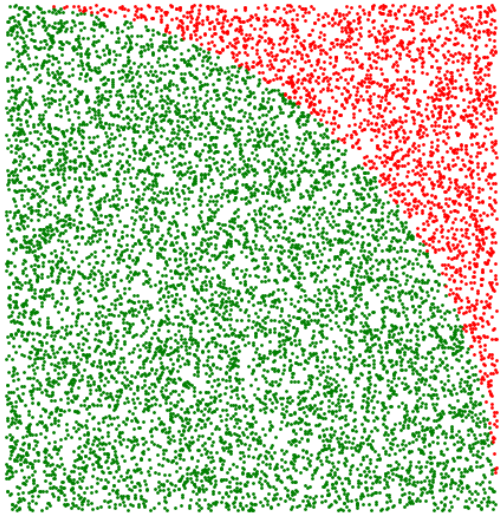
Karlsruher Institut für Technologie

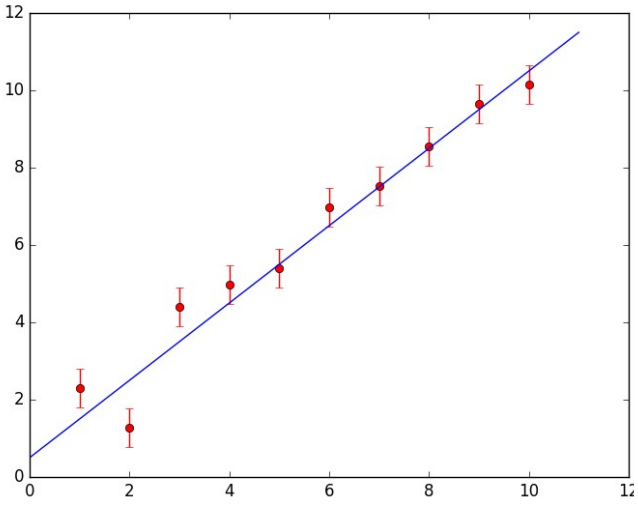
Rechnernutzung in der Physik

Vorlesung 3: Simulation mit der Monte Carlo-Methode

Günter Quast
Fakultät für Physik
Institut für Experimentelle Teilchenphysik

WS 23/24





Organisatorisches

Die Konten zur Nutzung von bwUniCluster2 sind seit Mo 30.10. nachmittags freigeschaltet.
Hinweise s. ILIAS-Link ILIAS-Link

Auszug aus E-Mail vom Rechenzentrum (Kurzanleitung):

„Für das Anmelden am bwUniCluster ist eine einmalige Registrierung auf der Seite https://bwidm.scc.kit.edu erforderlich.
Danach können Sie sich auf einem Login-Knoten des Clusters mit Ihrem KIT-Account einloggen (uc2.scc.kit.edu).

Informationen zum Systemzugang finden Sie auf der SCC-Webseite:
https://wiki.bwhpc.de/e/BwUniCluster_2.0_User_Access
http://www.scc.kit.edu/dienste/hpc.php
http://www.bwhpc-c5.de/wiki/index.php/BwHPC_Best_Practice_Repository

Falls Sie Fragen zum KIT-Kennwort haben, wenden Sie sich bitte an den SCC-Servicedesk (http://www.scc.kit.edu/hotline/index.php).“

Es sind Reservierungen für je 120 Cores am Mo. und Di. Nachmittag angelegt:
„etp-rn1“ (Mo.) und „etp-rn2“ (Di.)
In der Ressourcenauswahl beim Starten eines Jupyter Notebooks dafür "Advanced Mode" anklicken und bei "Reservation" den Namen der Reservierung angeben !

Da wir eine für Forschung vorgesehene Rechenressource nutzen, werden Sie um das Ausfüllen einer kurzen Umfrage gebeten. Als Projekt bitte:
„**Pilotprojekt Jupyter in der Lehre – Rechnernutzung Physik**“ angeben !

Bitte ausprobieren !

Bei Problemen: Beratungstutorium am Di, 07.11.2023 15:00-16:30

Übungsblatt

Das **erste Übungsblatt**

ist seit Ende letzter Woche on-line

1 Rechnernutzung in der Physik

Institut für Experimentelle Teilchenphysik
Institut für Theoretische Teilchenphysik
Prof. G. Quast, Prof. M. Steinhauser
Dr. A. Mildenberger, Dr. Th. Chwalek
Ilias Seite zum Kurs
WS 2023/24 – Blatt 01
Abgabe: Montag 13.11.2023 bzw. Dienstag 14.11.2023

Es handelt sich um eine Wiederholung von Bekanntem:

- Nutzung von Jupyter-Notebooks
- grafische Darstellung von diskreten und kontinuierlichen Verteilungen
- Berücksichtigung von Normierung und Binbreite

Block 1: Statistik und Datenanalyse

Themen:

31.10. Grundlagen der Statistik (Wiederholung CgDA);
python & friends, jupyter-Tutorials

07.11. Monte Carlo-Methode als numerisches Hilfsmittel
MC-Tools in PhyPraKit

14.11. Parameterschätzung mit der Maximum-Likelihood-Methode
Hilfsfunktionen in PhyPraKit, kafe2

21.11. Maximum Likelihood (2) und numerische Optimierung
Ensemble-Tests

28.11. Hypothesentests

Wiederholung VL 02

An dieser Stelle keine „Wiederholung der Wiederholung“

Übersicht

- Grundsätzliches zur MC-Methode
- MC als Integrationsverfahren
- Zufallszahlen aus dem Computer
- Zufallszahlen mit beliebiger Verteilungsdichte
- Anwendungsbeispiele

Monte Carlo - Methode

- **Numerisches Verfahren aus der Stochastik**,
  um analytisch nicht oder nur aufwändig lösbare Probleme
  mit Hilfe der Wahrscheinlichkeitstheorie zu lösen.
- Anwendungsgebiete: (1)
  - Numerische Mathematik (Integration, Optimierung, Faltung, …).
  - Angewandte Statistik (Bestimmung von Korrelationen, Fehlerfortpflanzung, Hypothesentests,…).
  - Nachbildung komplexer Prozesse mit statistischem Verhalten
    (Vielteilchensysteme, Teilchenphysik, …).
- Historie:
  - Erste Idee: Enrico Fermi 1930er Jahre.
  - Erste Ausführung: Stanislaw Ulam, John von Neumann 1947 (Los Alamos Projekt).
  - Namensgebung durch John von Neumann (als code Name innerhalb des Projektes).

Monte Carlo - Methode

- MC-Methode beginnt mit einer Reihe gleichverteilter Zufallszahlen.
- Diese Reihe kann entweder physikalisch bestimmt werden (z.B. Werfen eines Würfels, Zeitspanne Δt zwischen zwei Zerfällen eines radioaktiven Präparats, …), oder mit Hilfe eines Zufallszahlengenerators als Pseudo-Zufallszahlen.

(Pseudo-)Zufallszahlen

:

Ergebnis einer deterministischen Sequenz (insb. re-produzierbar)

Innerhalb eines vorgegebenen Intervalls nach bester Möglichkeit gleichverteilt.

- In einem zweiten Schritt werden die gleichverteilten Zufallszahlen in beliebige Wahrscheinlichkeitsdichte transformiert.

Grundsätzliches Verfahren

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

II. Transformiere diese zur Erzeugung einer anderen Sequenz $x_1, x_2, \ldots, x_n$, die einer Verteilungsdichte $f(x)$ folgen

(Anm.: x, x_i können auch Vektoren sein !)

III. Aus den x_i werden dann Eigenschaften von $f(x)$ bestimmt

(z.B. Anteil der x_i mit $a \le x_i \le b$ ergibt $\int_a^b f(x)\,dx$)

Grundsätzlich ist die MC-Methode eine **Integrationsmethode**

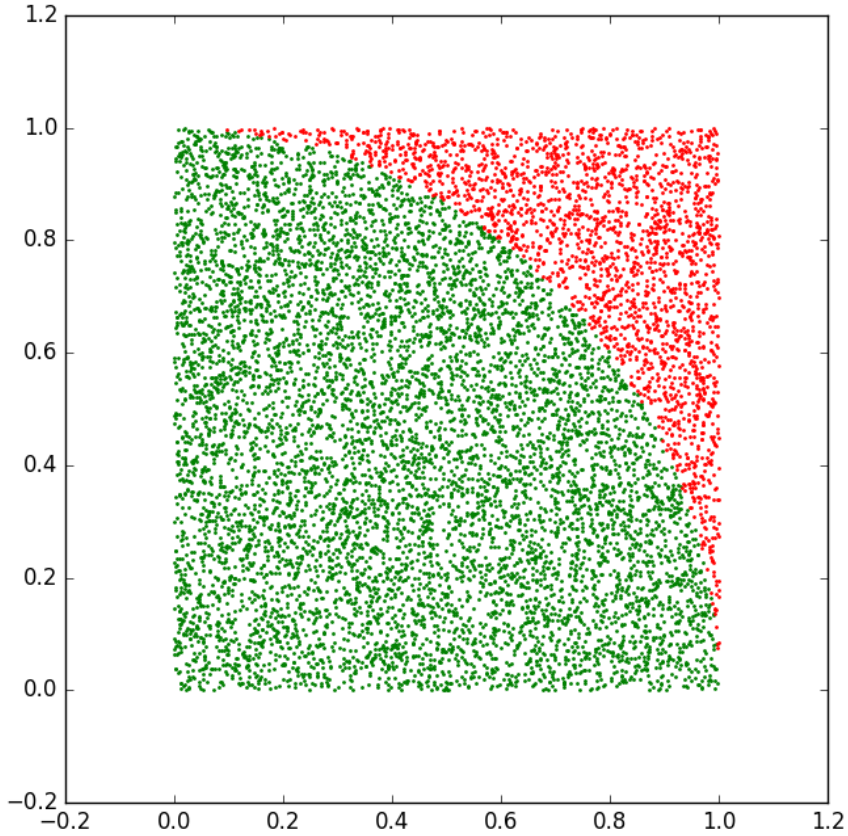
in einer Dimension natürlich anderen Verfahren unterlegen !

Beispiel: Bestimmung von π

Script `pi.py`

```
import numpy as np
import matplotlib.pyplot as plt
import sys

npoints=10000

inx=[]
iny=[]
outx=[]
outy=[]
for i in range(npoints):
  rvec=np.random.rand(2)
  if (np.sum(rvec**2)<=1.):
    inx.append(rvec[0])
    iny.append(rvec[1])
  else:
    outx.append(rvec[0])
    outy.append(rvec[1])

plt.scatter(inx, iny, s=1, color='g')
plt.scatter(outx, outy, s=1, color='r')

pi=4.*float(len(inx)) / \
    float((len(inx)+len(outx)))

plt.show()
```

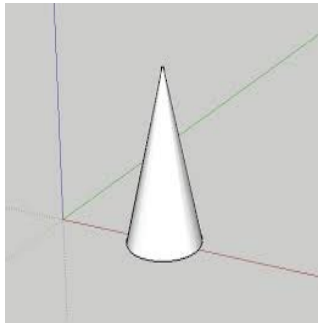
Gleichmäßig über die Einheitsfläche verteilte Zufallszahlen, dann abzählen, wie viele davon innerhalb bzw. außerhalb des Viertelkreises liegen

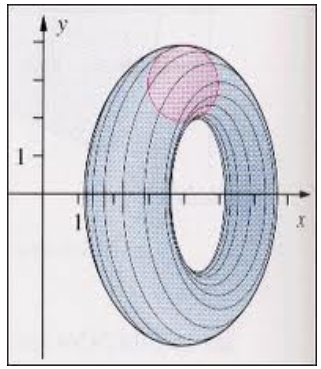



```
*==* script pi.py executing
    Total number of random points:   10000
     Points inside circle: 7848
     ==> pi= 3.139
```

Beispiel 2

Aus der Geometrie: **Schnittvolumen eines Kegels und eins Torus**





- definiere Quader, der Schnittvolumen einschließt

- zufällig gleichverteilte Punkte im Quadervolumen erzeugen

  - feststellen, ob Punkt im Schnittvolumen liegt,
    also im Torus UND im Kegel (relativ einfach zu machen!)

  - Anzahl der Punkte im Schnittvolumen zählen

    Verhältnis zur Gesamtzahl der Versuche entspricht dem Verhältnis des gesuchten Volumens zum Quader-Volumen

    – schon fertig !

Recht geringe intellektuelle Anstrengung, die Arbeit macht der Rechner !

Nachteil: die Methode hat einen statistischen Fehler,
berechenbar aus Binomialveterteilung P(N; n): **Genauigkeit steigt mit √N**

Methode ist ohne viel Aufwand auch auf mehr als drei Raumdimensionen zu erweitern, statistischer Fehler hängt nach wie vor nur von **√N** ab - bei anderen Integrationsverfahren steigt die Anzahl der notwendigen Rechenschritte exponentiell mit der Zahl der Dimensionen !

MC ist eine Integrationsmethode

Berechnung von Integralen: $$I = \int_a^b \phi(x)dx$$

Idee: Erwartungswert einer Stichprobe konvergiert gegen Erwartungswert einer Verteilung

Verwende **N gleichverteilte Zufallszahlen** $x_i \in [a,b]$, $f(x) = \frac{1}{b-a}$

Erwartungswert $$\langle\phi\rangle = \int_a^b \phi(x)f(x)dx = \int_a^b \frac{\phi(x)}{b-a}dx$$

Mittelwert $$\bar{\phi} = \frac{1}{N}\sum_{i=1}^{N}\phi(x_i) \to \langle\phi\rangle$$

Insgesamt also

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

Statistische Unsicherheit von 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)

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

statistische Unsicherheit

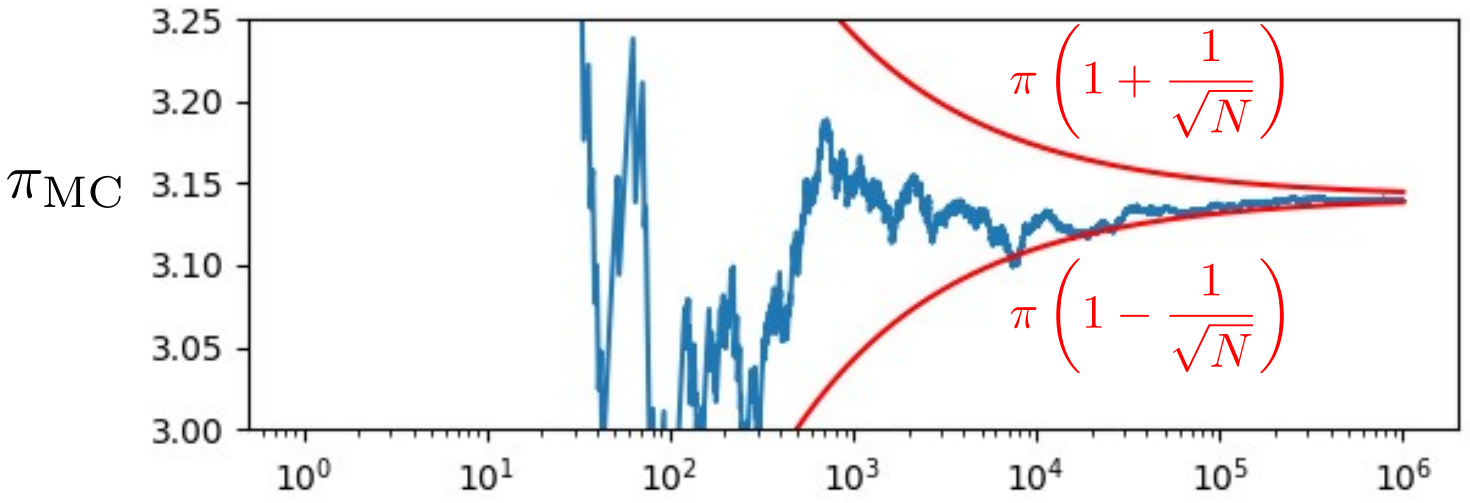
$$\frac{\delta I^2}{(b-a)^2} = \delta\bar{\phi}^2 = \frac{V[\phi]}{N}$$

V[I_{MC}] ist proportional zu V[Φ] und 1/N,
d. h. Genauigkeit verbessert sich mit 1/$\sqrt{N}$

Methode funktioniert auch in mehreren Dimensionen;
Unsicherheit skaliert weiter mit $\sqrt{N}$!

Beispiel: **Bestimmung von π über Kreisfläche**

Beispiel: calc_pi.py



Zahl der Punkte N

Genauigkeit wird mit 1/√N besser

Vergleich mit „Trapezmethode“

Trapezregel:

Integration durch Summation über Funktionswerte *f* in *n* Intervallen der Länge $h=(b-a)/n$

$$I_T = h\,(\frac{1}{2}f_0 + f_1 + \ldots + f_{n-1} + \frac{1}{2}f_n)$$

Abschätzung des Fehlers ΔI_T (durch Taylor-Entwicklung von I und I_T):

→ In einer Dimension: *n* Intervalle pro Dimension

ΔI_T nimmt mit $1/n^2 = n^{-2}$ ab

→ In d Dimensionen: $n^{1/d}$ Intervalle pro Dimension

ΔI_T nimmt mit $1 / n^{2/d} = n^{-2/d}$ ab.

MC-Methode ist dann besser, wenn gilt: $-2/d < -1/2$, **also für $d > 4$**

In hochdimensionalen Räumen ist Monte Carlo-Integration anderen Methoden überlegen

Blobel/Lohrmann Abschnitt 5.6

weitere Beispiele Anwendung MC-Methode

Aus Thermodynamik/Chemie:

Zwei Sorten von Molekülen in einem Behälter bei einer bestimmten Temperatur stoßen miteinander und erzeugen neue Moleküle, wenn die Schwerpunktsenergie ausreicht. Gesucht ist die Zusammensetzung des Gases als Funktion der Zeit.

Bei begrenzter Anzahl von Teilchen können Teilchenbahnen, Stöße und Umwandlungen verfolgt werden; durch Abzählen der Molekülsorten erhält man das Ergebnis.

Aus der Physik:

Messung der Lebensdauer von Teilchen
mit Digitaluhr bei bestimmter Genauigkeit und Maximalzeit

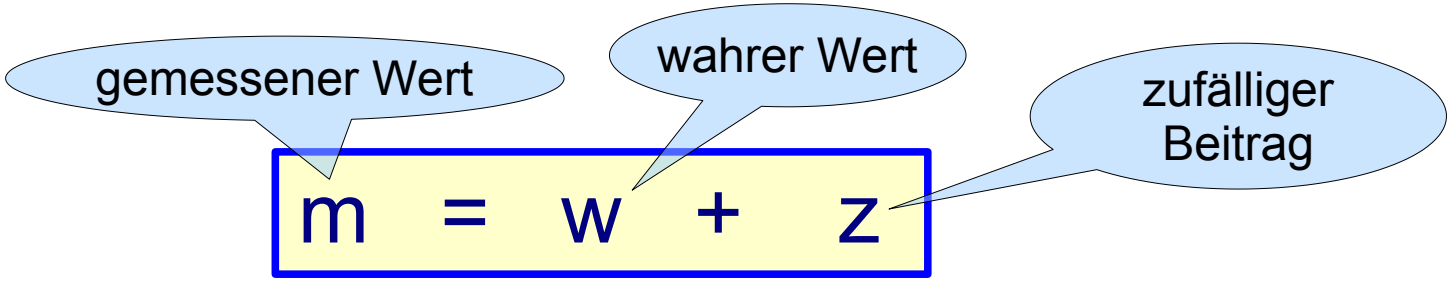
- zufällige Lebensdauer eines Teilchens aus Zerfallsgesetz (Exponentialverteilung)
- zufälligen Messfehler addieren → simulierter Messwert
- Messwerte innerhalb der Maximalzeit akzeptieren

Prozedur N-mal wiederholen

Aus dem Mittelwert der erhaltenen Messwerte und Vergleich mit der „wahren“ Lebensdauer erhält man eine Korrektur der durch das Messverfahren verfälschten Messwerte.

Simulierte Daten

Zufallskomponente z bei der Gewinnung empirischer Daten ebenfalls mit Zufallszahlen modellierbar:



Funktion (=Modell) des wahren Wertes, ***y = f(x),*** und Modellierung der Messwerte mit MC:

$$m_y = f(w_x + z_x) + z_y$$

z_x und z_y entsprechen den Unsicherheiten der Messgrößen x und y

Beispiel: Faltungsintegral

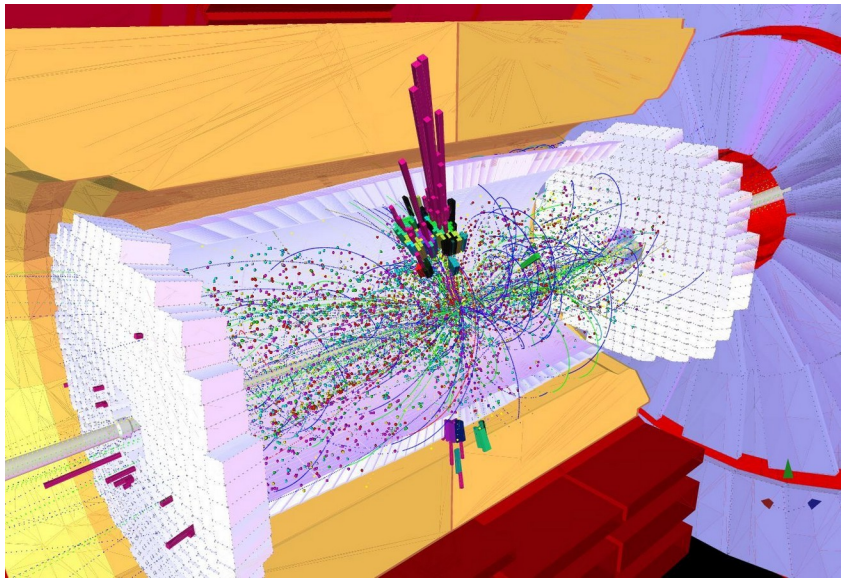
Grundsätzlich ist die **MC-Methode** ein **Integrationsverfahren**, bei der Simulation von Messwerten die Auswertung eines **„Faltungsintegrals“**:
statt eines „wahren“ Werts x wird ein durch Auflösungseffekte $t(x',x)$ „verschmierter“ Wert x' gemessen:

$$f'(x') = \int_{-\infty}^{\infty} t(x,x') f(x) dx$$

Z.B. ist bei einer **Lebensdauermessung** $f(x)$ die Verteilung der wahren Lebensdauern, $t(x,x')$ beschreibt die Auflösung und Systematik des Messapparates.

Bei der MC-Simulation werden zufällige Störungen der Reihe nach aufaddiert.
Vorteil: **einfach zu berechnende Summe von Zufallsvariablen entspricht einer Serie von komplizierten Faltungsintegralen**

In der Praxis werden Messwerte in der Regel von einer großen Zahl von Effekten verfälscht (Auflösung des Messgerätes, Rauschen, Digitalisierung, systematische Effekte usw.),
d.h. die zu bestimmenden Faltungs-integrale sind hoch-dimensional.



Simulierte Signale im CMS-Detektor

„Zufallszahlen“ aus dem Computer

Zufallszahlen aus dem Computer

Auf Computern (hoffentlich) nur deterministische Zahlenfolgen
=> auf Rechnern werden „**Pseudo-Zufallszahlen**“ verwendet

Anforderungen:
- „zufällig“ verteilt mit langer Periode
- k(l)eine Korrelation zwischen aufeinanderfolgenden Werten
- Erzeugung muss schnell sein
- Reproduzierbarkeit solcher Zahlenfolgen oft erwünscht (nicht bei Spielen …)

Einfacher Zufallszahlengenerator basiert auf

$X_{n+1} = (a\ x_n)\ mod\ 2^k$ d. h. die letzten k Bits (Bit-weises AND mit 2^k-1)

Allgemeiner (und besser):

$X_{n+1} = (a\ x_n + c)\ mod\ m$, $r_i = x_i/m$ $[0,1[$ **L**inear **C**ongruent **G**enerator „**LCG**“

m	Modulus
$0<a<m$	Multiplikator
$0<c<m$	Inkrement
$0 \leq x_0 < m$	Startwert oder „Seed“

Peridodenlänge ist höchstens m,
im ungünstigen Fall viel kleiner
a und c müssen geeignet gewählt werden !!!!

Es gibt eine Vielzahl von Algorithmen und Implementierungen **TIPP:**
„Professionellen“ Zufallszahlen-Generator aus Standard-Bibliothek verwenden

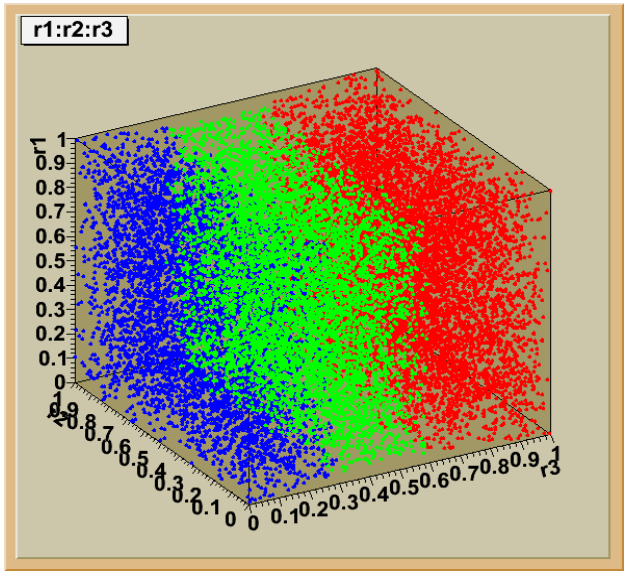
es kann auch schief gehen ....

Sind die Zufallszahlen „gut“ ?

- innerhalb der Folge zufällig verteilt ? Differenzen von Zufallszahlen anschauen
- gibt es Korrelationen zwischen aufeinander folgenden Zahlen ?

Betrachte Paare, Triplets, n-tuples von benachbarten Zahlen,
einfache Kontrolle durch Auftragen als n-dim. Histogramm, auf „Muster“ achten!

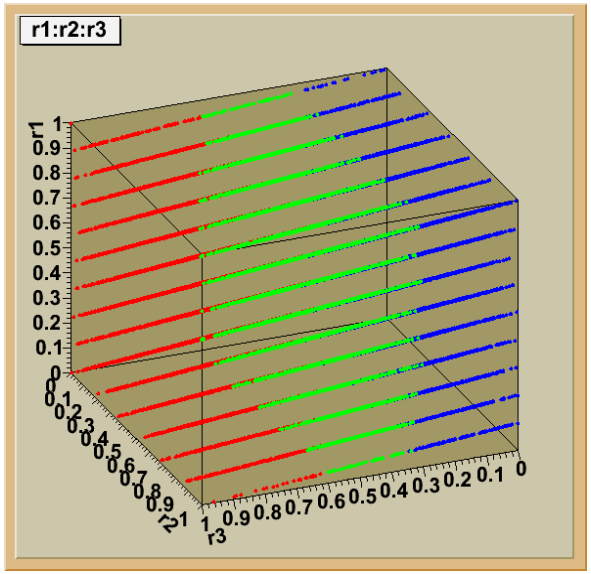
s. Beispiel lcg.py



Nur auf den ersten Blick ok

Bei genauerem Hinsehen (Drehen) liegen alle r_i auf Flächen im 3-dimensionalen Raum

– damit könnte man kein Kugelvolumen integrieren !



Allgemein gilt (siehe z.B. Brandt, Datenanalyse):

N-Tuples von so erzeugten Zufallszahlen liegen auf Hyperebenen im n-dim Raum mit Ebenenabstand $d \geq$ (ungefähr) $m^{-1/n}$

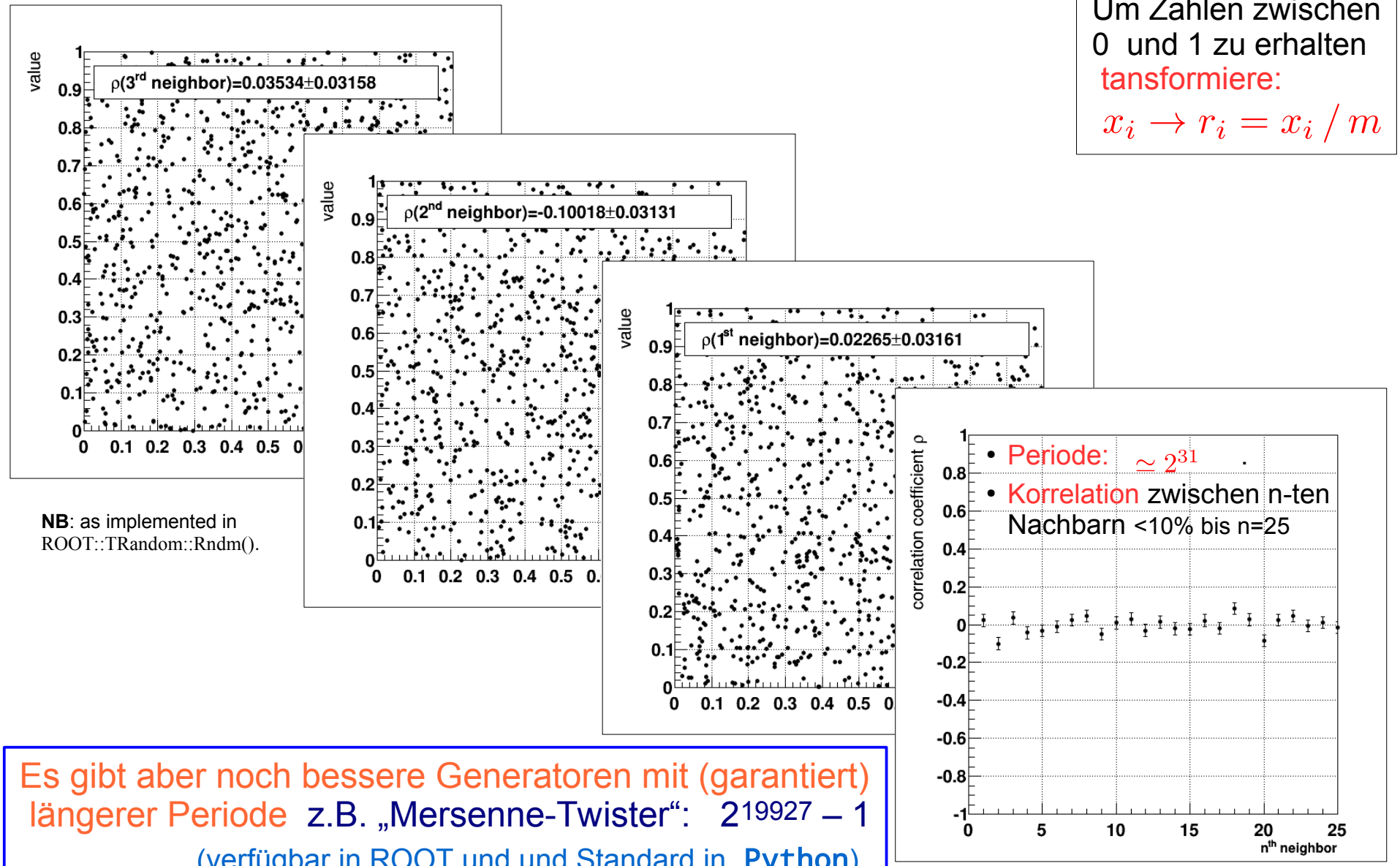
Im obigen Beispiel wird die theoretische Grenze aber weit unterschritten!

Beispiel LCG:

$$x_{i+1} = a \cdot x_i \mod m \qquad m = 2147483399 = 2^{31} - 249 \text{ (Primzahl)} \qquad a = 40692$$

Um Zahlen zwischen 0 und 1 zu erhalten tansformiere:

$$x_i \to r_i = x_i / m$$



NB: as implemented in ROOT::TRandom::Rndm().

Es gibt aber noch bessere Generatoren mit (garantiert) längerer Periode z.B. „Mersenne-Twister“: $2^{19927} - 1$

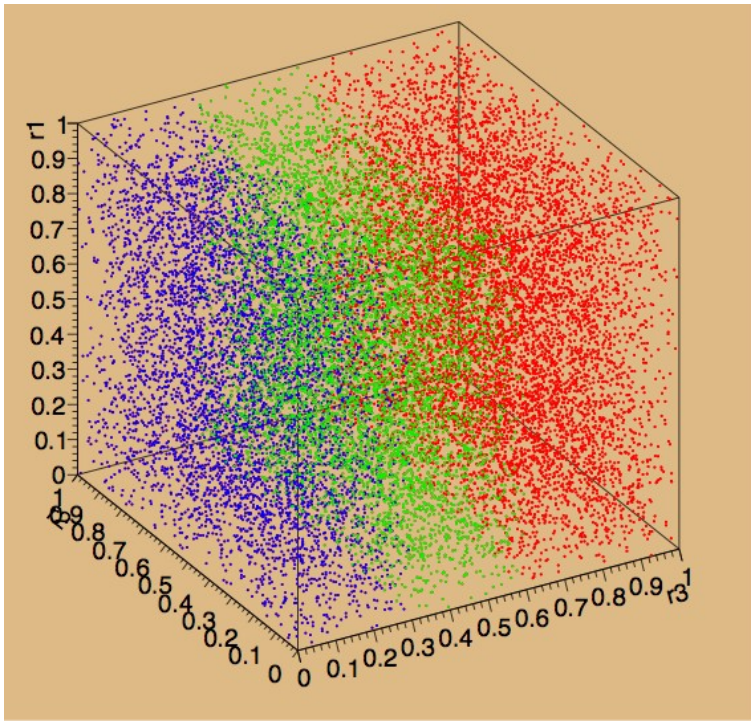
(verfügbar in ROOT und und Standard in **Python**)

Aktuelle Zufallsgeneratoren

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

PCG-Familie (permuted congruential generator) M. O'Neill, 2014

- Weiterentwicklung LCG:
  signifikanteste Bits in LCG-Ausgabe werden wird permutiert
- Wahl für Modulo-Operation: m = 2^k → schnell
- Standard in numpy.random

Qualität von Zufallszahlen

wirklich zufällige Zufallszahlen sind auch das Herz einer jeden Verschlüsselungstechnik

es gibt eine täglich wachsende Zahl
- neuer Generatoren, Algorithmen und Verfahren,
  bis hin zu physikalischen Quantensystemen
- **Testverfahren**, um Schwächen oder absichtliche Manipulation **(ja, auch die gibt es)** aufzudecken

Testverfahren (kleine Auswahl):

Trivialer Test: Histogramm der Zufallszahlen und Anpassung
Summe der Abstandsquadrate $\chi 2$ sollte nahe 1 / d.o.f sein

Moderne, fortgeschrittene Tests:

- G.Marsaglia, *Die Hard Battery of Tests of Randomness* (1995)
  15 verschiedene Tests
  darauf aufbauend: dieharder (R. Brown)
- P.L'Ecuyer und R.Simard: *TestU01 (2007)*
  Small Crush (10 Tests), *Crush* (96 Tests), *Big Crush* (106 Tests)

Es gibt Generatoren, die TestU01 überstehen

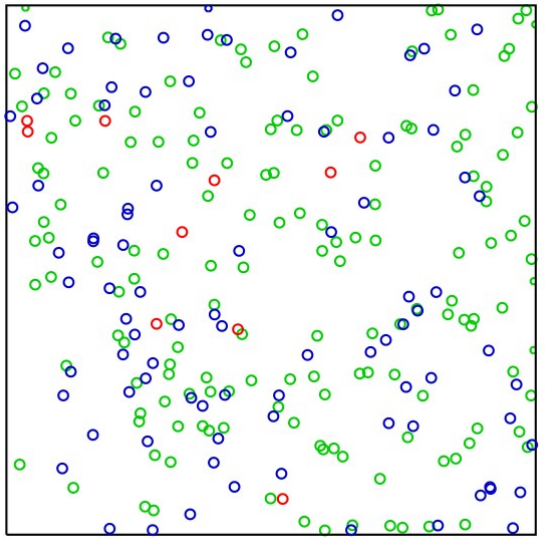
Mersenne-Twister erfüllt nicht alle Tests der Big Crush Suite,
gilt inzwischen als kryptographisch unsicher, da vorhersagbar

Diese Einschränkungen für MT sind für unsere
(wissenschaftlichen) Zwecke nicht relevant.

Quasi-Zufallszahlen

Reihen von (Pseudo-)Zufallszahlen verteilen sich statistisch im Raum

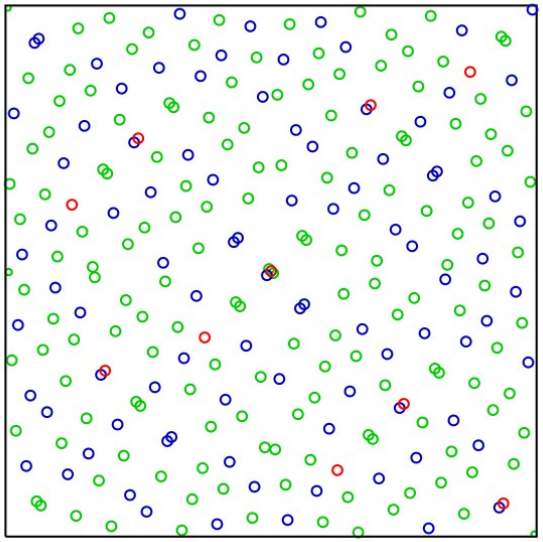
256 Pseudozufallszahlen (in 2d)



- 1 … 10
- 11 … 100
- 101 … 256

Wenn man eine bestimmten Phasenraum möglichst gleichmäßig „ausleuchten“ möchte, geht dies effiziente mit Hilfe **korrelierter,** sog. **Quasizufallszahlen**

256 erste Zahlen einer Sobol Sequenz



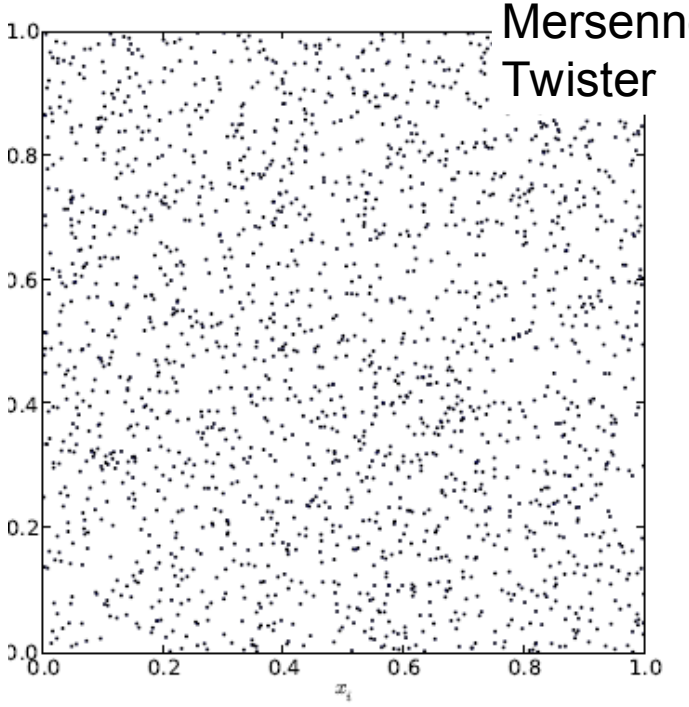
Beispiel:

Sobol Sequenz (schon 1964 zur effizienten Bestimmung numerischer Integrale eingeführt)

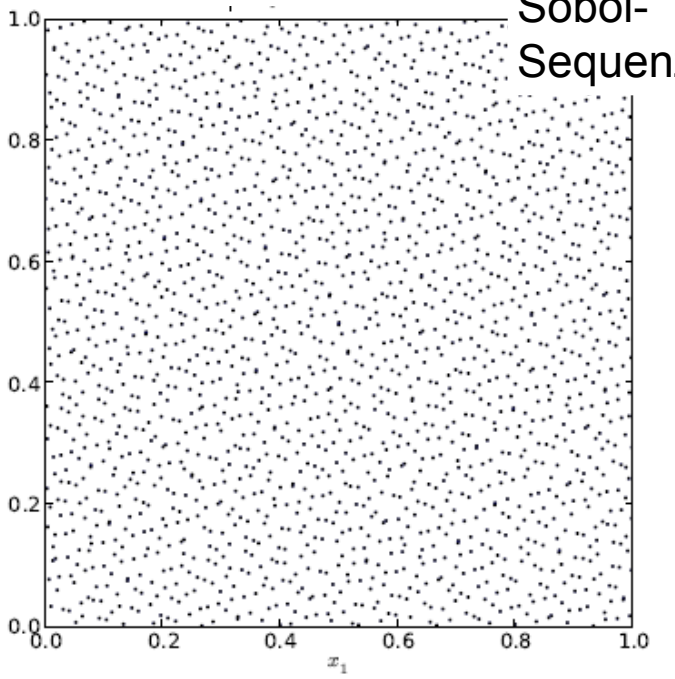
Quasi-Zufallszahlen

- Während Pseudo-Zufallszahlen „klumpen“ erreicht man eine gleichmäßigere Abdeckung eines n-dimensionalen Raumes mit Quasi-Zufallszahlen

Mersenne-Twister



Sobol-Sequenz



S.Press et al

- Konvergenz ~1/*N*, besser als für Pseudo-Zufallszahlen
- Quasi-Zufallszahlen sind stark korreliert. Nur für Integration!
- Sequenz muss fortgesetzt werden, Dimension nachträglich nicht veränderbar !

Erzeugung von Zufallszahlen mit beliebiger Verteilungsdichte

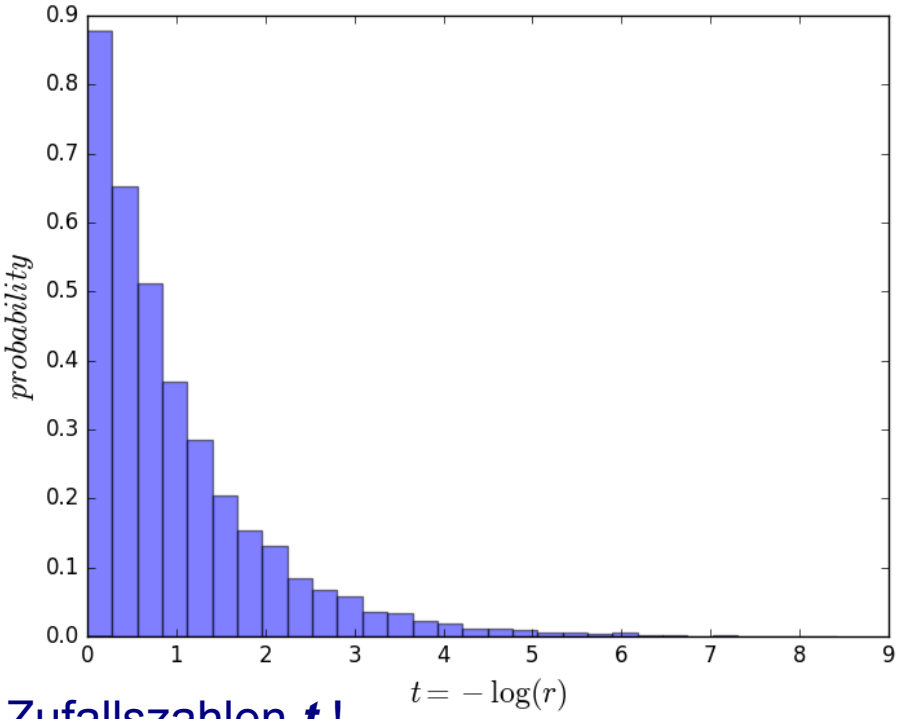
Funktionen von Zufallszahlen

Computerexperiment:

Häufigkeitsverteilung von

$t = -\log(1-r)$, r Zufallszahl $\in [0,1]$

```
 ...
r = np.random.rand(10000)
t = -np.log(1-r)

plt.hist(t, 30, normed=1)

plt.show()
```



Wir erhalten exponentiell verteilte Zufallszahlen ***t*** !

Erklärung:

$$g(t)dt = p(t) \equiv p(r) = f(r)dr \Rightarrow g(t) = f(r)\left|\frac{dr}{dt}\right|$$

Erhalt der Wahrscheinlichkeit

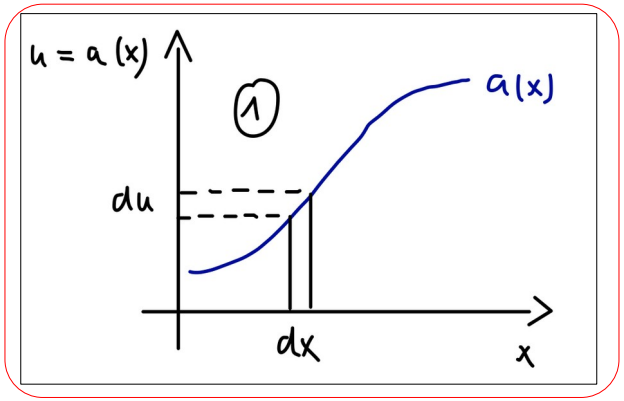
Regel für die Transformation von Verteilungsdichten

$$r(t) = 1 - \exp(-t),\ \frac{dr}{dt} = \exp(-t),\ \Rightarrow g(t) = 1 \cdot \exp(-t)$$

Anm.: *t*=log(*r*) statt *t*=log(*1-r*) ergibt identisches Ergebnis (warum?)

Funktionen von Zufallszahlen allgemein

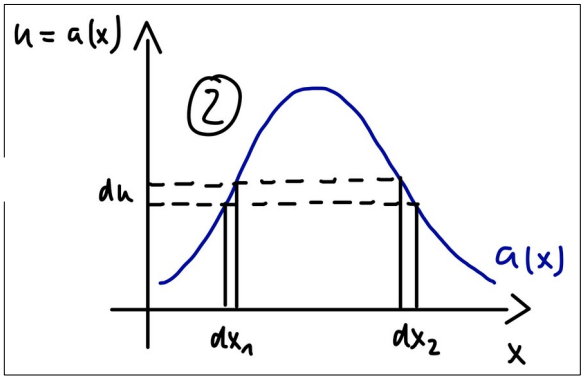
$u = a(x)$



$$g(u) = f(x) \left| \frac{dx}{du} \right|$$

Wahrscheinlichkeitsdichte von *x* multipliziert mit dem Betrag der Ableitung der Inversen Funktion $a^{-1}(u)$

$a(x)$ nicht (eindeutig) umkehrbar



Beiträge aller Äste der Umkehrfunktion aufaddieren !

Für mehrdimensionale Wahrscheinlichkeitsdichten

wird $\left| \frac{\mathrm{d}x}{\mathrm{d}a} \right|$

zur Jacobi-Determinante:

$$\begin{vmatrix} \frac{\partial x_1}{\partial a_1} & \frac{\partial x_1}{\partial a_2} & \cdots & \frac{\partial x_1}{\partial a_n} \\ \frac{\partial x_2}{\partial a_1} & \frac{\partial x_2}{\partial a_2} & \cdots & \frac{\partial x_2}{\partial a_n} \\ \vdots & & \vdots & \\ & \cdots & & \frac{\partial x_n}{\partial a_n} \end{vmatrix}$$

Ganz analog zur Transformation von Volumenelementen bei mehrdimensionalen Integralen

„Transformationsmethode“
zur Erzeugung von Zufallszahlen

Den Zusammenhang zwischen der ursprünglichen Verteilungsdichte *f(r)* und derjenigen einer Transformation *t(r)* können wir ausnutzen, um **beliebig verteilte Zufallszahlen** zu erzeugen:

Welche Transformation t(r) liefert g(t) ?

$$g(t)dt = U(r)dr \quad (\text{U: Gleichverteilung in } [0,1])$$

$$\int_{-\infty}^{t} g(t')dt' = \int_{0}^{r} U(r')dr' = r$$

$$G(t) = r \qquad \Rightarrow \qquad \boxed{t = G^{-1}(r)}$$

Die gesuchte Transformation ist die Inverse der Verteilungsfunktion *G* von *g*

Anm: *Das Verfahren ist nur dann (effizient) praktikabel, wenn die gesuchte Verteilungsdichte integrierbar und das Integral invertierbar ist !*

Beispiele Transformationsmethode

Dreieck-Verteilung

$$g(t) = 2t,\ 0 \le t \le 1$$
$$t(r) = \sqrt{r}$$

$$g(t) = (n+1)\,t^n,\ 0 \le t \le 1, n > -1$$
$$t(r) = r^{1/(n+1)}$$

Exponentialverteilung

$$g(t) = \gamma e^{-\gamma t}$$
$$t(r) = -\frac{1}{\gamma}\ln(1-r)$$

Log-Weibull -Verteilung

$$f(x) = e^{-x-e^{-x}}$$
$$x = -\ln(-\ln r)$$

Breit-Wigner-Verteilung

$$f(x) = \frac{1}{\pi\Gamma/2}\frac{(\Gamma/2)^2}{x^2+(\Gamma/2)^2}$$
$$x(r) = \frac{\Gamma}{2}\tan\left[\pi(r-\frac{1}{2})\right]$$

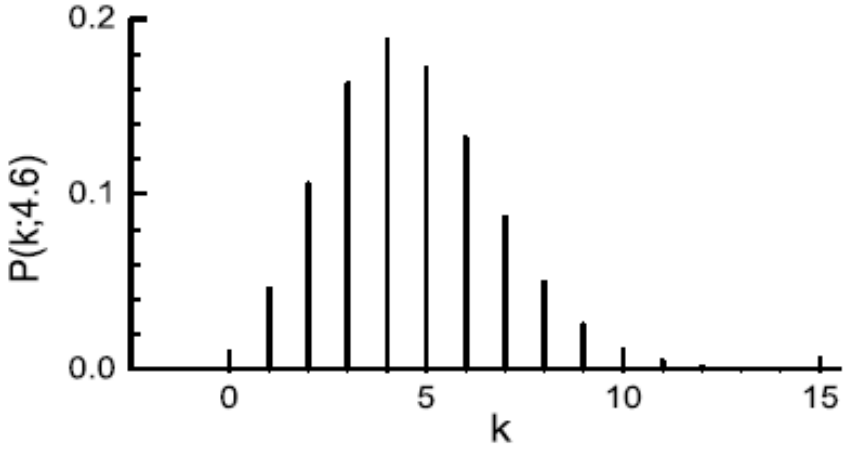
Paar von Gauß-verteilten Zahlen

$$f(x,y) = \frac{1}{2\pi}\exp\left[-\frac{x^2+y^2}{2}\right]$$
$$x(r_1,r_2) = \sqrt{-2\ln(r_1-1)}\cos(2\pi r_2)$$
$$y(r_1,r_2) = \sqrt{-2\ln(r_1-1)}\sin(2\pi r_2)$$

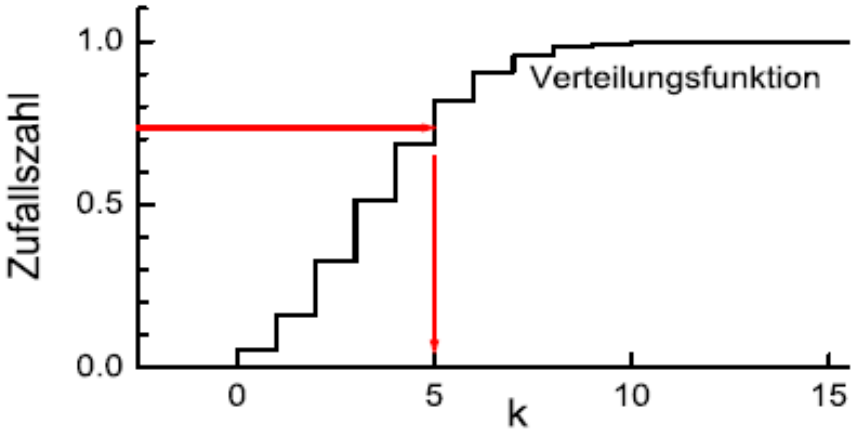
Anm.:
1-*r* kann durch
r ersetzt werden

Erzeugung diskreter Verteilungen

Das gezeigte Verfahren funktioniert auch für diskrete Verteilungen



Beispiel Poisson-Verteilung



Durch Summation aller Bins der diskreten Verteilung erhält man die Verteilungsfunktion, repräsentiert als eindimensionales Feld $S(j)$

Eine Zufallszahl r_i wird demjenigen Bin j zugeordnet, das dem kleinsten der $S(j)$ mit $r_i > S(j)$ entspricht

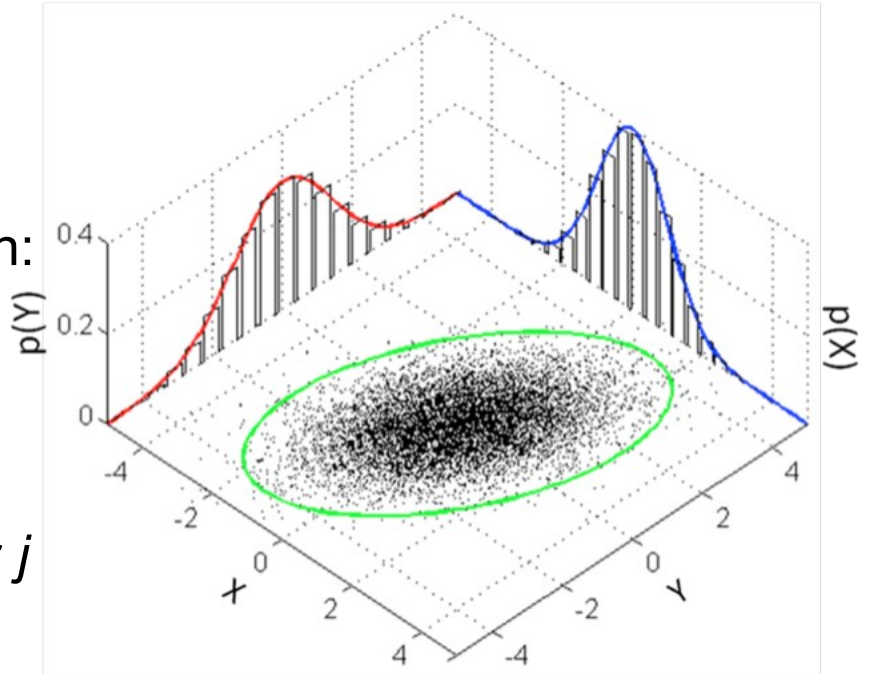
Verteilungen aus Histogrammen

Allgemein kann die Transformationsmethode auch zur Erzeugung von Zufallszahlenverteilungen aus vorgegebenen Histogrammen verwendet werden:

- gleichverteilte Zufallszahl r_i bestimmt das Bin k
- Der Rest r_i - $S(k\text{-}1)$ kann zur Interpolation innerhalb des Bins verwendet werden

Auch für mehr-dimensionale Verteilungen:

- bilde für jedes i:
  Randverteilung $g_i = \sum h_{ij}$
  (also Projektion von h_{ij} in j)
- berechne Summenverteilungen über j
- generiere i und dann für
  gegebenes i eine Bin-Nummer j

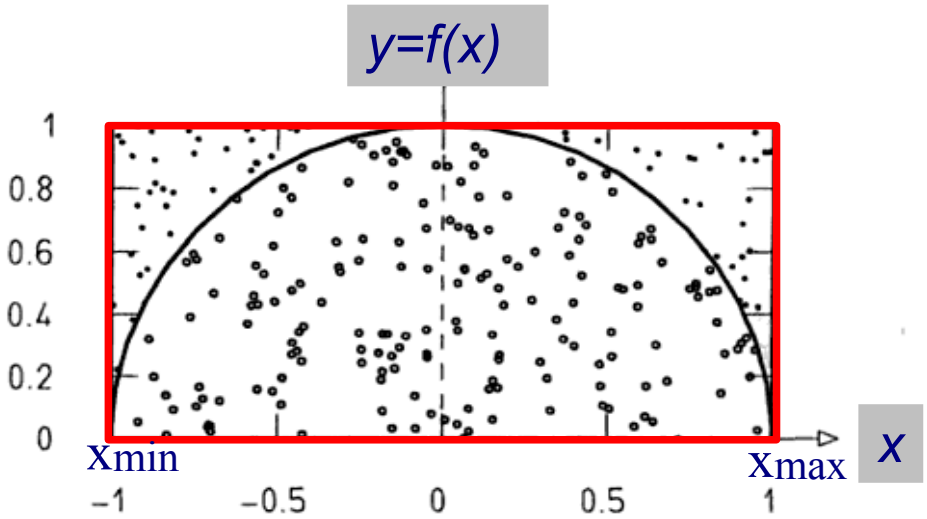


Quelle: Wikipedia

von Neumann'sches Rückweisungsverfahren

Ein allgemein anwendbares, aber nicht immer effizientes Verfahren geht so:

Beispiel:
Kreisbogen als
Verteilungsdichte f(x)



- Verteilungsdichte wird in Rechteck („Box“) eingeschlossen
- erzeuge gleichförmig in der Box verteilte Paare von Zufallszahlen (x_i, y_i)
- x-Werte aus Paaren mit $y < f(x)$ werden akzeptiert

(mit einer Wahrscheinlichkeit $f(x) / f_{max}$)

Die Häufigkeitsverteilung der akzeptierten x-Werte folgt der gewünschten Verteilung

Anm: *Im gezeigten Beispiel funktioniert das Verfahren effizient;*
Wenn die Verteilung steil abfällt und Werte für $x \to \infty$ benötigt werden,
kann die Zahl der verworfenen Zufallszahlen unakzeptabel groß werden.

Majoranten-Verfahren (engl. Importance Sampling)

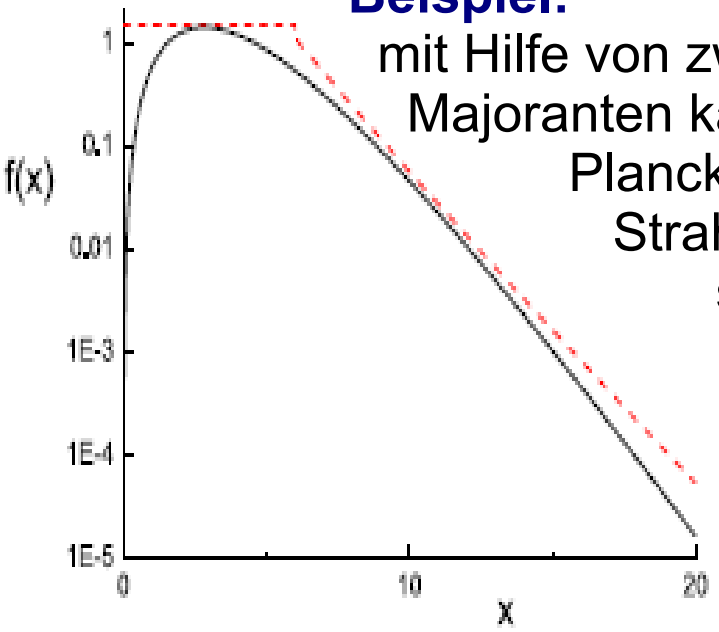
Verbesserte Wegwerfmethode:

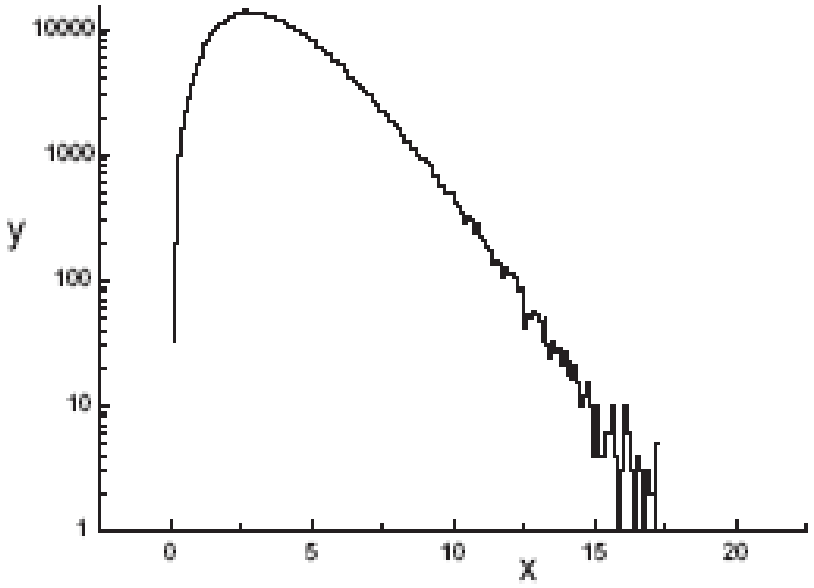
Wenn eine Funktion $m(x)$ mit $m(x) > f(x)$ für alle x existiert und $x=M^{-1}(r)$ hinreichend leicht bestimmbar ist (d.h.Stammfunktion von m ist invertierbar)

- generiere $x_i=M^{-1}(r2_i)$ und eine weitere Zufallszahl $r2_{i+1} \in [0,1]$
- akzeptiere x_i, wenn $r2_{i+1} \cdot m(x) < f(x)$ ist
  (für jeden Wert von x wird ein Bruchteil $(m(x) - f(x)) / f(x)$ an x-Werten verworfen)

Die akzeptierten Werte von x sind gemäß $f(x)$ verteilt.

Beispiel:
mit Hilfe von zwei Majoranten kann ein Planck'sches Strahlungs-spektrum effizient erzeugt werden.

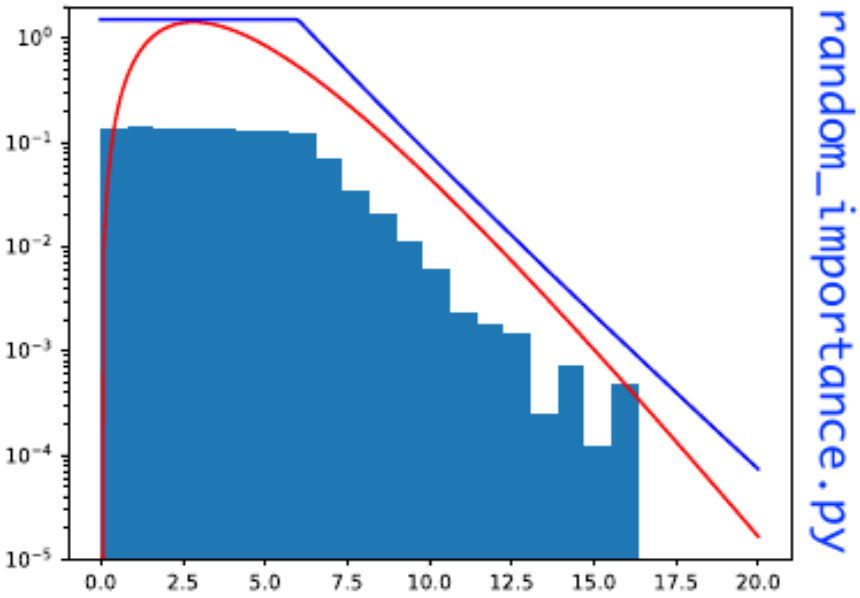




Beispiel Majorantenmethode

$$f(x) = \frac{x^{5}}{\exp[x] - 1}$$

$$m(x) = \begin{cases} c & x < 6 \\ d \cdot x^{-0,1} \cdot \exp\left[-x^{0,9}\right] & x \geq 6 \end{cases}$$



Numerisches Integral (Mathematica):	6,49394
10.000 Zufallszahlen (Straight Sampling):	6,396
10.000 Zufallszahlen (Importance Sampling):	6,52917

effizientes Integrationsverfahren: Markov-Chain Monte Carlo

Idee: statt zufällig gleichverteilter Zufallszahlen werden Zahlen aus einer sog. Markov-Kette verwendet: $x_{i+1} = f(x_i)$

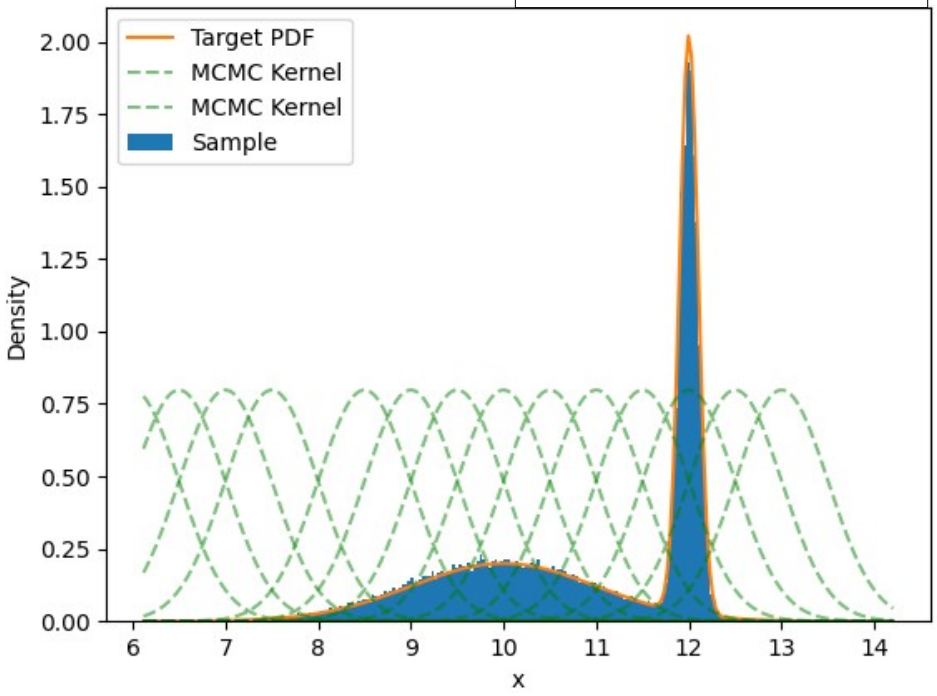
→ Bevorzugung von Stellen mit großen Werten der Verteilungsdichte

Verwende Kernel-Dichte, aus der die nächste Zufallszahl gewählt wird, bevorzugt an Stellen mit großen Werten der PDF

Algorithmus:

1. wähle anfänglichen Wert x_I für i=0
2. berechne $\mathcal{L}(x) = \text{TargetPDF}(x_i)$
3. ziehe neues x aus KernelPDF(x_i)
4. wenn $\mathcal{A}(x) / \mathcal{L}(x_i) < z = \text{rand}[0,1]$
   akzeptiere und speichere x als x_{i+1},
   sonst behalte x als x_{i+1}
5. gehe zu 2., bis Anzahl Werte erreicht

MCMC_example.py



Alle Werte nach Burn-in Phase akzeptiert !
→ **sehr hohe numerische Effizienz !**

Anmerkungen:

1. Verfahren benötigt „burn-in“ Phase, d.h. die ersten x_i werden nicht verwendet
2. Aufeinanderfolgende Zahlen sind stark korreliert (irrelevant bei z.B. Integration)

In hochdimensionalen Räumen ist MCMC die Methode der Wahl !

Nächste Vorlesung

Parameteranpassung mit der Likelihood-Methode

- Material:
  V04a_Parameterschaetzung.pdf