Statistische Methoden der Datenanalyse

Kapitel 3: Die Monte-Carlo-Methode

Professor Markus Schumacher

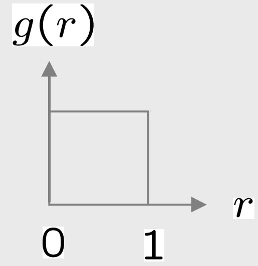
Freiburg / Sommersemester 2009

Basiert auf Vorlesungen und Folien von Glen Cowan und Abbildungen von L Feld, S, Zech, H. Kalinowski.

The Monte Carlo method

What it is: a numerical technique for calculating probabilities and related quantities using sequences of random numbers.

The usual steps:



(1) Generate sequence $r_1, r_2, \ldots, r_m$ uniform in [0, 1].

(2) Use this to produce another sequence $x_1, x_2, \ldots, x_n$ distributed according to some pdf $f(x)$ in which we're interested (x can be a vector).

(3) Use the x values to estimate some property of $f(x)$, e.g., fraction of x values with $a < x < b$ gives $\int_a^b f(x)\, dx$.

→ MC calculation = integration (at least formally)

MC generated values = 'simulated data'

→ use for testing statistical procedures

Random number generators

Goal: generate uniformly distributed values in [0, 1].

Toss coin for e.g. 32 bit number... (too tiring).

→ 'random number generator'

= computer algorithm to generate $r_1, r_2, \ldots, r_n$.

Example: multiplicative linear congruential generator (MLCG)

$n_{i+1} = (a\, n_i) \bmod m$, where

n_i = integer

a = multiplier

m = modulus

n_0 = seed (initial value)

N.B. mod = modulus (remainder), e.g. 27 mod 5 = 2.

This rule produces a sequence of numbers $n_0, n_1, \ldots$

Random number generators (2)

The sequence is (unfortunately) periodic!

Example (see Brandt Ch 4): $a = 3$, $m = 7$, $n_0 = 1$

$$\begin{aligned} n_1 &= (3 \cdot 1)\text{mod}7 = 3 \\ n_2 &= (3 \cdot 3)\text{mod}7 = 2 \\ n_3 &= (3 \cdot 2)\text{mod}7 = 6 \\ n_4 &= (3 \cdot 6)\text{mod}7 = 4 \\ n_5 &= (3 \cdot 4)\text{mod}7 = 5 \\ n_6 &= (3 \cdot 5)\text{mod}7 = 1 \end{aligned}$$

← sequence repeats

Choose a, m to obtain long period (maximum = $m - 1$); m usually close to the largest integer that can represented in the computer.

Only use a subset of a single period of the sequence.

Random number generators (3)

$r_i = n_i/m$ are in [0, 1] but are they 'random'?

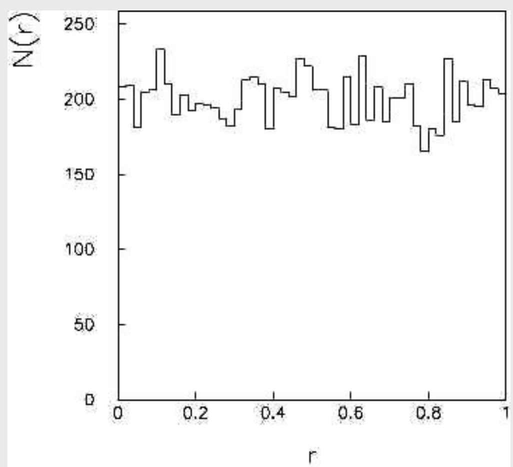
Choose a, m so that the r_i pass various tests of randomness:

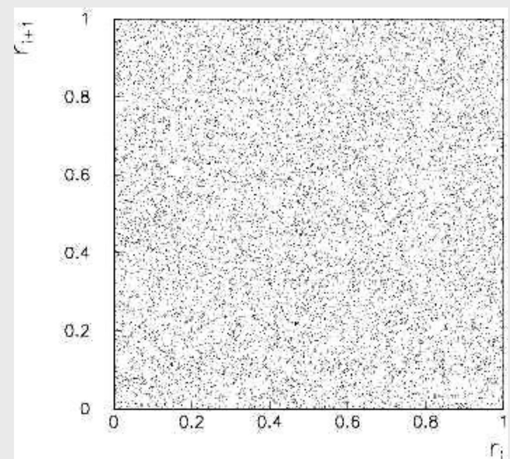
uniform distribution in [0, 1],

all values independent (no correlations between pairs),

e.g. L'Ecuyer, Commun. ACM **31** (1988) 742 suggests

$a = 40692$
$m = 2147483399$



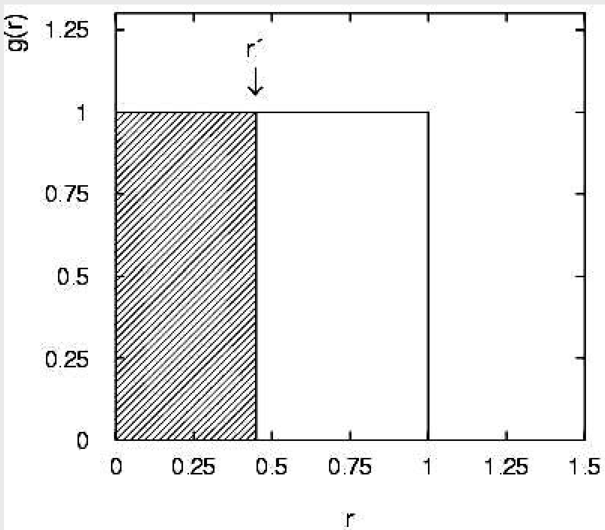


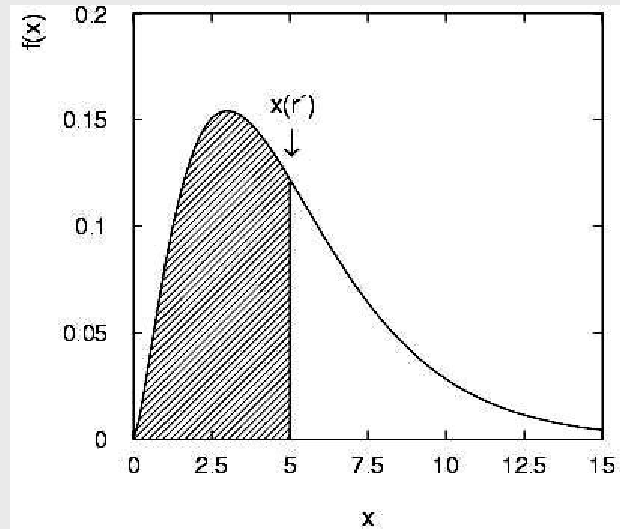
Far better algorithms available, e.g. **`TRandom3`**, period $\approx 10^{6000}$.

See F. James, Comp. Phys. Comm. 60 (1990) 111; Brandt Ch. 4

The transformation method

Given $r_1, r_2, \ldots, r_n$ uniform in [0, 1], find $x_1, x_2, \ldots, x_n$ that follow $f(x)$ by finding a suitable transformation $x(r)$.





Require: $P(r \leq r') = P(x \leq x(r'))$

i.e. $$\int_{-\infty}^{r'} g(r)\,dr = r' = \int_{-\infty}^{x(r')} f(x')\,dx' = F(x(r'))$$

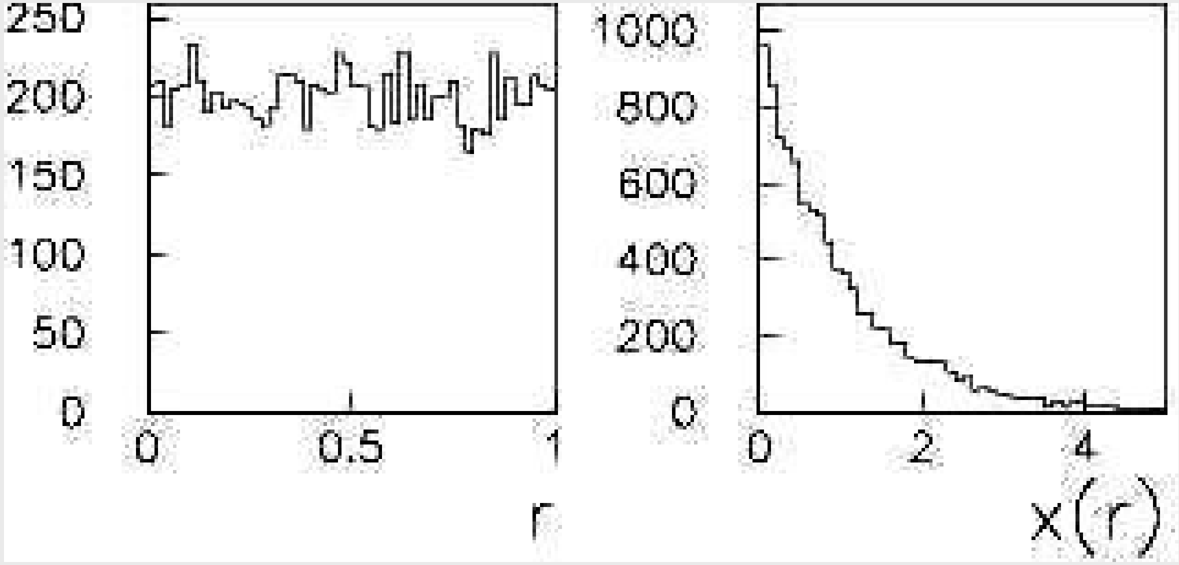
That is, **set $F(x) = r$ and solve for $x(r)$.**

Example of the transformation method

Exponential pdf: $f(x;\xi) = \frac{1}{\xi} e^{-x/\xi} \quad (x \geq 0)$

Set $\int_0^x \frac{1}{\xi} e^{-x'/\xi}\, dx' = r$ and solve for $x(r)$.

→ $x(r) = -\xi \ln(1-r)$ ($x(r) = -\xi \ln r$ works too.)

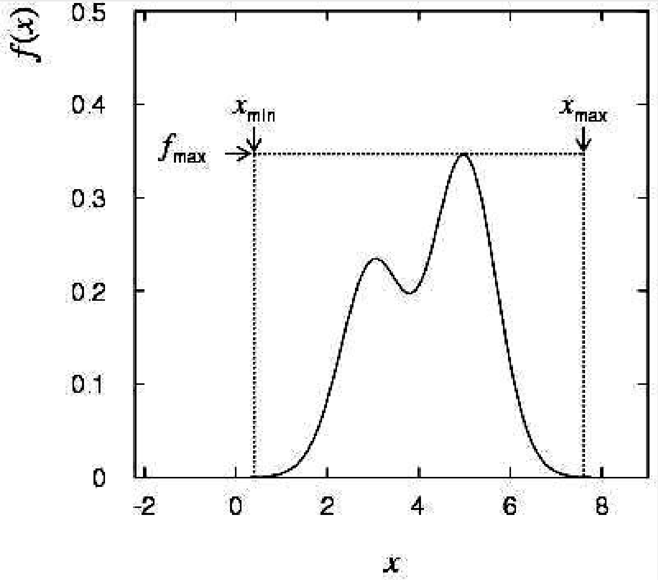


Further transformations

Wahrscheinlichkeitsdichte	Wertebereich	Algorithmus
$f(x) = \frac{1}{b-a}$	$[a,b[$	$x = (b-a)\cdot z + a$
$f(x) = 2x$	$[0,1[$	$x = \max(z_1, z_2)$ or $x = \sqrt{z}$
$f(x) \sim x^{r-1}$	$[a,b[$	$x = \left[(b^r - a^r)\cdot z + a^r\right]^{1/r}$
$f(x) \sim \frac{1}{x}$	$[a,b[$	$a \cdot (b/a)^z$
$f(x) = \frac{1}{x^2}$	$]1,\infty]$	$x = 1/z$
$f(x) = \frac{1}{k} e^{-x/k}$	$]0,\infty]$	$x = -k \ln z$
$f(x) = x e^{-x}$	$]0,\infty]$	$x = -\ln(z_1 \cdot z_2)$
$f(x) = -\ln x$	$[0,1[$	$x = z_1 \cdot z_2$
Gauss: $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-\frac{x^2}{2\sigma^2}}$	$[-\infty,\infty]$	$x = \sigma\sqrt{-\ln z_1^2} \cdot \cos(2\pi z_2)$
Breit–Wigner: $f(x) = \frac{\Gamma}{2\pi} \cdot \frac{1}{(x-\mu)^2 + (\Gamma/2)^2}$	$[-\infty,\infty]$	$x = [\tan\pi(z-0.5)] \cdot \Gamma/2 + \mu$

The acceptance-rejection method

Enclose the pdf in a box:



(1) Generate a random number x, uniform in $[x_{min}, x_{max}]$, i.e.

$$x = x_{\text{min}} + r_1(x_{\text{max}} - x_{\text{min}}) \,,$$ r_1 is uniform in [0,1].

(2) Generate a 2nd independent random number u uniformly distributed between 0 and f_{max}, i.e. $u = r_2 f_{\text{max}}$.

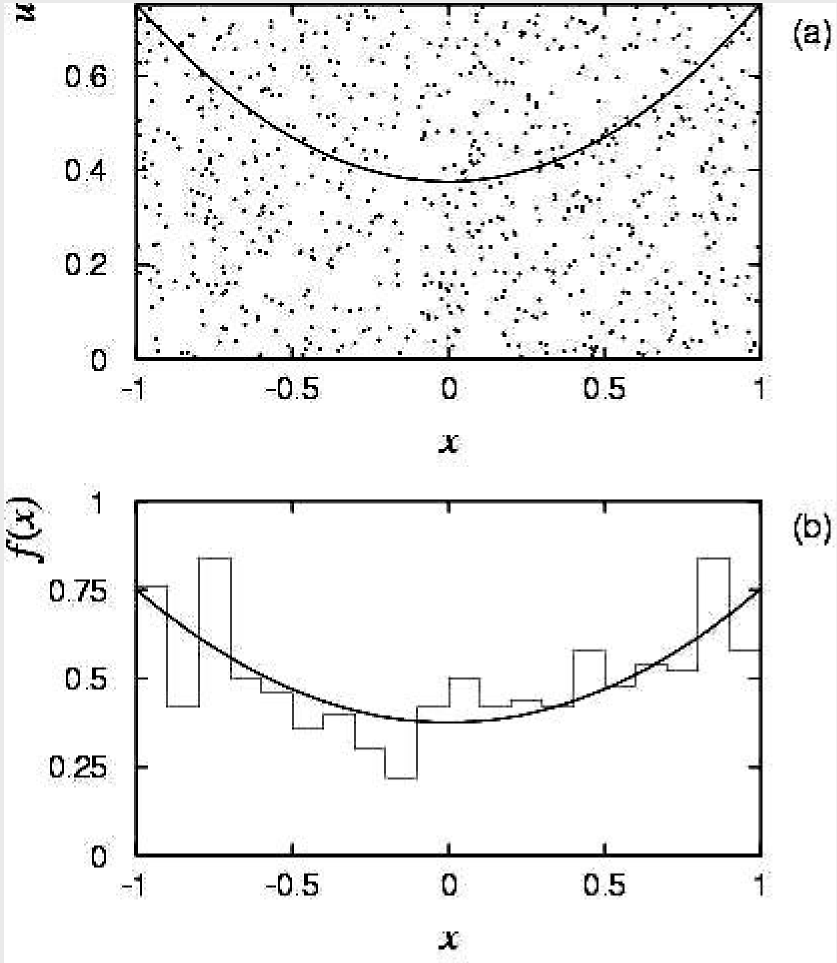
(3) If $u < f(x)$, then accept x. If not, reject x and repeat.

Example with acceptance-rejection method

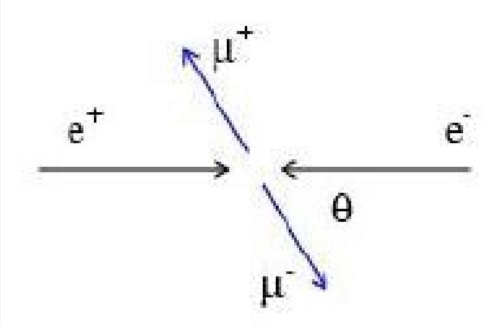
$$f(x) = \frac{3}{8}(1 + x^2)$$

$$(-1 \leq x \leq 1)$$

If dot below curve, use x value in histogram.



Monte Carlo event generators



Simple example: $e^+e^- \rightarrow \mu^+\mu^-$

Generate $\cos\theta$ and ϕ:

$$f(\cos\theta; A_{\mathrm{FB}}) \propto (1 + \frac{8}{3} A_{\mathrm{FB}} \cos\theta + \cos^2\theta) \,,$$

$$g(\phi) = \frac{1}{2\pi} \quad (0 \leq \phi \leq 2\pi)$$

Less simple: 'event generators' for a variety of reactions:

$e^+e^- \rightarrow \mu^+\mu^-$, hadrons, ...

pp $\rightarrow$ hadrons, D-Y, SUSY,...

e.g. PYTHIA, HERWIG, ISAJET...

Output = 'events', i.e., for each event we get a list of generated particles and their momentum vectors, types, etc.

Monte Carlo detector simulation

Takes as input the particle list and momenta from generator.

Simulates detector response:

multiple Coulomb scattering (generate scattering angle),
particle decays (generate lifetime),
ionization energy loss (generate Δ),
electromagnetic, hadronic showers,
production of signals, electronics response, ...

Output = simulated raw data $\rightarrow$ input to reconstruction software:

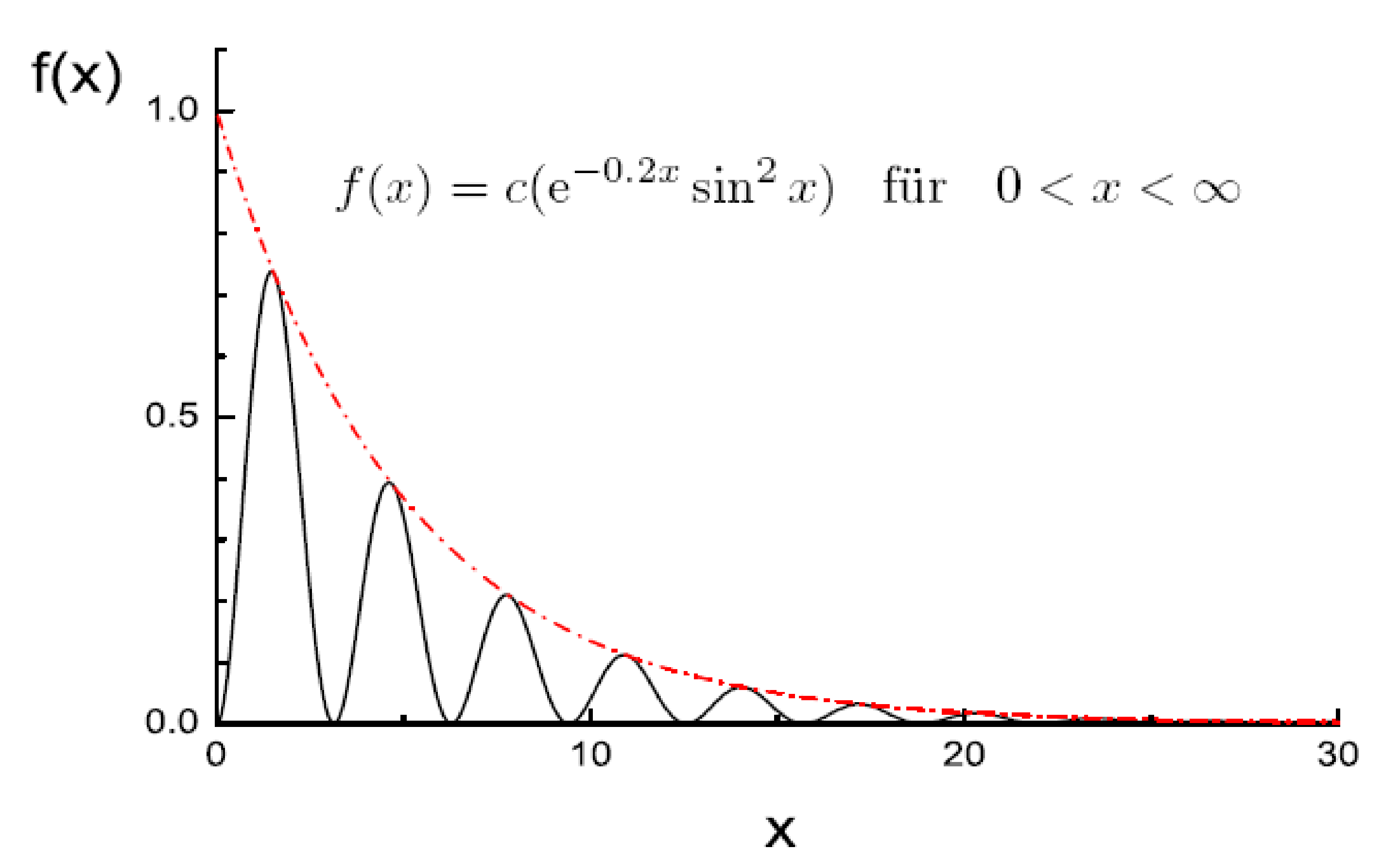
track finding, fitting, etc.

Predict what you should see at 'detector level' given a certain hypothesis for 'generator level'. Compare with the real data.

Estimate 'efficiencies' = #events found / # events generated.

Programming package: **GEANT**

Integration with importance sampling



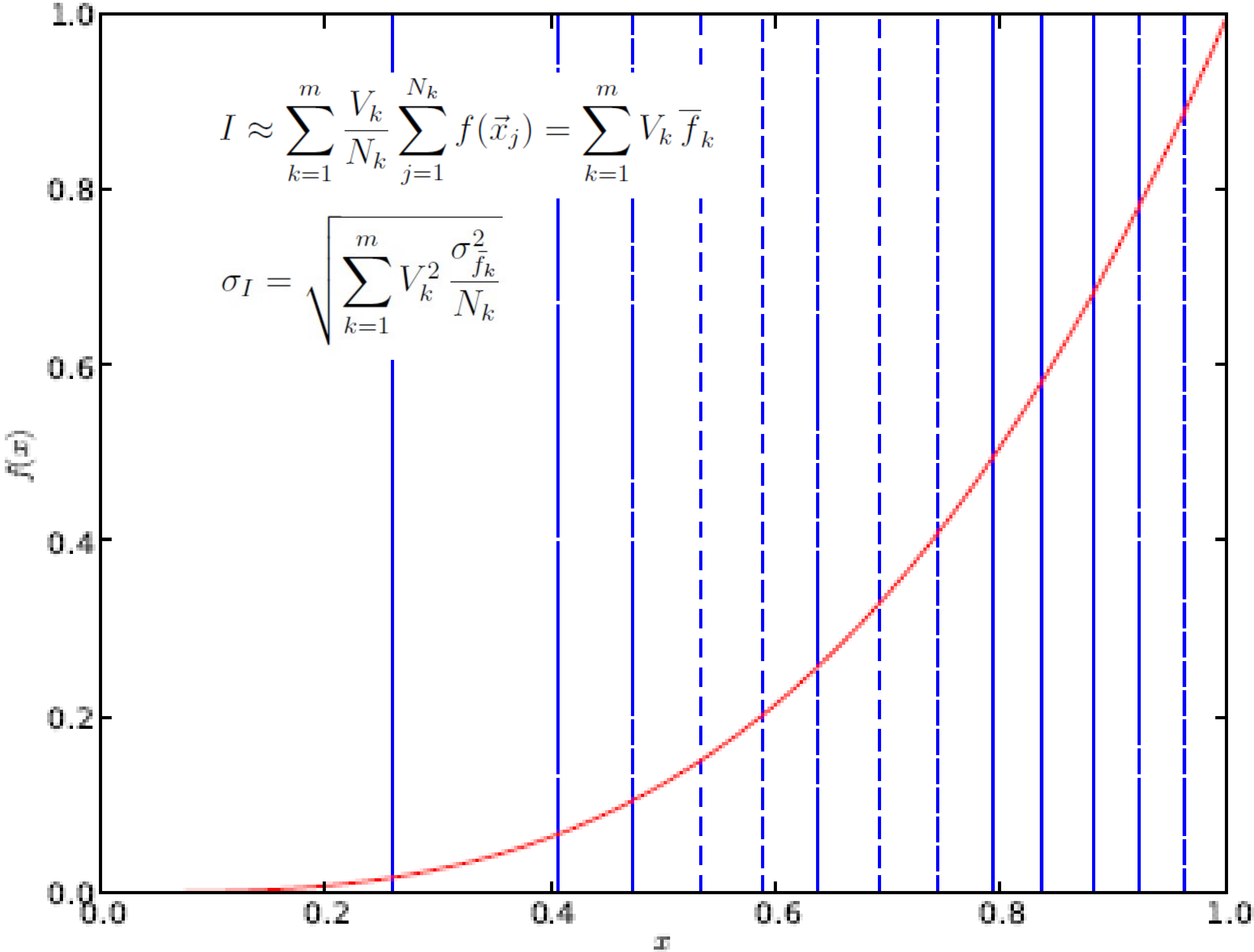
Majorante

$$m(x) = c\,e^{-0.2x}$$

$$x = -\frac{1}{0.2}\ln(1 - r_1)$$

für $r_2 < \sin^2 x \quad \rightarrow$ behalte x,
für $r_2 > \sin^2 x \quad \rightarrow$ verwerfe x

Integration with stratified sampling



$$I \approx \sum_{k=1}^{m} \frac{V_k}{N_k} \sum_{j=1}^{N_k} f(\vec{x}_j) = \sum_{k=1}^{m} V_k \, \overline{f}_k$$

$$\sigma_I = \sqrt{\sum_{k=1}^{m} V_k^2 \, \frac{\sigma_{\overline{f}_k}^2}{N_k}}$$