Karlsruher Institut für Technologie

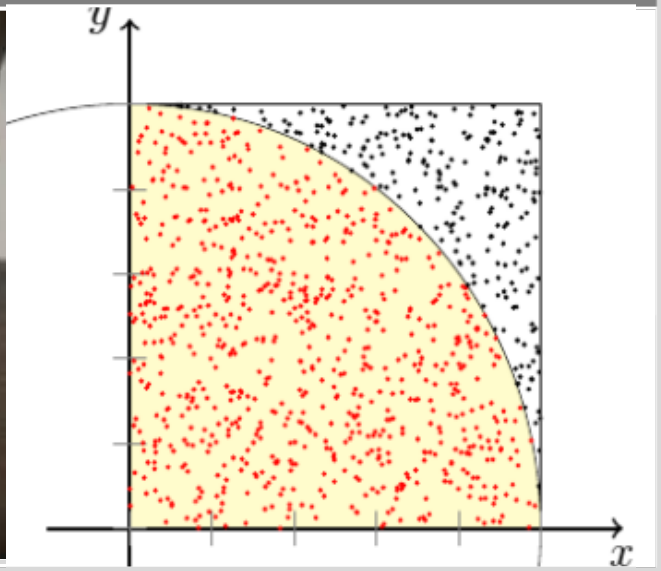
Moderne Methoden der Datenanalyse

Zusammenfassung
Zufallszahlen und Monte Carlo Methoden

Andreas B.Meyer

Fakultät für Physik
Institut für Experimentelle Kernphysik - IEKP

9. Mai 2017



Die Monte Carlo Methode

- Auf (Pseudo-)Zufallszahlen basierende numerische Methode
  - Bestimmung der Eigenschaften von Verteilungen von Zufallszahlen (PDF), z.B. für Messgrößen in Experimenten
  - Lösung viel-dimensionaler Integrale (Transport-, Differential und Integral-Gleichungen)
  - Beispiele bisher: Würfel, Fehlerfortpflanzung, Wettervorhersage, Ziegenproblem

- Vorteile der Methode:
  - Einfache Formulierung der Problemstellungen
  - Unabhängige Kontrolle analytischer Lösungen
  - Kontinuierliche Verbesserung der Genauigkeit (mit mehr CPU)

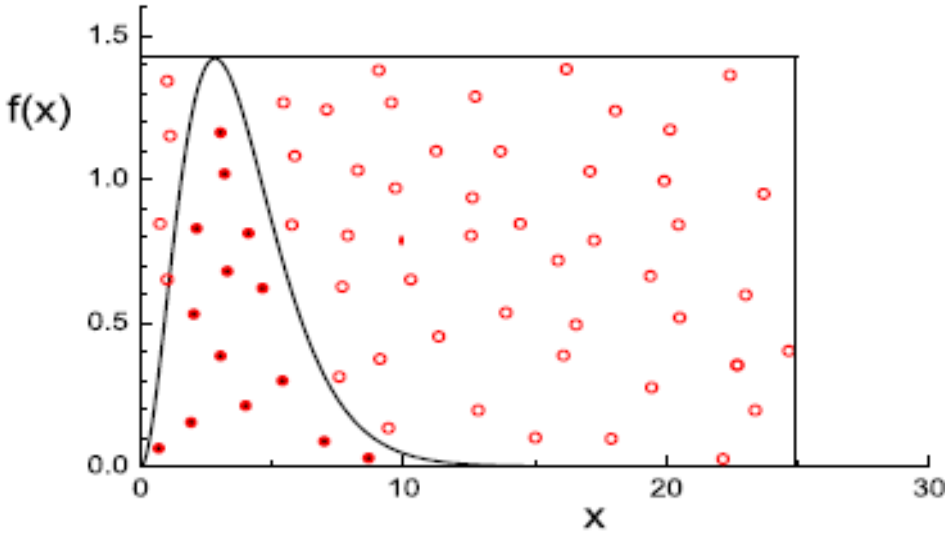
- Aus der Teilchenphysik ist MC nicht wegzudenken:
  - Simulation von Physik-Ereignissen und Detektoreigenschaften
  - Entwerfen und Planen von Experimenten
  - Test von Analyse-Methoden (statistische Tests / “Pseudo-Experimente”)
  - Matrix-Elemente und Phasenraum-Integrale

Grundsätzliches Verfahren

- Erzeuge Folge von Zufallszahlen $r_1, r_2, \ldots r_m$ gleichverteilt in]0,1]

- Transformiere r_i in $x_i = x_1, \ldots x_m$, so dass x_i der gewünschten Verteilungsdichte $f(x)$ folgen.

- Aus den x_i werden dann Eigenschaften von $f(x)$ bestimmt, z.B. das Integral $\int_a^b f(x)dx$ = Anteil der x_i mit $a<x_i<b$



Berechnung von Integralen

$$I = \int_a^b \phi(x)dx \simeq (b-a)\frac{1}{N}\sum_{i=1}^{N}\phi(x_i)$$

- Varianz von I_{MC}:

$$V(I_{MC}) = \sigma^2_{I_{MC}} = V\left[\frac{b-a}{N}\sum_{i=1}^{N}\phi(x_i)\right] = \left(\frac{b-a}{N}\right)^2 V\left[\sum_{i=1}^{N}\phi(x_i)\right]$$

(Zentraler Grenzwertsatz)

$$= \frac{(b-a)^2}{N} V[\phi(x_i)]$$

- V[I_{MC}] ist proportional zu V[φ] und 1/N
- D.h Genauigkeit σ verbessert sich mit 1/$\sqrt{N}$
- Gilt auch für mehrdimensionale Integrale

$$\sigma \propto 1/\sqrt{N}$$

Blobel/Lohrmann Abschnitt 5.6

Erzeugung von Pseudo-Zufallszahlen

- Computer liefern deterministische Zahlenfolgen “Pseudo”-Zufallszahlen:
  - Möglichst gute Verteilung (unkorrelierte Folgen)
  - Lange Periode
  - Schnelle Erzeugung
  - Reproduzierbarkeit (für vorgegebenen Startwert)

- Einfachster Algorithmus:

  $x_n = (a\ x_{n-1}) \bmod 2^k$ (wählt die letzten k bits aus, d.h. bit-weises UND mit 2^k-1)

- Allgemeiner und besser: (“linear kongruenter Generator, LCG”)

  $x_n = (a\ x_{n-1} + c) \bmod m$

  Dann ist $u_n = x_n/m$ gleichverteilt im Intervall [0,1[

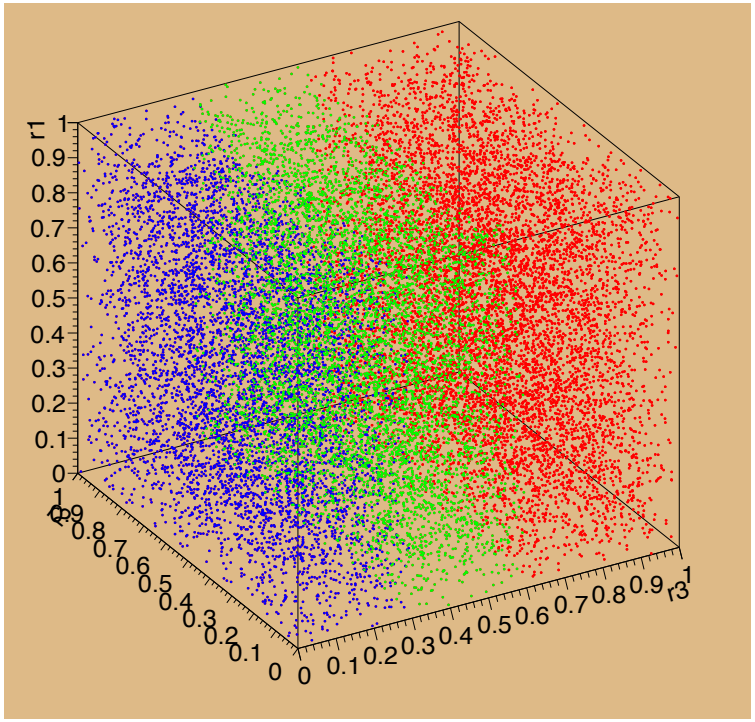
- Periode ist höchstens m. Die Parameter a und c müssen geeignet gewählt werden.

Blobel/Lohrmann Abschnitt 5.2

Mersenne-Twister Algorithmus

Matsumoto, Nishimura (1998)

- basiert auf Mersenne-Primzahlen (2^n-1)
- Zustand wird beschrieben durch 624 Integer Zahlen (32 bit), die als Startwerte mit einem einfachen Generator initialisiert werden können.
- Extrem lange Periode (2^{19937}-1 ≈ 10^{6000})
- Gute Gleichverteilung bis zu 623 Dimensionen (bewiesen)
- Hinreichend schnell



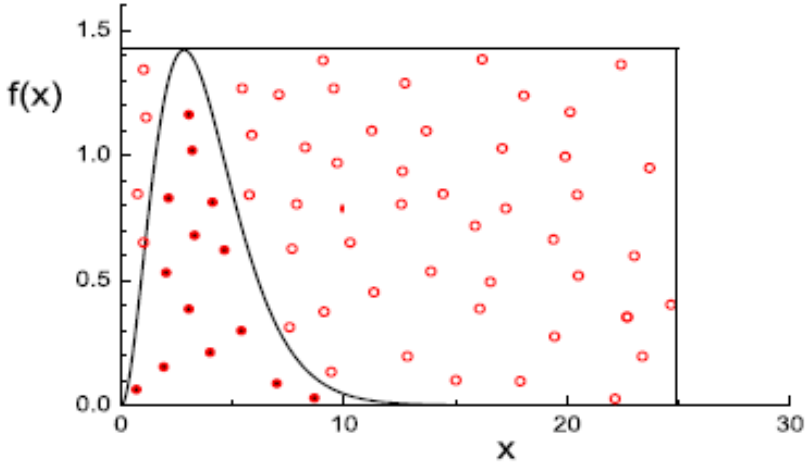
Root-Klasse TRandom3
Standard-Generator in Python

Beispiel: mertwist.C

Beliebig verteilte Zufallszahlen

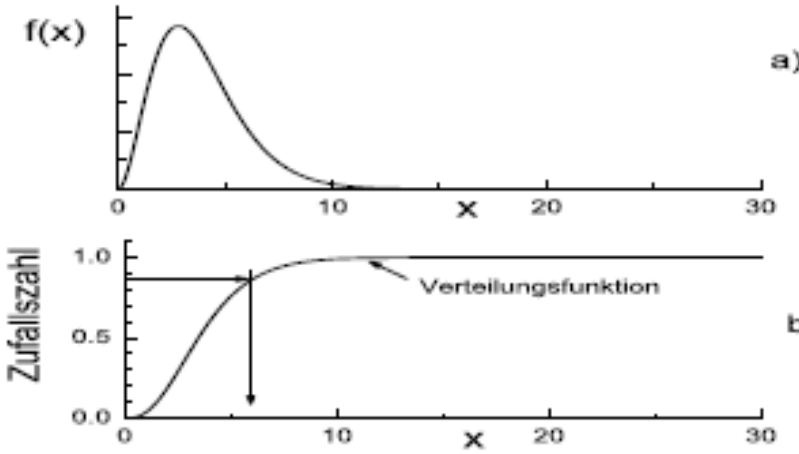
- Von Neumann'sches Rückweisungs-verfahren ("Wegwerfmethode")

  Generiere zwei Zufallszahlen r_1, r_2, akzeptiere $x=r_1$, wenn $r_2<f(x)$, x ist dann gemäß $f(x)$ verteilt



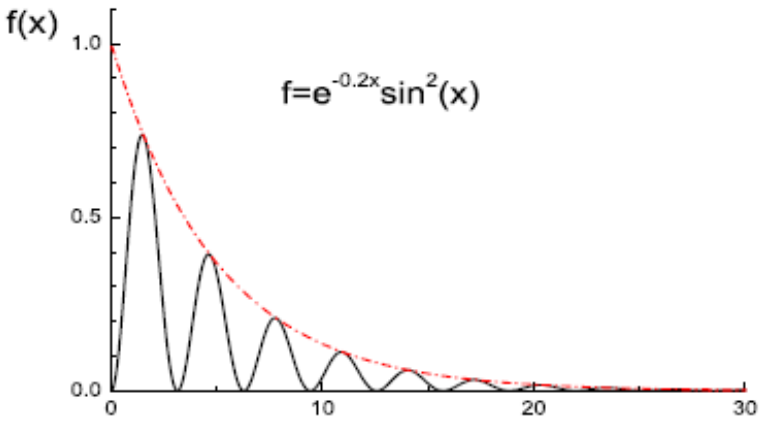
- Transformationsmethode

  Transformiere gleichverteilte $r_i \in [0,1[$, so dass für eine geeignete Funktion $t(r)$, $x_i=t(r)$ der Verteilung $f(x)$ folgt. $t(r)$ ist die Inverse der Verteilungsfunktion $F(x)$



- Majoranten-Verfahren (Importance Sampling)

  Kombination von 1) und 2): finde Einhüllende $m(x) \geq f(x)$, die mit 2) erzeugt werden kann. Erzeuge Zufallszahl x, und wende dann 1) an, d.h. erzeuge zweite Zufallszahl $r_2 \in [0,1]$ und akzeptiere x, wenn $r_2 \cdot m(x) < f(x)$

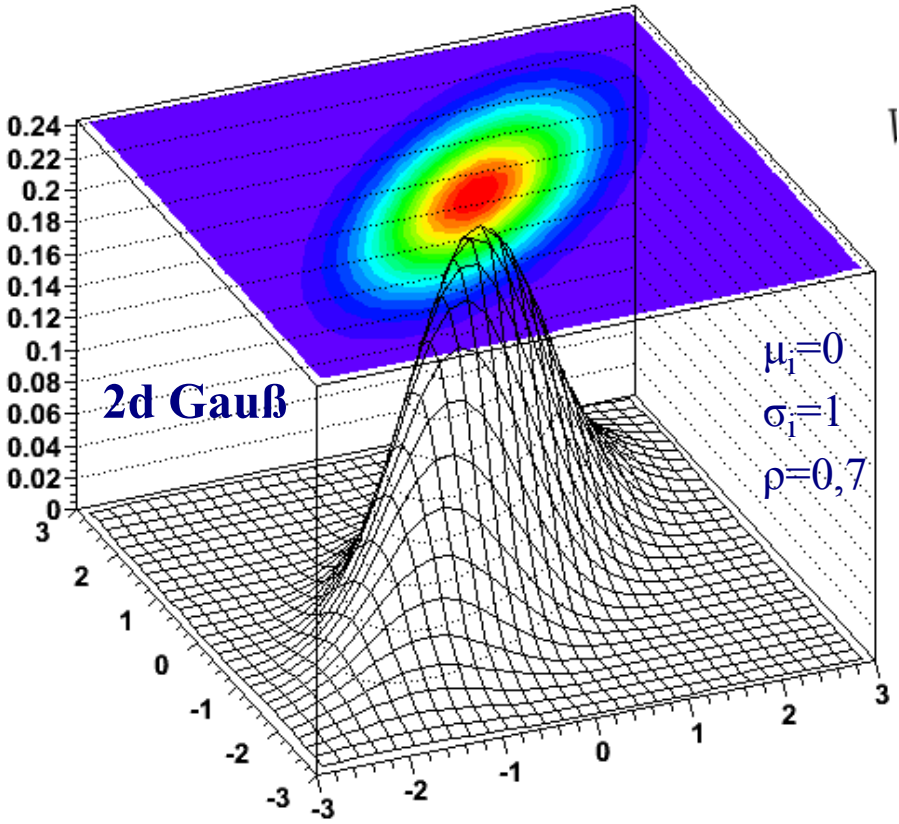


Korrelierte Gauß-verteilte Zufallszahlen

$$V = \begin{pmatrix} \sigma_1^2 & \rho\,\sigma_1\sigma_2 \\ \rho\,\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$$

$$D^{-1} = \begin{pmatrix} \sigma_1 & 0 \\ \rho\sigma_2 & \sigma_2\sqrt{1-\rho^2} \end{pmatrix}$$

$$\vec{x} = D^{-1}\vec{u} + \mu$$



- Verwende standard-normalverteilte unkorrelierte u_1 und u_2
- Dann sind x_1 und x_2 standard-normalverteilt und korreliert mit Korrelationskoeffizient ρ:

$$x_1 = \mu_1 + \sigma_1\,u_1$$

$$x_2 = \mu_2 + \sigma_2\left(\rho\,u_1 + \sqrt{1-\rho^2}\,u_2\right)$$

- Bemerke: u_1 trägt zu x_1 und x_2 bei → Korrelation

Beispiel: rangauss2d.C oder rangauss2d-pyroot.py

Blobel/Lohrmann Abschnitt 5.4