int y = 0; y < SizeY; ++y)
            Sample = Shade(x + 0.5, y + 0.5);
    Picture = Pixel(x, y) = Sample;
    SaveImage(Picture);
    return 0;
}
```

Observation: Aliasing

- The image contains jaggies.

Anti-Aliasing by Supersampling

- In fact the pixel is an area, not a point!
  \( \Rightarrow \) pixel color is average not a single sample

\[
\text{pixel color} = \frac{1}{|P|} \int_P L(x)dx \approx \frac{1}{N} \sum_{i=1}^{N} L(x_i)
\]

- Multiple samples instead of only pixel center
Supersampling

\[
\text{for (int } x = 0; x < \text{SizeX}; x++) \\
\text{for (int } y = 0; y < \text{SizeY}; y++) \\
\{ \\
\text{SumOfSamples } = \text{Black; } \\
\text{for (int } i = 0; i < 3; i++) \\
\text{for (int } j = 0; j < 3; j++) \\
\{ \\
\text{Sample } = \text{Shade}(x + ((\text{double}) i + 0.5) / 3.0, y + ((\text{double}) j + 0.5) / 3.0); \\
\text{SumOfSamples } = \text{SumOfSamples } + \text{Sample; } \\
\} \\
\text{Picture--Pixel}(x, y) = \text{SumOfSamples } / 9; \\
\}
\]

Observation 1: Reduced Aliasing

- Introducing Randomness
  - Jittering

\[\Rightarrow \text{use random numbers}\]

- Estimation by throwing the dice is superior!
  \[\Rightarrow\text{Monte Carlo algorithms}\]

Observation 2: Still Aliasing

- since the 9 points can behave like only 3

\[\begin{array}{|c|c|c|} \hline \\
\ast & \ast & \ast \\
\ast & \ast & \ast \\
\ast & \ast & \ast \\
\hline \\
\end{array}\]

Stochastic Supersampling

\[
\text{for (int } x = 0; x < \text{SizeX}; x++) \\
\text{for (int } y = 0; y < \text{SizeY}; y++) \\
\{ \\
\text{SumOfSamples } = \text{Black; } \\
\text{for (int } i = 0; i < 3; i++) \\
\text{for (int } j = 0; j < 3; j++) \\
\{ \\
\text{Sample } = \text{Shade}(x + ((\text{double}) i + \text{drand48}()) / 3.0, \\
\text{y } + ((\text{double}) j + \text{drand48}()) / 3.0); \\
\text{SumOfSamples } = \text{SumOfSamples } + \text{Sample; } \\
\} \\
\text{Picture--Pixel}(x, y) = \text{SumOfSamples } / 9; \\
\}
\]

Antialiasing by Stochastic Supersampling

- Noise instead of aliasing
The Effort of A Bug's Life

- Movies: 24 frames/sec ⇒ 90 minutes are 129,600 frames
- Each frame: 2048 x 872 pixels at 4 bytes ⇒ 7.1 MB/frame
- 129,600 frames x 7.1 MB/frame ⇒ 925 GB
- In addition: Many tests, the video, ...
  - A Bug's Life: 2TB of disk storage

- Render-farm: 1000 processors at 250MHz
  - ca. 15 hours per frame (mono-processor)

Scene Complexity

- Realism by details: Grass, stone, fur, hair, ...

⇒ natural look by complexity

The REYES-Architecture: Assumptions and Goals

- Scene complexity and variety
  - procedural models
  - online renderer
- Complex shading
  - procedural shaders
  - vast amount of textures
- Minimal ray casting
- Efficiency: Performance and image quality
  - anti-aliasing
- Flexibility

Design Principles

- Canonical frames
- Vectorization (streaming)
- Common graphic primitive
  - micro-polygons
- Locality
  - geometry
  - textures
- Linearity
- No limit on model size
- Backdoor for extensions

Principle: Spatial Locality

- Secondary rays
  - page thrashing
- Approximation of non-local effects by textures
  - reflections, by e.g. environment-maps
  - refractions
  - shadows

Principle: Point Sampling

- Simple, mighty, general
- Anti-aliasing
  - Monte Carlo integration by jittered sampling
- Visibility by Z-buffer
  - online algorithm
  - fits jittered sampling
  - backdoor for exact compositing
Principle: Micro-Polygon Nets
- Micro-polygon: Shaded quadrangles of 1/2 pixel size and one color
  - Nyquist limit
- Micro-polygon nets by splitting along \((u, v)\) coordinates
  - in camera frame
  - resolution by estimating the object size on the screen
- Shading of all micro-polygons of a net
  - before visibility
  - backface culling

Are Micro-Polygons Waste?
- No!
  - can be vectorized
  - texture locality
  - simple texture filtering
  - coherent subdivision (arrays)
  - no clipping
  - simple application of displacement mapping
  - no perspective projection computations

Principle: Texture Locality
- Important modelling mechanism: Textures
  - surface properties
  - approximation of reflections, refraction, and shadows
  - less ray casting
- Coherent access textures (CAT)
  - \((s, t) = (au + b, cv + d)\)
- Random access textures (RAT)
  - all other mappings \((s, t) = f(u, v)\)

The Algorithm
- Read objects from database
  - potential splitting into other primitives
- Dicing into micro-polygon nets
  - only one net in main memory at a time \(\Rightarrow\) memory control
  - Dicing along \((u, v)\) (natural)
- Shade each micro-polygon in the net
- Sample and potential entry into the \(Z\)-buffer
  - no return to object space required

Methods of the Primitives
- Bound
  - bound in camera space then in image space
  - consider displacements
- Split
  - split into other primitives
  - must stay in bounding box
  - spline patches, subdivision surfaces, ...
- DiceableTest
  - possible?
  - too many micro-polygons?
  - or split...
- Dice
  - dice into micro-polygons

Split Primitives and Dice into Micro-Polygons
- Split, Dice, DiceableTest
**Texturing**
- No back-transformation
- Mip-maps

**Sampling**
- Jittered sampling
- Z-buffer

**Culling**
- No clipping

**Extensions**
- Motion blur
  - each sample is attached a moment in time
  - move micro-polygons to that moment in time
- Depth of Field
  - sample on the lens
  - move micro-polygons according to these samples
- Constructive solid geometry and transparency
  - multiple Z-entries per sample
- Shadows
  - shadow maps, deep shadow maps

**Implementation**
- Tile image into buckets
  1. primitives are sorted into the buckets by their left upper corner
  2. process buckets from left to right, top down
     (a) Split or Dice, shade, sort again
     (b) sample the elements in a bucket
- Memory control by DiceableTest
- Rendering time proportional to the number of primitives

**Disadvantages**
- Shading before sampling
  - shading incorrect for motion blur
  - coherence of shading after sampling?
- Dicing to micro-polygons may be difficult
  - e.g. implicit surfaces like blobs
- Bounding box difficult
  - e.g. particle systems
- canonical frames difficult
  - e.g. polygons
  - simple for bicubic patches
Advantages

- No inversions
- No clipping
- Vectorization of computations
- No texture memory thrashing
- No prefiltering
- Micro-polygons as common primitive
- Rendering time proportional to scene complexity
- No limit on scene size
  - online algorithm

Monte Carlo and Beyond

- **Day 1**
  - computer graphics and visual effects
  - principles of rendering algorithms
- **Day 2**
  - Monte Carlo Integration
  - quasi-Monte Carlo points
- **Day 3**
  - quasi-Monte Carlo integration
  - Monte Carlo extensions of quasi-Monte Carlo
- **Day 4**
  - application to computer graphics

Visual Effects: Fur and Hair in Monsters Inc.

Probability Spaces, Random Variables and Random Fields

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

  $\begin{align*}
  0 \leq \text{Prob}(\omega_i) \leq 1 \quad \text{and} \quad \sum_{\omega \in \Omega} \text{Prob}(\omega) = 1.
  \end{align*}

- **Definition:** Given a probability space on the set of elementary events $\Omega$, a mapping $X : \Omega \rightarrow \mathbb{R}$
  $\omega \mapsto X_\omega$

  is called a random variable. $X_\omega$ is called a realization.

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

  $X : \Omega \rightarrow C(s, d)$
  $\omega \mapsto X_\omega$

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

Monte Carlo Integration

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

Discrete Random Variables

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

  $P_X : \mathbb{R} \rightarrow [0, 1]$ $z \mapsto \text{Prob}(X \leq z) = \sum_{x \leq z} \text{Prob}(X = x')$

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

Continuous Random Variables

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

\[ p_X : \mathbb{R} \to \mathbb{R}_+^0 \]

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

\[ \text{Prob}(A) = \int_A p_X(x) \, dx = \sum_{k} \int_{I_k} p_X(x) \, dx. \]

The cumulative distribution function (cdf) is

\[ P_X(x) = \text{Prob}(X \leq x) = \text{Prob} \{ \{ t \in \mathbb{R} | t \leq x \} \} = \int_{-\infty}^{x} p_X(t) \, dt. \]

Uniform Distribution \( \mathcal{U} \) on \([0, 1]^n\)

• Probability density function

\[ p_U(x) = \begin{cases} 1 & x \in [0, 1]^n \\ 0 & \text{else} \end{cases} \]

• Requirements for simulation, i.e. realization
  - fast, deterministic algorithms
  - mimic independence
  \[ \Rightarrow \text{pseudo-random numbers} \]

• Example: Linear congruential generators (starting value \( z_0 \))

\[ z_{i+1} = (az_i + c) \mod m \quad \in \{0, \ldots, m - 1\} \]

\[ z_{i+1} = \frac{z_{i+1}}{m} \]

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

The Multidimensional Inversion Method

• For \( p(x) > 0 \) for \( x \in \mathbb{R}^n \) and \( \int_{\mathbb{R}^n} p(x) \, dx < \infty \) realize \( p \)-distributed samples

\[ P^{-1}(x) := (y^{(1)}, \ldots, y^{(n)}) = y \]

from \( x \sim \mathcal{U} \) by successively determining

\[ y^{(1)} \quad \text{using} \quad x^{(1)} = F_1(y^{(1)}), \]

\[ y^{(2)} \quad \text{using} \quad x^{(2)} = F_2(y^{(1)}, y^{(2)}) \]

using the bijections

\[ F_j(t_1, \ldots, t_j) := \int_{t_{j-1}}^{t_j} \cdots \int_{t_1} \cdots \int_{t_1} p(t_1, \ldots, t_{j-1}, t_j, \ldots, t_n) \, dt_{j+1} \ldots dt_n \]

• If \( p(x) = \prod_{j=1}^n p_j(x^{(j)}) \)

\[ F_j(t_j) = \frac{\int_{t_j} \cdots \int_{t_2} \cdots \int_{t_1} p(t_1, \ldots, t_j, \ldots, t_n) \, dt_{j+1} \ldots dt_n}{\int_{t_1} \cdots \int_{t_1} p(t_1, \ldots, t_n) \, dt_{j+1} \ldots dt_n} \]

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

The Inversion Method

• Given a density \( p(x) > 0 \) on \([0, 1]\) generate samples \( y \) that are \( p \)-distributed

• Determine

\[ x = P(y) = \frac{\int \rho(r) \, dr}{\int \rho(r) \, dr} \in [0, 1] \]

and use

\[ y_i = P^{-1}(x_i) \]

if \( P \) is invertible.

Composition Method

• Simulation of composite probability density functions

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

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

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

i.e. simulate a discrete random variable with \( \text{Prob}(w_i) = w_i \)

2. efficiently simulate \( p_i \) by

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

using only one random number

• Note: The composition method can raise variance.

• Applications: Russian Roulette, stochastic evaluation of sums
Selection Methods

- Neumann rejection method, if \(|p|_\infty < b < \infty\)
  - Choose two independent realizations of uniform random numbers \(\xi, \zeta \sim \mathcal{U}\)
  - If \(p(\xi) > \zeta \) take \(\xi\) as a sample
  - else reject \(\xi\) and try again
- Efficiency depends on graph of \(p\)
- Generalized Neumann rejection method
  - density separable, i.e. \(p(x) = p_2(x_0) \cdot p_1(x_2)\)
  - multidimensional inversion method on invertible part \(p_2\)
  - Neumann rejection method on \(p_1\)
- Metropolis sampling algorithm
  - construct Markov chain with desired density \(p\) as stationary density
- Construction dimension, i.e. random numbers required for one realization
  - now only finite expectation

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

- Probability density function
  \[ f_{\mathcal{N}(\mu, \sigma)}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \]
  - expectation \(\mu\)
  - variance \(\sigma^2\)
- Trick: Simulate a pair \((X, Y) \sim \mathcal{N}(0,1) \times \mathcal{N}(0,1)\)
  \[ f_{\mathcal{N}(0,1)}(x) \cdot f_{\mathcal{N}(0,1)}(y) dx dy = \frac{1}{2\pi} e^{-\frac{x^2 + y^2}{2}} dx dy = \frac{1}{2\pi} e^{-\frac{r^2}{2}} rdrd\theta \]
- Polar method (Box-Müller)
  \((X, Y) = \sqrt{-2\ln(1 - \xi)} \cdot (\cos 2\pi \nu, \sin 2\pi \nu)\)
  where \(\xi, \nu \sim \mathcal{U}\) on \([0, 1)\)

Simulation of Periodic Random Fields

- Typical realization procedure of \(X : \Omega \rightarrow C(s, d)\)
  1. Realize Gaussian noise on \(s\)-dimensional regular grid \(K\)
     \[ N_{\omega}(k) \sim (\mathcal{N}(0,1) \times \mathcal{N}(0,1))^d, \quad k \in K \]
  2. Shape noise by spectrum \(S\) of phenomenon
     \[ \hat{X}_{\omega}(k) = S(k) N_{\omega}(k) \]
  3. Band limited evaluation by fast Fourier transform for each dimension
     \[ X_{\omega}(x) = \sum_{k \in K} \hat{X}_{\omega}(k) e^{2\pi k^T x}, \quad x \in C(s, d) \]
  - Standard tensor product approach is exponential in \(s = \dim x = \dim k\)
  \(\Rightarrow\) Curse of dimension

Curse of Dimension

- Theorem (Bakhvalov): Let \(C_M^r\) denote the set of functions on \([0,1]^r\) with \(r\) continuous, bounded derivatives, i.e.
  \[ \left| \frac{\partial^r f(x)}{\partial x_1^{\alpha_1} \cdots \partial x_r^{\alpha_r}} \right| \leq M \text{ for } f \in C_M^r \]
  for all \(\alpha_1, \ldots, \alpha_r\), such that \(\sum_{i=1}^{r} \alpha_i = r\). Then there exists a function \(f \in C_M^r\) such that the error of approximating the integral of \(f\) using any \(N\) point quadrature rule with weights \(w_i\) and function values \(f(x_i)\) is
  \[ \left| \int_{[0,1]^r} f(x) dx - \sum_{i=0}^{N-1} w_i f(x_i) \right| \geq k \cdot N^{-r} \]
  where the constant \(k > 0\) depends on \(M\) and \(r\).

Curse of Discontinuities

- Consider
  \[ f(x) = \begin{cases} 1 & \text{if } x < x^* \\ 0 & \text{if } x \geq x^* \end{cases} \]
  with \(x_i = \frac{i}{n}\) and \(x_i \neq x^*\). Then
  \[ \left| \int_{0}^{1} f(x) dx - \frac{1}{n} \sum_{i=0}^{n-1} f(x_i) \right| \approx \frac{1}{n} \]
  \(\Rightarrow \) \(N^{-1}\) error for \(s\) dimensions

Curse of Dimension from Regular Grids

- Lattices of rank \(s\) with \(N = n^s\) points from tensor product approach

\[
\begin{array}{cccccccc}
0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 \\
0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\
0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 \\
0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 \\
0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\
0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
\end{array}
\]
Information Based Complexity Theory

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

Information Based Complexity Theory

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

Monte Carlo Integration

- **Principle**: Construct random variable with desired functional as expectation
- **Numerical integration by random sampling**
  \[ \text{Prob} \left( \left| \int_{0}^{1} f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} f(x_i) \right| < \frac{3\sigma(f)}{\sqrt{N}} \right) \approx 0.997 \]
  \( z_i \sim U \)
- **Simple**, independent of dimension and smoothness, only \( f \in L^2 \)
- **Problems**
  - Noise, slow convergence, difficult parallelization and reproducability
  - No real random numbers
- **Computational complexity**
  \[ N \cdot \sigma^2(f) = N \cdot \sigma^2 \left( \int_{0}^{1} f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} f(x_i) \right)^2 \]
- **Increase efficiency**, not only variance reduction!!!
  \[ \frac{1}{t_9} \cdot \sigma^2(f) \]

Error Control

- **Unbiased estimator \( Y \)**
  \[ E[Y] = \int_{0}^{1} f(x)dx \]
- **Bias of estimator \( Y \)**
  \[ b_Y := E[Y] - \int_{0}^{1} f(x)dx \]
- **Consistent estimator \( Y \)**
  \[ \text{Prob} \left( \lim_{N \to \infty} \frac{1}{N} \sum_{i=0}^{N-1} y_i = \int_{0}^{1} f(x)dx \right) = 1 \]
- **Error estimate of the estimate**
  \[ \sigma^2 \left( \frac{1}{N} \sum_{i=0}^{N-1} f(x_i) \right) = \left( \frac{1}{N} \right) \left( \frac{N-1}{N} \right) \left( \frac{N-2}{N} \right) \]
  - adaptive sampling
**Correlated Sampling: Separation of the Main Part**

- Variance reduction by approximation, method of control variables
- Search \( g \) with
  \[ \| I - g \|_\infty < \tau \in \mathbb{R}^d \]
- Then
  \[ \int f(x) dx = \int g(x) dx + \int (f(x) - g(x)) dx \]
  \[ \approx \int g(x) dx + \frac{1}{N} \sum_{i=0}^{N-1} (f(x_i) - g(x_i)) \]

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

- Variance of Monte Carlo part
  \[ \sigma^2(f - g) \leq \int (f(x) - g(x))^2 dx \leq \tau^2 \]
- Lower bound \( \mathcal{O} \left( N^{-\frac{1}{2}} \right) \) for \( f \in C^0([0,1]^d) \) obtained by Newton-Cotes methods

**The Method of Dependent Tests**

- Principle: Construct random field with desired function as expectation
- Method of dependent tests (parametric Monte Carlo integration)
  \[ g(y) := \int f(x,y) dx \approx \frac{1}{N} \sum_{i=0}^{N-1} f(x_i,y) \]
  for integro-approximation problems
- Computational complexity
  \[ N \cdot 2^d \cdot \mathcal{E}(\sum_{i=0}^{N-1} f(x_i,y))^2 \]
  \[ = \text{exploit induced grid structure} \]
- Examples
  - accumulation buffer
  - multilevel method of dependent tests

**Hierarchical Function Representation**

- Use multilevel function representation [Heinrich 1998]
  \[ P_{0g} = P_0g + \sum_{i=1}^{N_i} (P_i - P_{i-1})g \]
  for an arbitrary sequence \( (P_i)_{i=0}^{N_i} \) of interpolation operators

**Multilevel Method of Dependent Tests**

- Linear Lagrange interpolation of \( g_k := g(y_k) = (P_{0g})(y_k) \) in \( y_k = \frac{i}{N_i} \)
- Method of dependent tests
  \[ G_{k}^1 := \frac{1}{N_i} \sum_{i=0}^{N_i-1} f(x_i,y_k) \text{ with } N_i := N \cdot 2^m + 1 \]
  \[ \text{refinement} \]
  \[ \tilde{g}_{(2k+1)2^m-1} \approx \tilde{g}_{(2k+1)2^m-1} \]
  \[ + \frac{G_{(2k+1)2^m-1}}{2} \]
  \[ \text{Predictor} \]
  \[ \approx \tilde{g}_{(2k+1)2^m-1} \]
  \[ + \frac{G_{(2k+1)2^m-1}}{2} \]
  \[ + \frac{G_{(2k+2)2^m-1}}{2} \]
  \[ \approx \text{Update} \]

**Implementation**

- In-place reconstruction

**Efficiency Issues**

- Individual functionals
  - same high variance
  - same sampling rate, even if correlated
  - converged samples
- One function
  - small detail contribution if correlated
    \[ \chi_k^2 = N_i \sum_{i=0}^{N_i-1} \left( f(x_i,y_{(2k+1)2^m-1}) - f(x_i,y_{(2k+2)2^m-1}) + f(x_i,y_{(2k+1)2^m-1}) \right) \]
  - adapt sampling rate \( N_i \) to support size
  => reduced computational cost by exploiting correlation
- Localization heuristics
  - range check
  - predictor-corrector difference
  - relative error
- With lifting scheme on arbitrary topology and boundaries
**Numerical Results**

<table>
<thead>
<tr>
<th>Number of (average) Samples per Pixel</th>
<th>L2 Distance to Reference Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>10</td>
<td>1.0</td>
</tr>
<tr>
<td>100</td>
<td>10.0</td>
</tr>
</tbody>
</table>

**Importance Sampling**

- Integral transformation by introducing a probability density $p$
  \[ \int f(x)dx = \int f(x) \frac{p(x)}{p(y)} dx = \int f(y) \frac{p(y)}{p(y)} dy \approx \frac{1}{N} \sum_{i=0}^{N-1} f(y_i) \quad y_i \sim p \]
- Variance
  \[ \sigma^2 \left( \frac{f}{p} \right) = \int f^2(x) \frac{1}{p(x)} dx - \left( \int f(x) \frac{1}{p(x)} dx \right)^2 \]
- Often $f(x) = g(x)p(x)$
  \[ \int f(x)dx = \int g(x)p(x)dx = \int g(y)p(y)dy \approx \frac{1}{N} \sum_{i=0}^{N-1} g(y_i) \quad y_i \sim p \]
- Often separating the main part is more efficient than importance sampling

**Replication: Independent and Dependent Sampling**

- Replication heuristic
  \[
  (w_j, R_j)^{M-1}_{j=0}
  \]
  - weight functions $w_j(x) : I^d \rightarrow \mathbb{R}$ and
  - mappings $R_j(x) : I^d \rightarrow I^d$
  such that
  \[
  \int f(x)dx = \sum_{j=0}^{M-1} w_j(x)f(R_j(x))dx = \sum_{j=0}^{M-1} \int w_j(x)f(R_j(x))dx
  \]
  - Either independent integral estimation
    \[
    \sum_{j=0}^{M-1} \frac{1}{N} \sum_{i=0}^{N-1} w_j(x_i)f(R_j(x_i))dx \approx \sum_{i=0}^{N-1} \frac{1}{N} \sum_{j=0}^{M-1} w_j(x_i)f(R_j(x_i)),
    \]
  or dependent, i.e. correlated sampling
  \[
  \int w_j(x)f(R_j(x))dx \approx \sum_{i=0}^{N-1} \frac{1}{N} \sum_{j=0}^{M-1} w_j(x_i)f(R_j(x_i)),
  \]

**Stratification**

- Partition of integration domain $I^d = \bigcup_{k=1}^{N} A_k$
- Monte Carlo integration on each of the disjoint strata $A_k$
  \[
  \int_{A_k} f(x)dx = \frac{N}{N_k} \int_{A_k} f(x)dx = \frac{N}{N_k} \sum_{i=0}^{N_k-1} f(x_{i,k})
  \]
- Variance reduction for standard choice $N_k = \lambda(A_k) N$
  \[
  \sum_{i=0}^{N_k} \frac{\lambda(A_k)}{N_k} \left( f(y_i) - \frac{1}{N_k} \int_{A_k} f(x)dx \right)^2 dy \leq \frac{\sigma^2(f)}{N}
  \]
  at least as good as uniform random sampling
- $\lambda(A_k) = \frac{1}{\mu}$ yields
  \[
  \int_{A_k} f(x)dx \approx \frac{1}{N} \sum_{k=0}^{N-1} f(x_{i,k}),
  \]
  - Lloyd-relaxation
  - Jittered sampling

**Replication Heuristics: Multiple importance sampling**

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

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>independent sampling</th>
<th>dependent sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power ($\beta \in \mathbb{R}^+$) $w_j(x) := \frac{N_j p_j(x)}{\sum_{j=0}^{N_k-1} N_j p_j(x)} \cdot \frac{1}{p_j(x)}$</td>
<td>$w_j(x) = \frac{N_j p_j(x)}{\sum_{j=0}^{N_k-1} N_j p_j(x)} \cdot \frac{1}{p_j(x)}$</td>
<td></td>
</tr>
<tr>
<td>Balance ($\beta = 1$) $w_j(x) := \frac{N_j}{\sum_{j=0}^{N_k-1} N_j}$</td>
<td>$w_j(x) = \frac{N_j}{\sum_{j=0}^{N_k-1} N_j}$</td>
<td></td>
</tr>
<tr>
<td>Uniform ($\beta = 0$) $w_j(x) := \frac{N_j}{p_j(x) \sum_{j=0}^{N_k-1} N_k}$</td>
<td>$w_j(x) = \frac{1}{\mu p_j(x)}$</td>
<td></td>
</tr>
</tbody>
</table>

- Problem of insufficient techniques

**Stratification by Lloyd-Relaxation**

- Algorithm (similar to vector quantization)
  - Take $N$ random initial points
  - Loop: Move each point into the center of gravity of its Voronoi-cell
- Periodic boundary conditions
- Fast convergence to regular patterns
  ⇒ Small number of relaxation steps yields blue-noise-samples
- Expensive iteration step
- No incremental sampling
### Replication Heuristics: Regularization

- **Antithetic variables**

\[
\int_a^b f(x) \, dx = 2 \int_0^1 f(x) \, dx - \frac{1}{N} \sum_{i=1}^{N-1} \left( f(x_i) + f(1-x_i) \right)
\]

- Sample points doubled and symmetrized.
- More efficient if variance reduced to less than half of original variance.
- Good for monotonic problems.
- Effect killed by independent sampling!

- **Combining stratification**

\[
f_{\text{strat, anti}}(x) = \frac{1}{2} \left( f\left( \frac{x}{2} \right) + f\left( 1 - \frac{x}{2} \right) \right)
\]

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

### Replication Heuristics: Dependent Splitting

- **Splitting** considered as a replication heuristic restricted to selected dimensions.

\[
\int_{a_1}^{b_1} \int_{a_2}^{b_2} f(x,y) \, dy \, dx = \int_{a_1}^{b_1} \int_{a_2}^{b_2} \sum_{j=0}^{M-1} w_j(x) f(x, R_j(x, y)) \, dy \, dx
\]

\[
\approx \frac{1}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} w_j(x, y_i) f(x_i, R_j(x_i, y_i)) \, dy \, dx
\]

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

### Summary

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

⇒ Use as few random numbers as possible.

### Monte Carlo and Beyond

- **Day 1**
  - Computer graphics and visual effects.
  - Principles of rendering algorithms.

- **Day 2**
  - Monte Carlo integration.
  - Quasi-Monte Carlo points.

- **Day 3**
  - Quasi-Monte Carlo integration.
  - Monte Carlo extensions of quasi-Monte Carlo.

- **Day 4**
  - Application to computer graphics.

### Quasi-Monte Carlo Points

- Discrepancy.
- Deterministic low discrepancy:
  - Halton and Hammersley points.
  - Scrambling.
  - \((t, m, s)\)-nets and \((t, s)\)-sequences.
  - Digital constructions.
  - Good lattice points.
Stratification by Lloyd-Relaxation

- Iteration 18

- Iteration 19

Stratification by Lloyd-Relaxation

- Iteration 20

Stratification: Jittered Sampling

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

- Increased efficiency by increased uniformity of distribution

- Problem: $N$ must be factorized

**Latin Hypercube Sampling ($N$-Rooks Sampling)**

- Using $s$ uniform random permutations $\sigma_{[1]}^{[s]}$ of size $N$ yields
  
  $$z_i = \left(\frac{\sigma_{[1]}^{[s]}(i) + \xi_{[1]}^{[s]}(i)}{N}, \ldots, \frac{\sigma_{[N]}^{[s]}(i) + \xi_{[N]}^{[s]}(i)}{N}\right)$$

  where $\sigma_{[1]}^{[s]}$ can be chosen as identity

- Cannot be much worse than uniform random sampling
  
  $$\sigma^2(f_{\text{LHS}}) \leq \frac{N}{N-1} \sigma^2(f_{\text{MC}})$$

**Replication Heuristics: Stratification**

- Heuristic with
  
  - weights $w_j = \lambda(A_j)$, and
  - mappings $R_j: I^s \to A_j$

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

  - Dependent sampling
    
    $$\int_{I^s} f(x) dx \approx \frac{1}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \lambda(A_j)f(R_j(x_{i,j}))$$
Stratification by Lloyd-Relaxation

- Iteration 6

Stratification by Lloyd-Relaxation

- Iteration 7

Stratification by Lloyd-Relaxation

- Iteration 8

Stratification by Lloyd-Relaxation

- Iteration 9

Stratification by Lloyd-Relaxation

- Iteration 10

Stratification by Lloyd-Relaxation

- Iteration 11
Stratification by Lloyd-Relaxation

- Iteration 12

- Iteration 13

- Iteration 14

- Iteration 15

- Iteration 16

- Iteration 17
**Discrepancy**

- Definition: The discrepancy
  \[ D(P_N, A) := \sup_{A \in \mathcal{A}} \lambda(A) - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i) \]
  is a measure of the uniform distribution of a given point set \( P_N = \{x_0, \ldots, x_{N-1}\} \) with respect to non-empty families \( \mathcal{A} \) of Lebesgue-measurable subsets of \( I^* \). \( \chi_A \) is the characteristic function of the set \( A \).
- \( D(P_N, A) \) is worst case integration error
- (Star-)discrepancy \( D'(P_N) := D(P_N, \{A \in \mathcal{A} : \mu(A) = \frac{1}{N} \}) \)
- Extreme discrepancy \( D(P_N) := D(P_N, \{A \in \mathcal{A} : \mu(A) = \frac{1}{N}, 1 \}) \)

**Discrepancy Bounds**

- Case \( s = 1 \): Discrepancy is size of largest gap
  \[ D'(P_N) \leq \frac{2}{N} \]
  \[ D(P_N) \leq \frac{1}{N} \]
- General case
  \[ D'(P_N) \geq B_s \log \frac{1}{N} \]
- Discrepancy of random points
  \[ D'(P_N) \in \mathcal{O} \left( \frac{\log \log N}{N} \right) \]
- Discrepancy of regular grids
  \[ D'(P_N) \in \mathcal{O} \left( \frac{1}{\sqrt{N}} \right) \]
  - includes points taken from space filling curves like e.g. the Hilbert curve

**Uniform and Completely Uniform Distribution**

- By the theory of uniform distribution
  \( (x_i) \) is uniformly distributed in \( I^* \)
  \[ \lim_{N \to \infty} D(P_N) = 0 \]
  \[ \lim_{N \to \infty} D'(P_N) = 0 \]

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

**Halton Sequence and Hammersley Points**

- Radical inverse (van der Corput sequence) in base \( b \)
  \[ i = \sum_{j=0}^{\infty} a_j(i)b^j \Rightarrow \Phi_b(i) := \sum_{j=0}^{\infty} a_j(i)b^{-j-1} \]
- Note: The radical inverses are not completely uniform distributed!!!
- Halton sequence \( x_i := (\Phi_b(i), \ldots, \Phi_b(i)) \) where \( b_i \) is the i-th prime number
  \[ D'(P_N)^{(\text{Halton})} < \frac{s}{N} + \frac{1}{N} \prod_{j=1}^{s-1} \left( \frac{b_j - 1}{2 \log b_j} \right) \]

- Hammersley point set \( x_i := (\frac{1}{N}, \Phi_{b_1}(i), \ldots, \Phi_{b_{s-1}}(i)) \)
  \[ D'(P_N)^{(\text{Hammersley})} < \frac{s}{N} + \frac{1}{N} \prod_{j=1}^{s-1} \left( \frac{b_j - 1}{2 \log b_j} \right) \]

**Quasi-Monte Carlo Point Sets**

- Low discrepancy means
  \[ D'(P_N) \in \mathcal{O} \left( \frac{\log^2 N}{N} \right) \]
- Low discrepancy sequences cannot be completely uniformly distributed

- Quasi-Monte Carlo points means
  - low discrepancy and
  - deterministic points
  \[ \Rightarrow \text{Discrete density approximation of uniform distribution} \]

**Algorithm: Radical Inversion**

```cpp
double RadicalInverse(const int Base, int i)
{
    double Digit, Radical, Inverse;
    Digit = Radical = 1.0 / (double) Base;
    Inverse = 0.0;
    while(i)
    {
        Inverse += Digit * (double) (i % Base);
        Digit *= Radical;
        i /= Base;
    }
    return Inverse;
}
```
Algorithm: Incremental Radical Inversion

double NextRadicalInverse(const double Radical, double Inverse)
{/n
  const double AlmostOne = 1.0 - le-10;
  double NextInverse, Digit1, Digit2;
  NextInverse = Inverse + Radical;
  if (NextInverse > AlmostOne)
    return NextInverse;
  else
    { Digit1 = Radical;
      Digit2 = Radical * Radical;
      while(Inverse + Digit2 >= AlmostOne)
        { Digit1 = Digit2;
          Digit2 = Digit2 * Radical;
        }
      return Inverse + (Digit1 - 1.0) + Digit2;
    }
}

Other Discrepancies
- Isotropic discrepancy $J(P_N)$
  - $A$ is family of all convex subsets of $I^d$
  - by
    $$D^*(P_N) \leq D(P_N) \leq 2D^*(P_N)$$
  - $D(P_N) \leq J(P_N) \leq 4sD(P_N)^{1/s}$
    - upper bound
      $$J(P_N) \leq 4sD(P_N)^{1/s} \leq 4s(2D^*(P_N))^{1/s} = 8sD^*(P_N)^{1/s}$$
    - lower bound
      $$J(P_N) \geq D(P_N) \geq D^*(P_N)$$
  - Triangle discrepancy
  - Edge discrepancy

Computing Discrepancies
- $L_2$-norm based discrepancy
  $$D_2^2(P_N) := \sqrt{\frac{1}{N} \int_0^1 \left( A(x) - \frac{1}{N} \sum_{i=0}^{N-1} \chi_{A}(x_i) \right)^2 dx}$$
  where $A(x) = [1, 0, x(1)]$
- Can be efficiently computed in contrast to $L_\infty$-norm based discrepancies
- Numerical example: Triangular discrepancy
  $$D(P_N, T) \leq J(P_N) \leq 16^{1/s}D^*(P_N)$$

<table>
<thead>
<tr>
<th>N</th>
<th>10000 random triangles</th>
<th>100000 random triangles</th>
<th>theoretical bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.539712</td>
<td>0.591708</td>
<td>16.971</td>
</tr>
<tr>
<td>16</td>
<td>0.18326</td>
<td>0.230355</td>
<td>9.381</td>
</tr>
<tr>
<td>64</td>
<td>0.0660696</td>
<td>0.0777368</td>
<td>5.099</td>
</tr>
<tr>
<td>256</td>
<td>0.032454</td>
<td>0.0364673</td>
<td>2.739</td>
</tr>
<tr>
<td>1024</td>
<td>0.0118665</td>
<td>0.0178952</td>
<td>1.456</td>
</tr>
<tr>
<td>4096</td>
<td>0.00521621</td>
<td>0.00715305</td>
<td>0.771</td>
</tr>
</tbody>
</table>

Scrambling Permutations by Faure
- Scrambled radical inverse
  $$i = \sum_{j=0}^{\infty} a_j(i)2^j \mapsto \sum_{j=0}^{\infty} a_j(i)2^{j-1},$$
  using permutations $a_j$ by Faure
  $$a_2 = (0, 1),$$
  $$a_3 = (0, 1, 2),$$
  $$a_4 = (0, 2, 1, 3),$$
  $$a_5 = (0, 3, 2, 1, 4),$$
  $$a_6 = (0, 2, 4, 1, 3, 5),$$
  $$a_7 = (0, 2, 5, 3, 1, 4, 6),$$
  $$a_8 = (0, 4, 2, 6, 1, 5, 3, 7),$$
- Construction rule
  - $b$ is even: Take $2a_b$ and append $2a_b + 1$
  - $b$ is odd: Take $a_{b-1}$, increment each value $\geq \frac{b-1}{2}$ and insert $\frac{b-1}{2}$ in the middle

Correlation Problems of Projections
- Dimensions 7 and 8 of the Halton sequence

Scrambled Halton Sequence and Hammersley Points
- Scrambled Halton sequence
  $$x_i := \left( \Phi_{k_1}(i, \sigma_{k_1}), \ldots, \Phi_{k_d}(i, \sigma_{k_d}) \right)$$
- Scrambled Hammersley point set
  $$x_i := \left( \frac{1}{N} \Phi_{k_1}(i, \sigma_{k_1}), \ldots, \Phi_{k_d-1}(i, \sigma_{k_d-1}) \right)$$
- Improvement by scrambling (scrambled Halton sequence dimensions 7 and 8)
(t, m, s)-Nets in Base b

- Elementary interval
  \[ E := \prod_{j=1}^{m} \left( \frac{a_j}{b^j}, \frac{a_j + 1}{b^j} \right) \subseteq \mathbb{I}^s \]
  for integers \( a_j \geq 0 \) and \( 0 \leq a_j < b^j \)
- Consequently its volume is
  \[ \lambda_s(E) = \prod_{j=1}^{m} \frac{1}{b^j} = \frac{1}{b^m} \]
- Definition: For two integers \( 0 \leq t \leq m \), a finite point set of \( b^m \) points in \( s \) dimensions is called a \((t, m, s)\)-net in base \( b \), if every elementary interval of volume \( \lambda_s(E) = b^t \) contains exactly \( b^t \) points.
- For \((t, m, s)\)-nets in base \( b \) we have
  \[ D^*(P_N) \leq \Theta(s, b) b^{(t-1)s} N + O \left( \left( \frac{b^{t-1} N}{N} \right)^s \right) \]
  \( t \) is the quality parameter.
- Note: So far the concept applies to random and deterministic points

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

- All elementary volumes of a \((0, 3, 2)\)-net in base \( b = 2 \):
  \[ \lambda_s(E) = b^{t-m} = 2^{0-3} = \frac{1}{8} \]
  with exactly \( b^t = 2^0 = 1 \) point
  \( \Rightarrow \) it cannot be a \((0, 3, 2)\)-net!

- All elementary volumes of a \((1, 3, 2)\)-net in base \( b = 2 \):
  \[ \lambda_s(E) = b^{t-m} = 2^{1-3} = \frac{1}{4} \]
  with exactly \( b^t = 2^1 = 2 \) points
  \( \Rightarrow \) it is only a \((1, 3, 2)\)-net...

(t, s)-Sequences in Base b

- Definition: For \( t \geq 0 \), an infinite point sequence is called a \((t, s)\)-sequence in base \( b \), if for all \( k \geq 0 \) and \( m \geq t \), the vectors \( x_{k+m} \),...\( x_{(k+1) m} \in \mathbb{I}^t \) form a \((t, m, s)\)-net.
- For \((t, s)\)-sequence in base \( b \) we have
  \[ D^*(P_N) \leq C(s, b) b^{(t-1)s} N + O \left( \left( \frac{b^{t-1} N}{N} \right)^s \right) \]
- Adding the component \( \frac{1}{N} \) to a \((t, s)\)-sequence yields a \((t, m, s+1)\)-net
- \((0, s)\)-sequences can only exist for \( b \geq s \)
- Examples
  - Van der Corput sequences are \((0, 1)\)-sequences in base \( b \)
  - Adding the component \( \frac{1}{N} \) with \( N = b^m \) yields a \((0, m, 2)\)-net
  - e.g. Hammersley point set for \( s = 2 \) and \( N = 2^m \) points
  - many applications in finance and particle transport problems

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

- \((t, m, s)\)-net in base \( b \):
  - Set \( P_0 \) of \( N = b^m \) \( s \)-dimensional points of low discrepancy
  - Every elementary interval of volume \( b^{t-m} \) contains exactly \( b^t \) points

- \((0, m, 2)\)-net in base \( b = 2 \):
  - Set \( P_0 \) of \( N = 2^m \) \( 2 \)-dimensional points of low discrepancy
  - Every elementary interval of volume \( 2^{2-t} \) contains exactly \( 1 \) point

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

- More general than stratification and Latin hypercube sampling

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

- \((t, m, s)\)-net in base \( b \):
  - Set \( P_0 \) of \( N = b^m \) \( s \)-dimensional points of low discrepancy
  - Every elementary interval of volume \( b^{t-m} \) contains exactly \( b^t \) points

- \((0, 2n, 2)\)-net in base \( b = 2 \):
  - Set \( P_0 \) of \( N = 2^m \) \( 2 \)-dimensional points of low discrepancy
  - Every elementary interval of volume \( 2^{2-t} \) contains exactly \( 1 \) point

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

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

- Fixed-point numbers with \( M \) digits in base \( b \)
  \[ [0, 1)_{b,M} := \{ k b^{-M} \mid k = 0, \ldots, b^M - 1 \} \subset [0, 1) \]
- Components \( A^{(j)} \) of a point set \( A = \{ A_0, \ldots, A_{N-1} \} \)
  \[ A^{(j)} := \sum_{k=1}^{M} a^{(j)}_{k} b^{-k} \text{ where } a^{(j)}_{k} \in [0, 1)_{b,M} \]
- Arithmetic in commutative ring \((R, +, \cdot)\) with \( |R| = b \) elements
- Bijections \( \eta_{(j)}^{(i)} : R \rightarrow Z_b \) and \( \psi_{(j)} : Z_b \rightarrow R \)

- If now \( A \) is a \((t, m, s)\)-net, it is called a digital \((t, m, s)\)-net
- If now \( A \) is a \((t, s)\)-sequence, it is called a digital \((t, s)\)-sequence
Deterministic Constructions of Digital Point Sets

- Generator matrix
  \[ c^{(j)} := \left( c^{(j)}_{ij} \right)_{i,j=0}^{M,M-1} \in \mathbb{R}^{M \times M} \]
- van der Corput, Sobol', Faure, Niederreiter, and Niederreiter-Xing
  - increased quality by decreased parameter \( t \)
  - difficult computation of the generator matrices
- Fast evaluation by
  - Gray codes
  - vectorization
  - buffering of invariants
  - rings implemented as lookup tables
- Very often
  \[ c^{(j)} = c^{(j)}d_{i} \]

Vectorization Example for Base \( b = 2 \)

- Ring \( R = ((0,1),+,\cdot) = \mathbb{Z}_{2} \) by bit vector operations
- One component at \( M \) bits precision
  \[ x_{i} = \frac{1}{2^{M}} \cdot \sum_{k=0}^{m-1} d_{k}(i)2^{k} \]
- Basic vectorized algorithm
  \[
  \begin{align*}
  \text{double x(int i)} \\
  &\{ \\
  &\quad \text{for(int y = 0; int k = 0; i /= 2, k++)} \\
  &\quad \quad \text{if(i & 1)} \\
  &\quad \quad \quad y = C[k]; \\
  &\quad \quad \text{return (double) y / (double) (1 << (M + 1));}
  \} \]

Examples Matrices for Base \( b = 2 \)

- \((0, m, 1)\)-nets at \( N = 2^{m} \)
  \[ C_{1} = \begin{pmatrix} 0 & 0 & \ldots & 0 & 1 \\
 0 & 0 & \ldots & 1 & 0 \\
 0 & 1 & \ldots & 0 & 0 \\
 1 & 0 & \ldots & 0 & 0 \\
 \end{pmatrix} \]
  implements \( z = \frac{1}{N} \)

Examples Matrices for Base \( b = 2 \)

- \((0, 1)\)-sequences: Sobol' scrambled radical inverse
  \[ C_{3} = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 & 0 \\
 1 & 1 & 0 & \ldots & 0 & 0 \\
 1 & 0 & 1 & \ldots & 0 & 0 \\
 1 & 1 & 1 & \ldots & 0 & 0 \\
 \end{pmatrix} \quad \text{mod 2} \]
- Algorithm
  \[
  \text{double SobolRadicalInverse(int i)} \\
  \{ \\
  \quad \text{int r, v;} \\
  \quad \text{v = 1 << M;} \\
  \quad \text{for(r = 0; i >>= 1)} \\
  \quad \quad \text{if(i & 1)} \\
  \quad \quad \quad v = v; \\
  \quad \quad \quad r = r << 1; \\
  \quad \quad \} \\
  \quad \text{return (double) r / (double) (1 << (M + 1));}
  \}

Examples Matrices for Base \( b = 2 \)

- \((0, 1)\)-sequences: Larcher-Pillichshammer scrambled radical inverse
  \[ C_{4} = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 & 0 \\
 1 & 1 & 0 & \ldots & 0 & 0 \\
 1 & 1 & 1 & \ldots & 0 & 0 \\
 1 & 1 & 1 & \ldots & 1 & 1 \\
 \end{pmatrix} \]
- Algorithm
  \[
  \text{double LarcherPillichshammerRadicalInverse(int i)} \\
  \{ \\
  \quad \text{int r, v;} \\
  \quad \text{v = 1 << M;} \\
  \quad \text{for(r = 0; i >>= 1)} \\
  \quad \quad \text{if(i & 1)} \\
  \quad \quad \quad r = r; \\
  \quad \quad \quad v = v << 1; \\
  \quad \quad \} \\
  \quad \text{return (double) r / (double) (1 << (M + 1));}
  \}
Digital \((0, m, s)\)-Nets and \((0, s)\)-Sequences in Base \(b = 2\)

- \((0, m, 2)\)-nets at \(N = 2^m\)
  - Hammersley points (worst constant)
    
    \((C_1, C_2)\)
  - Larcher-Pillichshammer points (best constant)
    
    \((C_1, C_4)\)

- \((0, 2)\)-sequence: Sobol' LP0-sequence
  
  \((C_2, C_3)\)

- \((0, m, 3)\)-net at \(N = 2^m\): Sobol' LP0-net

- \((C_1, C_2, C_3)\)

- Very useful in particle transport, especially computer graphics

Good Lattice Points: Rank-1 Lattices

- Definition: A discrete subset

  \[ L := F_N + \mathbb{Z}^* \subseteq \mathbb{R}^n \]

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

- Rank-1 lattice

  \[ x_i := \frac{i}{N} g \]

  by suitable generating vector \(g \in \mathbb{N}^n\)

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

- No explicit construction - only tables

Example: Fibonacci Rank-1 Lattice

- Fibonacci numbers: \(F_1 = F_2 = 1, F_k = F_{k-1} + F_{k-2} \text{ for } k > 2\)

- Fibonacci lattice by generator vector \(g = (1, F_{k-1})\) at \(N = F_k\) points

  \[ x_i := \frac{i}{F_k} (1, F_{k-1}) \]

  - Low discrepancy

- Example: \(N = F_{10} = 55, x_i := \frac{i}{55} (1, 34)\)

Software

- Numerical Recipes
  - Sobol' sequence

- http://www.mcqmc.org/Software.html
  - Sobol' sequence
  - Faure sequence
  - Niederreiter sequence

- http://www.multitjets.caltech.edu/software/libseq/index.html
  - general package
  - several sequences (Halton, Niederreiter, ...)

- http://www.dismat.oeaw.ac.at/pirs/niedxing.html
  - generator matrices for the Niederreiter-Xing sequence

Lattice Sequences

- Rank-1 lattice

  \[ x_i = \frac{i}{N} g \]

- Hide \(N\) by choosing \(N = b^m\) and

  \[ x_i = \phi_b(i) \cdot g \]

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

- Shift \(\Delta\) in the \(k + 1\)st run for \(N = b^m\)

  \[
  \phi_b(i + kb^m) \cdot g = \phi_b(i) \cdot g + \phi_b(kb^m) \cdot g = \phi_b(i) \cdot g + \phi_b(k) b^{m-1} g = \Delta
  \]
Summary

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

Monte Carlo and Beyond

- Day 1
  - computer graphics and visual effects
  - principles of rendering algorithms
- Day 2
  - Monte Carlo integration
  - quasi-Monte Carlo points
- Day 3
  - quasi-Monte Carlo integration
  - Monte Carlo extensions of quasi-Monte Carlo
- Day 4
  - application to computer graphics

Quasi-Monte Carlo Integration

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

Theorem: The Koksma-Hlawka Inequality

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

- Proof for \( s = 1 \): Decompose
  \[ f(x) = f(1) - \int_x^1 f'(u) du = f(1) - \int_{x_0,u}(x)f'(u) du \]
  and define
  \[ V(f) := \int \left| \frac{\partial f(x)}{\partial u} \right| du \]
- Note:
  \[ \chi_{[0,u]}(x) = \begin{cases} 1 & x \in [0,u) \\ 0 & \text{else} \end{cases} \]

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

\[ = \int f(1) - \int \chi_{[0,u]}(x)f'(u) du dx + \frac{1}{N} \sum_{i=0}^{N-1} \left( \int \chi_{[0,u]}(x_i)f'(u) du \right) \]

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

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

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

\[ \leq \int |f'(u)| du \sup_{x \in [1]} \left| \frac{1}{N} \sum_{i=0}^{N-1} \chi_{[0,u]}(x_i) - \int \chi_{[0,u]}(x) dx \right| \]

\[ = V(f) D^*(P_N) \quad \text{q.e.d.} \]
### Variation in the Sense of Vitali

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

\[
\Delta(f, A) := \sum_{j_1=0}^{1} \cdots \sum_{j_n=0}^{1} (-1)^{j_1+\cdots+j_n} f(x_{j_1} \cdots x_{j_n}) + (1 - j_1) b_1 \cdots (1 - j_n) b_n
\]

- Variation in the sense of Vitali

\[
V^{(1)}(f) := \sup_{P} \sum_{A \in P} |\Delta(f, A)|
\]

where $P$ is the set of partitions of $I^*$ into subintervals $A$ as above

- If $f$ has a continuous derivative

\[
V^{(1)}(f) = \int_{I^*} \left| \frac{\partial f}{\partial x_1} \right| dx
\]

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

\[
\Rightarrow \Delta(f, A) = 0 \Rightarrow V^{(1)}(f) = 0
\]

### Variation Reduction

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

- Quasi-Monte Carlo important sampling

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

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

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

### Discrepancy Bounds for Transformed Points

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

\[
D^*(p, C_N) := \sup_{x \in P} \left| \int_{I^*} x_i(p) dx - \frac{1}{N} \sum_{i=0}^{N-1} x_i(y_i) \right|
\]

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

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

\[
D^*(p, C_N) = D^*(p_N)
\]

otherwise

\[
D^*(p, C_N) \leq c (D^*(p_N))^\frac{1}{2} \quad c \in \mathbb{R}^+
\]

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

- Generalized Koksma-Hlawka inequality

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

### Approximating Continuous by Discrete Measures

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

- Avoid weighting by small probabilities

\[
\int_{I^*} f(x) dx = \int_{I^*} g(x) p(x) dx = \int_{I^*} g(y) dP(y)
\]

- Approximate measure $P$ by discrete measure

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

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

- Then

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

### Discrete Density Approximation

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

- Note: Assigning dimensions is crucial
Discrete Density Approximation

Random

Halton

Hammersley

Far Too Pessimistic Bounds by Isotropic Discrepancy

- Restrict \( f \) to convex domains \( C \), where \( f|_C \) is of bounded variation

\[
\int C f(z)dz - \frac{1}{N} \sum_{i=0}^{N-1} x_C(x_i) f(x_i) \leq (V(f) + |f(1, 1, 1)|) f(P_N) - (V(f) + |f(1, 1, 1)|) s D^*(P_N)
\]

- Bound worse than the Monte Carlo rate for \( s > 2 \)

- Numerical experiments tell a different story...

- Justification by discrete density approximation

  - using low discrepancy sequences always is better

- Which function class other than bounded variation?

The Spirit of the Numerical Recipes' Argument

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

**Proof:**

- \( I^s \) partitioned into \( N = \prod_{i=1}^N N_i \) voxels \( v_i \), \( \lambda_s(v_i) = \frac{1}{N}, 1 \leq i \leq N \)

- Jittered sampling for

\[
\int A \chi_A(z)dz = \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i)
\]

- Three sets of voxel indices

\[
V_i = \{v_i|v_i \cap A = v_i\}
\]

\[
V_i = \{v_i|\emptyset \neq v_i \cap A \
eq v_i\}
\]

\[
V_0 = \{v_i|v_i \cap A = \emptyset\}
\]

- **Assumption:** \(|V_i| \in O(N)\)

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

Infinite Variation

- Quasi-Monte Carlo is roughly quadratically faster than random sampling

- Case \( s = 1 \): \( \lambda(\mathbb{R}^s) < \infty \) for piecewise continuous functions

- General case: Usually infinite variation for piecewise continuous functions

- In computer graphics: Triangles and edges

\[
V(f) = \infty \quad \sigma^2(f) = \frac{1}{4}
\]

- Proof for the Hammersley points at \( N = 2^i \)

\[
\int f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} f(x_i) \leq \left(\frac{\sigma^2}{N^2} + \frac{1}{N^2}\right) \text{if even}
\]

\[
\left(\frac{\sigma^2}{N^2} + \frac{1}{N^2}\right) \text{else}
\]

Convergence

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

- Observed rate for discontinuous functions \( O\left(N^{-\frac{s}{2}}\right) \)

- Argument in "Numerical Recipes"

  - **Weak assumption:** The behavior of low discrepancy samples at the border of characteristic sets is uncorrelated.

  - in fact true for jittered sampling [Mitchell]

  - generalized by Szirmay-Kalos

- Argument by [MC95]

  - **Weak assumption:** Rate of random sampling used as upper bound for low discrepancy sampling, i.e. it is assumed, that low discrepancy sampling deterministically (!) does not behave worse than random sampling.

  - there exist proofs for some special cases for \( s = 2 \)

- Random sample \( x_i \in v_i \in V_0 \) is Bernoulli random variable with

\[
\lambda_s(A \cap v_i) = \lambda_s(v_i) \quad \text{and} \quad \sigma^2(\chi_A|v_i) \leq \frac{1}{4}
\]

- Then

\[
\sigma^2 \left(\frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i)\right) = \sigma^2 \left(\frac{1}{N} \sum_{i=0}^{N-1} \chi_A|v_i(x_i)\right)
\]

\[
= \sigma^2 \left(\frac{1}{N} \sum_{i \in V_i} \chi_A|v_i(x_i) + \frac{1}{2} \sum_{\emptyset \neq v_i \cap A = v_i} \chi_A|v_i(x_i) + \frac{1}{N} \sum_{v_i} \chi_A|v_i(x_i)\right)
\]

\[
= \sigma^2 \left(\frac{1}{N} |V_i| + \frac{1}{2} \sum_{\emptyset \neq v_i \cap A = v_i} \chi_A|v_i(x_i) + 0\right)
\]

\[
= \sigma^2 \left(\frac{1}{N} \sum_{i \in V_0} \chi_A|v_i(x_i)\right) = \sum_{i \in V_0} \sigma^2(\chi_A|v_i(x_i)) \leq \left|V_0\right| \frac{\sigma^2}{N^2} = cN^{s-1}N^{-2} = cN^{-\frac{s}{2}}\]
By the H"older inequality the error is expected to be
\[ \left| \sum_{i=0}^{N-1} \chi_A(e_i) \right| \leq \sqrt{cN} N^{-\frac{1}{2}} \in O(N^{-\frac{1}{2}}) \]
q.e.d.

**Note:**
\[ \lim_{N \to \infty} N^{-\frac{1}{2}} \cdot \frac{1}{N} = 0 \]

---

**Error Control**
- Determinism: Variance of estimate is zero!
- No cheap error estimate from samples
- No efficiency - complex analysis by information based complexity theory
- Quasi-Monte Carlo integration is "biased" but "consistent"

- Adaptive sampling by using low discrepancy sequences
- Convergence is rather smooth due to intrinsic stratification properties
- Choose fixed distance \( \Delta N \) of samples
- Compare difference of averages all \( \Delta N \) to a threshold
- Must be below the threshold \( T \) times

- The points "know" where to fall
- Consider local minima for \( \Delta N \)
  - e.g. \((s, t)\)-sequences at \( \Delta N = b^m \)
  - e.g., Hammersley in \( s = 2 \)

---

**From Monte Carlo to Quasi-Monte Carlo Integration**
- The basic algorithms transfer
  - Integration
  - Integro-approximation
  - Separation of main part and multilevel method of dependent tests
- Faster convergence by deterministic low discrepancy sampling
  - Intrinsically stratified, Latin hypercube, regularized, antithetic, ...
- The simulation of random variables becomes discrete density approximation
  - No independence required due to averaging
  - Importance sampling carries over
  - Rejection modeling impossible
- Adaptive sampling by difference comparison

- What about splitting?

---

**Efficient Design of Quasi-Monte Carlo Algorithms**
- Write down the integral
- Transform onto unit cube \( f^* \)
- Separate the main part
- Apply (multiple) importance sampling
- Use quasi-Monte Carlo points
  - Sample size \( N \)
  - Assigning dimensions
- Use dependent splitting

---

**Quasi-Monte Carlo Integration using Lattice Points**
- Originally developed for the class \( E_n(c) \) with \( c > 0, \alpha > 1 \), where
  \( f \in E_n(c) \Leftrightarrow |f(h)| \leq \frac{c}{(h_1 \ldots h_s)^\alpha}, \quad h_j := \max\{|1, |h_j|, h \in \mathbb{Z}^s \}
- Error bound
  \[ \left| \frac{1}{N} \sum_{i=0}^{N-1} f \left( \frac{i}{N} \right) - \int f(x) dx \right| \leq \sum_{h \neq 0 (\mod N)} \frac{1}{(h_1 \ldots h_s)^\alpha} \]
- Generalized to class of bounded variation

---

**Curse of Dimension from Regular Grids**
- Lattices of rank \( s \) with \( N = n^s \) points from tensor product approach
  \[ \begin{array}{ccccccccccccccccccc}
  0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 \\
  0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\
  0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
  0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 \\
  0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 \\
  0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\
  0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\
  0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
  \end{array} \]
- \( O(n^s \log n) \) for fast Fourier transforms
Fourier Transform on Rank-1 Lattices

- Choice of wave vectors
  \[ K_N := \{k_0, \ldots, k_{N-1}\} \subset \mathbb{Z}^s \]
  such that
  \[ k_m \in \mathbb{Z}_m := \{k \in \mathbb{Z}^s \mid k^T \cdot g \equiv m \pmod{N}\} \]
  since then
  \[ k_m^T \cdot x_n = k_m^T \cdot \frac{n}{N} g = (m + f_m N) \frac{n}{N} \]
- Evaluate
  \[ f(x_n) = \sum_{k \in K_N} \hat{f}(k) e^{2\pi i k^T x_n} = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \hat{f}(k_m) e^{2\pi i (m \frac{n}{N} + l_m n)} \]
  \[ = \sum_{n=0}^{N-1} \hat{f}(k_m) e^{2\pi i m \frac{n}{N}} \]
  by one-dimensional Fourier transform ⇒ way to break curse of dimension !

Determining the Wave Vectors

- Many possible choices for
  \[ k_m \in \mathbb{Z}_m := \{k \in \mathbb{Z}^s \mid k^T \cdot g \equiv m \pmod{N}\} \]
- Choose largest waves first
  \[ \|k_m\|_2 = \min_{k \in \mathbb{Z}_m} \|k\|_2. \]
- Enumerate along lines of constant \( \| \cdot \|_1 \)-norm

Summary

- Quasi-Monte Carlo simpler and faster than Monte Carlo integration
- Most Monte Carlo techniques transfer
- However, no rejection sampling !
- Works fine on \( L^2 \), too
  - justification by discrete density approximation
- Breaks curse of dimension even for discrete Fourier transform

Use whenever you can write the problem as an integral

Monte Carlo and Beyond

- Day 1
  - computer graphics and visual effects
  - principles of rendering algorithms
- Day 2
  - Monte Carlo integration
  - quasi-Monte Carlo points
- Day 3
  - quasi-Monte Carlo integration
  - Monte Carlo extensions of quasi-Monte Carlo
- Day 4
  - application to computer graphics

Monte Carlo Extensions of Quasi-Monte Carlo

- Random field synthesis on good lattice points
- Randomized quasi-Monte Carlo integration
- Randomized replications
- Restricted randomized replications

Periodic Random Field Synthesis on Good Lattice Points

- Applications of Periodic Random Fields \( \mathcal{L}_s(x) = \mathcal{L}_s(x + z) \) for \( z \in \mathbb{Z}^s \) (Period 1)
  - height fields: Waves, terrain
  - caustics
  - turbulent wind fields
- Typical procedure
  1. Realize Gaussian noise
     \[ N_s(k) \sim (\mathcal{N}(0, 1) \times \mathcal{N}(0, 1))^d \]
  2. Filter noise by spectrum \( S \) of phenomenon
     \[ \mathcal{L}_s(k) = S(k) N_s(k) \]
  3. Band limited evaluation by fast Fourier transform
     \[ \mathcal{L}_s(x) = \sum_{k \in K_N} \hat{\mathcal{L}}_s(k) e^{2\pi i k^T x} \]
Fourier Transform on Rank-1 Lattices

- Choice of wave vectors \( K_N := \{k_0, \ldots, k_{N-1}\} \subset \mathbb{Z}^d \) such that
  \( k_m \in \mathbb{Z}^d := \{k \in \mathbb{Z}^d | k^T \cdot g \equiv m \pmod{N}\} \)
  
  hence with \( x_n = \frac{n}{N} \epsilon \)
  
  \( k_m \cdot x_n = k_m \cdot \frac{n}{N} = (m + l_m N) \frac{n}{N} \)

- By one-dimensional Fourier transform evaluate
  \[
  f(x_n) = \sum_{k \in K_N} \xi_k e^{2\pi ik^T x_n} = \sum_{m=0}^{N-1} \xi_m e^{2\pi i(m \frac{n}{N} + h_n)}
  \]
  \[
  f(x_n) = \sum_{m=0}^{N-1} \xi_m e^{2\pi i m \frac{n}{N}}
  \]

Example: Ocean Waves on Fibonacci Rank-1 Lattices

- Fibonacci numbers: \( F_1 = F_2 = 1, F_k = F_{k-1} + F_{k-2} \) for \( k > 2 \)

- Fibonacci lattice by generator vector \( g = (1, F_{k-1}) \) at \( N = F_k \) points

  \( x_n := \frac{n}{F_k} (1, F_{k-1}) \)

  - Low discrepancy

  - Example: \( N = F_{10} = 55, x_n := \frac{n}{F_{10}} (1, 34) \)

  - Barycentric interpolation on periodic Delaunay triangulation

Application: Ocean Wave Simulation

- Ocean height field synthesis

  1. Realize Gaussian noise random field \( \xi_{r,m}, \xi_{t,m} \sim N(0, 1) \)
  2. Fourier coefficients by filtering with Philips spectrum \( P_M(k_m) \)

  \[
  h_{w}(k_m, t) = \sqrt{P_M(k_m)} \left( (\xi_{r,m} + i \xi_{t,m}) e^{i \omega(k_m)t} + (\xi_{r,m} - i \xi_{t,m}) e^{-i \omega(k_m)t} \right)
  \]

  3. Height field \( h_w : \mathbb{R}^3 \rightarrow \mathbb{R} \) and normals by \( \nabla h_w : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \)

  \[
  h_w(x_n, t) = \sum_{m=0}^{N-1} h_m(k_m, t) e^{2\pi im \frac{n}{N}}
  \]

  \[
  \nabla h_w(x_n, t) = \sum_{m=0}^{N-1} 2\pi im h_m(k_m, t) e^{2\pi im \frac{n}{N}}
  \]

  \( \Rightarrow \dim x_n = 2 \), but evaluation by one-dimensional fast Fourier transform

Periodic Tiling

- Breaking the Curse of Dimension

  - Point set \( P_N = \{x_0, \ldots, x_{N-1}\} \)

  - Monte Carlo Integration: Random points \( P_N \)

  \[
  \text{Prob} \left( \left\{ \int_{\mathbb{R}^2} f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} f(x_i) \right\} < \frac{3}{2 \sqrt{N} \sigma(f)} \right) \approx 0.997
  \]

  - slow

  - cheap error estimate

  - easy math for \( L^2 \)

  - Quasi-Monte Carlo Integration: Quasi-Monte Carlo points \( P_N \)

  \[
  \left| \int_{\mathbb{R}^2} f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} f(x_i) \right| < D(f)(P_N) \sqrt{V(f)}
  \]

  - fast

  - no error estimate

  - heavy math for \( BV \)

  - Combine and take the best!

  - Price: A little bit of convergence, problems of random number generators
**Randomized Quasi-Monte Carlo Integration**

- Randomized replications of a QMC point set \( A := \{A_0, \ldots, A_{n-1}\} \)

\[
X_k := \{X_{k,0}, \ldots, X_{k,n-1}\} \quad \text{for} \quad 1 \leq k \leq r
\]
such that

1. Uniformity: \( X_{k,i} \sim U[0,1) \) for fixed \( i \)

2. Equidistribution: \( X_1, \ldots, X_r \) are low-discrepancy point sets with probability one

- Monte Carlo estimate

\[
I_{r,n,f} := \frac{1}{r} \sum_{k=1}^{r} \sum_{i=0}^{n-1} f(X_{k,i})
\]

with error estimate

\[
\sigma^2(I_{r,n,f}) \approx \frac{1}{r(r-1)} \sum_{k=1}^{r} \left( \frac{1}{n} \sum_{i=0}^{n-1} f(X_{k,i}) - I_{r,n,f} \right)^2
\]

- Presmoothing of the integrand by correlated sampling

**Randomized Replications**

- Random bijections

\[
R_x : I^r \rightarrow I^r
\]

- in fact dependent sampling replication heuristics

- Cranley-Patterson rotations

- originally designed for error estimation with lattice points

- very simple

- Owen-Scrambling

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

- advanced

**Randomized Replications by Cranley-Patterson Rotations**

- Random shifts on the torus \( I^r \) applied to \( A \)

\[
X_{k,j} := A_{j}^{(i)} + U_{j}^{(i)} \mod 1 \quad \text{for} \quad 1 \leq j \leq s
\]

- Originally \( A \) was a lattice of low discrepancy

- Note: Cranley-Patterson rotations work with any arbitrary point set \( A \)

  - still unbiased Monte Carlo scheme

  - especially for \((t,s)\)-sequences and \((t,m,s)\)-nets

  - however discrepancy can be affected due to shifting

  - example: Padded replications sampling

- pad \( A \) by low dimensional point sets, apply random shifts

- exploit problem structure, e.g. in transport problems

- cheaper point sets than quasi-Monte Carlo points in high dimensions

**Randomized Replications by Owen-Scrambling**

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

- Algorithm: Start with \( H = I^r \) and for each axis

  1. slice \( H \) into \( b \) equally sized volumes \( H_1, H_2, \ldots, H_b \) along the axis

  2. randomly permute these volume

  3. for each \( H_j \) recursively repeat the procedure with \( H = H_j \)

- Algorithm gets finite by finite precision of computation, i.e. digital constructions

- Net and sequence parameters remain untouched

  - contrary to random shifts by Cranley-Patterson

- Much faster convergence for \( N > s^t \)

\[
O \left( \frac{\log_2 N}{N^2} \right)
\]

due to extinction effects by full stratification

**Replication by Scrambling**

- Unit square \([0,1)^2\)

  - Bit 1 of \( x \)
**Replication by Scrambling**

- Bit 2 of $x$

**Replication by Scrambling**

- Bit 3 of $x$

**Replication by Scrambling**

- All bits of $x$

**Replication by Scrambling**

- All bits of $x$ and $y$

---

**Formalization of Scrambling**

- Given a digital $(t, m, s)$-net $A = \{A_0, \ldots, A_{N-1}\}$ in base $b$ with components

$$A^{(j)} = \sum_{k=1}^{M} a^{(j)}_k b^{-k} = a^{(j)}_1 a^{(j)}_2 \ldots a^{(j)}_M$$

- A scrambled replicate $X$ of $A$ is obtained by

$$X^{(j)} = \sum_{k=1}^{M} x^{(j)}_k b^{-k} = x^{(j)}_1 x^{(j)}_2 x^{(j)}_3 \ldots x^{(j)}_M$$

where

$$x^{(j)}_1 := \pi^{(j)}(a^{(j)}_1)$$

$$x^{(j)}_2 := \pi^{(j)}(a^{(j)}_2)$$

$$\vdots$$

$$x^{(j)}_M := \pi^{(j)}(a^{(j)}_M)$$

- Independent random permutations $\pi^{(j)} \in S_b$

- Permutation depends on the $k - 1$ leading digits of $A^{(j)}$ ⇒ permutation tree

---

**Efficient Implementation of Scrambling**

- Main ideas for efficient scrambling:
  - keep only one path of the permutation tree in memory
  - traverse permutation tree paths that way, that each permutation is used only once
  - implies reordering of the points that should be scrambled
    - sorting the components
      $$A^{(j)} = \{A^{(j)}_0, \ldots, A^{(j)}_{N-1}\} \Rightarrow A^{(j)}_0 \leq \ldots \leq A^{(j)}_{N-1}$$
      - in this order scramble the components
      - each branch of the permutation tree is traversed at most once
      - undo the sorting using the inverse permutation $\sigma^{-1}$

- Main ideas for efficient scrambling:
  - keep only one path of the permutation tree in memory
  - traverse permutation tree paths that way, that each permutation is used only once
  - implies reordering of the points that should be scrambled
    - sorting the components
      $$A^{(j)} = \{A^{(j)}_0, \ldots, A^{(j)}_{N-1}\} \Rightarrow A^{(j)}_0 \leq \ldots \leq A^{(j)}_{N-1}$$
      - in this order scramble the components
      - each branch of the permutation tree is traversed at most once
      - undo the sorting using the inverse permutation $\sigma^{-1}$
**Example: Scrambled \((0, m, 2)\)-Nets in Base \(b = 2\)**

- \(N = 2^m\) points \(A = \{A_0, \ldots, A_{N-1}\}\)
- The components correspond to the inverse permutations \(\sigma^{-1}_j(i) = N \cdot A^{(j)}_i\)
  - e.g. Hammersley: \(\sigma_0^{-1}(i) = 2^m \cdot i\) and \(\sigma_1^{-1}(i) = 2^m \cdot \Phi_2(i)\)
- Random permutations on \(\mathbb{Z}_2\) are random bit flips and can be vectorized
  - i.e. applying a path of permutations means XORing the bit vector of bit permutations

Scrambling the component \(j\):
- start out with a random bit vector and save it in \(X^{(j)}_0(0)\)
- permutation tree traversal by enumerating \(i = 1, \ldots, 2^m - 1\)
  - detect where tree ramifies: Number \(f\) of leading shared digits of \(i - 1\) and \(i\)
  - XOR a bit vector with \(f\) leading zeros followed by a \(1\) filled by random bits
  - change the branch and choose new random permutations
- store result in \(X^{(j)}_{f}(i)\)

**Example: Instance of a Randomly Scrambled \((0, 4, 2)\)-Net**

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...

**Implementation: Scrambled Hammersley Point Set**

```c
N = 1 << m;
Digits = get32_random_bits();
P(0, 0) = (double) Digits / (double) 0x10000000;

for (i = 1; i < N; i++)
{
    Difference = i - 1 >> i;
    Shift = Log - Bits;
    Digits = (0x80000000 | get31_random_bits()) >> Shift;
    P(i, 0) = (double) Digits / (double) 0x10000000;
    Digits2 = (0x80000000 | get31_random_bits()) >> Shift;
    P(i, 1) = (double) Digits2 / (double) 0x10000000;
}
```

**Example: Instance of a Randomly Scrambled \((0, 4, 2)\)-Net**

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...

**Example: Instance of a Randomly Scrambled \((0, 4, 2)\)-Net**

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...

**Example: Instance of a Randomly Scrambled \((0, 4, 2)\)-Net**

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...
Example: Instance of a Randomly Scrambled $(0, 4, 2)$-Net
- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...

Example: Another Instance of a Randomly Scrambled $(0, 4, 2)$-Net
- All instances are of low discrepancy
- Not all instances are equally good...

Example: Instance of a Randomly Scrambled $(0, 4, 2)$-Net
- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...

Example: Another Instance of a Randomly Scrambled $(0, 4, 2)$-Net
- All instances are of low discrepancy
- Not all instances are equally good...

Trajectory Splitting and Dependent Sampling
- Increase efficiency by splitting
  \[
  \frac{1}{N} \sum_{i=0}^{N-1} f(x_i, y_i) \approx \int_{I^2} \int_{I^2} f(x, y) dx dy = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(x_i, y_j)
  \]
  depending on the correlation coefficient of \(f(\xi, \eta)\) and \(f(\xi, \eta')\)
- Exploit smoothness by correlated sampling
  \[
  \sum_{j=0}^{M-1} \sum_{i=0}^{N-1} f_j(x_i) y_j \approx \sum_{j=0}^{M-1} \sum_{i=0}^{N-1} f_j(x_i)
  \]
  \[
  = \int_{I^2} f_j(x) dx = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} f_j(x_i)
  \]
  e.g. separation of the main part

Trajectory Splitting by Dependent Sampling
- Integrals invariant under Cranley-Patterson rotation by \(z_j \in I^2\)
  \[
  R_j : I^2 \rightarrow I^2 \quad (y + z_j) \mod 1 \Rightarrow \int_{I^2} g(y) dy = \int_{I^2} g(R_j(y)) dy
  \]
- Presmoothing of selected dimensions by replication
  \[
  \int_{I^2} \int_{I^2} f(x, y) dx dy = \int_{I^2} \int_{I^2} \frac{1}{M} \sum_{j=0}^{M-1} f(x, R_j(y)) dy dx
  \]
  \[
  \approx \frac{1}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} f(x_i, R_j(y))
  \]
  \[
  = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} f(x_i, (y_j + z_j) \mod 1)
  \]
  - global quadrature rule \(P_{N^2+M+1} = (x_i, y_j)_{i,j=0}^{N,M-1}\)
  - local quadrature rule \(P_{M+1} = (x_i)_{i=0}^{M-1}\)
  \(\Rightarrow\) Trajectories split by dependent sampling
Further Randomization Techniques

- Padding quasi-Monte Carlo points for high dimensions
  - by random numbers
  - by Latin hypercube samples

- Jittered quasi-Monte Carlo point sets
  - Latin hypercube samples, however deterministic permutation
  - Note: Rate of randomly permuted Latin hypercube samples does not apply!
    - e.g. \((0, m, 2)\)-net with jitter of size \(b^{-m}\)

- Latin supercube sampling
  - biased
  - unbiased if used for decorrelating padded replications sampling

Summary

- Random field synthesis on good lattice points
- Randomized quasi-Monte Carlo integration
  - error estimate
  - \(L^2\)
  - almost as fast as pure quasi-Monte Carlo integration
  - concept of randomized replications
- Dependent splitting

Monte Carlo and Beyond

- Day 1
  - computer graphics and visual effects
  - principles of rendering algorithms

- Day 2
  - Monte Carlo integration
  - quasi-Monte Carlo points

- Day 3
  - quasi-Monte Carlo integration
  - Monte Carlo extensions of quasi-Monte Carlo

- Day 4
  - application to computer graphics

Applications to Computer Graphics

- Interleaved sampling
  - interleaved method of dependent tests
- Volume rendering
  - dependent splitting by restricted Cranley-Patterson rotations
- Bidirectional path tracing
  - padded replications sampling for cheap high-dimensional samples
- Distribution ray tracing
  - strictly deterministic
  - dependent splitting by restricted Cranley-Patterson rotations

- Note: Discontinuous integrands and high dimension !!!

Sampling

- Regular grids
  - fast rasterizers
  - aliasing potential
  - slow convergence
- Irregular patterns
  - no rasterizers
  - optimal spectral properties
  - low discrepancy

Accumulation Buffer

**Interleaved Sampling**

- Precomputed Max-Lloyd relaxation points as basis pattern
  - periodically tile seamlessly
  - blue noise spectral characteristics (minimum distance property)
  - low discrepancy (correlated)
  - for arbitrary problem dimension
- Size $N'$ of irregular basis pattern
  - blend between regular and irregular sampling
- Choice of interleaving ratio by $\chi_P$
  - spread out aliasing artifacts

**Consequences and Theoretical Considerations**

- Aliasing by pattern repetition
  - spread out by larger-than-pixel-size patterns
  - arbitrary interleaving
- Method of dependent tests (parametric Monte Carlo integration)
  - Accumulation buffer
    \[ g_N(y) = \int_{[0,1]^N} f(x,y) dx \approx \frac{1}{N} \sum_{i=0}^{N-1} f(x_i,y) \]
  - Interleaved sampling
    \[ g_N(y) = \int_{[0,1]^N} \chi_P(x)f(x,y) dx \approx \frac{1}{N'} \sum_{i=0}^{N'-1} \chi_P(x'_i)f(x'_i,y) \]
- Exploit intrinsic high coherence
  - new hardware
  - new software parallelization paradigm

**Sampling Patterns for Interleaved Sampling**

**Application: Antialiasing**

- Reduced aliasing at only 4 samples per pixel
  - artifacts spread out
  - artifacts from repetition, not from deterministic sampling
- Simple to implement by hardware (current and future)

**Application: Motion Blur**

- Artifacts replaced by noise at 16 samples per pixel
- Exactly one moment in time for each subimage
  - finite number of time samples and consequently instances of the scene
  - finally correct implementations of REYES/RenderMan and the photon map
  - Sobol' $(0, m, 3)$-net optimally can replace stratified random sampling

**Other Applications**

- All accumulation buffer techniques
  - weighted sampling
  - extended light source and the $N$-shadow problem
  - deep shadow maps
  - global illumination by instant radiosity
- CCD chip design
  - high dynamic range capturing
"One-Dimensional" Integration in Computer Graphics

• Linear light sources, spectral effects, volumetric effects
• The (example) problem
\[ \int_{x, y} f(x, y, z) dx dy dz \]
- \( x, y \) for ray from the eye through a point in the pixel
- \( z \) for integrating the density \( f \) along the ray
• The \( z \)-component requires presmoothing
• Bad: Using one-dimensional stratified Monte Carlo for
\[ g(x, y) = \int f(x, y, z) dz \]
- uncorrelated ray marching: Fur, photon map with participating media, ...
• Good: Use dependent splitting, e.g. by restricted Cranley-Patterson rotations
\[ \int_{x, y} \sum_{k=0}^{M-1} f(x, y, R_k(z)) dx dy dz \]
- correlated ray marching: Less random numbers and faster convergence

Volume Rendering

• Dependent sampling saves \( \approx 20\% \) rendering time wasted for random numbers
• Equidistant, i.e. correlated, samples have lower discrepancy than stratified samples
• Combine with interleaved sampling: Coherent ray marching

Application: Volume Rendering

• Much improved depth antialiasing (unbiased)
• Simply interleaving images
  - coherent ray marching

Remember...

• Discontinuous integrand mainly in \( x, y \)
• For \( s = 1 \)
  - lattices and \((0, m, 1)\)-nets become identical, in fact the rectangle rule
  - the best discrepancy is \( D^s(P_N) \geq \frac{1}{N} \)
    • obtained by equidistant set of samples (correlated)
    • (stratified) random sampling \( D^s(P_N) \in O\left( \frac{1}{N} \right) \) (uncorrelated)
  
=> Never use one-dimensional stratified Monte Carlo!!!
=> Use randomized quasi-Monte Carlo instead

Application: Volume Rendering
**Application: Volume Rendering**

**Path Integral Formulation**
- Path space and path measure
  \[ \mathcal{P}_k = \{ \mathbf{z} = z_0 z_1 \ldots z_k \mid z_i \in S \} \quad d\mu_k(\mathbf{z}) = \prod_{i=0}^k dA(z_i) \]
- Measurement contribution function
  \[ f_j = L e \cdot G \cdot f_s \cdot G \cdot f_s \cdot G \cdot w_j(z) \]
- Integral for path length \( k \)
  \[ \int_{\mathcal{P}_k} f_j(\mathbf{z}) d\mu_k(\mathbf{z}) \]

**Bidirectional Path Tracing**
- Generation of path space samples
  1. generate light subpath
  2. generate eye subpath
  3. connect deterministically and use all possibilities
- Techniques and probability density functions
  \[ P_i \quad P_3.1 \quad P_3.2 \quad P_3.3 \]
- Estimator
  \[ \int_{\mathcal{P}_k} f_j(\mathbf{z}) d\mu_k(\mathbf{z}) \approx \frac{1}{n \cdot m} \sum_{i=1}^n \sum_{j=1}^m f_j(z_{i,j}) \quad \text{where } z_{i,j} \sim P_{k,i} \]

**The Global Illumination Problem**
- Three-point form of the light transport equation
  \[ L(y \to z) = L_e(y \to z) + \int_{S \times S} L(x \to y) f_s(x \to y \to z) G(x \to y) dA(z) \]
- Measurement equation
  \[ I_j = \int_{S \times S} W^g_j(z \to y) L(z \to y) G(x \to y) dA(y) dA(z) \]
- Path integral formulation
  \[ I_j = \sum_{k=1}^N \int_{\mathcal{P}_k} f_j(\mathbf{z}) d\mu_k(\mathbf{z}) = \int_{\mathcal{P}_k} f_j(\mathbf{z}) d\mu_k(\mathbf{z}) \]
- Bidirectional path tracing
  - Multiple importance sampling for quasi-Monte Carlo integration
  - How much is sacrificed by randomized quasi-Monte Carlo integration?
  - Adapt to two-dimensional structure of integral equation
    - padded replications sampling

**Multiple Importance Sampling**
- \( N \) techniques to generate samples with associated probability density functions
  \[ P_1, P_2, \ldots, P_N : D \to \mathbb{R}_d^+ \]
- Heuristic
  \[ w_1, w_2, \ldots, w_N : D \to \mathbb{R}_d^+ \]
  \[ \sum_{i=1}^N w_i(x) = 1 \text{ for all } x \in D \text{ with } f(x) \neq 0 \]
  \[ w_i(x) = 0 \text{ for all } x \in D \text{ with } p_i(x) = 0 \]
- Estimator
  \[ \int_D f(x) dx \approx \frac{1}{n \cdot m} \sum_{i=1}^N \sum_{j=1}^m f(x_{i,j}) \]
- Example: Balance heuristic
  \[ w_i(x) : P_i(x) \]

**Randomized Quasi-Monte Carlo Integration**
- Low discrepancy point set
  \[ A = \{ a_1, a_2, \ldots, a_m \} \text{ where } a_i \in I^d \]
  \[ r \text{ randomized replications of } A \]
  \[ x_{1,1}, \ldots, x_{1,1}, \ldots, x_{m,1}, \ldots, x_{1,r}, \ldots, x_{1,r}, \ldots, x_{m,r} \]
  \[ U(I^d) \text{ independent} \]
- Estimator
  \[ \int_I f(x) dx \approx \frac{1}{r} \sum_{i=1}^m \sum_{j=1}^r f(x_{i,j}) \]
- Example: Cranley-Patterson rotation (1976)
  \[ x_{i,j} = a_i \oplus \xi_j := (a_i + \xi_j) \mod 1 \quad \text{where } \xi_j \sim U(I^d) \text{ independent} \]