Simulation mit der Monte-Carlo-Methode

- Grundsätzliches
- gleichverteilte (Pseudo-)Zufallszahlen
- Verfahren zur Erzeugung beliebiger Verteilungen
- Beispiele

Grundsätzliches zur MC-Methode

Grundsätzliches zur MC-Methode

Die Monte-Carlo-Methode (auch „MC-Simulation“)

- abgeleitet vom Namen der durch ihr Spielkasino berühmten Stadt Monte Carlo -

ist eine auf (Pseudo-)Zufallszahlen basierende numerische Methode zur

- Integration in hochdimensionalen Räumen

- Bestimmung der Eigenschaften von Verteilungen von Zufallszahlen (z.B. von Messgrößen in Experimenten)

- Nachbildung von komplexen Prozessen

(z.B. in Thermodynamik, Wirtschaft oder von experimentellen Apparaturen, u.v.a.)

Grundsätzliches Verfahren

- Erzeuge Folge von Zufallszahlen $r_1, r_2, \ldots, r_m$, gleichverteilt in]0,1].
- Transformiere diese zur Erzeugung einer anderen Sequenz $x_1, x_2, \ldots, x_n$, die einer Verteilungsdichte $f(x)$ folgen (Anm.: x_i können auch Vektoren sein !)
- Aus den x_i werden dann Eigenschaften von $f(x)$ bestimmt

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

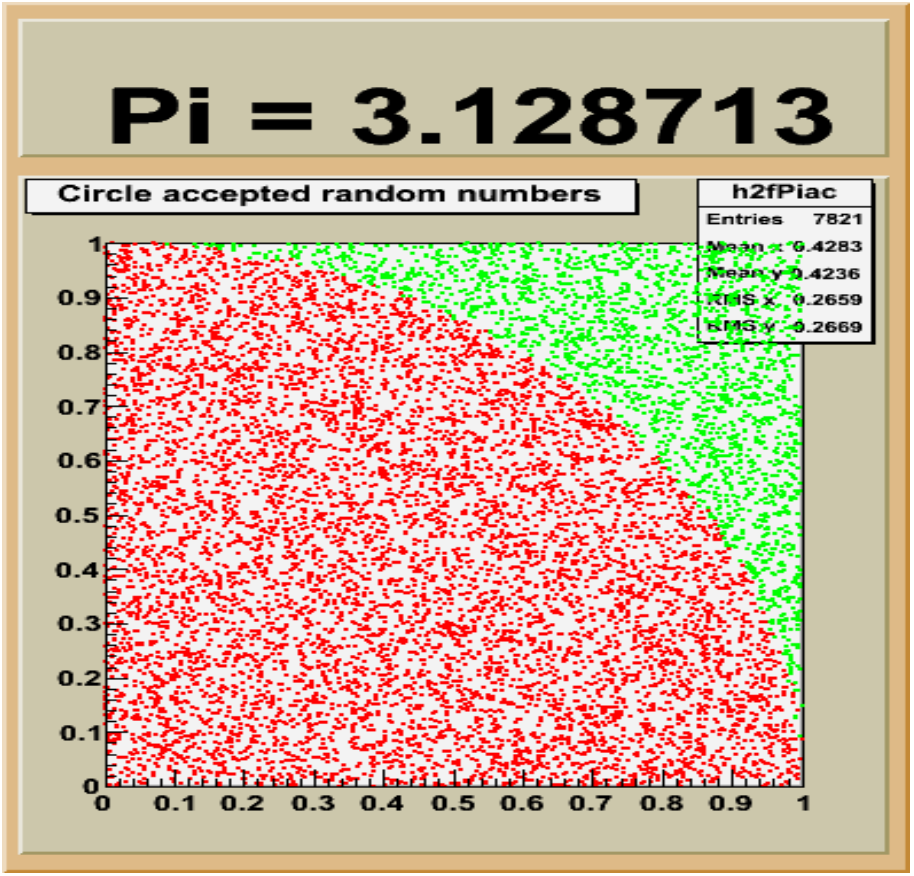
Grundsätzlich ist die MC-Methode eine **Integrationsmethode**

in einer Dimension natürlich anderen Verfahren unterlegen !

Beispiele

Beispiel:

Bestimmung der Fläche eines (Viertel-) Kreises (s. Root-Beispiel calc_pi.C)



Verhältnis der roten Punkte zur Gesamtzahl entspricht
Verhältnis der Flächen von Viertelkreis und Quadrat

Beispiele (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 $\sqrt{N}$**

Dieses Beispiel ist ohne viel Aufwand auch auf mehr als drei Raumdimensionen zu erweitern, statistischer Fehler hängt nach wie vor nur von $\sqrt{N}$ ab - bei anderen Integrationsverfahren steigt die Anzahl der notwendigen Rechenschritte exponentiell mit der Zahl der Dimensionen !

Beispiele (3)

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.

Beispiele (4)

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

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

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

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

Insgesamt also

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

mit statistischem Fehler

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

Funktioniert auch in mehreren Dimensionen; Fehler bleibt dabei gleich!

Beispiele (5)

Aus der Physik: **Messung der Lebensdauer von Teilchen**
mit Digitaluhr bei bestimmter Genauigkeit und Maximalzeit

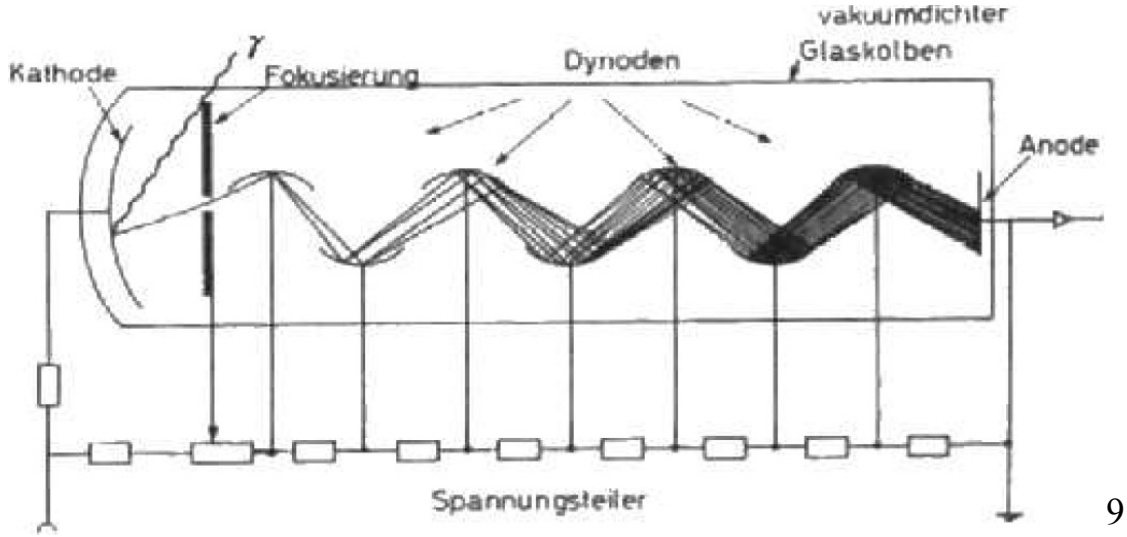
- zufällige Lebensdauer eines Teilchens aus Zerfallsgesetz (Exponentialverteilung)
- zufälligen Messfehler addieren → simulierter Messwert
- Messwerte innerhalb der Maximalzeit akzeptieren

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.

Simulation der Zahl der Elektronen in der letzten Stufe einer „Sekundär-elektronenvervielfacher-Röhre"
(Photomultiplier)

(Übungsaufgabe …)



Grundsätzliches zur MC-Methode

Grundsätzlich ist die **MC-Methode** ein **Integrationsverfahren**,
bei der Simulation von Messwerten die Auswertung eines „Faltungsintegrals“:

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

Im Beispiel **Lebensdauermessung** ist f(x) die Verteilung der wahren Lebensdauer, t(x,x') beschreibt die Auflösung des Messapparates; das oben skizzierte Verfahren liefert auf einfache Weise die Verteilung f ' (x') der Messwerte x'.

Vorteil der MC-Methode:
einfache Summe von Zufallsvariablen entspricht kompliziertem Faltungsintegral

In der Praxis werden Messwert fast immer von einer großen Zahl von Effekten verfälscht (Auflösung des Messgerätes, Rauschen, Digitalisierung, systematische Effekte usw.), d.h. die zu bestimmenden Faltungsintegrale sind hoch-dimensional.
In der MC-Methode werden die Störeinflüsse einfach der Reihe nach aufaddiert !

Erzeugung von gleichverteilten (Pseudo-) Zufallszahlen

Erzeugung von (Pseudo-)Zufallszahlen

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

Anforderungen:

- „zufällig“ verteilt mit langer Periode
- Erzeugung muss schnell sein
- Reproduzierbarkeit solcher Zahlenfolgen oft erwünscht (nicht bei Spielen …)

Einfacher Zufallszahlengenerator basiert auf

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

Allgemeiner (und besser):

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

m	Modulus
0<a<m	Multiplikator
0<c<m	Inkrement
$0 \le x_0 < m$	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
unter Linux: von `/dev/random` kann man Zufallszahlen lesen ...

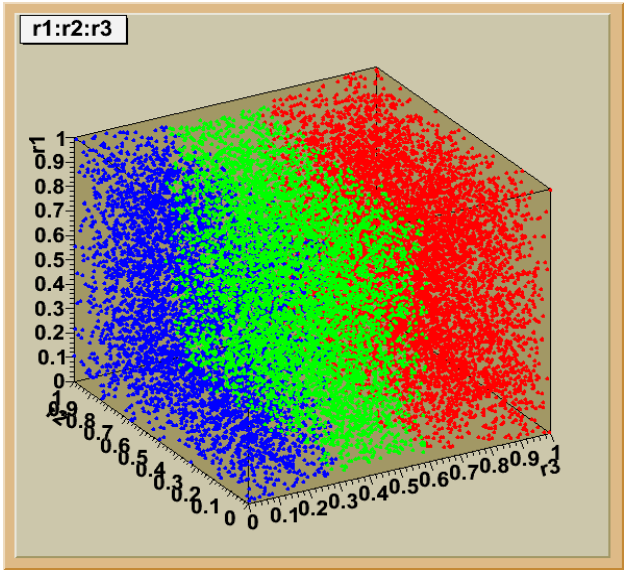
Erzeugung von Zufallszahlen (2)

Sind die Zufallszahlen „gut“ ?

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

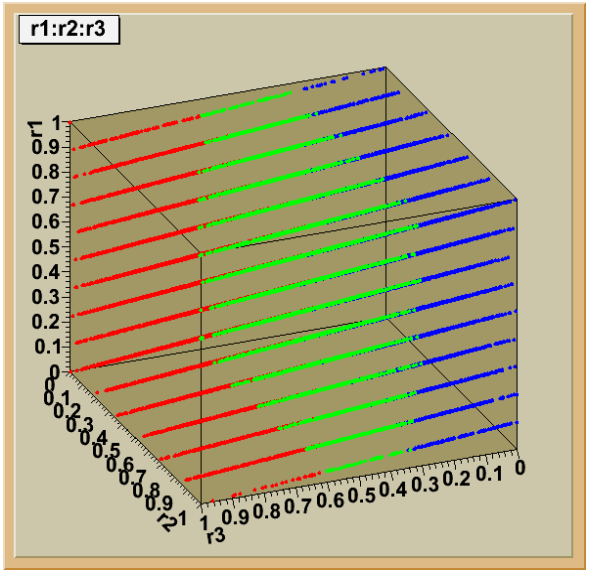
Betrachte Paare, Triplets n-tuples von benachbarten Zahlen,
einfache Kontrolle durch Auftragen als n-dim Histogramm, auf „Muster“ achten!

s. Beispiel `myrand`



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 weit überschritten !

Erzeugung von Zufallszahlen (3)

Mersenne-Twister Algorithmus

- basiert auf Mersenne-Primzahlen (d.h. Zweierpotenz -1)
- extrem lange Periode
- gute Gleichverteilung bis zu 623 Dimensionen (bewiesen)
- Zustand beschreiben durch 624+1 Integer-Zahlen (32 bit),
  die aus Saat mit einfachem linear-kongruenten Generator
  initialsiert werden
- hinreichend schnell

==> Mersenne-Twister ist Pseudo-Zufallszahlengenerator der Wahl

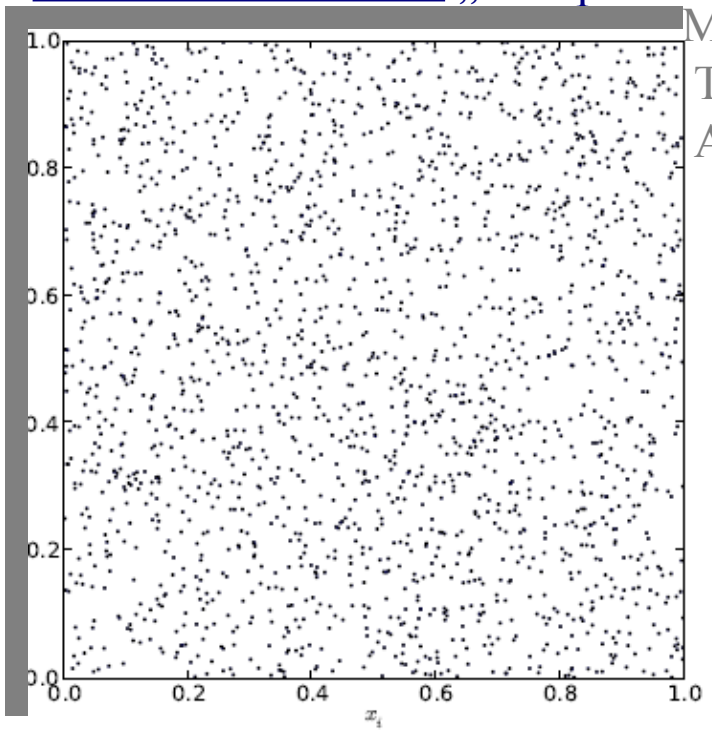
Vorhanden in GNU Scientific Library
und in Root: Klasse `TRandom3`

Erzeugung von Zufallszahlen (4) Quasi-Zufallszahlen

Eine gleichmäßigere Abdeckung eines n-dimensionalen Raumes
als mit Pseudo-Zufallszahlen lässt sich mit **„Quasi“-Zufallszahlen** erreichen

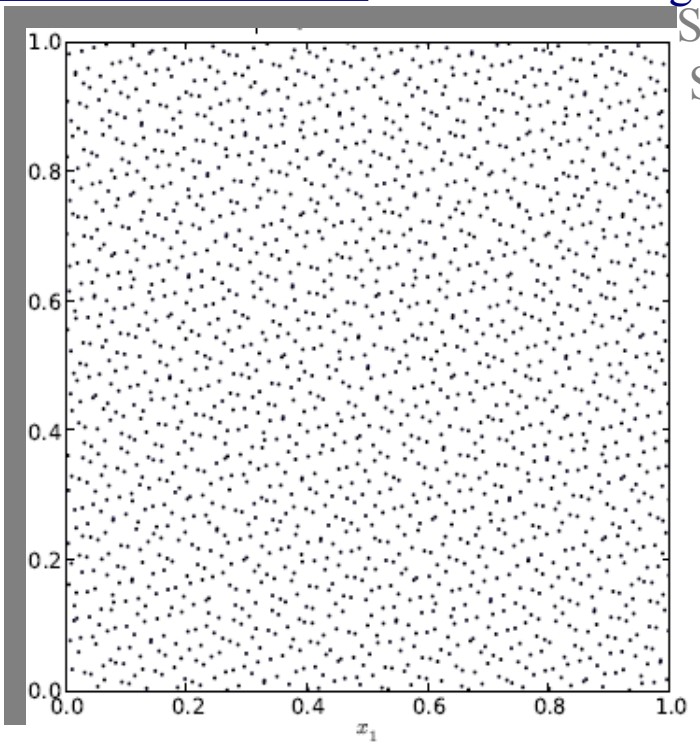
Spezielle Sequenz von nTupeln: neue Punkte fallen immer zwischen die vorherigen Punkte

Pseudozufallszahlen „klumpen“



Mersenne-Twister-Algorithmus

Quasi-Zufallszahlen decken Fläche gut ab



Sobol-Sequenz

- in 2-14 Dimensionen überlegen (Konvergenz ~ n^{-1})
- Dimensionalität nachträglich nicht veränderbar
- Sequenz muss bei Erhöhung der Genauigkeit an der gleichen Stelle fortgesetzt werden

Nur zur Integration geeignet!

Beliebig verteilte Zufallszahlen

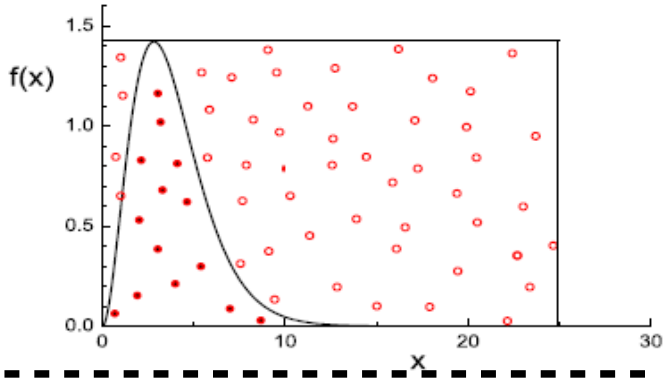
Beliebig verteilte Zufallszahlen

1) Von Neumann' sches Rückweisungsverfahren

Wegwerfmethode, engl. „accept-reject method“

Generiere zwei Zufallszahlen r_1, r_2,

akzeptiere $x=r_1$ wenn $r_2 < f(x)$; x ist dann gemäß f(x) verteilt.

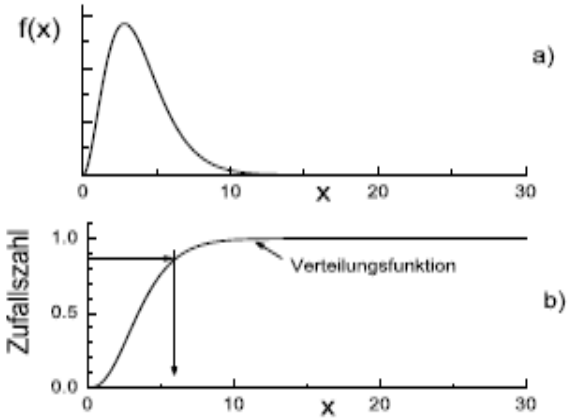


2) Transformationsmethode

transformiere gleichverteilte $r_i \in [0,1[$, so dass

für geeignete Funktion t() $x_i=t(r_i)$ der Verteilung f(x) folgt

t() ist die Inverse der Verteilungsfunktion F()



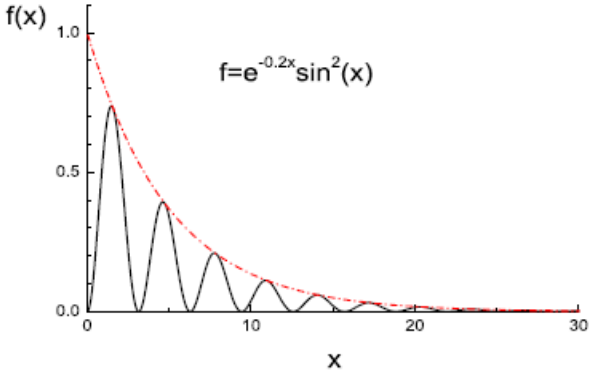
3) Majorantenmethode (engl. „Importance Sampling“)

Kombination aus 1) und 2) : finde Einhüllende $m \geq f \ \forall \ x$

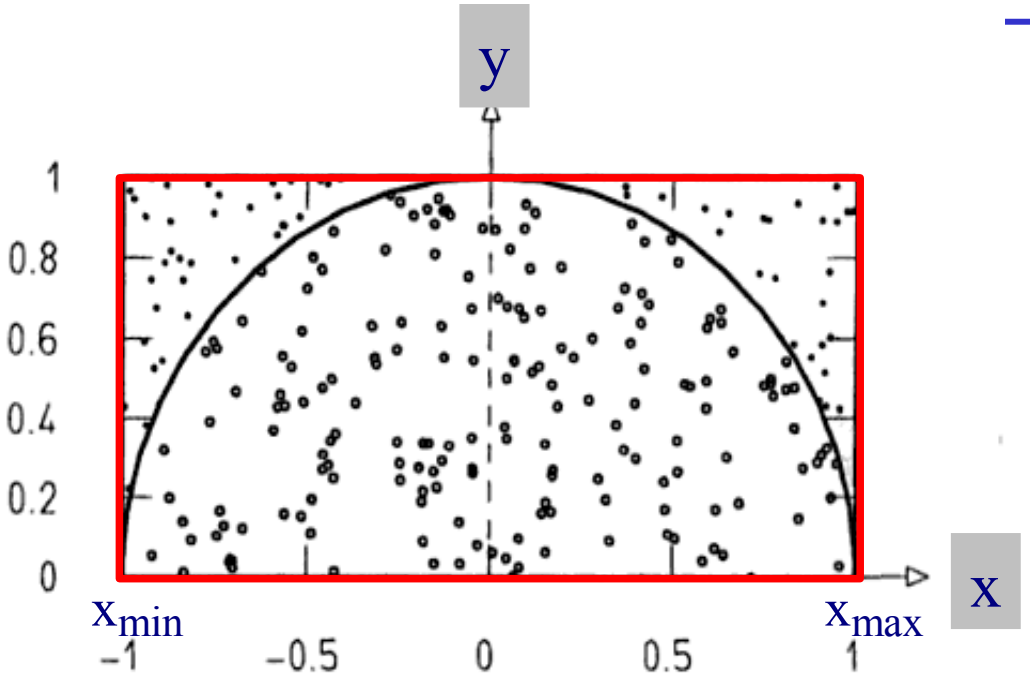
die mit 2) erzeugt werden kann, und erzeuge Zufallszahl x;

wende dann 1) an, d.h. erzeuge zweite Zufallszahl r_2 zwischen

0 und m(x) und akzeptiere x, wenn $r_2 < f(x)$.



von Neumann'sches Rückweisungsverfahren



Beispiel:
Kreis als Verteilungsfunktion f(x)

aus gleichverteilten Zufallszahlen
r_{2i} , $r_{2i+1} \in]0,1[$ erzeuge

$x_i = x_{min} + r_{2i}\,(x_{max}-x_{min})$

$y_i = r_{2i+1}\, f_{max}$

- Verteilungsdichte wird in Rechteck („Box“) eingeschlossen
- erzeuge gleichförmig in der Box verteilte Paare von Zufallszahlen (x_i,y_i) (s.o.)
- 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;
im oben 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.

Transformation der Gleichverteilung

Aus gleichverteilten Zufallszahlen $r_i \in]0,1[$ wird durch

Variablentransformation $x_i = x(r_i)$ eine Verteilungsdichte f(x) gewonnen

$$f(x)dx = u(r)dr$$

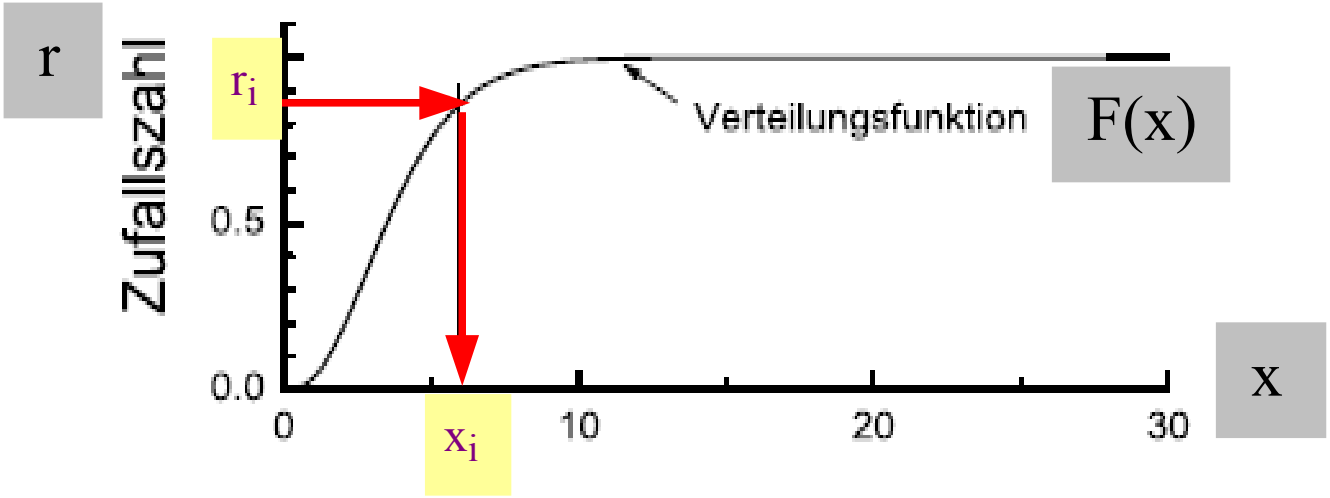
$$\int_{-\infty}^{x} f(x')dx' = \int_{0}^{r} u(r')dr' = r$$

$$F(x) = r$$

$$x = F^{-1}(r)$$

s. Abschnitt Variablentrasnsforrmation

Die Verteilungsfunktion der gewünschten Verteilungsdichte muss (mit vertretbarem numerischem Aufwand) integrierbar und invertierbar sein !



Beispiele für die Transformationsmethode

Dreieck-Verteilung

$$f(x) = 2x \quad 0 \le x \le 1$$
$$x(r) = \sqrt{r}$$

$$f(x) = (n+1)x^n \quad 0 \le x \le 1,\ n > -1$$
$$x = r^{1/(n+1)}$$

Exponentialverteilung

$$f(x) = \gamma e^{-\gamma x}$$
$$x(r) = -\frac{1}{\gamma}\ln(1-r)$$

Breit-Wigner-Verteilung

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

Log-Weibull -Verteilung

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

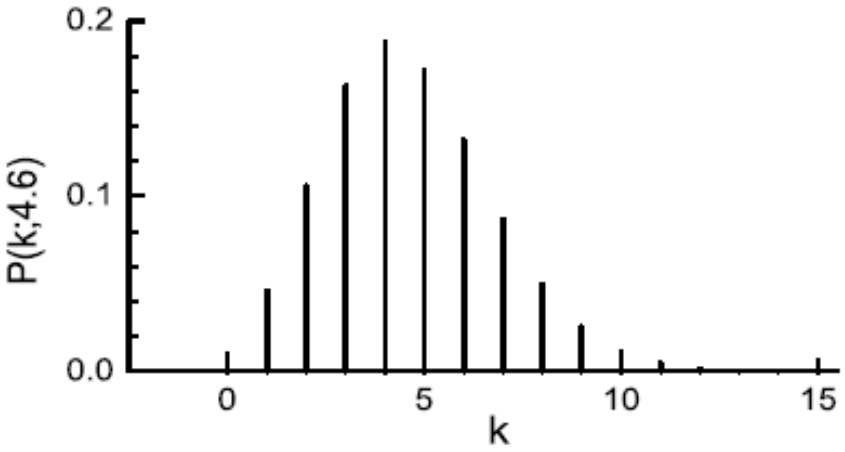
Paar von Gauß-Zahlen (Herl. s. später)

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

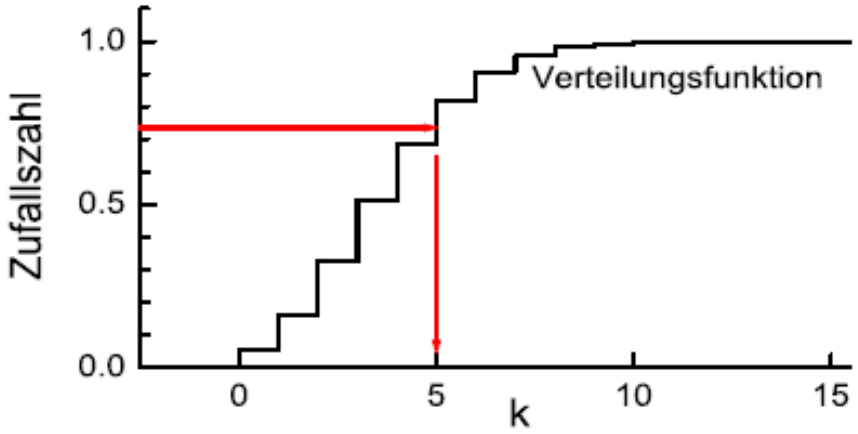
Anm.:
r kann auch durch 1-r ersetzt werden

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

Verteilungen aus Histogrammen

Dieses Verfahren funktioniert auch zur Erzeugung von Zufallszahlen,
die gemäß einer empirischen Verteilung aus einem Histogramm verteilt sind

- r_i bestimmt das Bin j;
- der Rest r_i-S(j-1) wird zur Interpolation innerhalb des Bins verwendet

Auch zwei-dimensionale Verteilungen möglich:
- bilde Randverteilungen $g_i = \sum_j h_{ij}$
- generiere zunächst i und dann für gegebenes i eine Bin-Nummer j
(benötigt für jedes i die Summenverteilung über j)

Majoranten-Verfahren (engl. Importance Sampling)

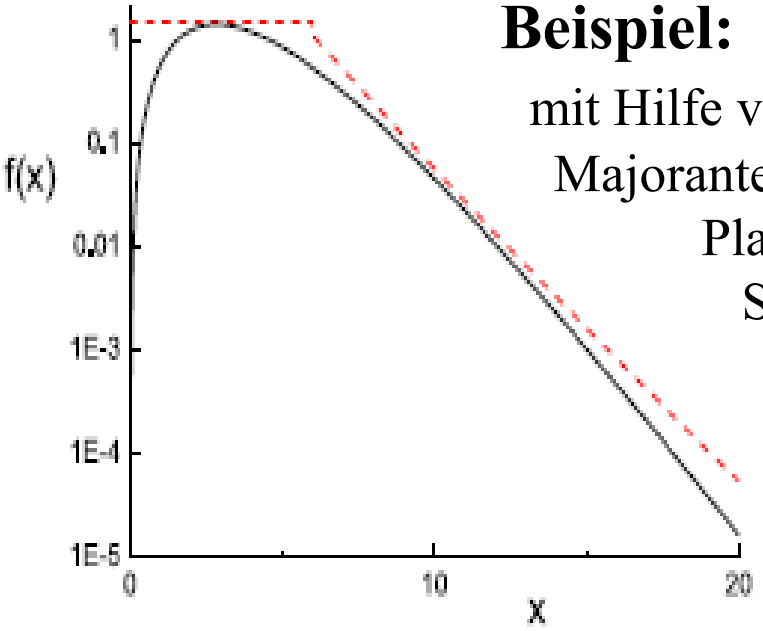
Verbesserte Wegwerfmethode:

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 $x_i=M^{-1}(r_{2i})$ und eine weitere Zufallszahl $r_{2i+1} \in [0,1]$
- akzeptiere x_i, wenn $r_{2i+1} \cdot m(x) > f(x)$ ist

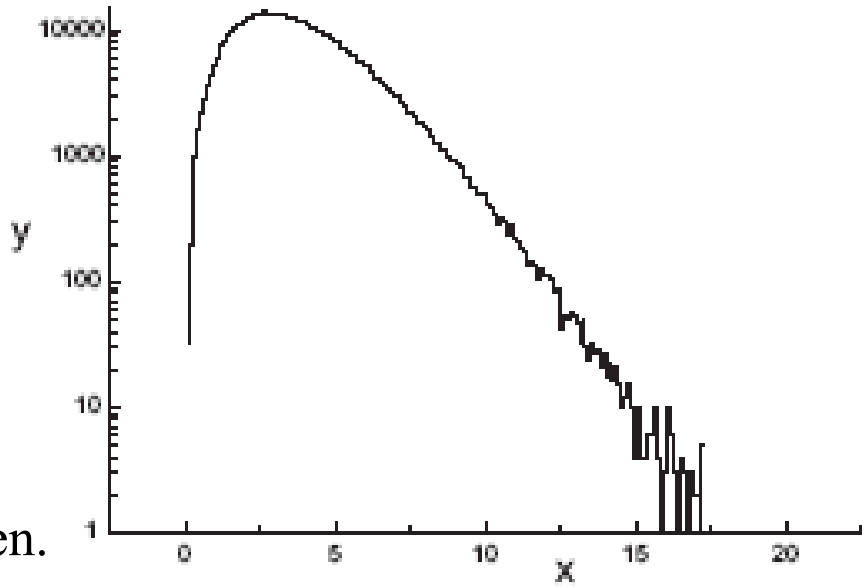
(für jeden Wert von x wird ein Bruchteil (m(x) -f(x)) / f(x) an x-Werten verworfen)

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.



Gaußverteilte Zufallszahlen

Problem: $\int \exp(-x^2)\,dx$ nur numerisch bestimmbar

(in C, C++ über die Funktion $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$ verfügbar)

Gauß-Funktion in 2 Dimensionen in Polarkoordinaten analytisch integrierbar:

$$\int\int \frac{1}{2\pi} \exp\{-(x^2+y^2)/2\} dx\, dy \quad \rightarrow$$

$$\int_0^{2\pi} \frac{1}{2\pi} d\phi \cdot \int_0^r \exp\{-r^2/2\}\, r\, dr$$

Gleichverteiung in Φ * Radialverteilung integrierbar mit $r\,dr = \frac{1}{2}\,dr^2$

$$\int_0^r \exp\{-r^2/2\}\, \tfrac{1}{2} dr^2 = (1 - exp\{-r^2/2\})$$

Verteilungsfunktion des Radialteils lässt sich nach r auflösen –
d.h. können Transformationsmethode anwenden !

=>

Erzeuge zwei gleichverteilte Zufallszahlen $u_1, u_2 \in]0,1]$,

setze $x_1 = \sqrt{(-2\ln u_1)}\cos(2\pi u_2)$

$x_2 = \sqrt{(-2\ln u_1)}\sin(2\pi u_2)$,

x_1 und x_2 sind standardnormalverteilt

Effizienteres Verfahren zur Erzeugung von zwei normalverteilten Zufallszahlen (s. V. Blobel)

(Beispiel für die Kombination aus Transformations- und Rückweisungsverfahren)

Trigonometrische Funktionen sind numerisch aufwänig,
ein etwas effizienteres Verfahren geht so:

1. erzeuge zwei gleichverteilte Zufallszahlen $u_1, u_2 \in [0,1[$
   setze $v_1 := 2u_1 - 1$, $v_2 := 2u_2 - 1$
2. $v^2 = v_1^2 + v_2^2$
3. falls $v^2 \geq 1$ → gehe zu 1.
   (akzeptierte Werte von v_1, v_2 gleichverteilt innerhalb des Einheitskreises)
4. $z_1 = v_1 \sqrt{(-2 \ln(v^2) / v^2)}$
   $z_2 = v_2 \sqrt{(-2 \ln(v^2) / v^2)}$

z_1 und z_2 sind unabhängig voneinander standard-normalverteilt

Normalverteilte Zufallszahlen mit Erwartungwert μ und Standardabweichung σ erhält man durch die Transformation $x = \sigma z + \mu$

Erzeugung korrelierter, Gauß-verteilter Zufallszahlen

Gaußverteilung in n Dimensionen: $G(\vec{x};\vec{x}_0,V) = \frac{1}{(2\pi)^{\frac{n}{2}}|V|^{\frac{1}{2}}} \cdot \exp\{-\frac{1}{2}(\vec{x}-\vec{x}_0)^T V^{-1}(\vec{x}-\vec{x}_0)\}$

x_0: n Erwartungswerte

V: Kovarianz-Matrix mit $(n^2+n)/2$ unabhängigen Parametern

|V|=det V ist die Determinante von V

Sei B = V^{-1}, schreibe B als Produkt aus Dreiecksmatrizen: $B=D^T D$ (Cholesky-Zerlegung)

Setzte dann **u**=D (**x**-$\mathbf{x}_0$) => $G(\vec{x};\vec{x}_0,V) = \frac{1}{(2\pi)^{\frac{n}{2}}|V|^{\frac{1}{2}}} \cdot \exp\{-\frac{1}{2}(\vec{u}^T \vec{u}\}$

Das ist die Verteilung von n unkorrelierten , Gauß-verteilten Zufallszahlen **u**

Damit ist die Vorgehensweise klar:

1. würfle n unabhängige Gauß-verteilte Zufallszahlen **u**
2. transformiere Vektor **u**: $\mathbf{x} = D^{-1}\mathbf{u} + \mathbf{x}_0$

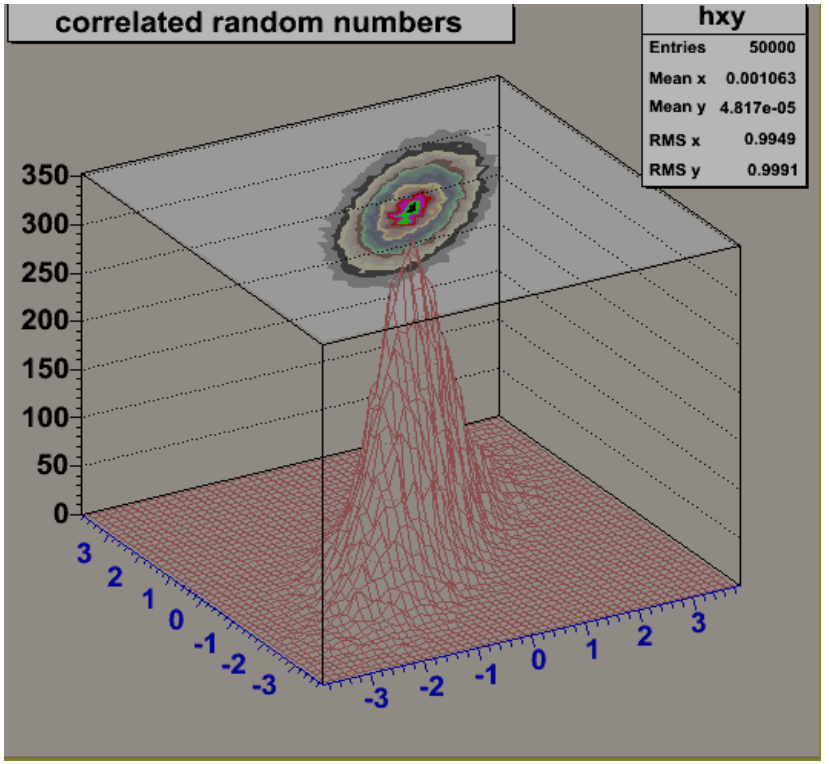
Korrelierte Gauß-verteilte Zufallszahlen in 2 Dimensionen

Daraus ergibt sich für die Erzeugung von
korrelierten Zufallszahlen x_1, x_2 mit

- Erwartungswerten μ_1 , μ_2
- Standardabweichungen σ_1 , σ_2 und
- **Korrelation** ρ:

$$u_1 \text{ und } u_2 \text{ standard} - \text{normalverteilt}$$
$$x_1 = \mu_1 + \sigma_1 \, u_1$$
$$x_2 = \mu_2 + \sigma_2 \left(\rho \, u_1 + \sqrt{1 - \rho^2} \, u_2 \right)$$

Details s. z.B. Blobel / Lohrmann



Darstellung eines 2d-Histogramms, gefüllt mit 75% korrelierten, standard-normal-verteilten Zufallszahlen
(Beispiel rangauss2d.C)

Genauigkeit der MC-Methode

Vergleich mit Trapez-Methode:

in 1 Dimension Trapez: n = Zahl der Stützstellen

Genauigkeit $\propto 1/n^2$

MC: N Zufallszahlen

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

In d Dimensionen: Trapezoid: Genauigkeit $\propto 1/n^{2/d}$

MC Genauigkeit $\propto 1/\sqrt{N}$ **unabhängig von d**

MC gewinnt für d>4;

andere Verfahren besser als Trapezoid-Methode,

aber für genügend großes d klarer Vorteil für MC !

Zufallszahlen in Root

Root-Klasse **TRandom** zur Erzeugung von Zufallszahlen

– einfacher, linear kongruenter Generator mit Periodenlänge 10^9

– schnell

– niedrigste Bits nicht unkorreliert !

Nicht in statistischen Studien verwenden !

Statt dessen

`TRandom1` (RANLUX-Algorithmus, Periodenlänge ~10^{171}, langsam)

`TRandom2` (Periodenlänge ~10^{25}, schnell)

`TRandom3` (Mersenne-Twister Alg., Periodenlänge ~10^{600}, akzeptabel schnell)

verwenden !

Methoden liefern Zufallszahlen gemäß

Exp(tau) Integer(imax) Gaus(mean,sigma)

Rndm() Uniform(x1) Landau(mpv,sigma)

Poisson(mean) Binomial(ntot,prob)

und einige mehr

Zufallszahlen in Root (2)

Methode SetSeed(UInt_t seed=0) zur Initialisierung
- seed=0: Systemuhr zur Initialisierung (jede Zahlenfolge anders)
- seed !=0: feste Zahlenfolge, abhängig vom Wert von seed

Globaler Zeiger TRandom *gRandom in Root initialisiert
Sehr nützlich, um immer die gleiche Folge von Zufallszahlen inherhalb eines Root-Programms zu verwenden.

```
// TRandom durch TRandom3 ersetzen
 delete gRandom;
 gRandom = new TRandom3(seed);
```

Man kann auch auch einen eigenen (lokalen) Generator initialisieren:

```
 TRandom3 *myrandom=new TRandom3(seed);
```

Zufallszahlen in Root (3)

Root-Klassen enthalten bequeme Möglichkeit, um
Zufallszahlen zu erzeugen, die gemäß einer Funktion
TF1, TF2
oder
einem Histogramm
TH1, TH2
verteilt sind: Methoden
GetRandom() für **TH1, TF1**
GetRandom2() für **TH2, TF2**
GetRandom3() für **TH3, TF2**

Achtung: für Funktionen TFi werden die Zufallszahlen ebenfalls über Histogramme mit einer Anzahl von Punkten fNpx erzeugt.