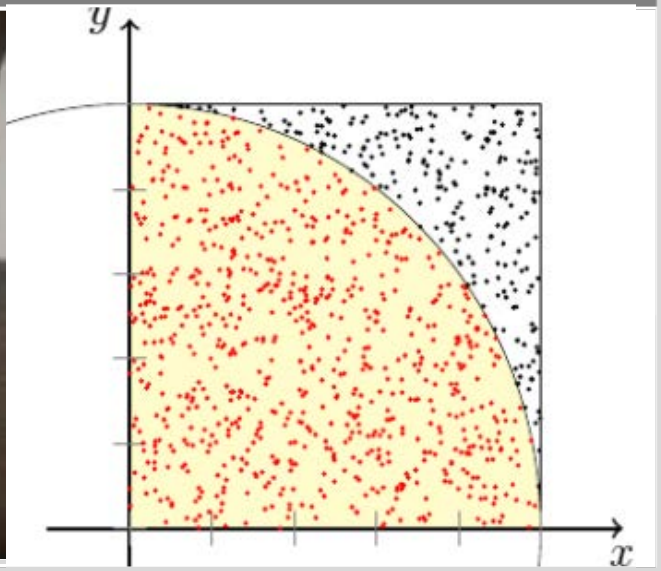
Moderne Methoden der Datenanalyse

Zufallszahlen und Monte Carlo Methoden

Andreas B.Meyer

Fakultät für Physik
Institut für Experimentelle Kernphysik - IEKP

9. Mai 2017



Vorlesungsprogramm

- Einführung: Überblick und grundlegende Konzepte (1)
- Einführung: Überblick und grundlegende Konzepte (2)
- Zufallszahlen und Monte-Carlo Methoden

A.MEYER

- Hypothesentests (1)
- Parameterschätzung
- Parameterschätzung (Goodness-of-fit)
- Optimierungs- und Parametrisierungsmethoden

R.ULRICH

- Konfidenzintervalle
- Hypothesentests (2)
- Klassifikation
- Klassifikation
- Klassifikation
- Messen und Entfalten
- Systematische Unsicherheiten

A.MEYER

2.

ZUFALLSZAHLEN UND MONTE-CARLO METHODEN

2. Zufallszahlen und Monte Carlo Methoden

- Grundsätzliches
- Beispiele und Anwendungen

2.1
DIE MC METHODE

- Gleichverteilte (Pseudo-)Zufallszahlen
- Tests und Eigenschaften

2.2
ZUFALLSZAHLEN

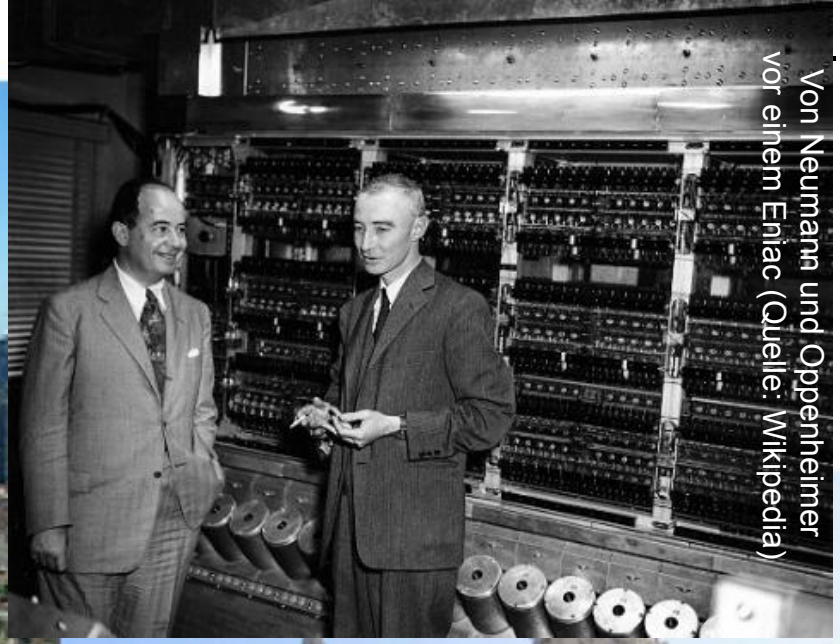
- Verfahren zur Erzeugung von Verteilungen
- Korrelierte Zufallszahlen

2.3
ERZEUGUNG VON VERTEILUNGEN

- Literatur:
  - Blobel/Lohrmann: *Statistische und numerische Methoden der Datenanalyse (Kap. 5)*
  - Bohm/Zech: *Einführung in die Statistik und Messwertanalyse (Kap. 4)*
  - Brandt: *Datenanalyse (Kap. 4)*
  - Cowan: *Statistical Data Analysis (Kap. 3)*

2.1

DIE MONTE-CARLO METHODE



Von Neumann und Oppenheimer vor einem Eniac (Quelle: Wikipedia)

Fermi, Ulam, van Neumann, Metropolis, ...

Ursprung in den 1940er Jahren: siehe z.B. http://lib-www.lanl.gov/la-pubs/00326866.pdf

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)

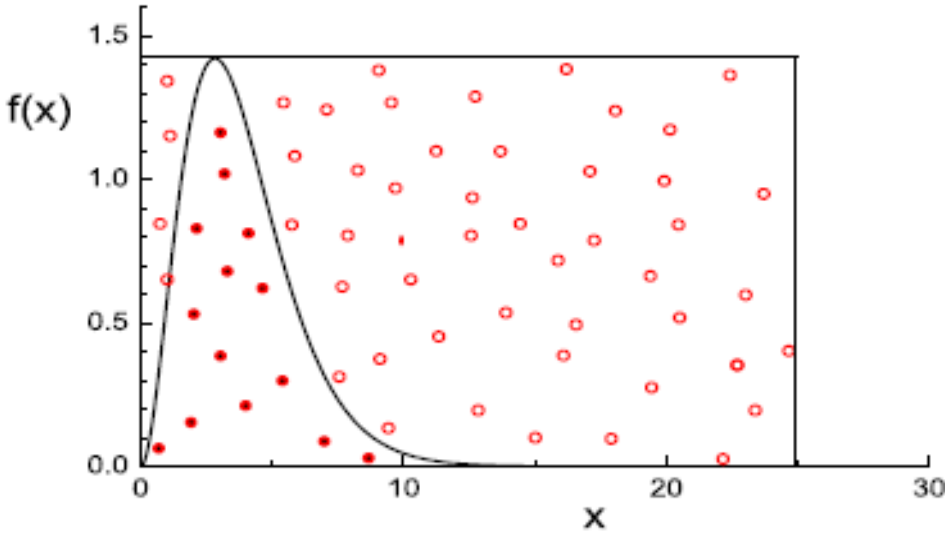
- Aus der Teilchenphysik ist MC nicht wegzudenken:
  - Simulation von Physik-Ereignissen und Detektoreigenschaften
  - Entwerfen und Planen von Experimenten
  - Test von Analyse-Methoden (statistische Tests / “Pseudo-Experimente”)
  - Matrix-Elemente und Phasenraum-Integrale

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 $a<x_i<b$



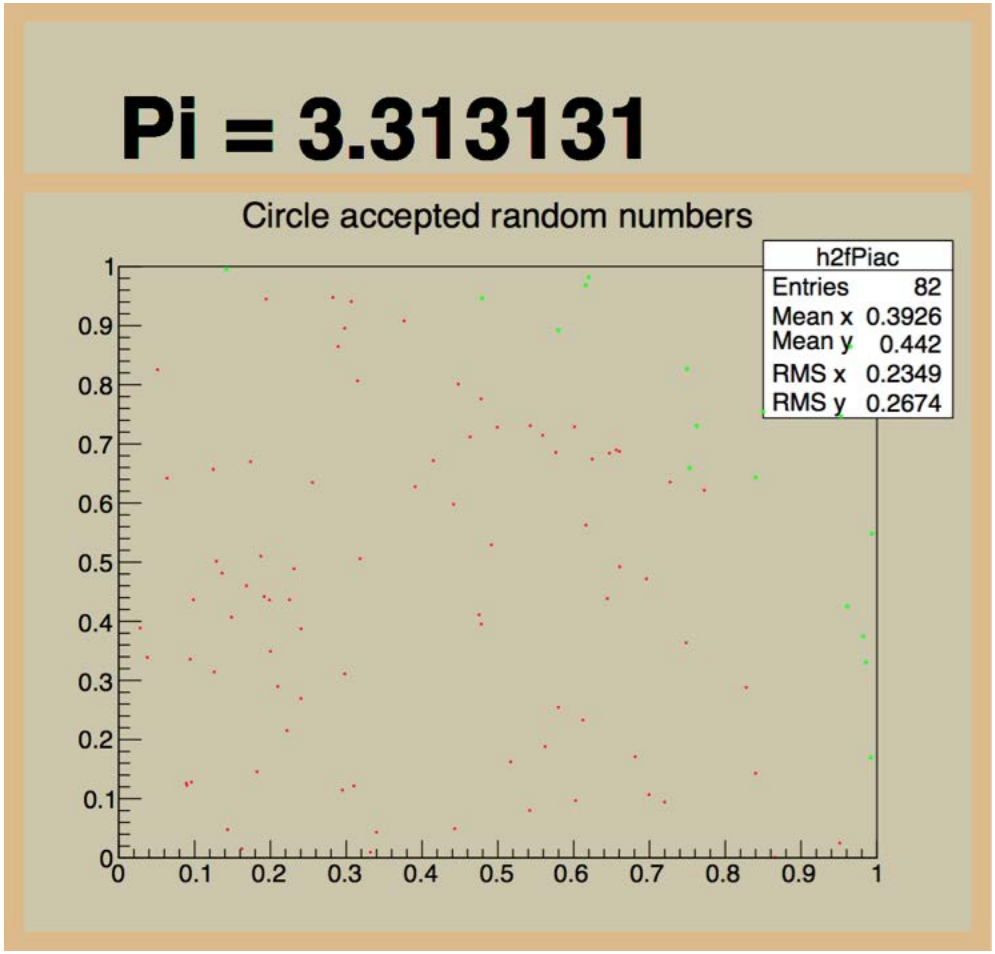
Beispiel: Bestimmung der Fläche eines Kreises

- Rot: Punkte, für die $x^2+y^2 < 1$

- Ersetzt Quadratur des Kreises:

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

  $N = 10^2$: 3.313131



Beispiele: calc_pi.C animate_pi.py

Beispiel: Bestimmung der Fläche eines Kreises

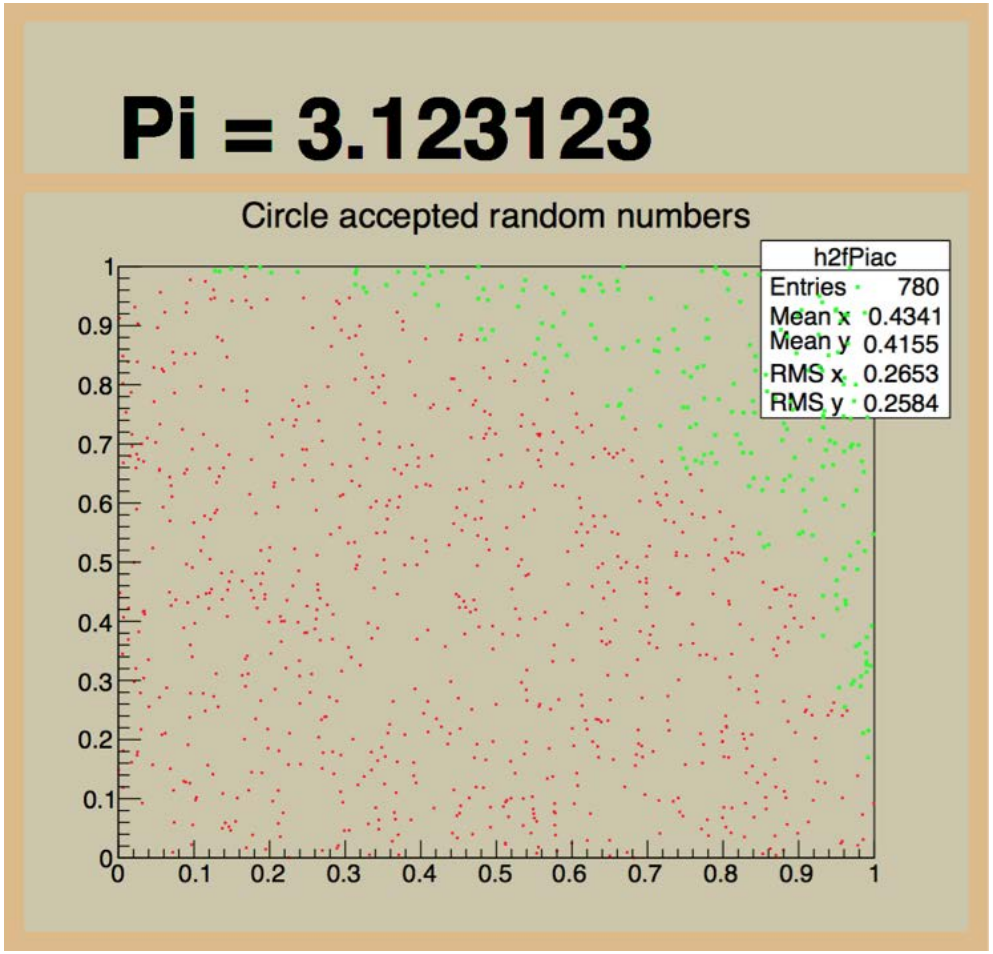
- Rot: Punkte, für die $x^2+y^2 < 1$

- Ersetzt Quadratur des Kreises:

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

  $N = 10^2$: 3.313131

  $N = 10^3$: 3.123123



Beispiele: calc_pi.C animate_pi.py

Beispiel: Bestimmung der Fläche eines Kreises

- Rot: Punkte, für die $x^2+y^2 < 1$

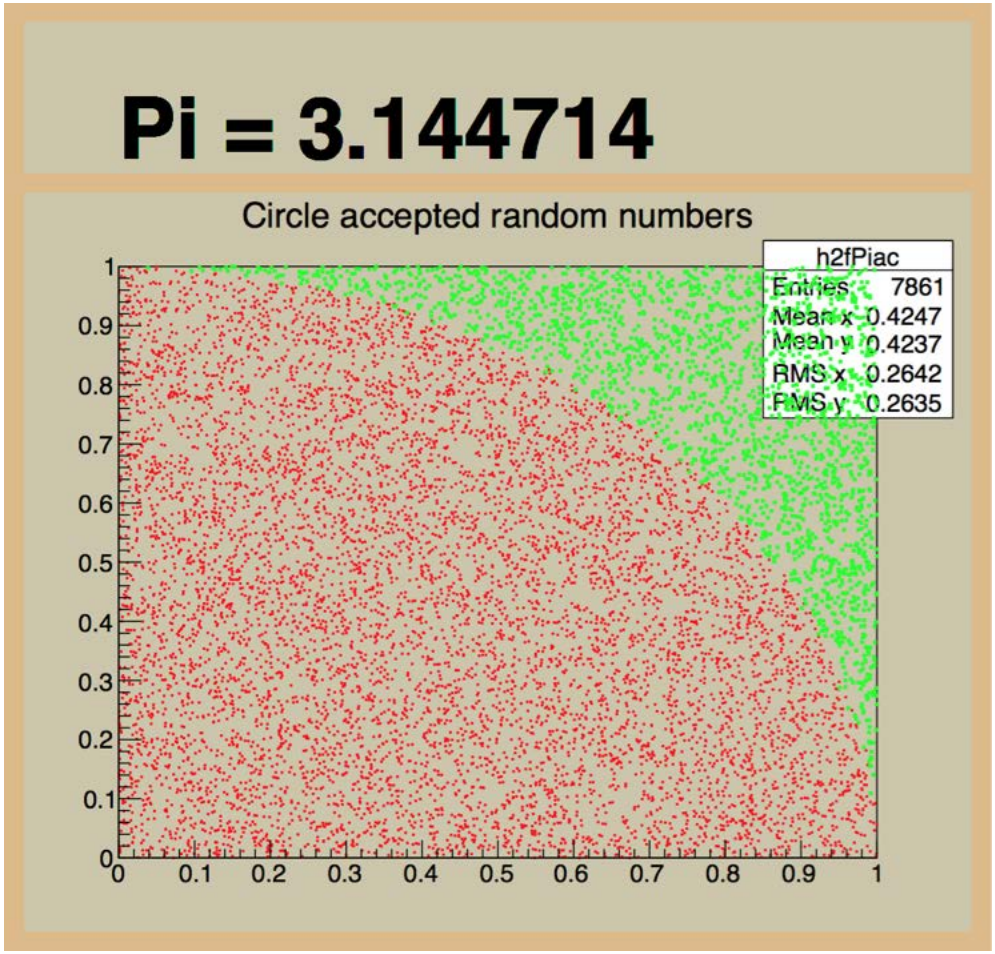
- Ersetzt Quadratur des Kreises:

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

  $N = 10^2$: 3.313131

  $N = 10^3$: 3.123123

  $N = 10^4$: 3.144714



Beispiele: calc_pi.C animate_pi.py

Beispiel: Bestimmung der Fläche eines Kreises

- Rot: Punkte, für die $x^2+y^2 < 1$

- Ersetzt Quadratur des Kreises:

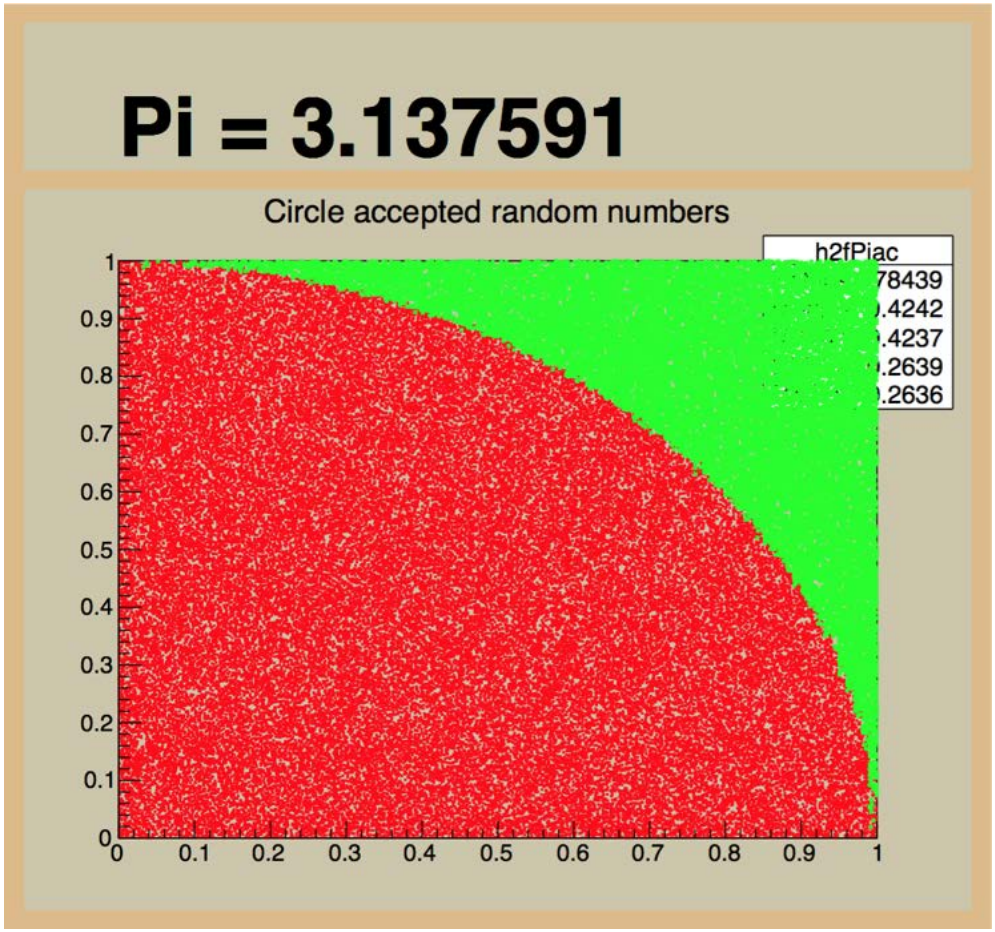
  Verhältnis der roten Punkte zur Gesamtzahl aller Punkte entspricht Verhältnis der Flächen von Viertelkreis und Quadrat

  $N = 10^2$: 3.313131

  $N = 10^3$: 3.123123

  $N = 10^4$: 3.144714

  $N = 10^5$: 3.137591



Beispiele: calc_pi.C animate_pi.py

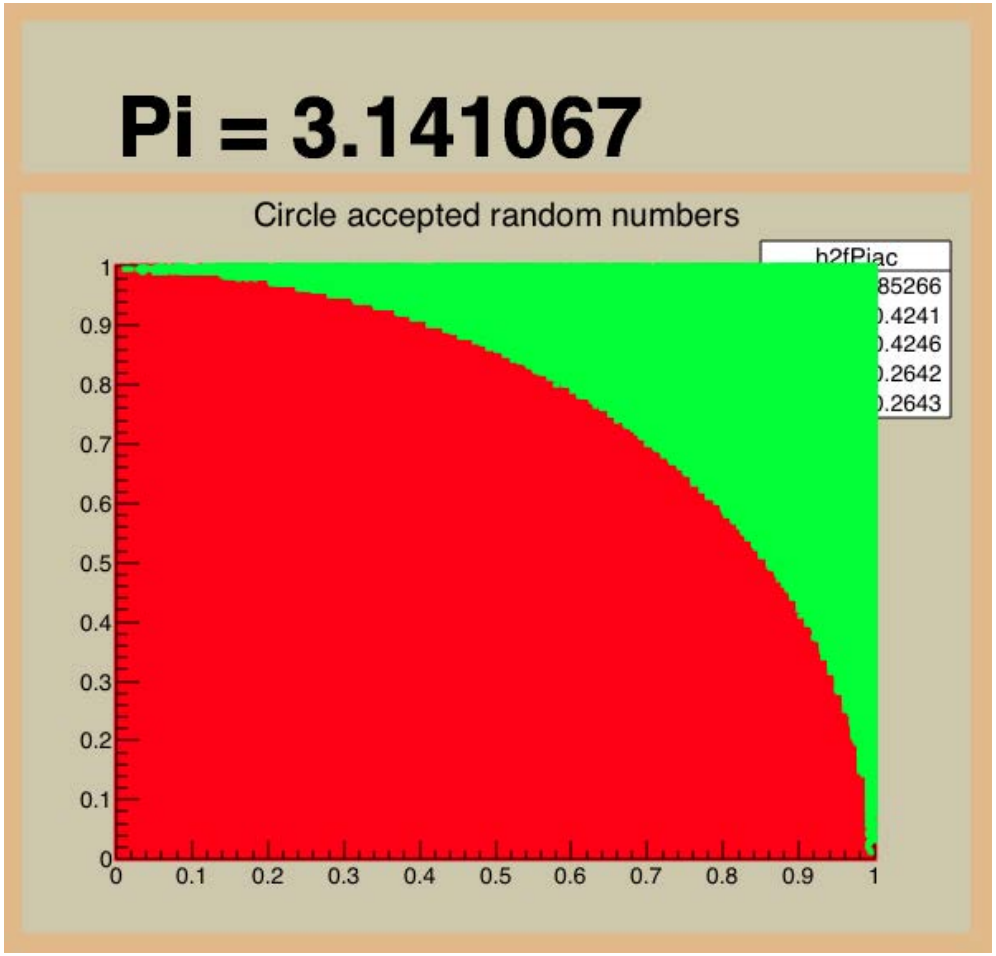
Beispiel: Bestimmung der Fläche eines Kreises

- Rot: Punkte, für die $x^2+y^2 < 1$

- Ersetzt Quadratur des Kreises:

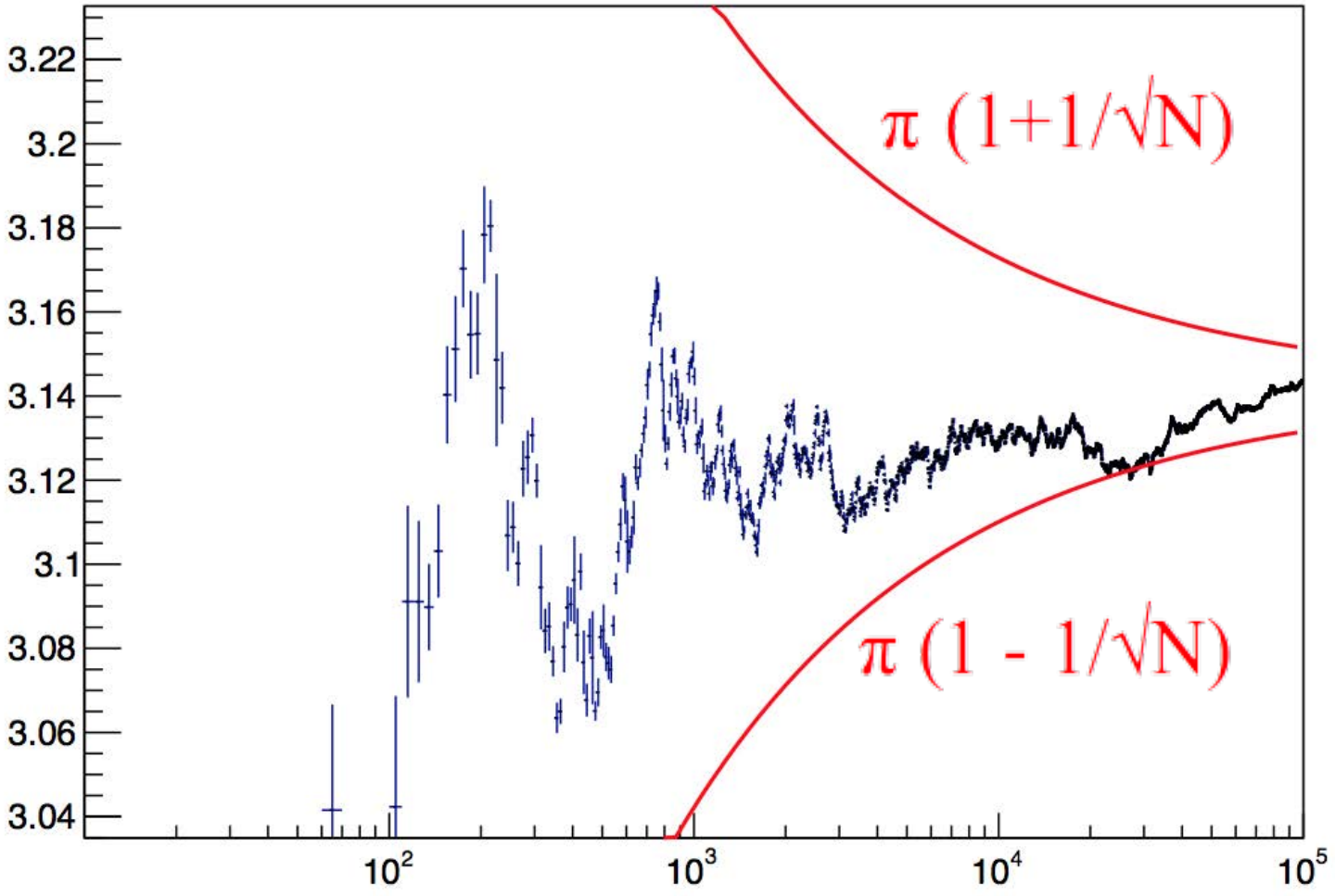
  Verhältnis der roten Punkte zur Gesamtzahl aller Punkte entspricht Verhältnis der Flächen von Viertelkreis und Quadrat

$N = 10^2$: 3.313131
$N = 10^3$: 3.123123
$N = 10^4$: 3.144714
$N = 10^5$: 3.137591
$N = 10^6$: 3.141067



Beispiele: calc_pi.C animate_pi.py

Beispiel: Bestimmung der Fläche eines Kreises

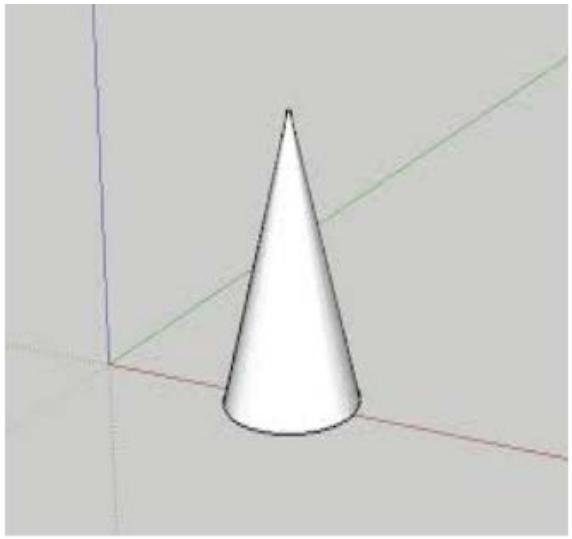


- Genauigkeit wird mit 1/√N besser

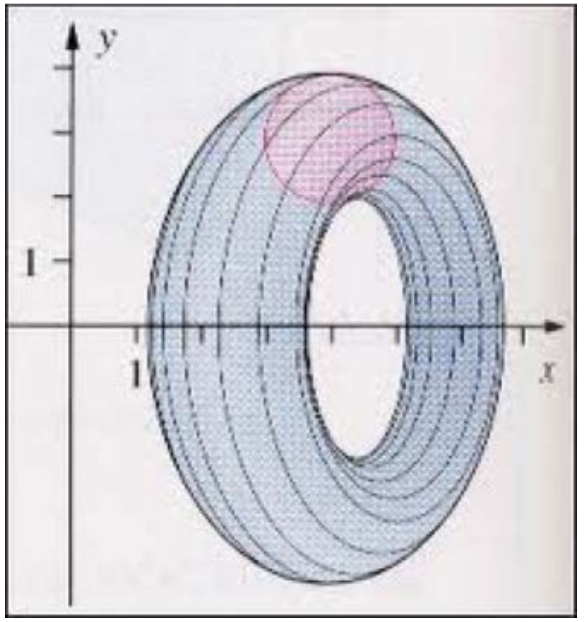
Beispiel: calc_pi.C

Beispiel: Schnittvolumen eines Kegels mit einem Torus

- Definiere Quader, der Schnittvolumen einschließt
- Erzeuge zufällig gleichverteilte Punkte im Quadervolumen
- Teste, ob Punkt im Schnittvolumen liegt, also im Torus und im Kegel.
- Zähle Zahl der Punkte im Schnittvolumen.



Verhältnis der Punkte zur Gesamtzahl der Versuche entspricht dem Verhältnis des Schnittvolumens zum Quadervolumen



- Extrem einfache Rechenvorschrift
- Funktioniert auch für viel-dimensionale Probleme

Statistischer Fehler (Binomialverteilung).
Genauigkeit steigt mit $\sqrt{N}$

Berechnung von Integralen

$$I = \int_a^b \phi(x) dx$$

- Erwartungswert von $\phi(x)$ für eine zwischen a und b gleichförmige PDF:

$$\int_a^b 1 \cdot \phi(x) dx = (b-a) \langle \phi \rangle$$

- Für N gleichverteilte Zufallszahlen $x_i \in [a,b]$ und N groß geht der Mittelwert gegen den Erwartungswert

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

- Also ist:

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

Blobel/Lohrmann Abschnitt 5.6

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

Vergleich MC-Methode mit Trapezregel

- 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: $n^{1/d} = \sqrt[d]{n}$ 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 $d > 4$

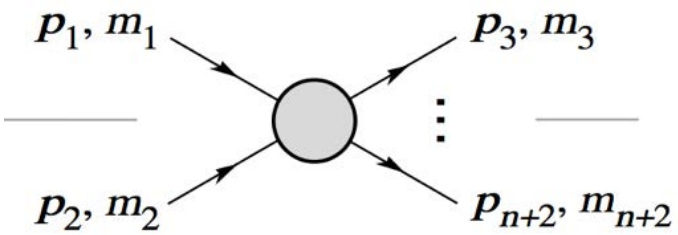
In hochdimensionalen Räumen ist Monte Carlo Integration anderen Methoden überlegen

Blobel/Lohrmann Abschnitt 5.6

MC-Methode: Einige Beispiele und Anwendungen

- Matrix-Elemente und Phasenraum-Integrale
- Vollständige Simulation von Physik-Ereignissen und Detektor
- Test und Optimierung von Analyse-Methoden (verdeckte Analyse)
- Design von Experimenten
- Korrekturfaktoren, Untergrundabschätzungen
- Statistische Tests / Pseudo-Experimente

Beispiel: Berechnung eines Wirkungsquerschnitts



Matrixelement

$$d\hat{\sigma} = \frac{(2\pi)^4 |\mathcal{M}|^2}{4\sqrt{(p_1 p_2)^2 - m_1^2 m_2^2}} \times d\Phi_n(p_1 + p_2; p_3 \ldots p_{n+2})$$

Flussfaktor

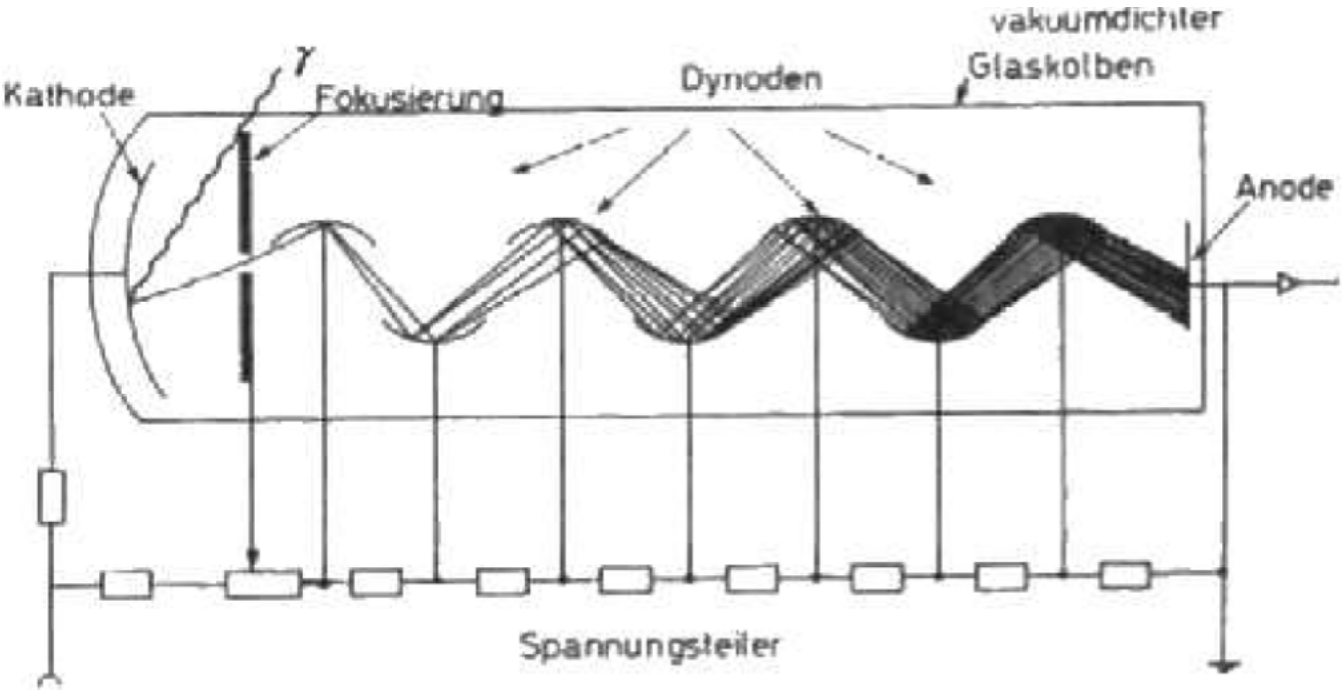
Phasenraum

- Multidimensionales Integral, analytische Integration nur in einfachsten Fällen möglich.
- Hadron-Kollisionen (z.B. LHC): Zusätzlich Integration über Anfangszustände (Partonverteilungsfunktionen)

$$\sum_{ij} \int dx_1 dx_2 f_i(x_1, Q^2) f_j(x_2, Q^2) \hat{\sigma}(Q^2)$$

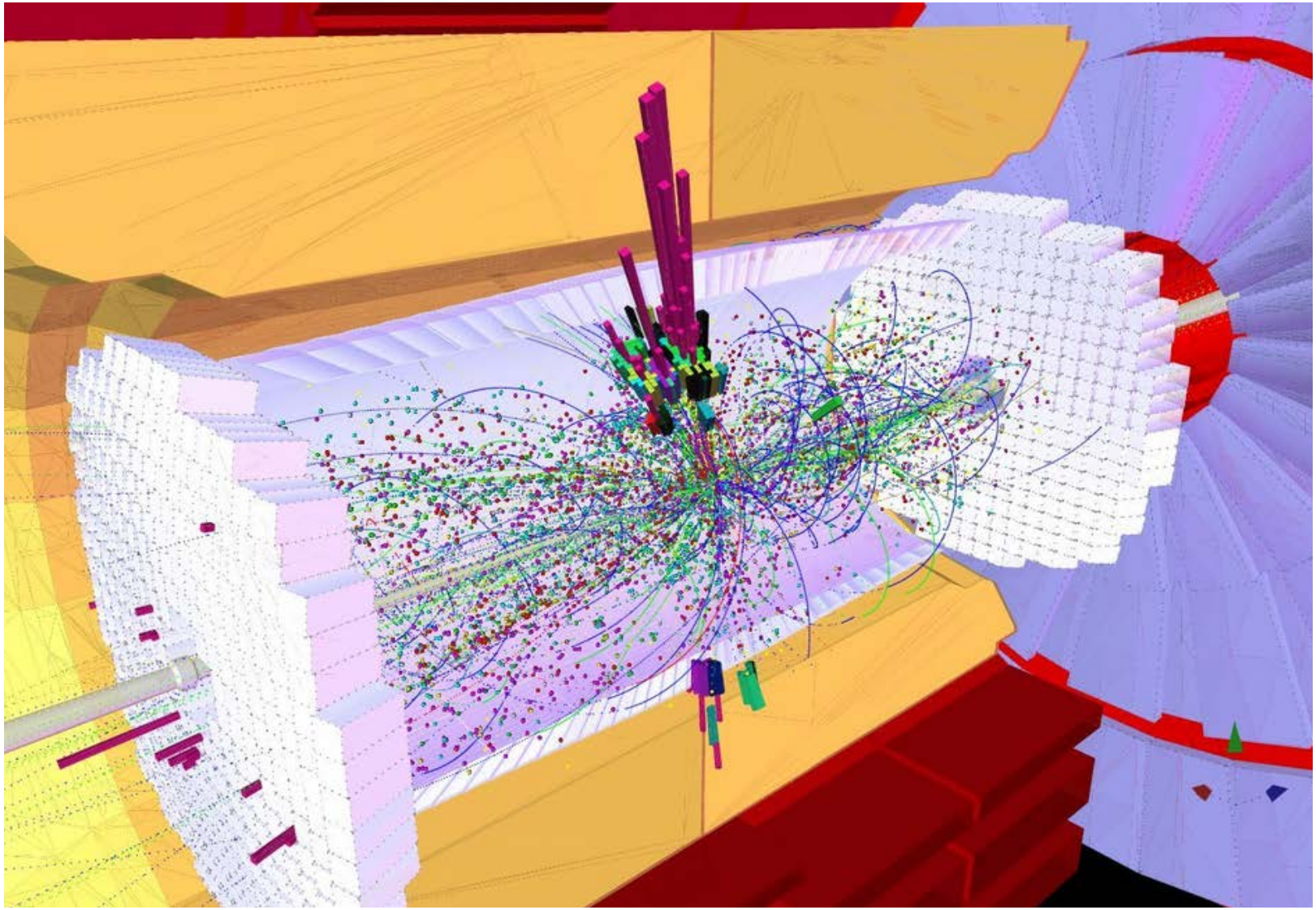
Beispiel: Detektorsimulation

- Simulation der Zahl der Elektronen in der letzten Stufe einer “Sekundärelektronenvervielfacher-Röhre” (Photomultiplier)



- Zahl der an einer Dynode ausgelösten Elektronen folgt einer Poisson-Verteilung
- Signal: Zahl der Elektronen an der Anode
- Hoher Rechenaufwand bei großer Anzahl von Teilchen → Parametrisierung

Beispiel: Teilchenphysikexperiment



Beispiel: Faltungsintegral

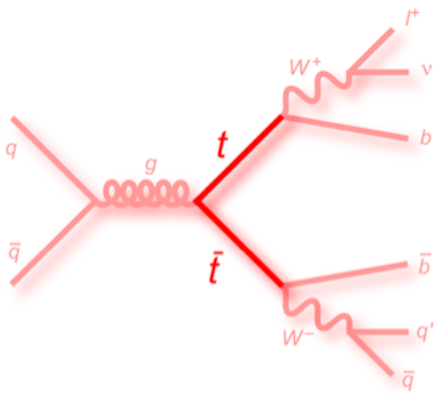
- Detektorauflösung wird i.a. durch ein Faltungsintegral beschrieben: Statt des wahren Werts x wird ein durch Auflösungseffekte t(x,x') verschmierter Wert x' gemessen.

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

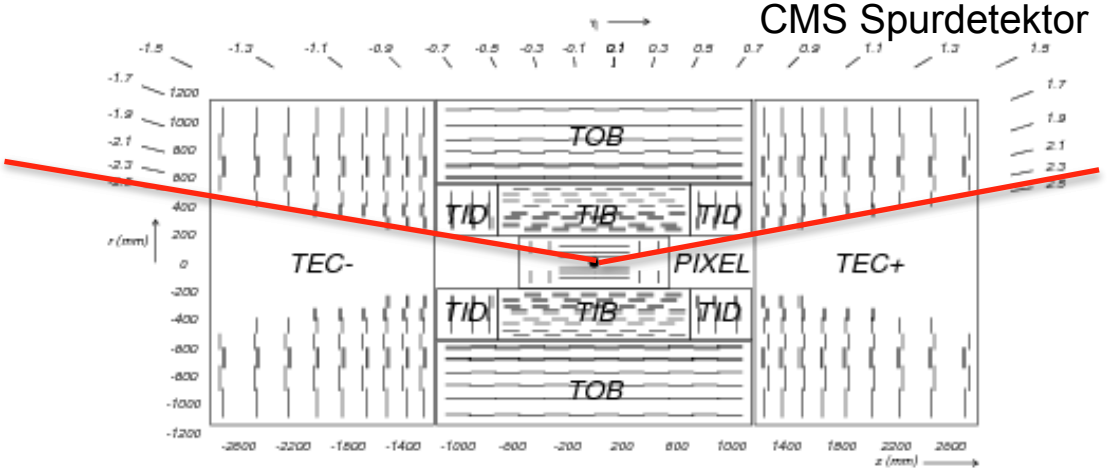
- In der Praxis werden Messwerte durch eine große Zahl von Einzel-Effekten verfälscht (Auflösung, Rauschen, Digitalisierung, systematische Effekte usw.) d.h. die zu bestimmenden Faltungsintegrale sind hochdimensional.

- Bei der MC-Simulation werden die Effekte der Reihe nach aufaddiert. Serie von Faltungsintegralen wird zu einer Summe von Zufallsvariablen

Beispiel: Detektorakzeptanz

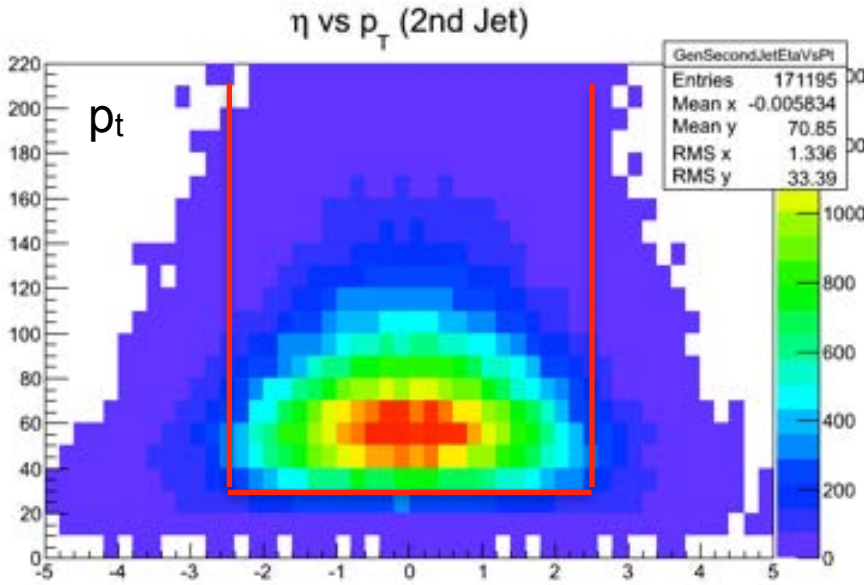


CMS Spurdetektor



- Vollständige Simulation von
  - Physik-Prozess
  - Detektorgeometrie, Ansprechverhalten und Auflösung

  ermöglicht Messung, Korrektur für Verluste und Abschätzung der systematischen Fehler



$\eta = -\ln \tan(\theta/2)$

Karlsruher Institut für Technologie

2.2

ZUFALLSZAHLEN

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[

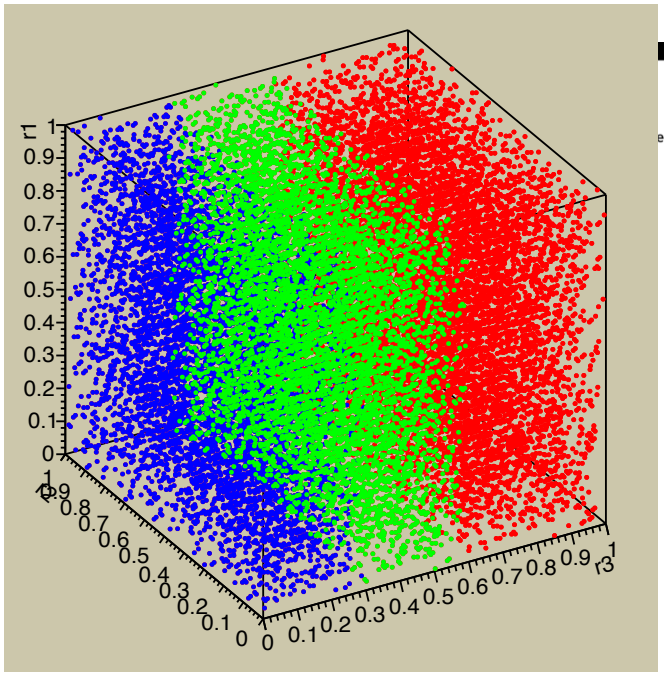
- Periode ist höchstens m. Die Parameter a und c müssen geeignet gewählt werden.

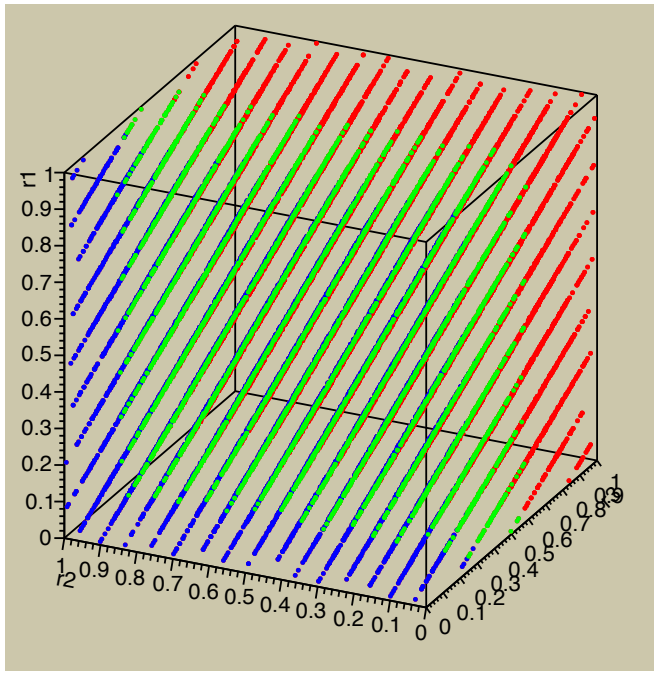
Blobel/Lohrmann Abschnitt 5.2

Zufallszahlen mit LCG

- Sind diese Zufallszahlen gut?
- Gibt es Korrelationen zwischen aufeinanderfolgenden Zahlen?
- Betrachte *k*-tuples (d.h. Paare, Tripletts, …) von *n* benachbarten Zahlen
- Allgemein gilt für LCG: *k*-Tuple aufeinander folgender Zahlen liegen auf *k*-1 dimensionalen Hyperebenen.

  G.Marsaglia, PNAS 1968 61(1) 25-28
- Gute Wahl von *a*,*c* und *m* liefert gleichmäßige Verteilung der Ebenen





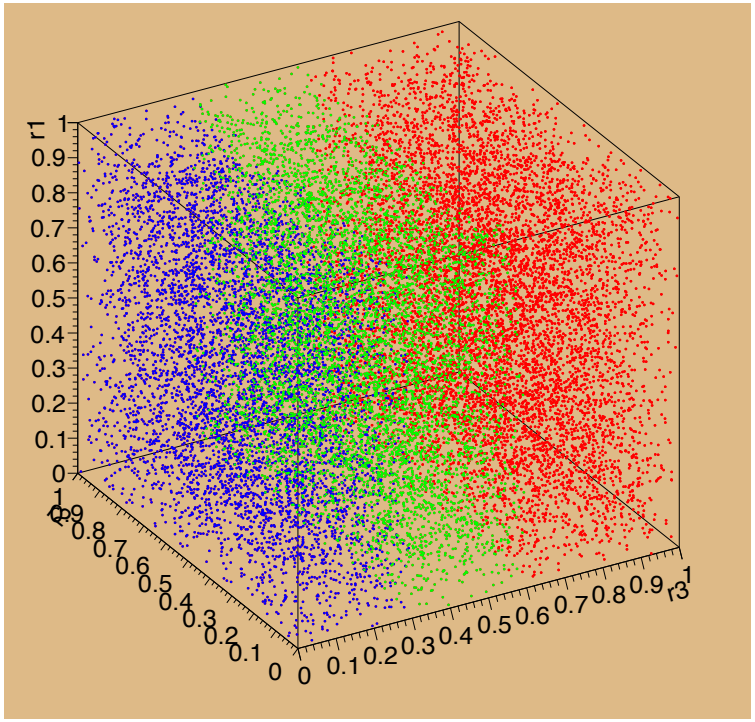
siehe z.B. Brandt, Datenanalyse

Beispiel: myrand.C

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

Tests für Zufallszahlengeneratoren

- Trivialer Test: Histogram der Zufallszahlen und Anpassung. Summe der Abstandsquadrate χ^2 sollte nahe 1/d.o.f sein

- Moderne, avancierte Tests:
  - G.Marsaglia: “Die-Hard Battery of Tests of Randomness” (1995)
    15 verschiedene Tests (einschl. eines Satzes von guten Zufallszahlen)
  - P.L.Ecuyer und R.Simard: TestU01 (2007)
    Small Crush (10 Tests), Crush (96 Tests), Big Crush (106 Tests)
    http://dl.acm.org/citation.cfm?doid=1268776.1268777

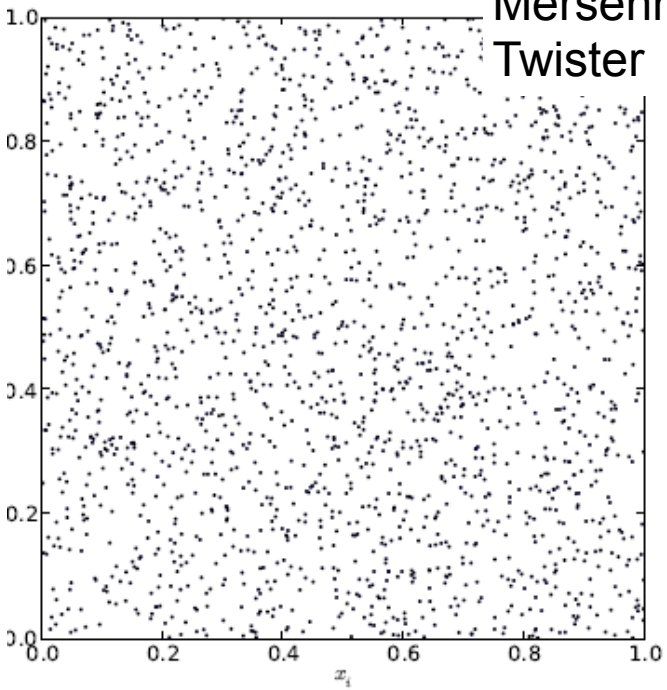
- Es gibt Generatoren, die TestU01 überstehen.
- Ergebnisse für Mersenne-Twister:
  - erfüllt nicht alle Tests der “Big Crush” Suite
  - gilt inzwischen als kryptographisch unsicher, da vorhersagbar

Diese Einschränkungen von MT sind aber für unsere Zwecke nicht relevant

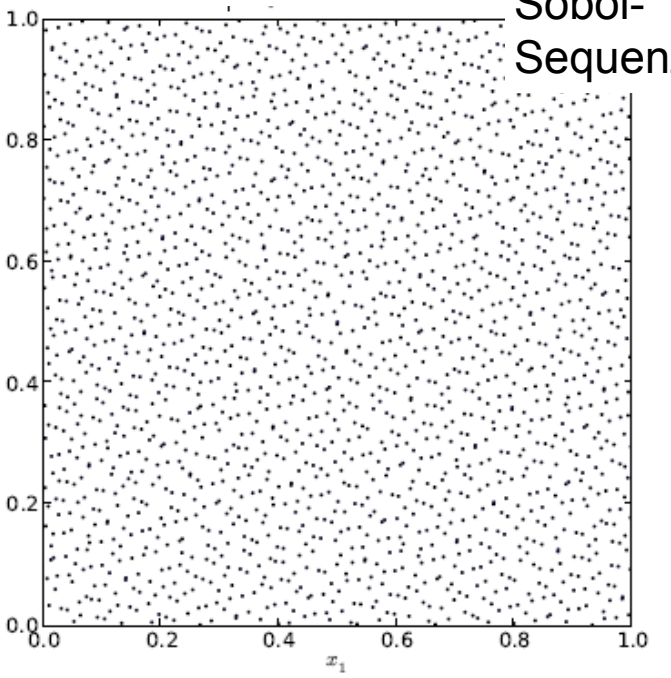
Quasi-Zufallszahlen

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

Mersenne-Twister



Sobol-Sequenz



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

S.Press et al

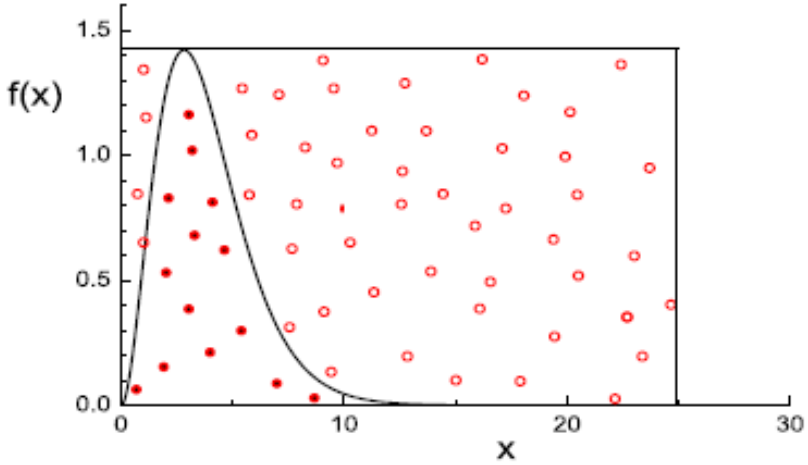
2.3

ERZEUGUNG VON VERTEILUNGEN

Beliebig verteilte Zufallszahlen

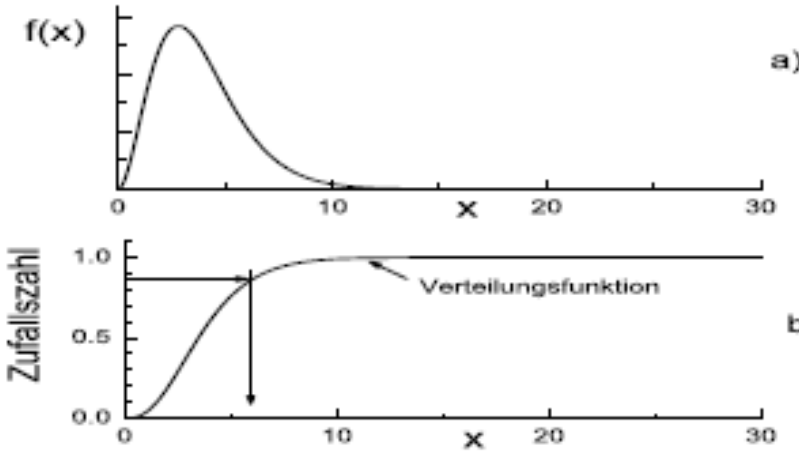
- Von Neumann'sches Rückweisungs-verfahren ("Wegwerfmethode")

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



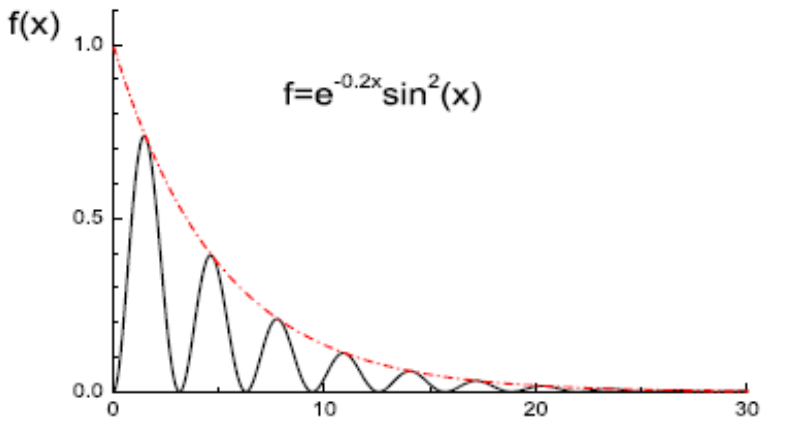
- Transformationsmethode

  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 $F(x)$



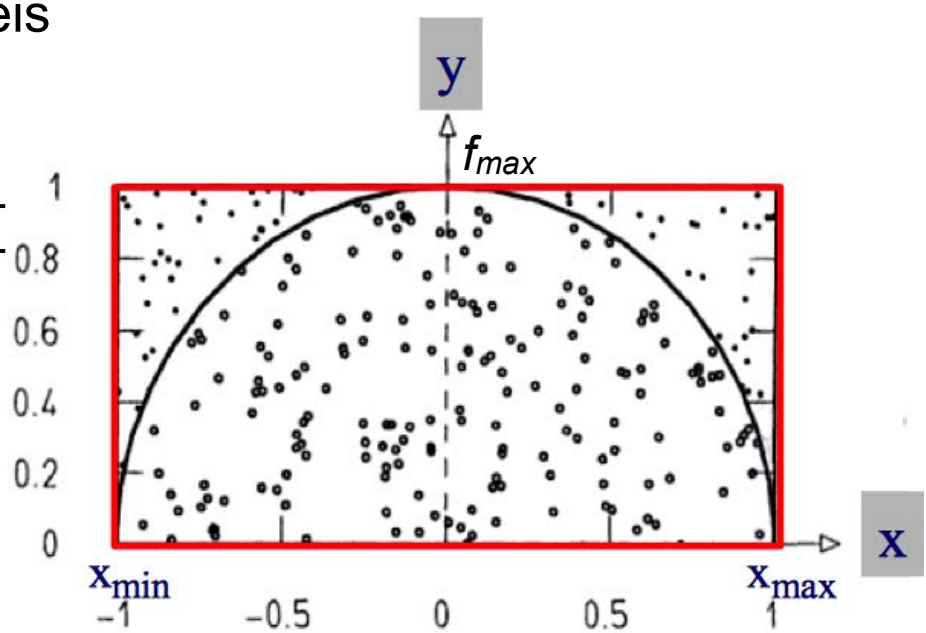
- Majoranten-Verfahren (Importance Sampling)

  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 $r_2 \cdot m(x) < f(x)$



Von Neumannsches Rückweisungsverfahren

- Beispiel: Verteilungsfunktion $f(x)$ Kreis
- Erzeuge gleichmäßig in der Box verteilte Zufallszahlen r_{2i}, $r_{2i+1} \in]0,1[$
  - $x_i = x_{min} + r_{2i} \, (\, x_{max} - x_{min} \,)$
  - $y_i = r_{2i+1} \, f_{max}$
- Akzeptiere x-Werte aus Paaren mit $y_i < f(x_i) = \sqrt{(R^2 - x_i^2)}$

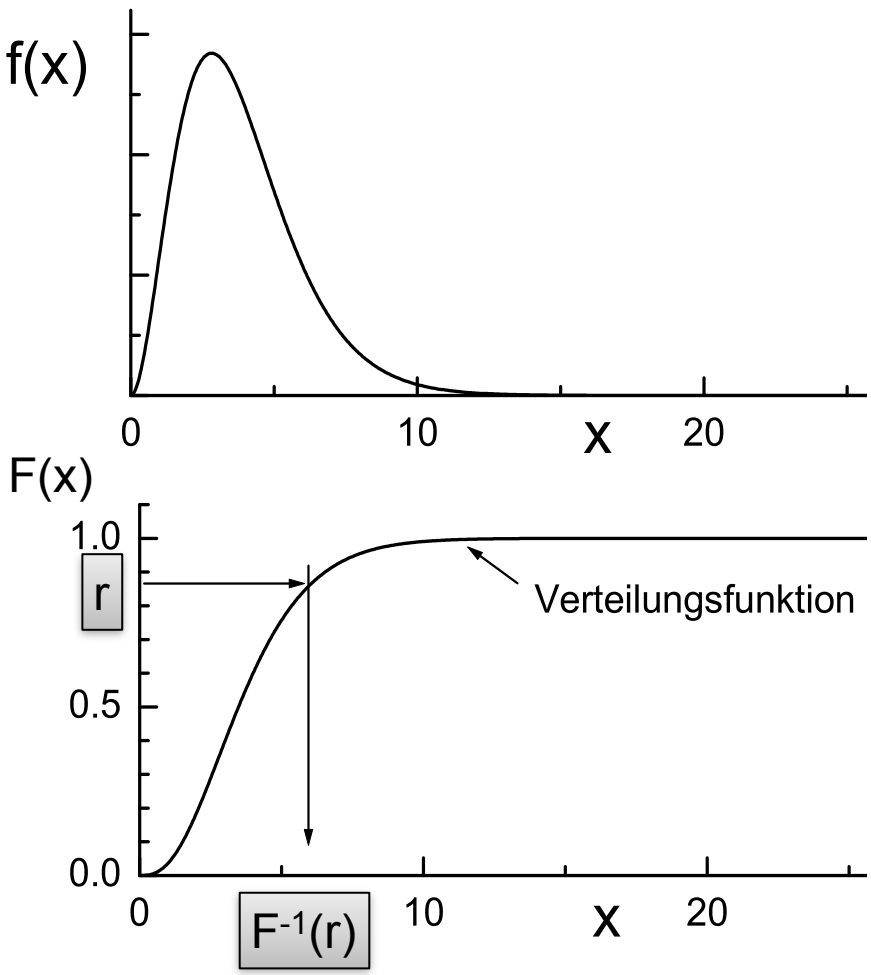


- Die Häufigkeitsverteilung der akzeptierten x-Werte folgt der gewünschten Verteilung $f(x)$
- Für dieses Beispiel funktioniert das Verfahren gut. Wenn die Verteilung steilt abfällt, und Werte von $x \rightarrow \infty$ benötigt werden, kann die Zahl der verworfenen Zufallszahlen aber unakzeptabel groß werden

Transformationsmethode

- Erzeuge gleichverteilte Zufallszahlen $r_i \in]0,1[$
- Variablentransformation: $r \rightarrow x$

$$
\begin{aligned}
f(x)dx &= R(0,1)dr \\
\int_{-\infty}^{x} f(x')dx' &= \int_{0}^{r(x)} dr' = r(x) \\
F(x) &= r \\
x &= F^{-1}(r)
\end{aligned}
$$



- Die Zufallszahlen $x_i = F^{-1}(r_i)$ folgen der PDF $f(x)$
- Anschaulich: Dichte $f(x)$ entspricht Steigung (Ableitung) $F'(x) = dF(x)/dx$

- Bedingung: $f(x)$ muss integrierbar und $F(x)$ invertierbar sein

Bohm/Zech Abschnitt 4.2
Blobel/Lohrmann Abschnitt 5.3

Verteilungen für die Transformationsmethode

Dreieck-Verteilung

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

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

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

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

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

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

Anmerkung:
r kann auch durch 1-r ersetzt werden

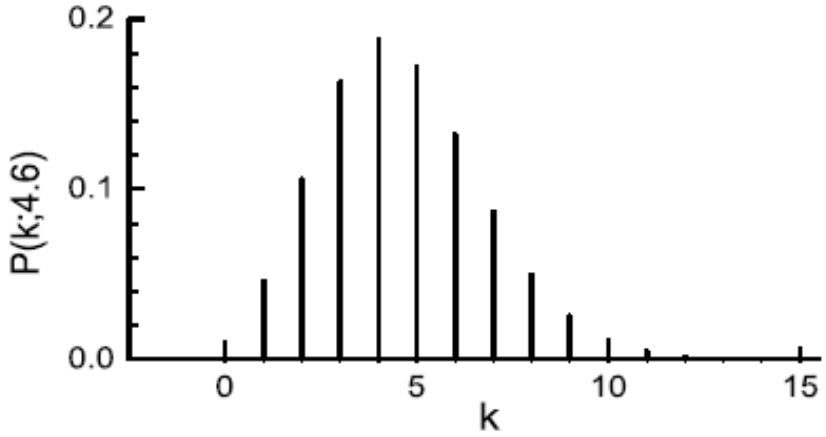
z.B. Bohm/Zech Abschnitt 4.2

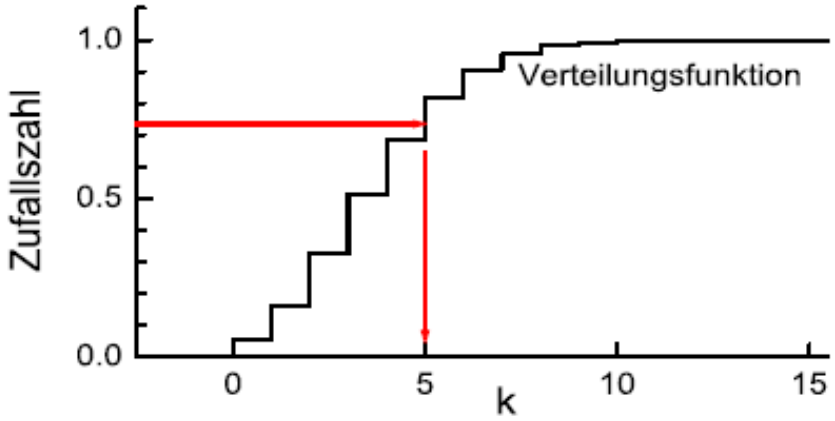
Erzeugung diskreter Verteilungen

- Transformationsmethode funktioniert auch für diskrete Verteilungen (Beispiel: Poisson-Verteilung)

- Durch Summation aller Bins k der diskreten Verteilung erhält man die Verteilungsfunktion $S(k)$

- Eine Zufallszahlen r_i wird dem Bin k zugeordnet, das dem kleinsten $S(k)$ mit $r_i > S(k)$ entspricht



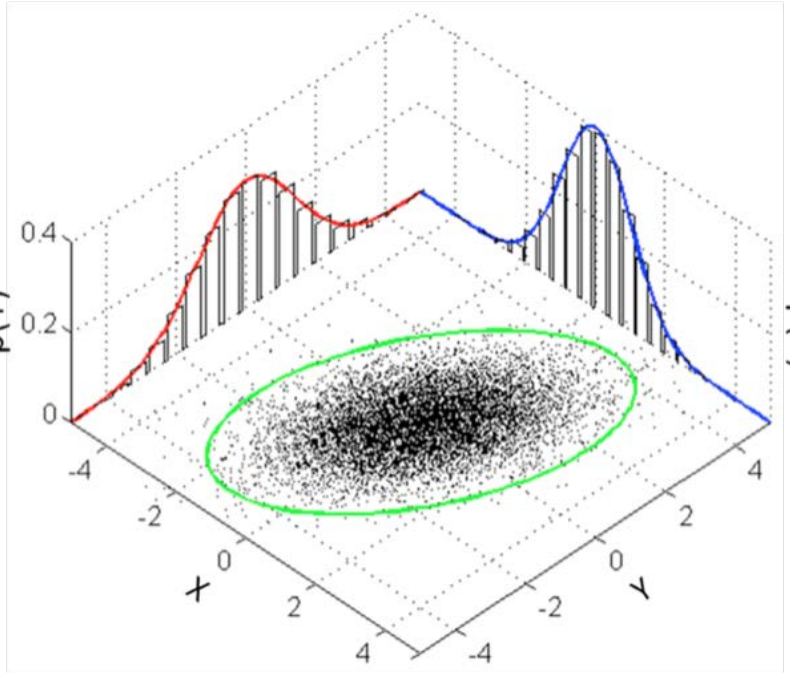


Blobel/Lohrmann Abschnitt 5.3

Verteilungen aus Histogrammen

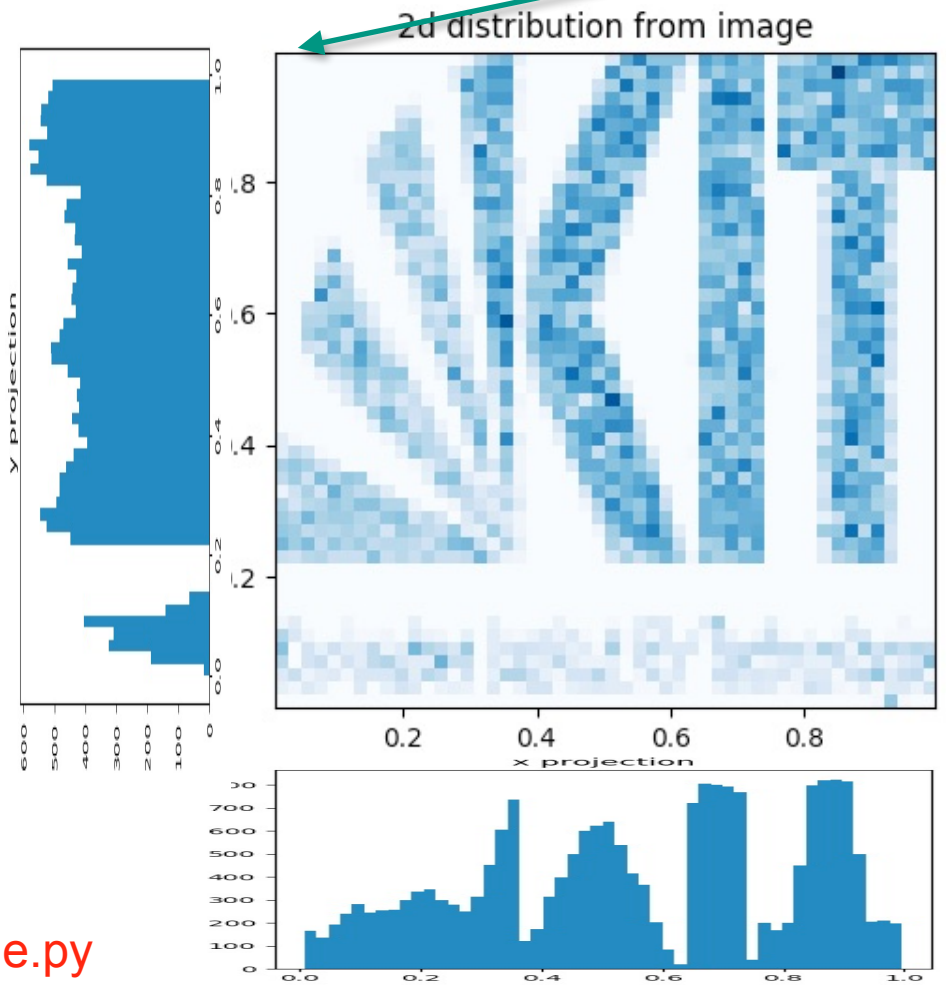
- Allgemein kann die Transformationsmethode zur Erzeugung von Zufallszahlenverteilungen aus vorgegebenen Histogrammen verwendet werden
  - r_i bestimmt das Bin k
  - Der Rest r_i - $S(k\text{-}1)$ kann zur Interpolation innerhalb des Bins verwendet werden

- 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

Beispiel: 2D Verteilung aus Bild



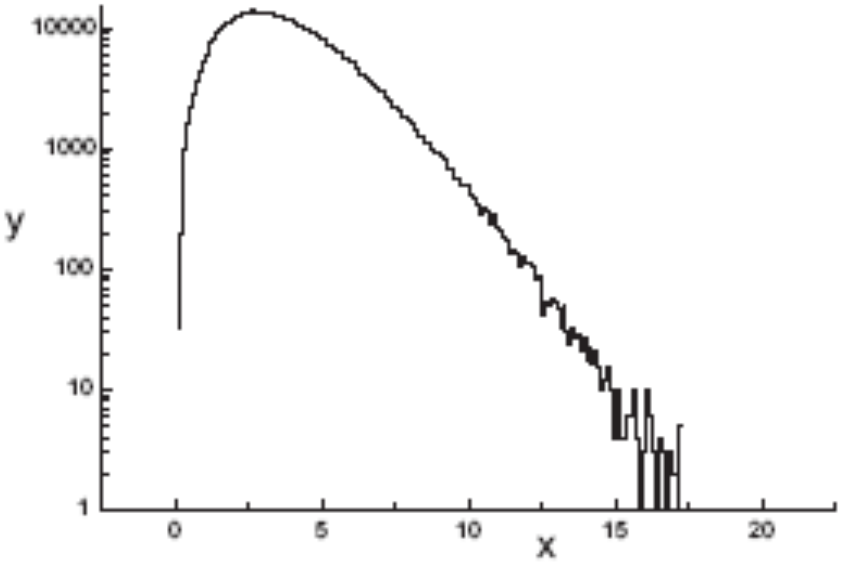
Beispiel: Distribution-fromImage.py

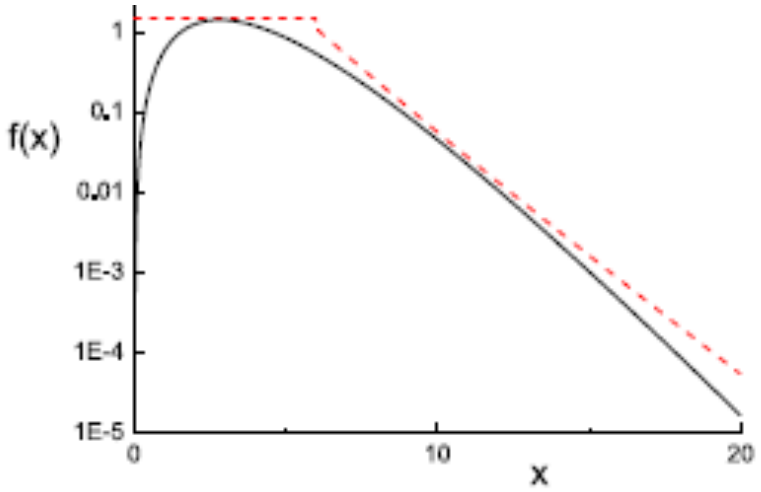
- Helligkeit des Pixels wird als Wahrscheinlichkeitsdichte verwendet

Varianzreduzierende Verfahren

- Ziel: Verwende generierte Ereignisse möglichst effizient, z.B. bei steil abfallenden Verteilungen (d.h. strebe optimales Verhalten $\sigma \propto 1/\sqrt{N}$ an)

- Majorantenverfahren (Importance-Sampling):
  Wenn eine Funktion $m(x)$ mit $m(x)>f(x)$ existiert und $x=M^{-1}(r)$ bestimmbar ist:
  - Generiere $x_i = M^{-1}(r_{2i})$ und eine weitere Zufallszahl $r_{2i+1} \in [0,1]$
  - Akzeptiere x_i, wenn $r_{2i+1} \cdot m(x) < f(x)$
  - Die akzeptierten Werte von x sind gemäß $f(x)$ verteilt





Bohm, Zech Abb, 4.7

Andere übliche Verfahren zur Varianz-Reduktion

- Subtraktion eines analytisch lösbaren Funktionsanteils *f*(*x*)

$$\int_a^b g(x)dx = \int_a^b f(x)dx + \int_a^b |g(x) - f(x)|dx$$

- Stratified Sampling (geschichtete Stichprobe)
  - Hohe (niedrige) Ereignisrate in Bereichen von hohem (niedrigem) Interesse (z.B. in der Nähe von Singularitäten und in Schwänzen)

- Integration für eine leicht modifizierte Funktion $g_m(x_i)$
  - Verwende Gewichte $w_i = g_m(x_i)/g(x_i)$ zur Berechnung

$$I' = \int g_m(x)\,\mathrm{d}x \approx \frac{b-a}{n}\sum g_m(x_i) = \frac{b-a}{n}\sum g(x_i)\,w_i$$

  - Wird in der Teilchenphysik extrem häufig verwendet, z.B. Abschätzung der Auswirkung systematischer Unsicherheiten auf das Endergebnis
  - Funktioniert auch mit negativen Gewichten

Bemerkung: Gewichtung von Ereignissen

- Betrachte Bin eines Histograms mit gewichteter Ereigniszahl N_w

$$N_w = \sum_i w_i k_i \quad \text{und} \quad V[N_w] = \sum_i \left(\frac{\partial N_w}{\partial k_i}\right)^2 V[k_i] = \sum_i w_i^2 V[k_i]$$

- Ein einzelnes Ereignis k_i ist Poisson-verteilt mit $E[k_i] = V[k_i] = 1$

$$V[N_w] = \sum_i w_i^2$$

- Deshalb müssen bei gewichteten Einträgen auch die Quadrate der Gewichte mitgenommen werden (siehe auch Vorl.1, Folie 43).
- Stark unterschiedliche Gewichte führen zu großer Varianz:

$$\frac{\delta\left(\sum w_i\right)}{\sum w_i} = \frac{\sqrt{\sum w_i^2}}{\sum w_i}$$

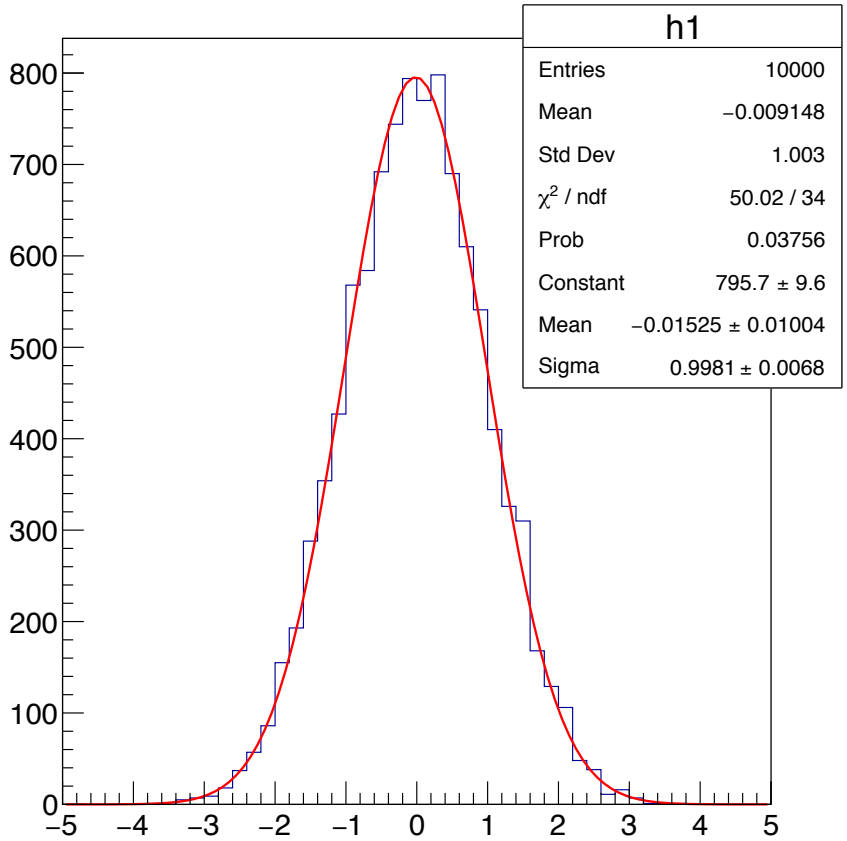
Gauß-verteilte Zufallszahlen

- Mit zentralem Grenzwertsatz:
  Für große *n* geht

$$w = \sum_{i=1}^{n} x_i$$ über in eine

Gauß-Verteilung mit

$$\langle w \rangle = n\langle x \rangle \text{ und } V[w] = n\sigma^2$$



- Gute Gauß-Verteilung schon für relativ kleine *n*, z.B. *n* = 12

$$z_i = \sum_{j=1}^{12} u_j - 6$$

Blobel/Lohrmann Abschnitt 5.5.2

Gauß-Verteilung mit Transformationsmethode

- Problem: Benötigen kumulierte Verteilungsfunktion der Gauß-Verteilung $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$ - in 1D nur numerisch berechenbar.

- In zwei Dimensionen ist Gauß analytisch integrierbar (Polarkoordinaten)

$$\int\int \frac{1}{2\pi} \exp\left(-\frac{x^2+y^2}{2}\right) dxdy \quad \rightarrow \quad \int_0^{2\pi} \frac{1}{2\pi} d\phi \int_0^r \exp\left(-\frac{r^2}{2}\right) r dr$$

- Gleichverteilung in φ, Radialverteilung integrierbar mit $r\,dr = 1/2\,dr^2$

$$\int_0^r \exp\left(-\frac{r^2}{2}\right) \frac{1}{2} dr^2 = 1 - \exp\left(-\frac{r^2}{2}\right)$$

- Lässt sich nach r auflösen, d.h. Transformationsmethode ist anwendbar
- Erzeuge zwei gleichverteilte Zufallszahlen $u_1, u_2 \in]0,1]$. Dann sind z_1, z_2 standard-normalverteilt:

$$z_1 = \sqrt{-2 \ln u_1} \cos(2\pi u_2)$$
$$z_2 = \sqrt{-2 \ln u_1} \sin(2\pi u_2)$$

Effizienteres Verfahren

- Trigonometrische Funktionen sind numerisch aufwändig.
- Kombination aus Transformations- und Rückweisungsverfahren:

1. Erzeuge zwei gleichverteilte Zufallszahlen $u_1, u_2 \in [0,1[$ und setze $v_1 := 2u_1 - 1$ und $v_2 := 2u_2 - 1$
2. Berechne $v_2 = v_1^2 + v_2^2$
3. Falls $v^2 \geq 1 \rightarrow$ gehe zu 1.)
   Akzeptierte Werte von v_1, v_2 sind gleichverteilt innerhalb des Einheitskreises
4. Berechne:

$$z_1 = v_1 \sqrt{-2\ln(v^2)/v^2}$$

$$z_2 = v_2 \sqrt{-2\ln(v^2)/v^2}$$

- z_1 und z_2 sind dann unabhängig voneinander standard-normalverteilt.
- Gauß-Verteilung mit Erwartungswert μ und Standardabweichung σ erhält man durch die Transformation $x = \sigma z + \mu$

Blobel/Lohrmann Abschnitt 5.5

Korrelierte Gauß-verteilte Zufallszahlen

- Gauß-Verteilung in *n* Dimensionen:

$$P(\vec{x},\vec{\mu},V) = \frac{1}{(2\pi)^{n/2}\sqrt{|V|}} \exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu})^T V^{-1}(\vec{x}-\vec{\mu})\right)$$

  - $\vec{\mu}$: *n* Erwartungswerte,
  - *V*: Kovarianzmatrix mit $(n^2+n)/2$ unabhängigen Parametern.
  - $|V|$ = det *V* ist die Determinante von *V*

Cholesky-Zerlegung

- Schreibe V^{-1} als Produkt von Dreiecksmatrizen $V^{-1} = D^T D$

- Mit $\vec{u} = D(\vec{x}-\vec{\mu})$ folgt: $G(\vec{x},\vec{\mu},V) = \frac{1}{(2\pi)^{n/2}\sqrt{|V|}} \exp\left(-\frac{1}{2}\vec{u}^T\vec{u}\right)$

- Vorgehensweise also:
  - Würfle *n* unabhängige Gauß-verteilte Zufallszahlen *u*
  - Transformiere Vektor *u*: $\vec{x} = D^{-1}\vec{u} + \mu$

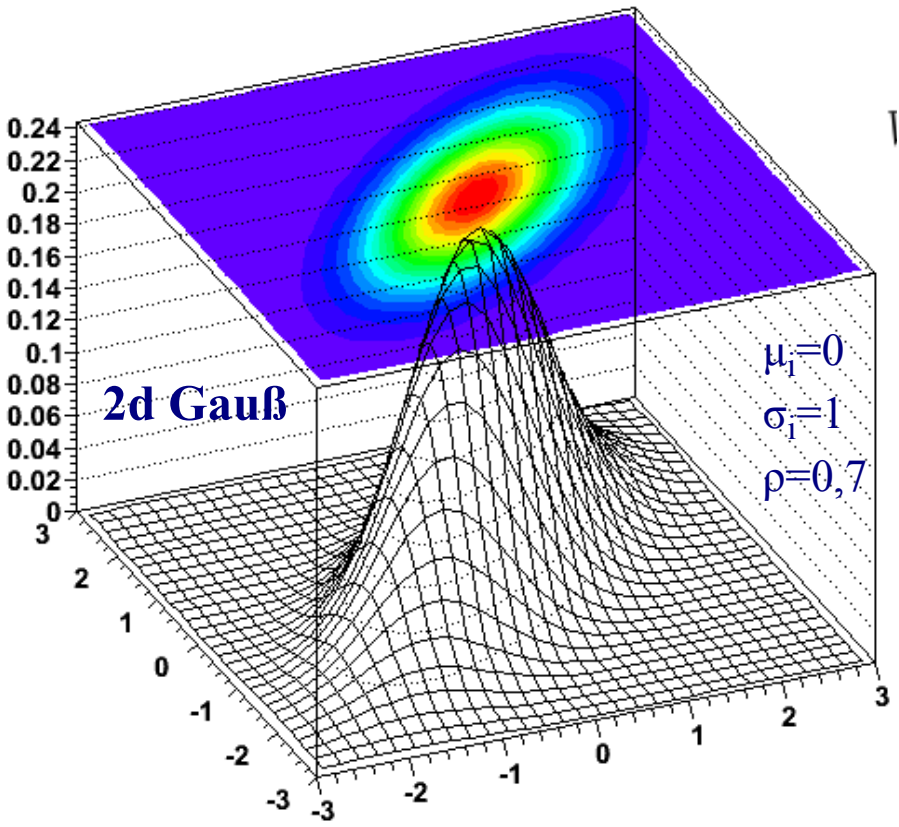
Blobel/Lohrmann Abschnitt 5.4

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

Zwischenrechnung

- Berechne aus V:

$$V^{-1}V = 1$$

$$V^{-1} = D^T D \quad \text{mit } D_{21}=0$$

$$D^{-1}D = 1$$

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

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

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

$$V^{-1} = \frac{1}{\sigma_1^2\sigma_2^2(1-\rho^2)} \times \begin{pmatrix} \sigma_2^2 & -\rho\,\sigma_1\sigma_2 \\ -\rho\sigma_1\sigma_2 & \sigma_1^2 \end{pmatrix}$$

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

Blobel/Lohrmann Abschnitte 5.4 und 3.6

Zusammenfassung

- Monte Carlo Methode (auch “MC-Simulation”)
  - Integration in hochdimensionalen Räumen
  - Nachbildung von komplexen Prozessen (z.B. experimentelle Apparaturen)
  - Bestimmung der Eigenschaften von Verteilungen von Zufallszahlen

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

- Einfache Operationen mit den x_i ersetzen sehr viel komplexere Operationen auf den Verteilungsdichten.

Zufallszahlen in Python

- Paket numpy.random:
  - binomial(n, p[, size])
  - chisquare(df[, size])
  - exponential([scale, size])
  - lognormal([mean,
  - sigma, size])
  - multinomial(n, pvals[, size])
  - multivariate_normal(mean, cov[, size])
  - normal([loc, scale, size])
  - poisson([lam, size])
  - power(a[, size])
  - standard_cauchy([size])
  - standard_exponential([size])
  - standard_normal([size])
  - standard_t(df[, size])
  - triangular(left, mode, right[, size])
  - uniform([low, high, size])

https://docs.scipy.org/doc/numpy/reference/routines.random.html

Zufallszahlen in Root

- TRandom: LCG
  - schnell, kurze Periode: 10^9
  - niedrigste Bits nicht unkorreliert, nicht verwenden!
- TRandom1: RANLUX (Lüscher, James '94)
  - langsam, lange Periode: 10^{171}
  - übersteht TestU01 Suite (auf höchstem Level)
  - http://arxiv.org/abs/hep-lat/9309020

- TRandom2: Tausworthe (P.L'Ecuyer '96)
  - schnell, Periodenlänge ok 10^{26}

- TRandom3: Mersenne-Twister ('98)
  - hinreichend schnell, Periodenlänge 10^{6000}
  - Default: gRandom points to TRandom3

- Methoden: Exp(tau), Integer(imax), Gaus(mean,sigma), Rndm(), Uniform(x1), Landau(mpv, sigma), Poisson(mean), Binomial(ntot, prob) u.v.m.

Zufallszahlen in Root

- 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 innerhalb 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);
```