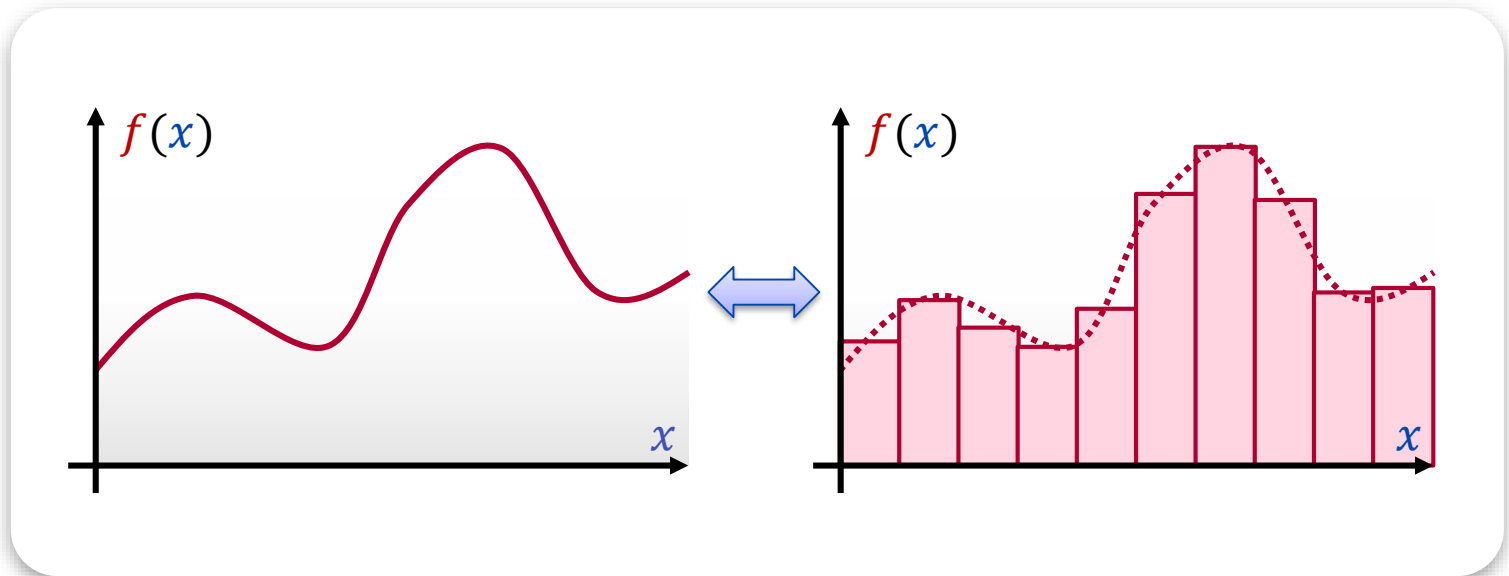


Modelling 1

SUMMER TERM 2020



LECTURE 5

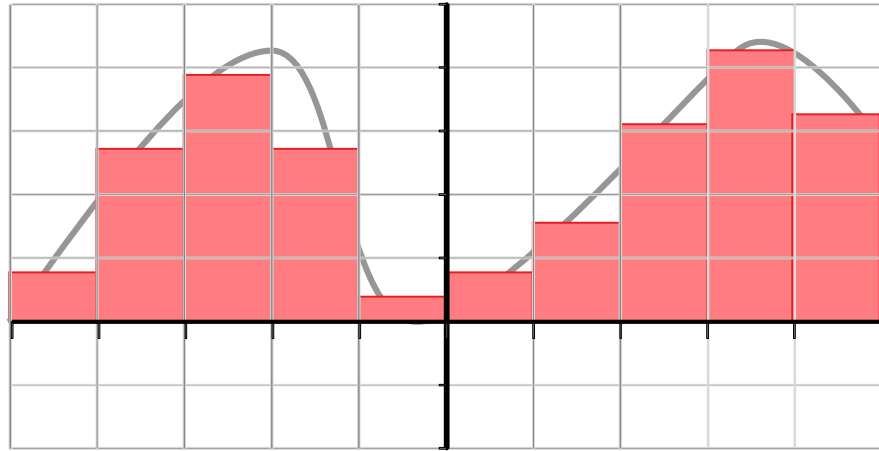
Basic Numerical Calculus

Overview

Objective

- Simple “finite-difference” numerics
 - Helps understanding the theory
- More precision: Interpolation
- More sophisticated: “linear ansatz”
 - Basis for “finite elements”
- Monte-Carlo integration
 - Accuracy mediocre
 - Suitable for high-dimensional problems
 - No “aliasing problems” (later chapter)

Approximation of Function Spaces



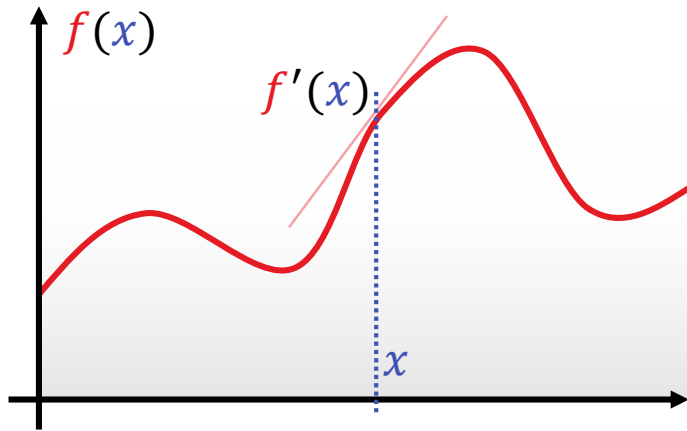
Parametrization as array of numbers

- Sample function f on discrete grid
- Store sample values
- Use this as intuition: $\int \rightarrow \Sigma$, $\frac{d}{dx} \rightarrow \frac{f_i - f_{i-1}}{h}$

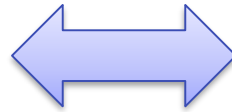
Differentiation

Discrete Representation

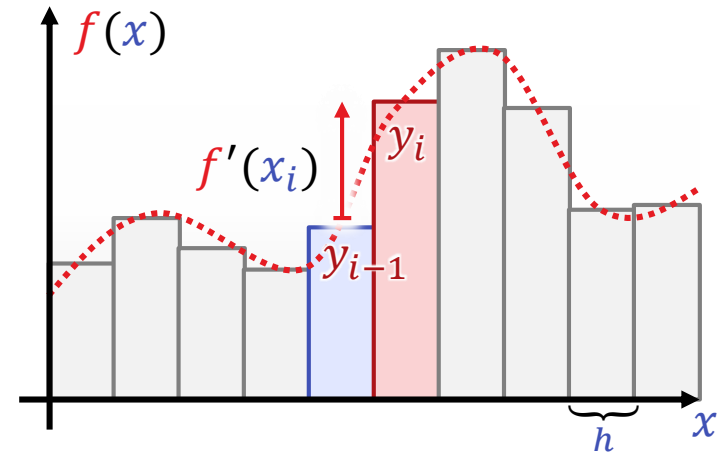
Function f



tangent slope



Think of this:



neighborhood differences

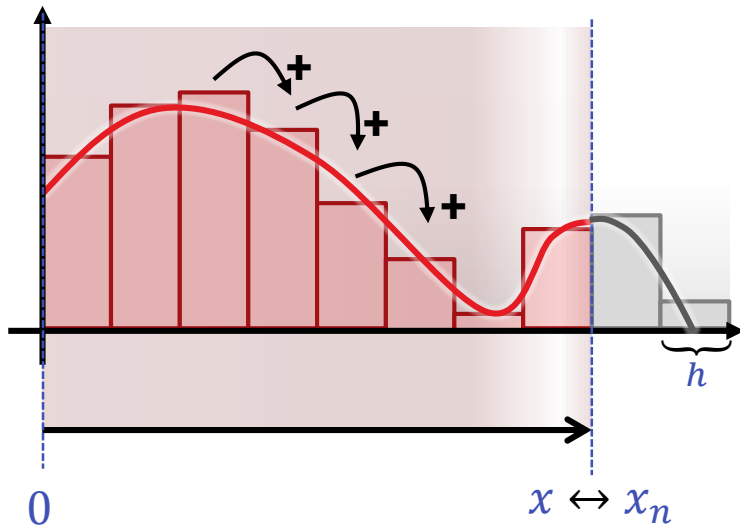
$$f: \mathbb{R} \rightarrow \mathbb{R}$$

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

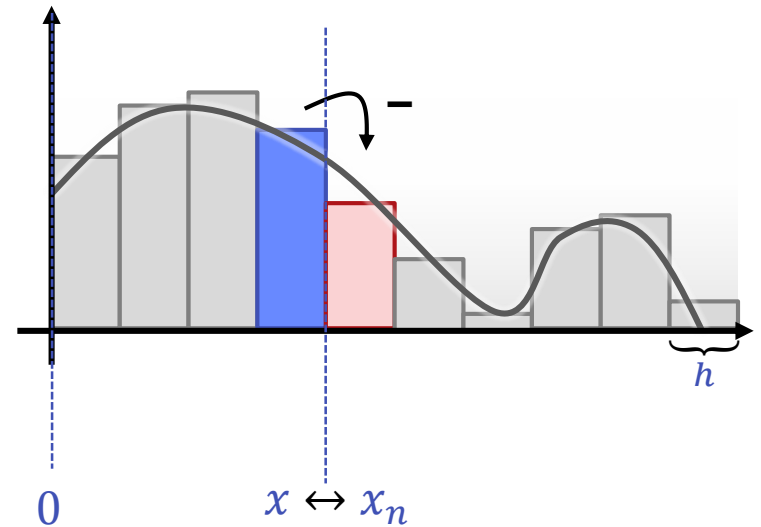
$$f = (y_1, \dots, y_n)$$

$$f'(x_i) \approx \frac{y_i - y_{i-1}}{h}$$

Integration and Differentiation

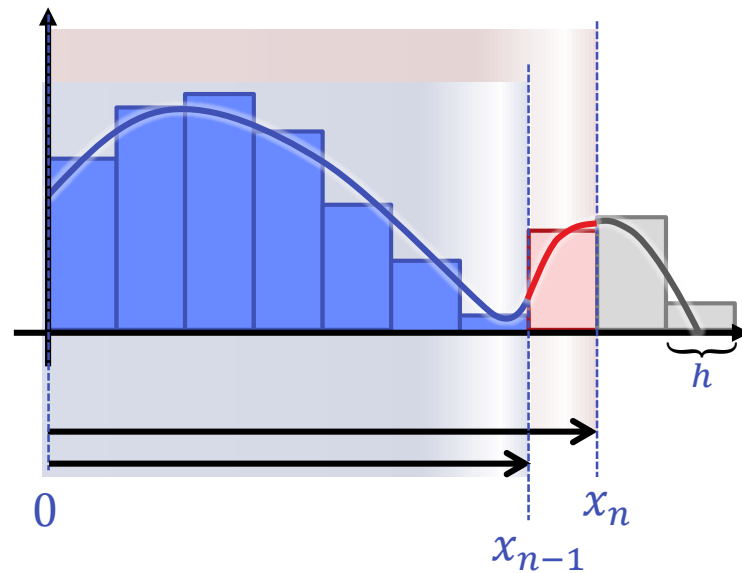


$$F(x_n) \approx h \cdot \sum_{i=0}^n y_i$$



$$f'(x_n) \approx \frac{y_n - y_{n-1}}{h}$$

Fundamental Theorem of Calculus



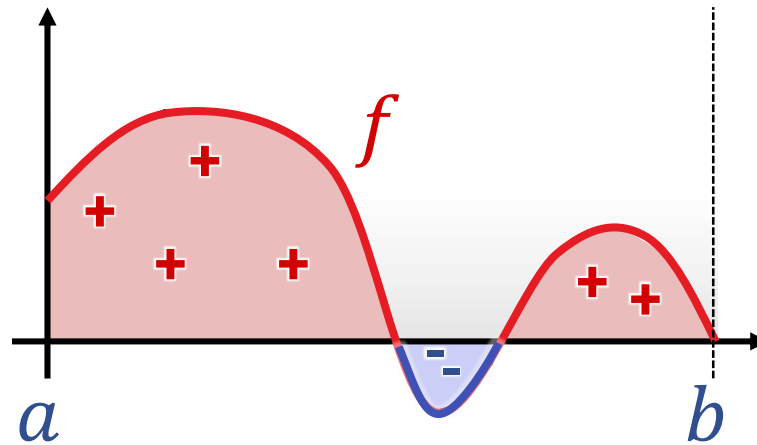
$$\frac{d}{dx} F(x) = f(x)$$

discrete:

$$\frac{1}{h} \left[h \cdot \sum_{i=0}^n y_i - h \cdot \sum_{i=0}^{n-1} y_i \right] = y_n$$

Integration

Integral



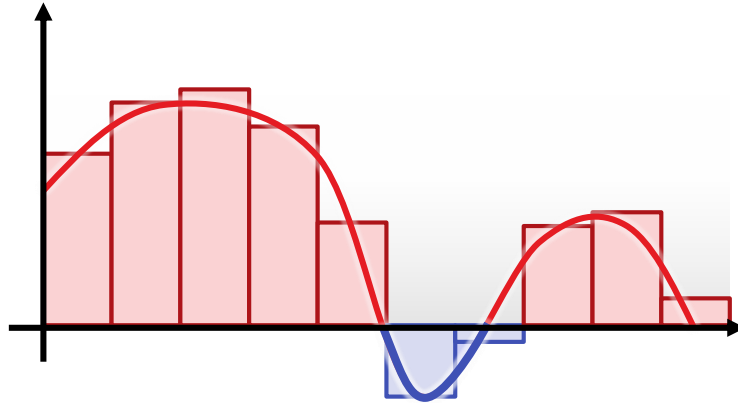
Integral of a function

- Function $f: \mathbb{R} \rightarrow \mathbb{R}$
- Integral

$$\int_a^b f(x) dx$$

measures signed area under curve

Integral



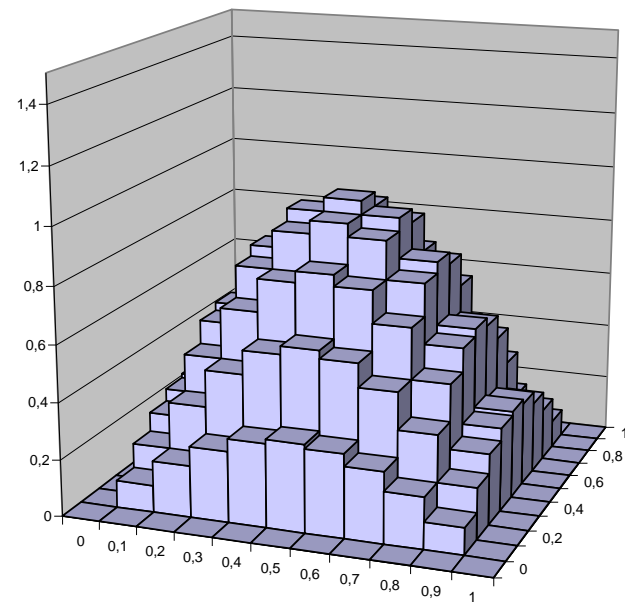
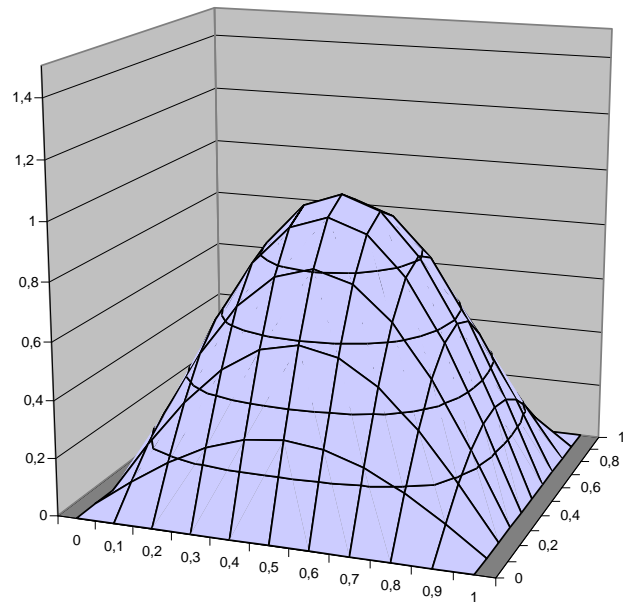
Numerical Approximation

- Sum up a series of approximate shapes
- (Riemannian) Definition: limit for baseline \rightarrow zero
- Intuition: Sum of numbers in array

Multi-Dimensional Integral

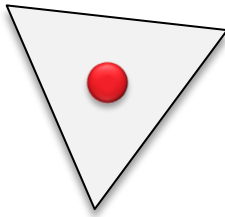
Integration in higher dimensions

- Functions $f: \mathbb{R}^n \rightarrow \mathbb{R}$
- Tessellate domain and sum up cuboids

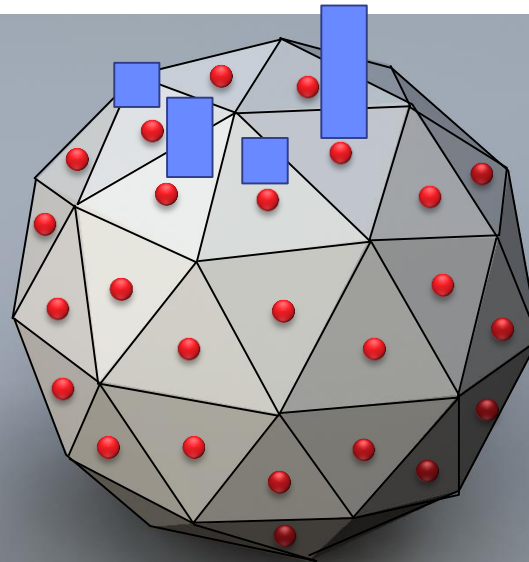
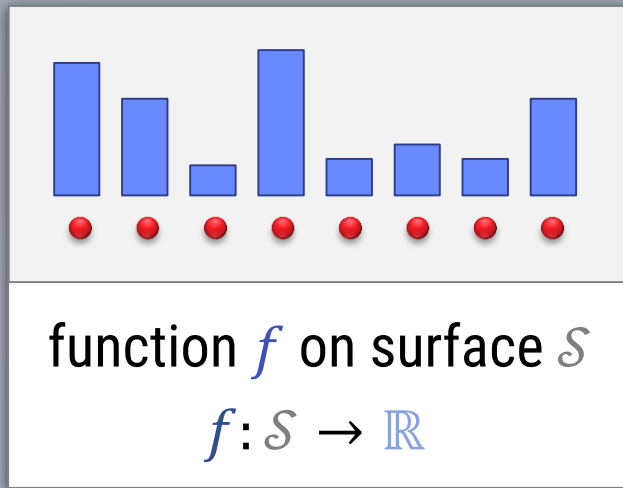


Surface Integrals

Line / Surface / Volume / Hypervolume Elements



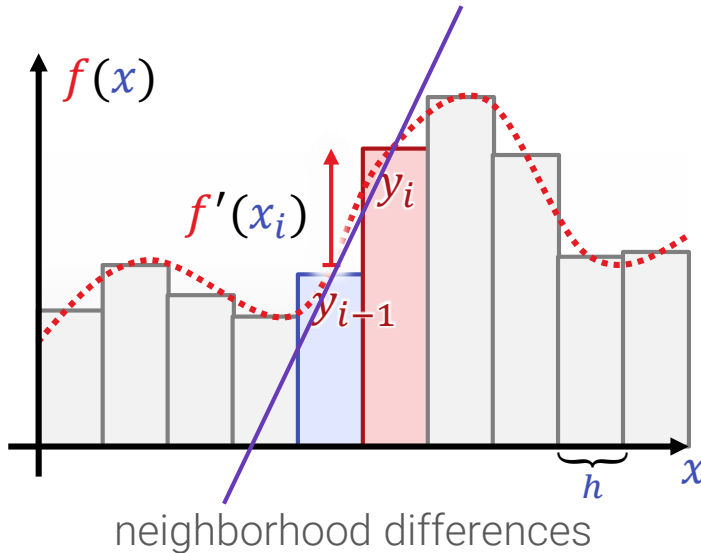
$$\int_{\mathcal{S}} f(\mathbf{x}) d\mathbf{x} = \lim_{\text{smaller } \triangle} \sum_{i=1}^n f(\mathbf{x}_i) \cdot |\triangle_i|$$



More Precision
(higher consistency order)

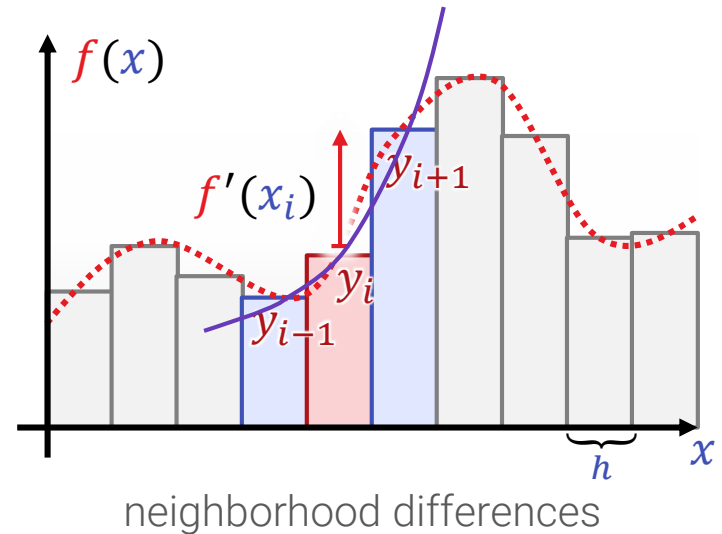
Discrete Representation

Linear:



$$f = (y_1, \dots, y_n)$$
$$f'(x_i) \approx \frac{y_i - y_{i-1}}{h}$$

Quadratic:

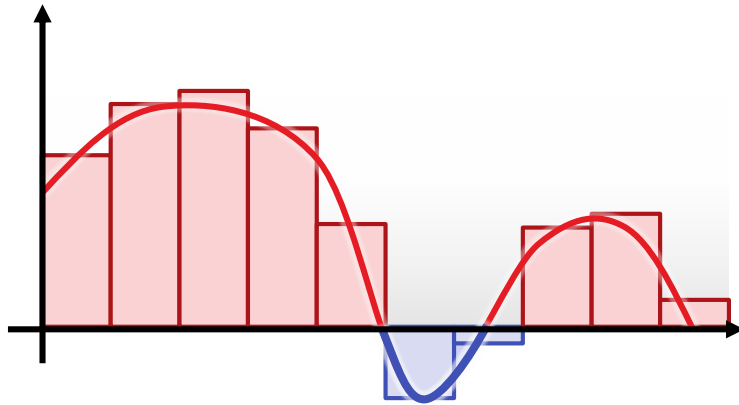


$$f = (y_1, \dots, y_n)$$
$$f'(x_i) \approx \frac{y_{i+1} - y_{i-1}}{2h}$$

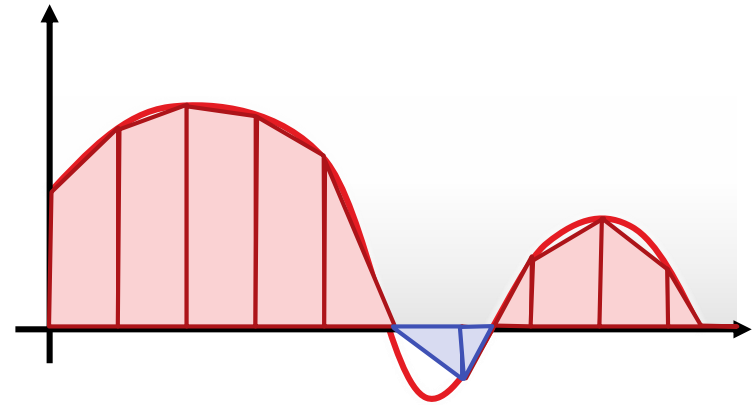
Recipe: Fit polynomial, then take its (analytic) derivative

Integral

Constant



Linear



Recipe

- Same: Fit polynomial locally
- Compute its integral (analytically)
- Converges more quickly for smooth enough functions

Consistency Order

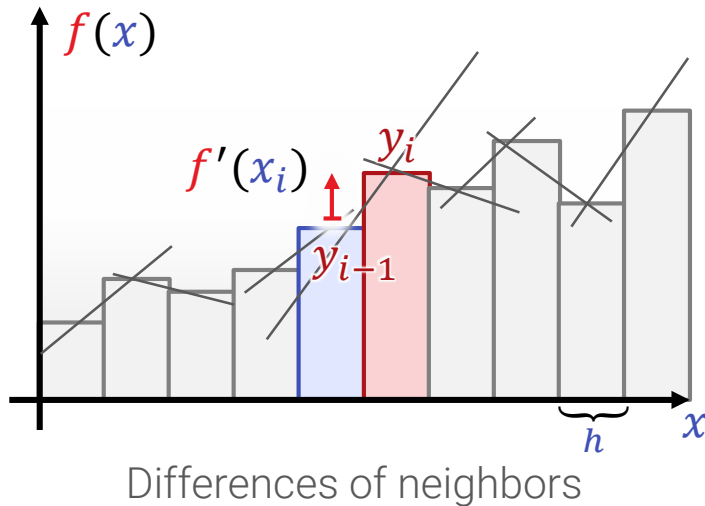
Numerical approximation

- “has consistency order k ”
means
- For polynomials of degree k ,
the result is exact
- For smooth functions, convergence is faster
 - In non-smooth case, high consistency order can have
adverse effect
- Too high order becomes unstable (we’ll see later why)
 - Stay in the lower single digits

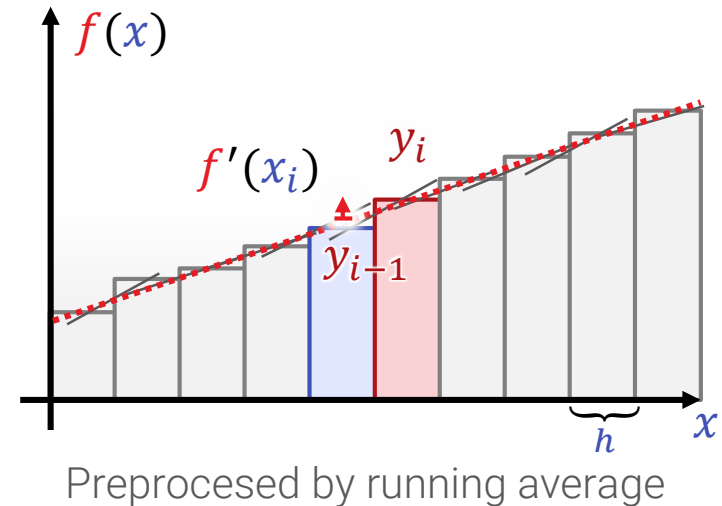
Noisy Data

Differentiation is ill-posed (sensitive)

Raw Data



Smoothed

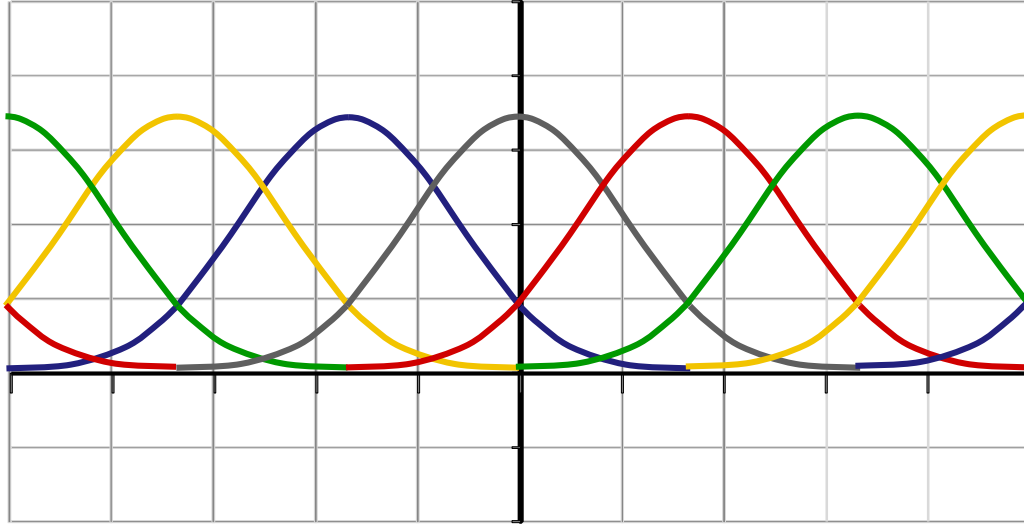


Estimate derivatives in noisy data

- Filter the data first
- E.g., running average over neighbors
- For example: Gaussian filter kernel

Linear Ansatz

Approximation of Function Spaces



Parametrization with a linear ansatz

$$\blacksquare f(x) = \sum_{i=1}^d \lambda_i b_i(x) \quad \rightarrow \quad \begin{cases} \frac{d}{dx} f(x) = \sum_{i=1}^d \lambda_i b_i'(x) \\ \int_{\Omega} f(x) dx = \sum_{i=1}^d \lambda_i \int_{\Omega} b_i(x) dx \end{cases}$$

precompute (often)

precompute (often)

Monte-Carlo Integration

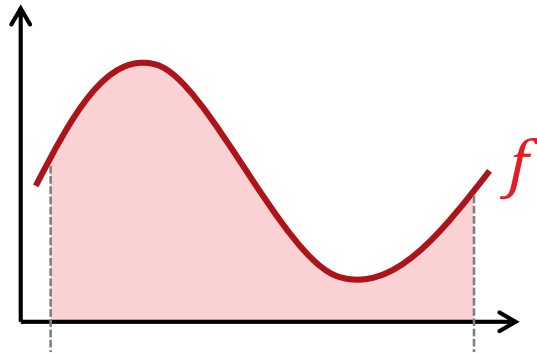


Monte-Carlo Integration

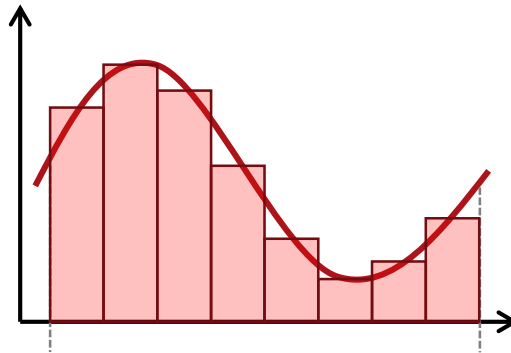
Monte-Carlo integration

- Black-box integrator
- Easy to code/understand,
 - Generally poor accuracy
- But suitable for high-dimensional problems
 - Often the only option for large dimension!
- No “aliasing problems”
 - We will need this later (signal processing)
- Conceptually interesting / important
 - “Importance sampling” is in general a useful idea

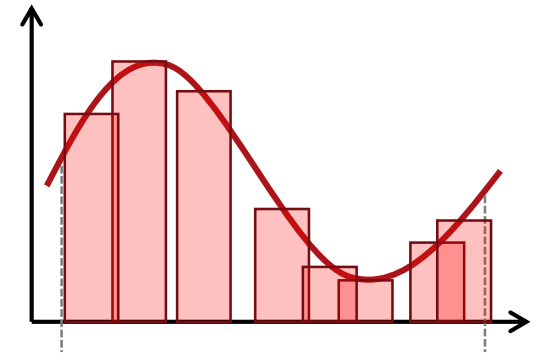
Numerical Integration



integral



Riemann-sum



Monte-Carlo

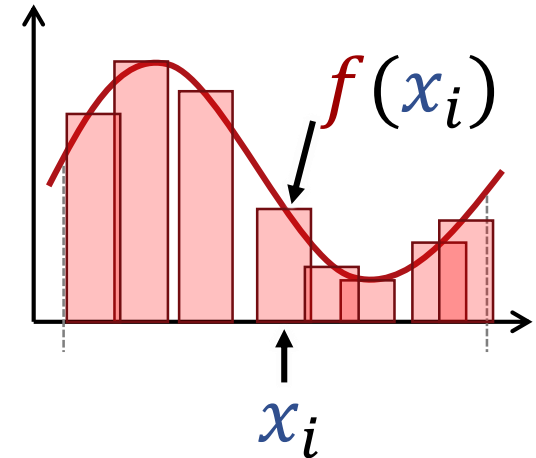
Numerical Integration

- **Standard (Riemann integral):**
 - Grid in Ω , sum of cuboids
- **Monte-Carlo:**
 - Random points $x_i \in \Omega$
 - Compute average $\times |\Omega|$

Monte-Carlo Approach

Formally:

$$\int_{\Omega} f(x) dx \approx \underbrace{\frac{|\Omega|}{n}}_{\text{"secondary estimator"}} \sum_{i=1}^n \underbrace{f(x_i)}_{\text{"primary estimator"}}$$



Sampling

- n random points
 - independently chosen (iid)
 - uniformly distributed on Ω
- Expected value is true integral

Question: How good is the estimate?

Quality of the estimate

Law of large numbers

- Probability of finite deviations $\epsilon > 0$ from expected value converges to zero

Variance

$$\text{Var} \left(\frac{|\Omega|}{n} \sum_{i=1}^n f(x_i) \right) \in \mathcal{O} \left(\frac{\sigma^2}{n} \right)$$

Standard deviation

$$\sigma \in \mathcal{O} \left(\frac{\sigma}{\sqrt{n}} \right)$$

Proof

Computing the Variance:

$$\begin{aligned}\text{V ar} \left(\frac{|G|}{n} \sum_{i=1}^n f(x_i) \right) &= \frac{|G|^2}{n^2} \text{V ar} \left(\sum_{i=1}^n f(x_i) \right) \\ &= \frac{|G|^2}{n^2} \sum_{i=1}^n \text{V ar} (f(x_i)) \quad (\text{Independence}) \\ &= \frac{|G|^2}{n^2} n \text{V ar} (f) \\ &= \frac{|G|^2}{n} \text{V ar} (f) \in \mathcal{O} \left(\frac{1}{n} \right)\end{aligned}$$

(remark: here $G = \Omega$)

Result

Error estimate

$$\sigma \in \mathcal{O}\left(\frac{\sigma}{\sqrt{n}}\right)$$

- Quadruple sample size ($\times 4$) to half error ($1/2$)
- Typical convergence behavior:
 - Quick first estimate
 - Long computation for good (noise-free) result

Does It Make Sense?

When is Monte-Carlo integration useful?

- Error depends on variance of primary estimator
- Then goes down with sample size

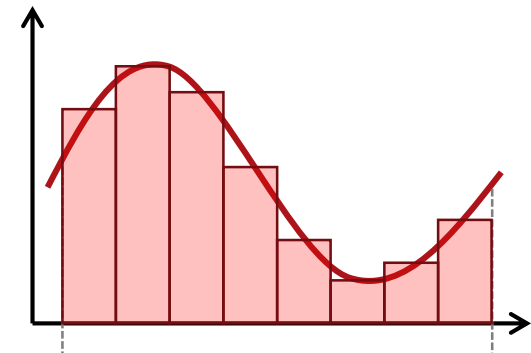
Error is totally independent of

- Dimension of Ω
- Structure of f (discontinuities etc.)
- Structure of Ω
 - Just need to be able to sample it

Higher Dimensions

Classic application domain

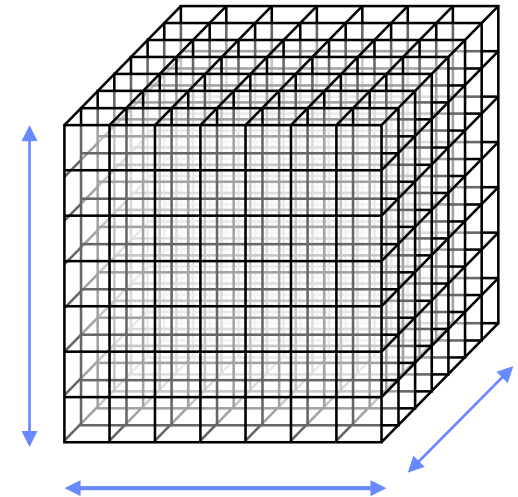
- High-dimensional integration domains
- Let's say, $\Omega = [0,1]^{20}$



Rieman-sum

Standard Integration

- Regular grid, k^{20} samples
- Don't even try this...



k subdivisions
per axis

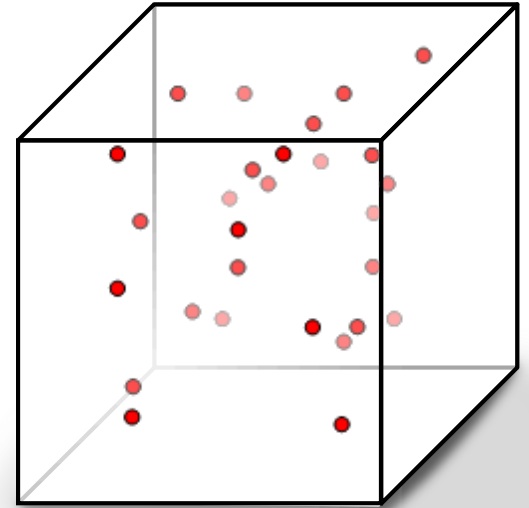
Higher Dimensions

Monte-Carlo Approach:

- Sample n points
- Compute average
- Multiply with domain volume

Property

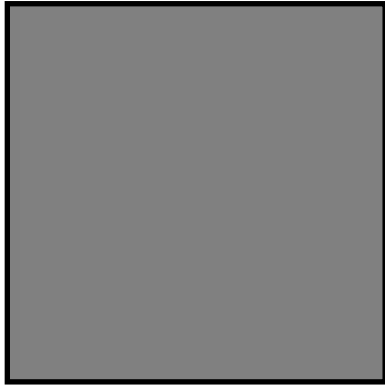
- Works if variance is not too large
- Dimension irrelevant



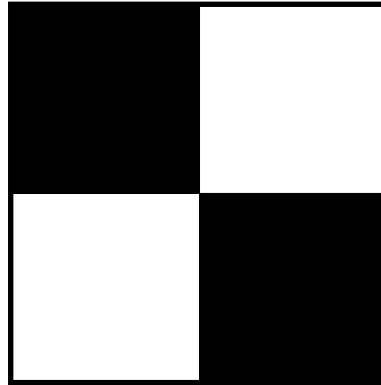
n sample points
(irregular)

Example

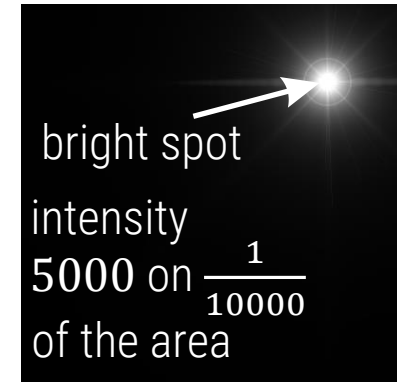
When is Monte-Carlo integration possible?



optimal –
no variance



moderate variance –
MC-int. possible

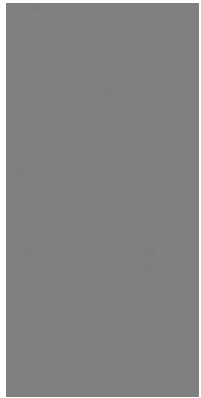


large variance –
not efficient

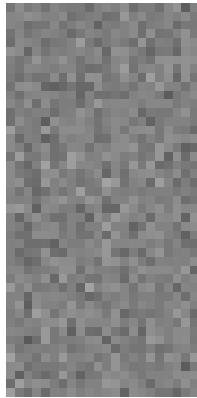
General observation

- Randomized algorithms are efficient if the *crucial information* is *easy to find* by random trials

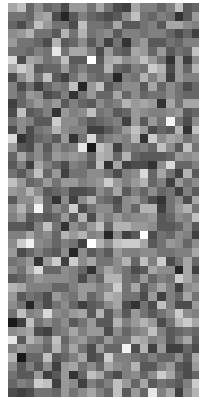
Numerical Example



$q = 1$



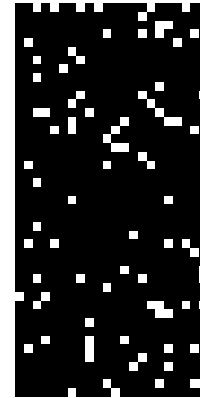
$q = 0.5$



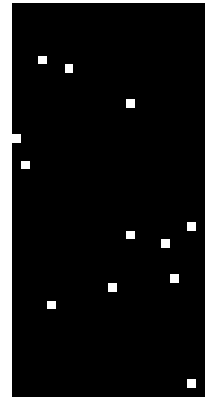
$q = 0.1$



$q = 0.01$



$q = 0.001$



$q = 0.0001$

Averaging Samples:

- $n = 100$ samples
- Fraction q of the domain with value $0.5/q$
- Showing multiple pixels

Example

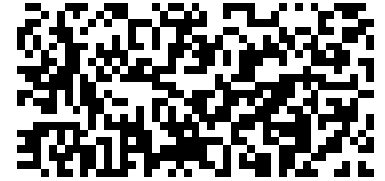
Speed of convergence:

- Now growing n
- Pixel: 50% black / 50% white
- Growing sample size

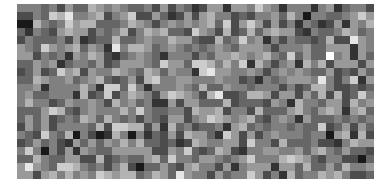
Observation

- Large sample size required before noise becomes invisible

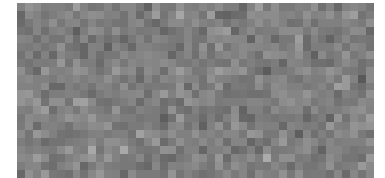
$n = 1$



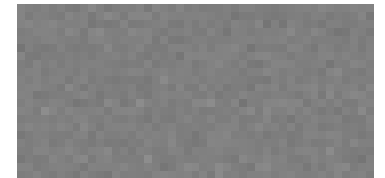
$n = 10$



$n = 100$



$n = 1000$



$n = 10000$

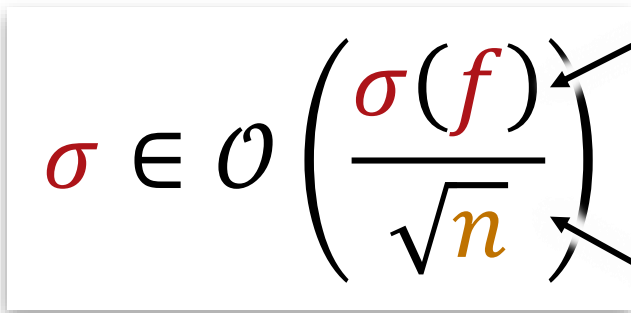


Variance Reduction



Variance Reduction

Two reasons for long compute times

$$\sigma \in \mathcal{O} \left(\frac{\sigma(f)}{\sqrt{n}} \right)$$


Problem #1

primary estimator
variance

Problem #2

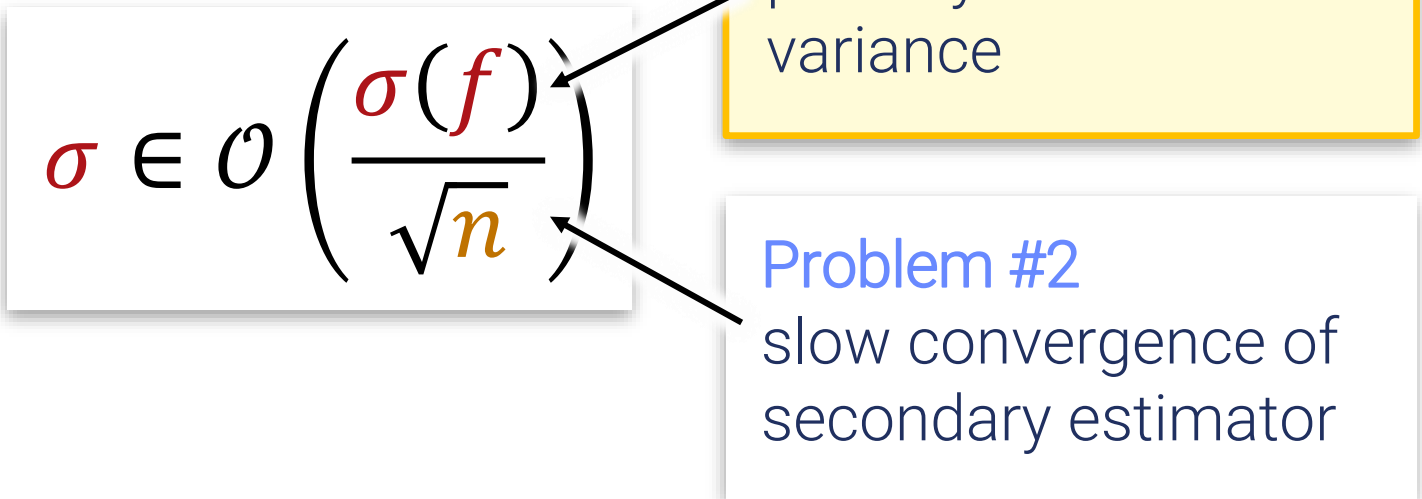
slow convergence of
secondary estimator

Problem #1: *Importance Sampling*

Problem #2: *Stratification*

Variance Reduction

Two reasons for long compute times

$$\sigma \in \mathcal{O} \left(\frac{\sigma(f)}{\sqrt{n}} \right)$$


Problem #1

primary estimator
variance

Problem #2

slow convergence of
secondary estimator

Problem #1: *Importance Sampling*

Problem #2: *Stratification*

Importance Sampling

Importance Sampling

- Idea: More samples in important regions
 - Need to weight differently to avoid bias
- New estimator
 - Choose sampling density p on Ω

$$\int_{\Omega} f(x) dx \approx \frac{1}{n} \sum_{i=1}^n \frac{f(x_i)}{p(x_i)} \quad (p(x) > 0 \forall x \in \Omega)$$

- (Note: $|\Omega|$ factor not required here.)
- Sampling density p controls importance

How to choose p ?

What is a good importance function?

- Idea: Minimize errors
- Large values lead to bigger errors

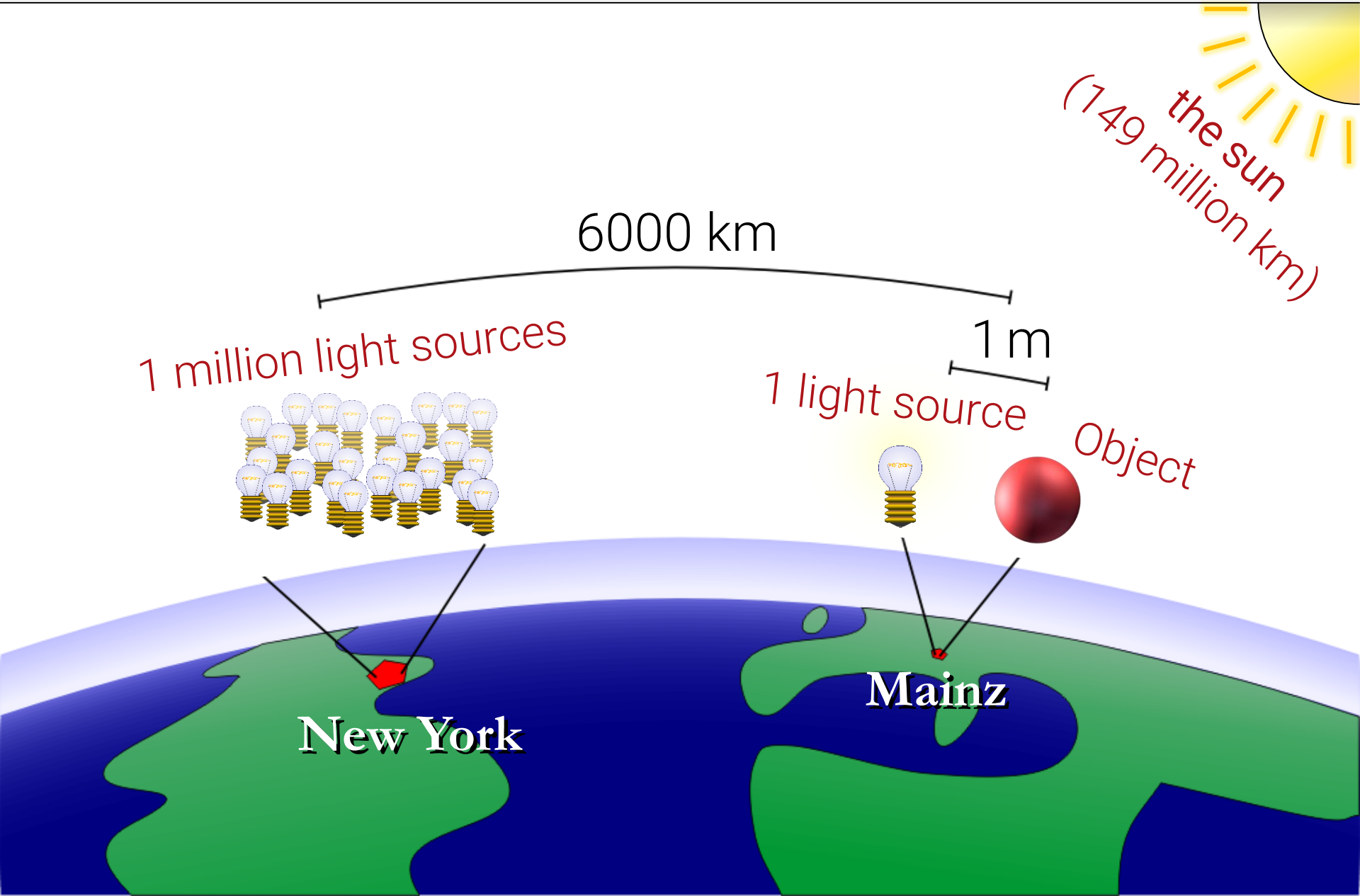
Hypothetical optimum

- $p \sim |f|$ (zero variance for positive f)
 - Not practical
 - Would need to know integral already

In practice

- p similar to f
- Often: $f = g \cdot h$ with known g . Choose $p \sim g$.

Illustrative example



Variance Reduction

Two reasons for long compute times

$$\sigma \in \mathcal{O} \left(\frac{\sigma(f)}{\sqrt{n}} \right)$$

Problem #1

primary estimator
variance

Problem #2

slow convergence of
secondary estimator

Problem #1: *Importance Sampling*

Problem #2: *Stratification*

Stratification

Problem

- Variance of average of *iid* samples always converges with $\mathcal{O}(\sqrt{n})$

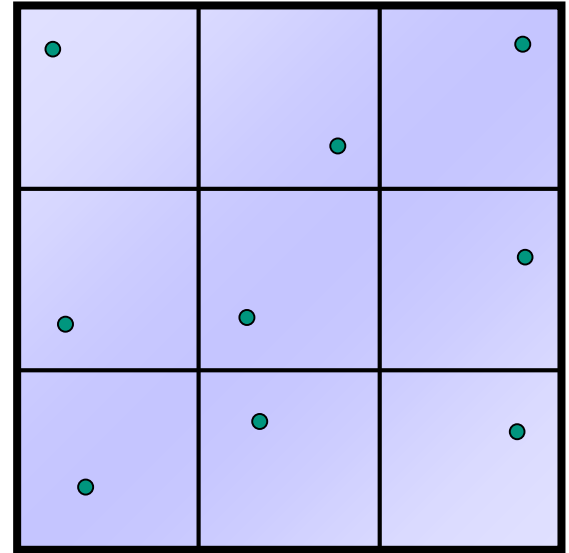
Hence

- Non-independent sampling
- Targeting samples
 - Goal uniform coverage
- Divide Ω in regions („*strati*“)
- One sample point per region

Jittered Sampling

Example: „Jittered Grid“

- Divide Ω in regular grid
 - For example: subpixels
- Random point per cell
- Rest of the algorithm unchanged

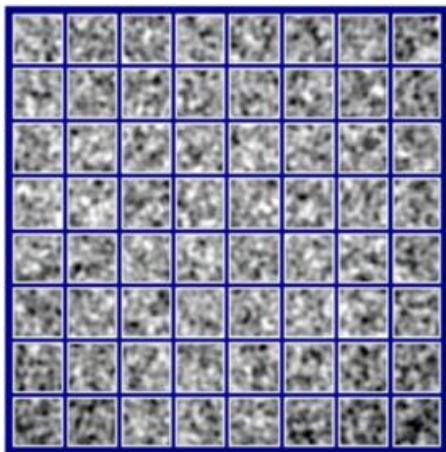


Improved versions

- Poisson-disc sampling
- Halton sequences
- Combination with importance sampling & adaptive sampling

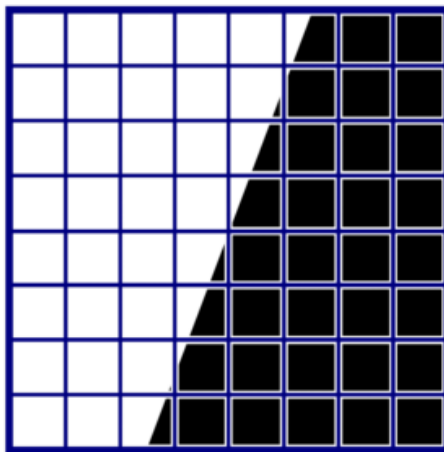
Consequences

Looking at one pixel



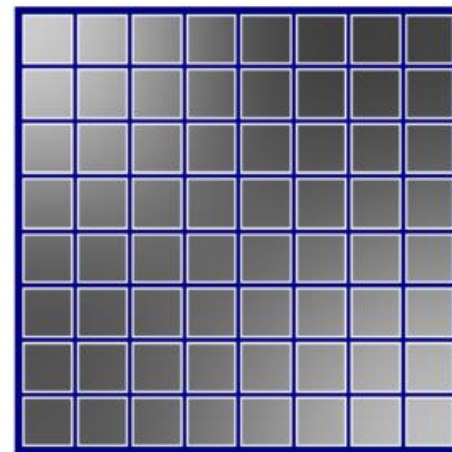
Random image

$$\mathcal{O}\left(n^{-\frac{1}{2}}\right)$$



sharp edge

$$\mathcal{O}\left(n^{-\frac{3}{4}}\right)$$



smooth image

$$\mathcal{O}(n^{-1})$$

(Lipschitz-smooth)

(c.f. Mitchel: „Consequences of Stratified Sampling in Graphics“, Siggraph 96)

Example: sharp boundary

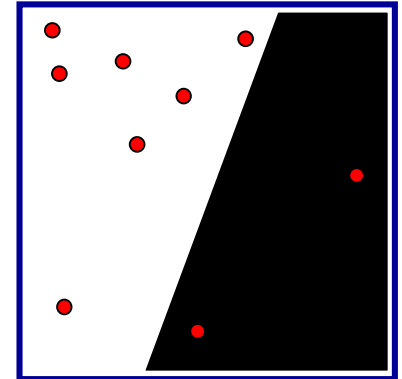
Standard MC-Integration

$$\text{Var}\left(\frac{1}{n}\sum_{i=1}^n f(x_i)\right) = \frac{1}{n}\text{Var}(f(x_i)) \Rightarrow \mathcal{O}(n^{-\frac{1}{2}})$$

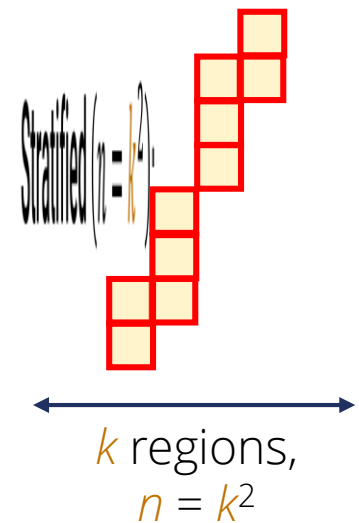
Stratified ($n = k^2$):

$$\begin{aligned} \text{Var}\left(\frac{1}{k^2}\sum_{i=1}^n f(x_i)\right) &= \frac{1}{k^4}\left(\underbrace{\sum_{i=1}^{\mathcal{O}(k)} \text{Var}(f(x_{j_i}))}_{\leq c \text{ (const.)}} + \underbrace{\sum_{i=1}^{\mathcal{O}(k^2)} \text{Var}(f(x_{j_i}))}_{=0}\right) \\ &= \frac{1}{k^3}c \in \mathcal{O}(n^{-\frac{3}{2}}) \Rightarrow \sigma \in \mathcal{O}(n^{-\frac{3}{4}}) \end{aligned}$$

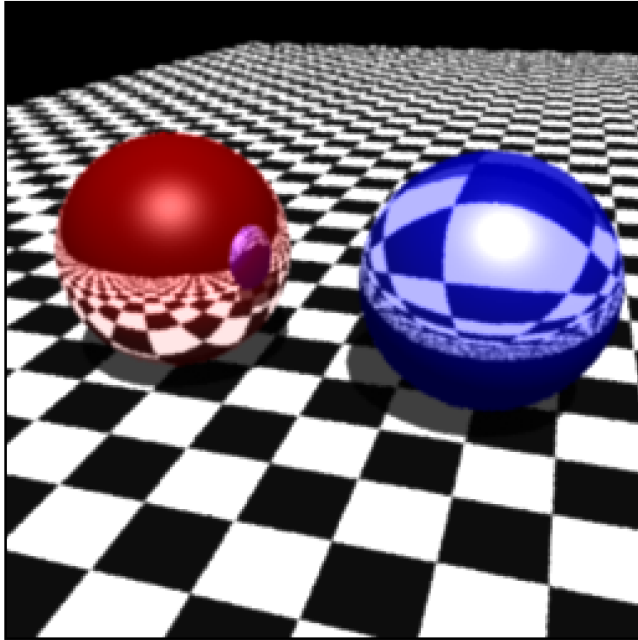
purely random



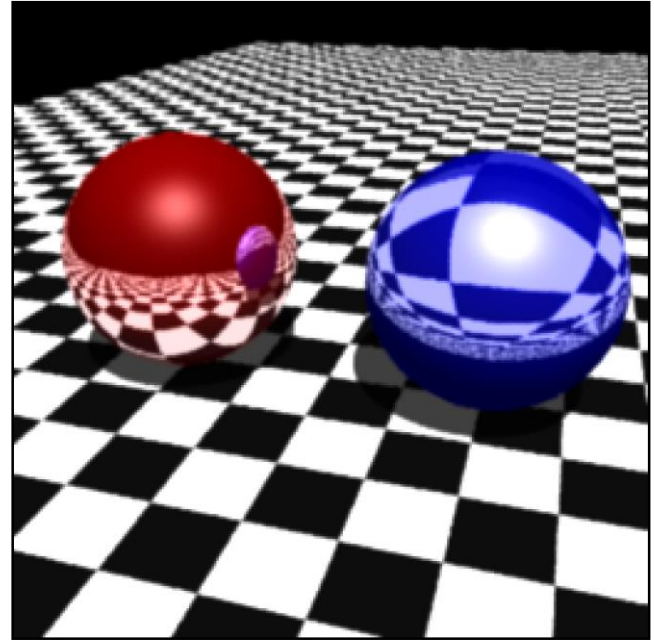
stratified



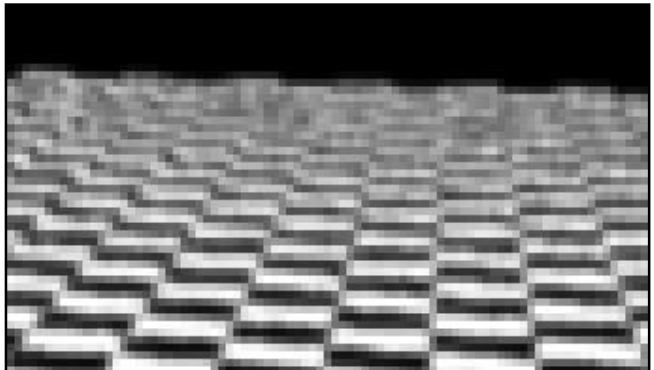
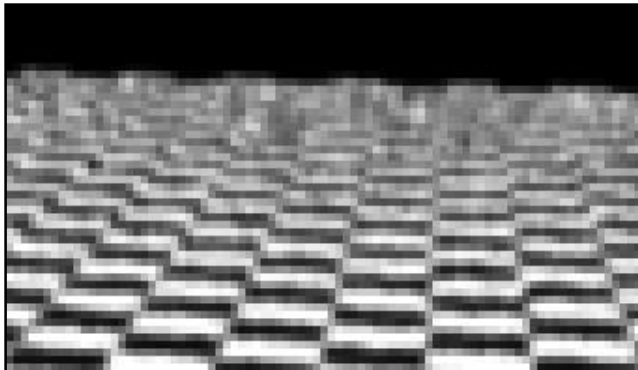
Example

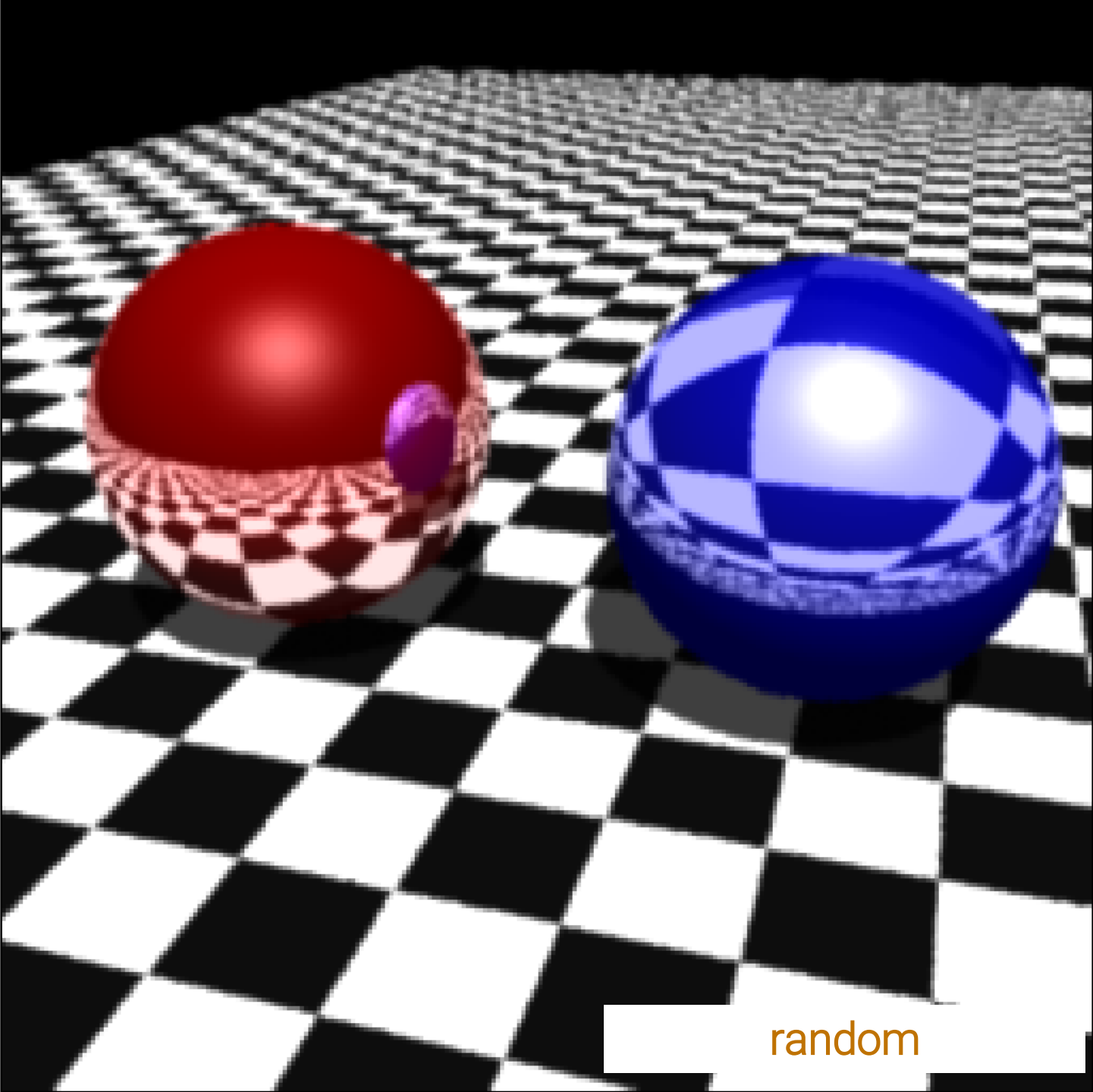


random rays (per pixel)

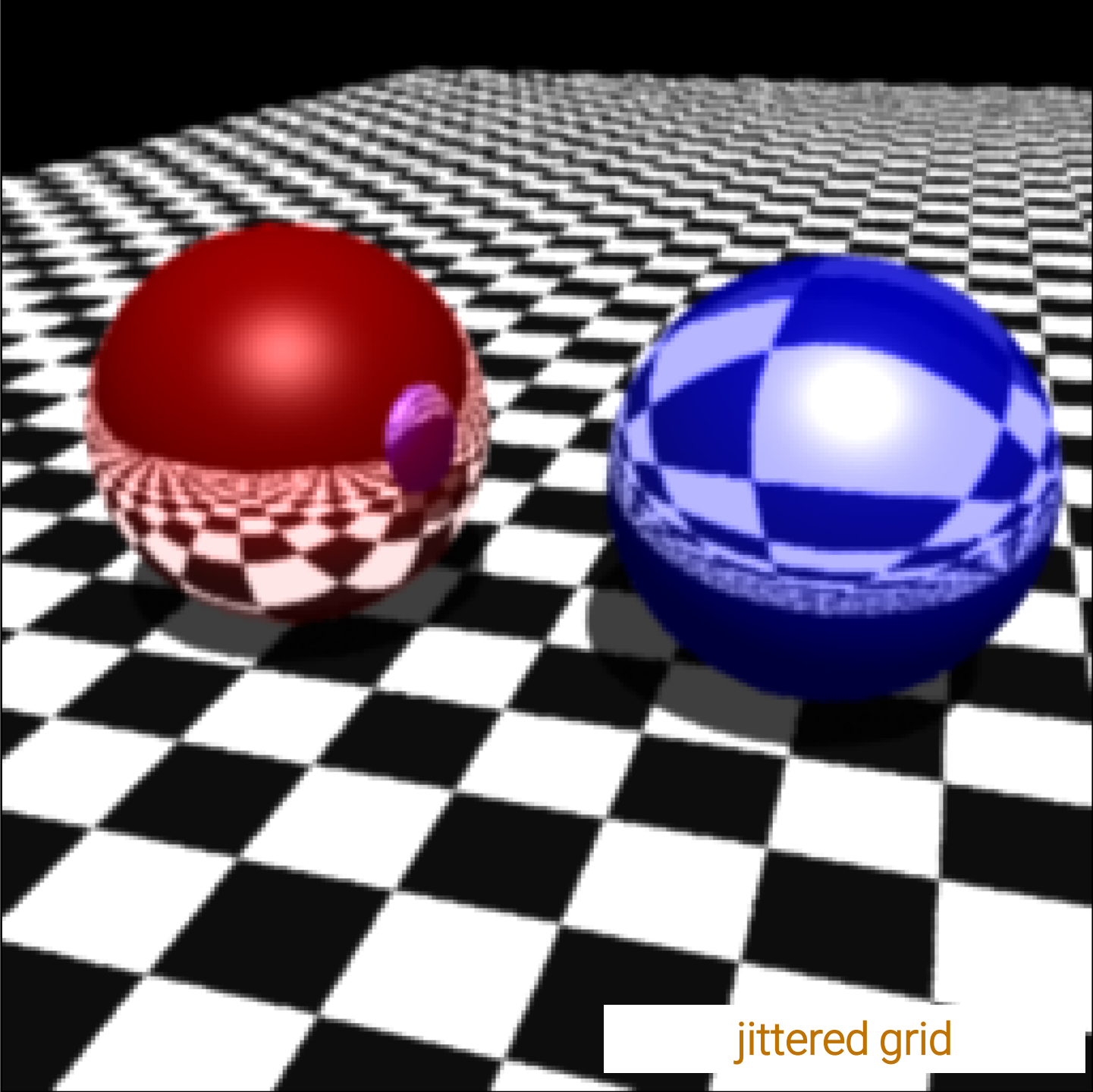


jittered grid



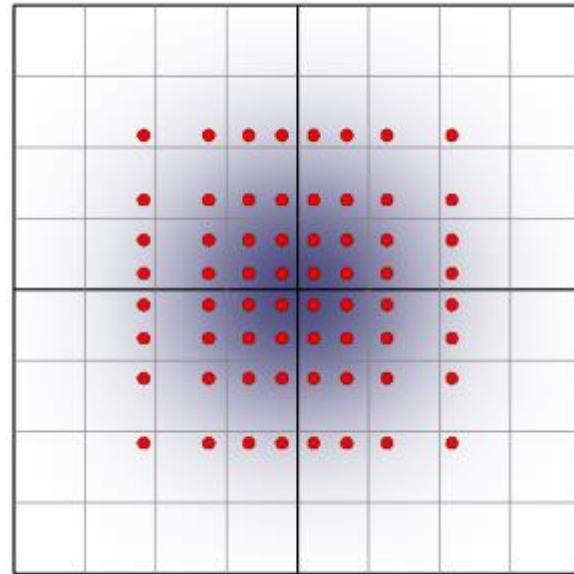
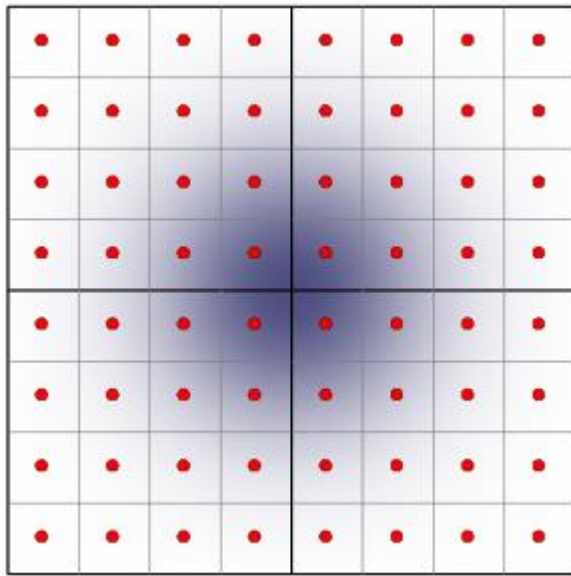


random



jittered grid

Combination



Combination of Stratification & Importance

- Varying sampling density
- Example:
 - 1D grid morphed according to distribution function
 - Tensor product grid (i.e., apply to x and y coordinate)
 - Adaptive jittering