

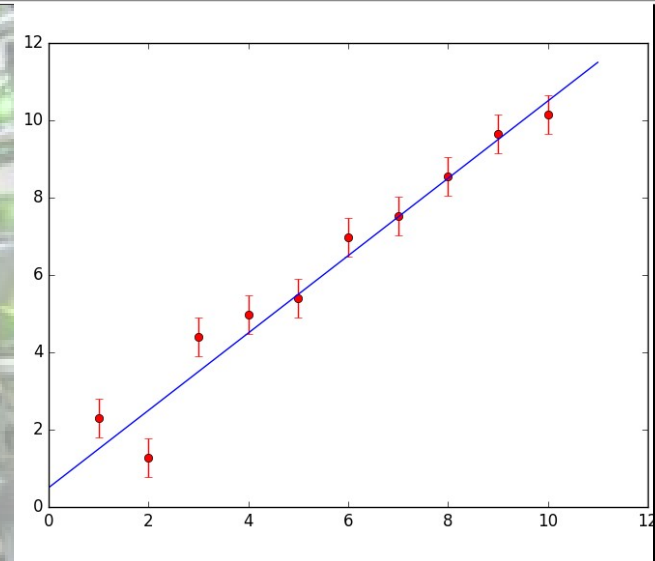
Rechnernutzung in der Physik

Vorlesung 03b: Monte Carlo-Methode: Beispiele und Anwendungen

Günter Quast

Fakultät für Physik
Institut für Experimentelle Teilchenphysik

WS 2023/24



Beispiele

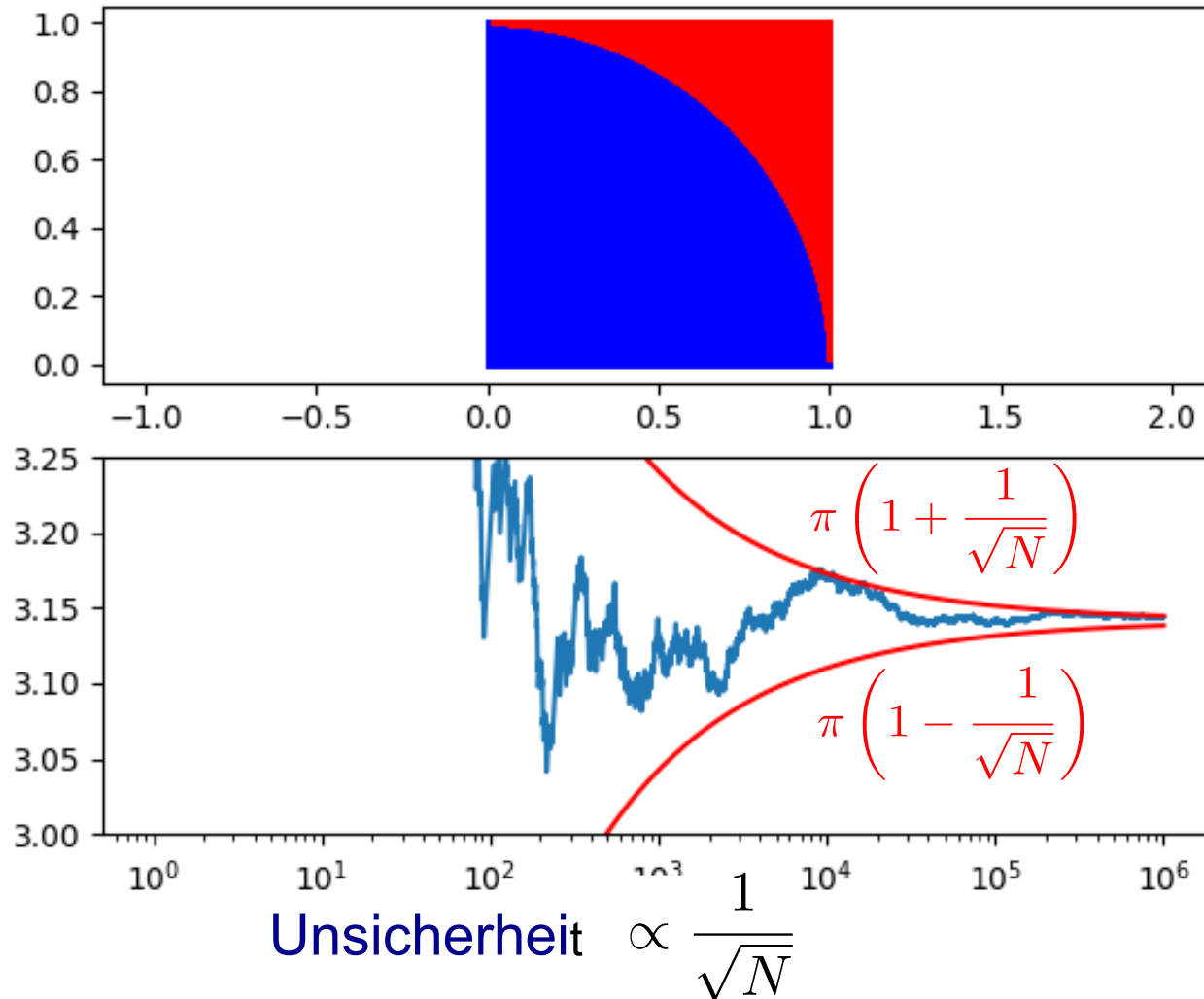
Einfache MC-Integration

Kreisfläche mittels einfachem Rückweisungsverfahren:

Scripte:

[animate_pi.py](#)

[calc_pi.py](#)



Anwendungsbeispiel: 2d-Verteilung aus Bild

Auch aus Histogrammen lassen sich Stichproben erzeugen, die gemäß der vorgegebenen Häufigkeit verteilt sind.

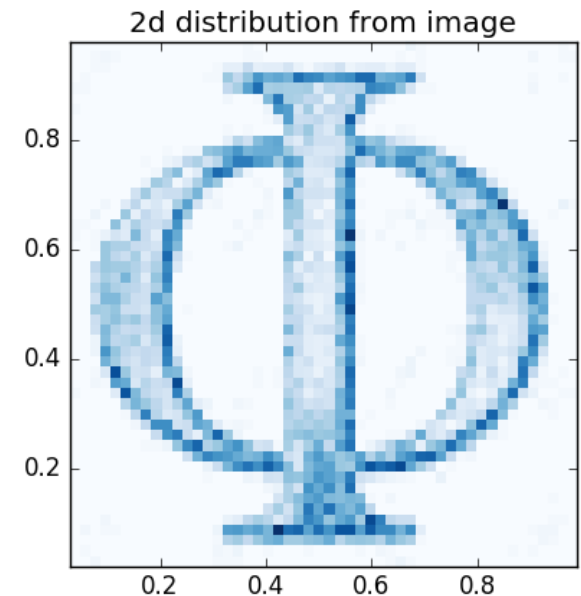
Hier ein Beispiel, bei dem die Wahrscheinlichkeit der Helligkeit eines Pixels in einem Bild entspricht:

```
...
#open an image
img=Image.open('image.jpg')
# img.show()
print "image size:", img.size[0], img.size[1]
pix=img.load()

def rand_fromImage():
    # return pixel coordinate with a probability
    #   proportional to brightness of the pixel
    r0=np.random.uniform()
    r1=np.random.uniform()
    i=int(img.size[0] * r0 - 0.5)
    j=img.size[1] -1 - int(img.size[1] * r1 - 0.5)
    r,g,b = pix[i,j]          # brightness of pixel
    if(3*255*np.random.uniform() > r+b+g):
        return r0, r1
    ...
```

Script [Distribution-fromImage.py](#)

Rückweisungs-Schritt



Welche **Transformation** $t(u)$

angewendet auf **gleichverteilte Zufallszahlen** u aus $]0, 1]$

liefert **Zufallszahlen** x ,

die der gewünschten **Verteilung** $f(x)$ folgen ?

Lösung:

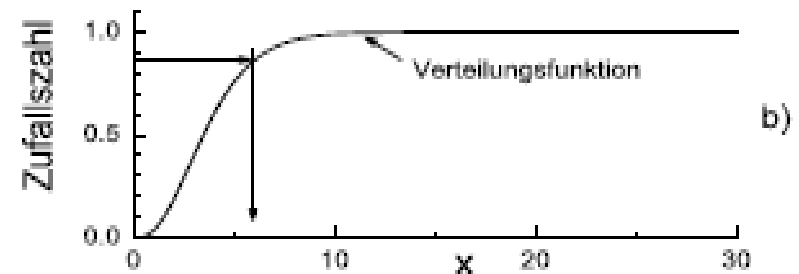
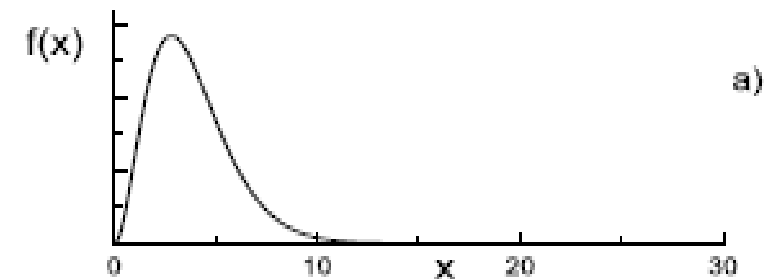
Umkehrfunktion der
Verteilungsfunktion von $f(x)$

$$t() = F^{-1}()$$

mit $F(x) = \int_{-\infty}^x f(x') dx'$

$$x = t(u) = F^{-1}(u)$$

grafisch:



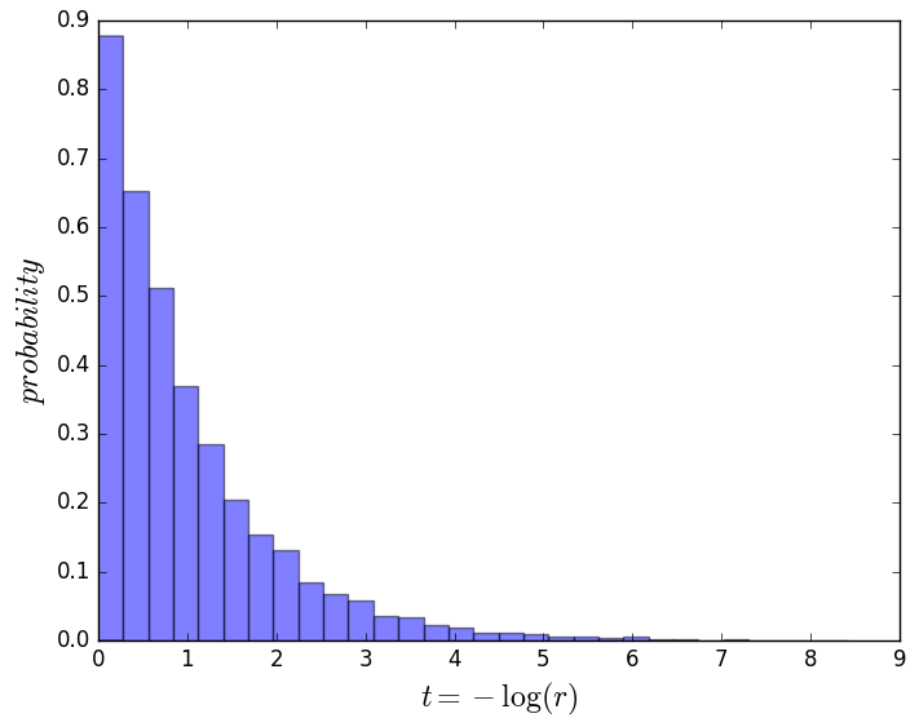
Beispiel Transformationsmethode

Computorexperiment:

Häufigkeitsverteilung von

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

```
...  
r = np.random.rand(10000)  
t = -np.log(1-r)  
  
plt.hist(t, 30, normed=1)  
  
plt.show()
```



Wir erhalten exponentiell verteilte Zufallszahlen **t** !

Beispiel Majorantenmethode

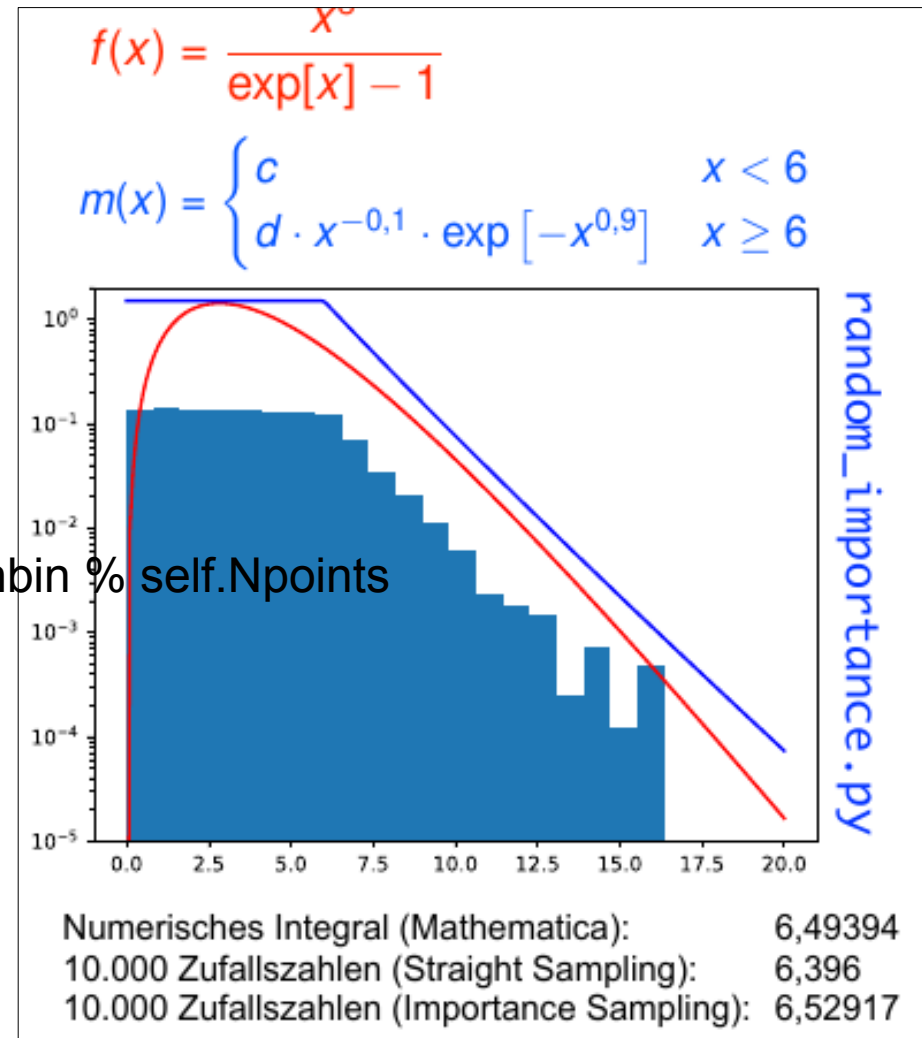
Verbesserte „Wegwerfmethode“:

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 nur ein Bruchteil $(m(x) - f(x)) / f(x)$ der x -Werte verworfen)

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

`k = hbin % self.Npoints`



Allgemeine Bemerkung:

mit etwas Überlegung können numerische Verfahren effizienter gemacht werden (denken Sie z. B. auch an den CO₂-Fußabdruck der Rechnung!)

Beispiel gaußverteilte Zufallszahlen

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

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

Gauß-Funktion in 2 Dimensionen in Polarkoordinaten analytisch integrierbar:

$$\varphi(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \quad (\text{gewünschte Wahrscheinlichkeitsdichte})$$

o.B.d.A. setzen wir $\mu = 0$. Das Integral von φ lässt sich in 2d analytisch lösen:

$$\mathcal{Q}(R) = \int \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2\sigma^2}(x_1^2+x_2^2)} dx_1 dx_2 = \frac{1}{2\pi\sigma^2} \int_0^{2\pi} d\phi \int_0^R e^{-\frac{r^2}{2\sigma^2}} r dr ;$$

$$\text{mit: } r = \sqrt{x_1^2 + x_2^2}; \quad dr^2 = 2r dr$$

$$\mathcal{Q}(R) = \frac{1}{2\pi\sigma^2} \int_0^{2\pi} d\phi \int_0^R e^{-\frac{r^2}{2\sigma^2}} r dr = \frac{1}{\sigma^2} \int_0^R e^{-\frac{r^2}{2\sigma^2}} \frac{dr^2}{2} = \frac{1}{\sigma^2} \left[-\sigma^2 e^{-\frac{r^2}{2\sigma^2}} \right]_0^R = \left(1 - e^{-\frac{R^2}{2\sigma^2}} \right)$$

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

Beispiel gaußverteilte Zufallszahlen (2)

Umkehrfunktion (in 2d) für $u \in U_{[0,1]}$:

$$u(R) = \left(1 - e^{-\frac{R^2}{2\sigma^2}}\right) = \mathcal{Q}(R)$$

$$R(u) = \sqrt{-2\sigma^2 \ln(1-u)} \equiv \sqrt{-2 \ln(u)} \cdot \sigma = \mathcal{Q}^{-1}(u)$$

Wählen Sie $u \in U_{[0,1]}$ und $\phi \in U_{[0,2\pi]}$ gleichverteilt erhalten Sie zwei voneinander unabhängige normalverteilte Zufallszahlen:

$$x_1 = \sqrt{-2 \ln(u)} \cdot \sigma \cos \phi$$

$$x_2 = \sqrt{-2 \ln(u)} \cdot \sigma \sin \phi$$

Verschiebung um μ erfolgt über $x_i \rightarrow x'_i = x_i - \mu$

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 standard-normalverteilt

(Beispiel für die Kombination aus Transformations- und Rückweisungsverfahren)

Trigonometrische Funktionen sind numerisch aufwändig,
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 $v^2 \geq 1 \rightarrow$ gehe zu 1 (Rückweisungsschritt)
(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 dann unabhängig voneinander standard-normalverteilt

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

Erzeugung korrelierter, gaußverteilter Zufallszahlen

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

$\vec{x}_0$: 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“)

Setze dann $\vec{u} = D(\vec{x} - \vec{x}_0)$ $\Rightarrow G(\vec{x}; \vec{x}_0, V) = \frac{1}{(2\pi)^{\frac{n}{2}} |V|^{\frac{1}{2}}} \cdot \exp\left\{-\frac{1}{2}(\vec{u}^T \vec{u})\right\}$

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**: $\vec{x} = D^{-1} \vec{u} + \vec{x}_0$

Cholesky-Zerlegung:

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

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

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

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

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

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

Verfahren zur Varianzreduktion

- 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 Ausläufern)

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

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

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

Markov-Chain Monte Carlo

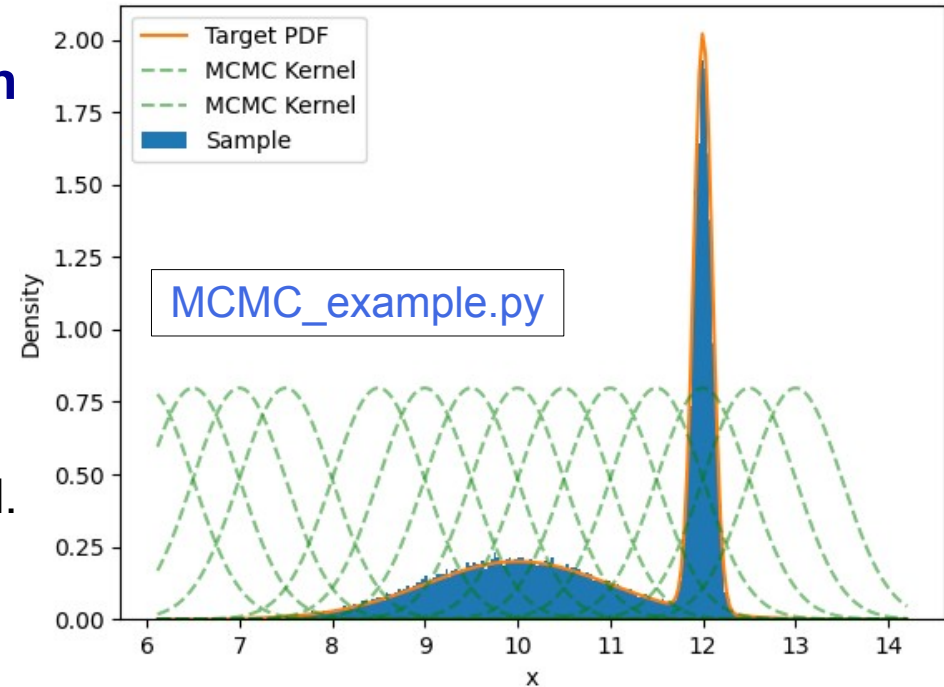
Effizientes Verfahren zur Integration in hochdimensionalen Räumen

Markov-Kette:

Zufallszahl $z_{i+1} = f(z_i)$

Markov-Chain Monte Carlo:

vorgeschlagen 1953 von Metropolis et al.



Markov-Kette als Stichprobe aus einer Wahrscheinlichkeitsdichte-Verteilung;
Wahl des nächsten Stichproben-Punktes erfolgt aus einer lokalisierten, im
Stichprobenraum verschiebbaren Verteilungsdichte („MCMC Kernel“).

Punkte mit höheren Werten der Zielverteilung $f(z_{i+1}) > f(z_i)$ werden immer akzeptiert,
Werte mit $f(z_{i+1}) < f(z_i)$ werden nur mit einer gewissen Wahrscheinlichkeit akzeptiert.

Stichprobe der Kette sind nach einer Burn-in-Phase produzierte Werte

Markov-Chain Monte Carlo: einfacher Python Code

Mit gewisser Wahrscheinlichkeit akzeptierte Zufallsbewegungen („random walk“).

Code für den Metropolis-Hastings Algorithmus

Algorithmus:

1. wähle anfänglichen Wert x_1 für $i=0$
2. berechne $\mathcal{L}(x) = \text{TargetPDF}(x_i)$
3. ziehe neues x aus $\text{KernelPDF}(x_i)$
4. wenn $\mathcal{L}(x) / \mathcal{L}(x_i) < z = \text{rand}[0,1]$
akzeptiere und speichere x als x_{i+1} ,

sonst behalte x als x_{i+1}
5. gehe zu 2., bis Anzahl Werte erreicht

```
def mcmc_updater(curr_state,
                  curr_likeli,
                  target_pdf,
                  proposal_distribution
                  ):
    """ Propose a new state and compare the likelihoods
    global Nupd # count number of state updates

    # Generate a proposal state using the proposal distribution
    # Proposal state == new guess state to be compared to current
    proposal_state = proposal_distribution(curr_state)

    # Calculate the acceptance criterion
    prop_likeli = target_pdf(proposal_state)
    accept_crit = prop_likeli / curr_likeli

    # Generate a random number between 0 and 1
    accept_threshold = np.random.uniform(0, 1)

    # If the acceptance criterion > random number,
    # accept the proposal state as the current state
    if accept_crit > accept_threshold:
        return proposal_state, prop_likeli
    return curr_state, curr_likeli
```

Markov-Chain Monte Carlo: Anmerkungen

- Unabhängigkeit vom Startwert erst nach einer „burn-in“-Phase !
- Vollständige Überdeckung des gesamten Stichprobenraums nicht garantiert
→ mehrere Ketten an verschiedenen Stellen starten
- Mit MCMC produzierte, aufeinanderfolgende Werte sind stark korreliert
ok für Integration, nicht verwendbar für viele andere Anwendungen
- Das Verfahren ist numerisch hocheffizient !
In hochdimensionalen Räumen vielfach angewandt,
insb. auch beim maschinellen Lernen

Simulierte Daten

Simulierte Daten

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

The diagram shows the equation $m = w + z$ inside a yellow box with a blue border. Three blue speech bubbles point to the terms: 'gemessener Wert' points to m , 'wahrer Wert' points to w , and 'zufälliger Beitrag' points to z .

$$m = w + z$$

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

Bei mehreren Messungen m_i können

- die z_i unabhängig oder auch
- untereinander korreliert sein !

Arten von Unsicherheiten

- unkorrelierte (statistische) Unsicherheiten
Ablesegenauigkeit, Rauschen, Quanteneffekte, ...
- absolute korrelierte Unsicherheiten
apparative Offsets, ...
- relative statistische oder korrelierte Unsicherheiten bezogen auf den
 - beobachteten Wert
eher selten (**Achtung: Verzerrung bei Anpassungen!**)
 - oder den wahren Wert
Poissonstatistik, Kalibrationsunsicherheiten (**praktisch alle Messgeräte!**)
- bei der Überprüfung von funktionalen Zusammenhängen $y_i = f(x_i)$
sind im Regelfall sowohl die y_i als auch die x_i mit Unsicherheiten behaftet!

$$\Rightarrow z = z_s \oplus z_{\text{abs}} \oplus z_{\text{rel,b}} \oplus z_{\text{rel,w}}$$

$$z^x = z_s^x \oplus z_{\text{abs}}^x \oplus z_{\text{rel,b}}^x \oplus z_{\text{rel,w}}^x$$

**Insgesamt 8 Arten von
Unsicherheiten zu berücksichtigen !**

Beispiel aus Datenblatt Multimeter

Bereich Multimeter A DC	Genauigkeit	Auflösung
200 mV	$\pm(0,5\% + 8)$	0,1 mV
2000 mV		1 mV
20 V		0,01 V
200 V		0,1 V
250 V	$\pm(0,8\% + 8)$	1 V

Simulation von Messdaten

Zur **Überprüfung der Qualität von Analyseverfahren**

oder zur Erstellung und Test von Analyse-Software parallel zum Experimentaufbau werden **Simulationen der erwarteten Daten** eingesetzt.

Vorgehensweise:

1. Berechnung der „wahren“ Werte
2. Einfaches Fehlermodell („Toy-Monte Carlo“) oder detaillierte Simulation der erwarteten Detektorantworten auf ein Signal
3. wahre Werte mit den auf Grund der Detektorauflösung erwarteten Unsicherheiten verschmieren.

$$m_i = w_i + \sum_{\zeta} z_{i,\zeta} \quad \zeta : \text{Index der Zufallseffekte}$$

- **Unabhängige Effekte:** je eine Zufallszahl für jedes i , ζ
- **korrelierte Effekte:** eine Zufallszahl z_{ζ} für die betroffenen Indizes i

Aus Vorbereitungsmaterial:

das entspricht der Auswertung eines komplexen Faltungsintegrals mit MC

Beispiel: Toy-MC für Physikalische Praktika

Annahme rein Gauß-förmiger Unsicherheiten,
die auf Eingabedaten addiert werden („Smearing“)

Implementiert im Paket `PhyPraKit`

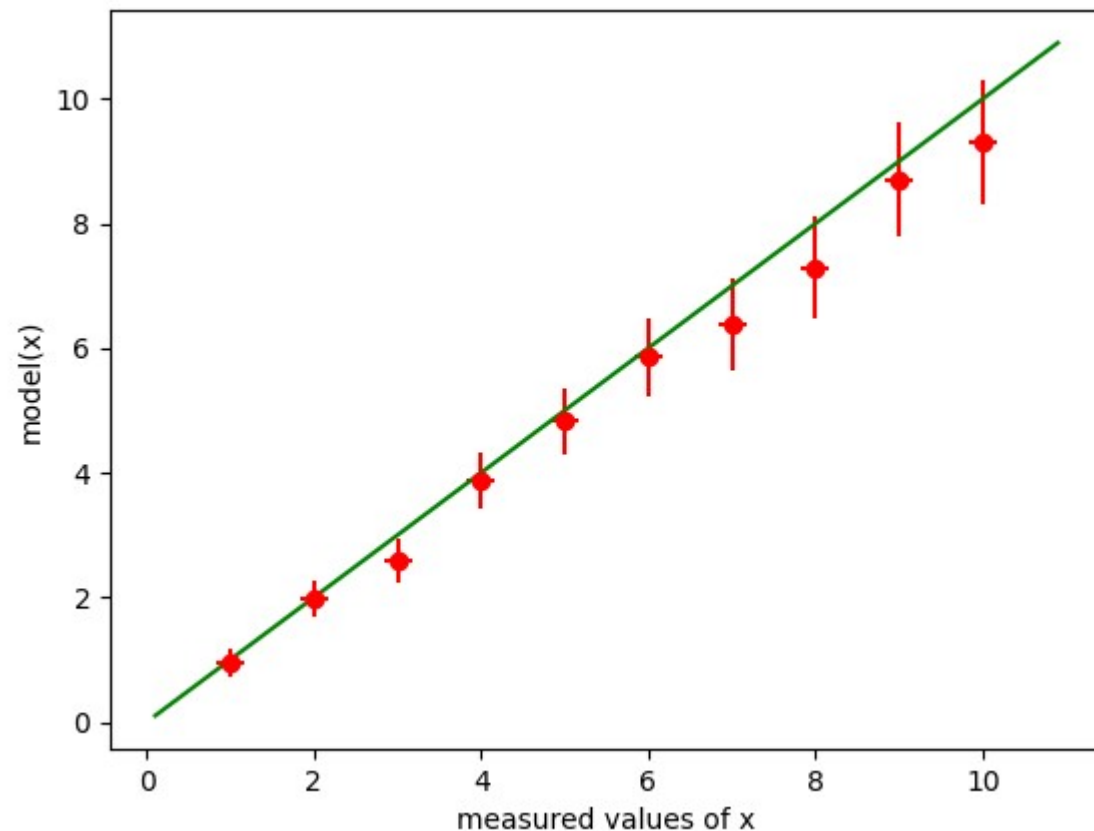
Funktionen

`smearData()` Addieren von zufälligen
Unsicherheiten auf Eingabedaten

und

`generateXYdata()` Erzeugen
simulierter Datenpunkte
($x + \Delta x$, $y + \Delta y$)

PhyPraKit Beispiel: `test_generateData.py`



Implementierung in PhyPraKit

`PhyPraKit.smearedData(d, s, srel=None, abscor=None, relcor=None)`

Generate measurement data from “true” input by adding random deviations according to the uncertainties

Args:

- d: np-array, (true) input data

the following are single floats or arrays of length of array d

- s: Gaussian uncertainty(ies) (absolute)
- srel: Gaussian uncertainties (relative)

the following are common (correlated) systematic uncertainties

- abscor: 1d np-array of floats or list of np-arrays: absolute correlated uncertainties
- relcor: 1d np-array of floats or list of np-arrays: relative correlated uncertainties

Returns:

- np-array of floats: dm, smeared (=measured) data

`PhyPraKit.ustring(v, e, pe=2)`

v +/- e as formatted string with number of significant digits corresponding to precision pe of uncertainty

Args:

- v: value
- e: uncertainty
- pe: precision (=number of significant digits) of uncertainty

Returns:

- string: <v> +/- <e> with appropriate number of digits

Implementierung in PhyPraKit

`PhyPraKit.generateXYdata(xdata, model, sx, sy, mpar=None, srelx=None, srely=None, xabscor=None, yabscor=None, xrelcor=None, yrelcor=None)`

Generate measurement data according to some model assumes xdata is measured within the given uncertainties; the model function is evaluated at the assumed “true” values xtrue, and a sample of simulated measurements is obtained by adding random deviations according to the uncertainties given as arguments.

Args:

- xdata: np-array, x-data (independent data)
- model: function that returns (true) model data (y-dat) for input x
- mpar: list of parameters for model (if any)

the following are single floats or arrays of length of x

- sx: gaussian uncertainty(ies) on x
- sy: gaussian uncertainty(ies) on y
- srelx: relative Gaussian uncertainty(ies) on x
- srely: relative Gaussian uncertainty(ies) on y

the following are common (correlated) systematic uncertainties

- xabscor: absolute, correlated error on x
- yabscor: absolute, correlated error on y
- xrelcor: relative, correlated error on x
- yrelcor: relative, correlated error on y

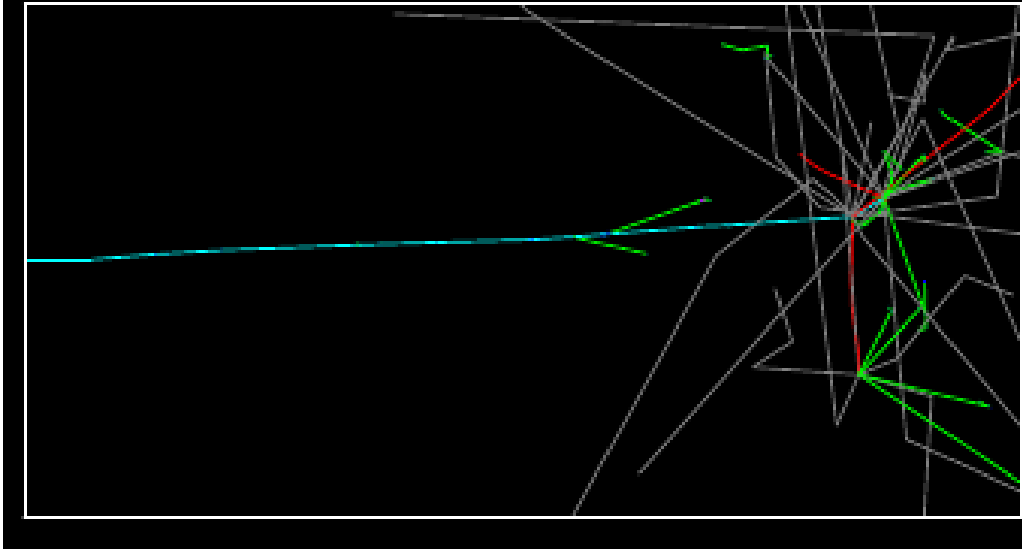
Returns:

- np-arrays of floats:
 - xtrue: true x-values
 - ytrue: true value = model(xtrue)
 - ydata: simulated data

Beispiel: Wechselwirkung von Strahlung mit Materie

BA-Arbeit M. Burkart, ETP (2017)

Wechselwirkung eines Pions in Blei



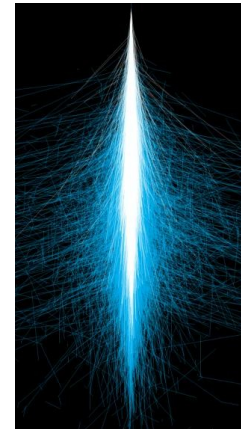
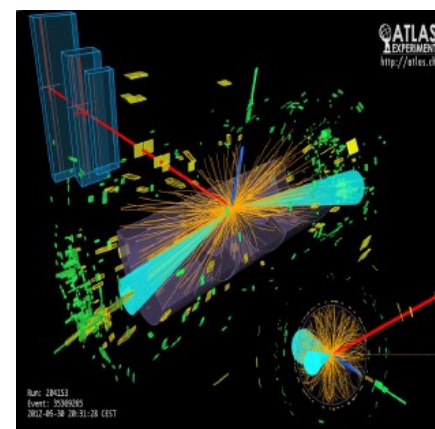
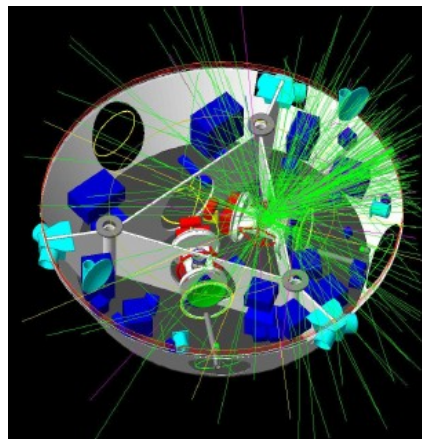
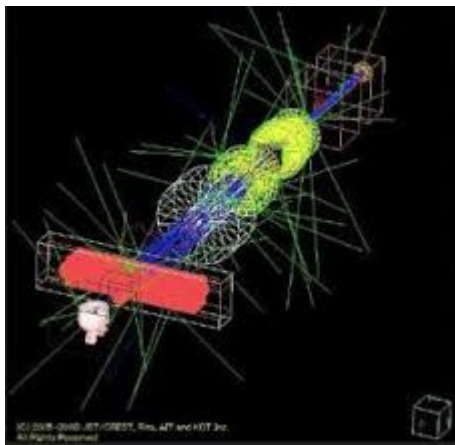
Methode:

- Verfolgen von Teilchen und aller produzierten Sekundärteilchen durch Materie
- Bestimmung der Energie-deposition in allen Detektorelementen
- Umwandlung in elektrisches Signal („mV in detector cell“)

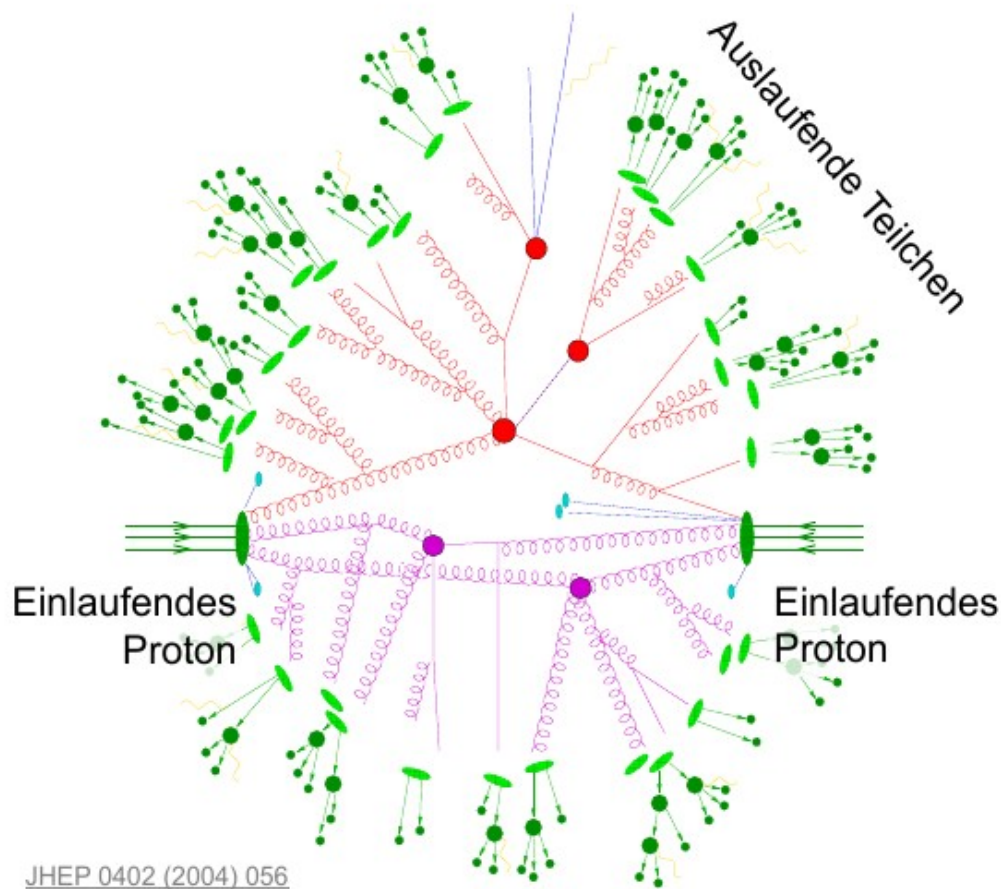
z.B. Programmpaket

GEANT4 - A Simulation Toolkit

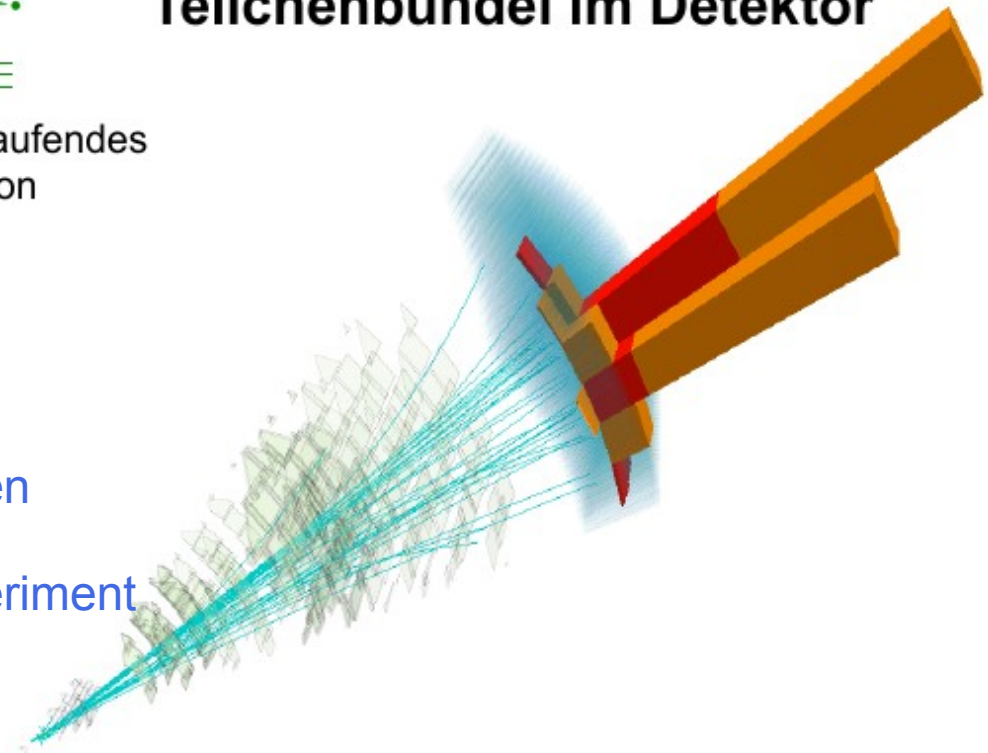
Anwendungen in Medizin, Raumfahrt, Kern-, Teilchen- und Astroteilchenphysik



Physikprozess und Detektorantwort



Teilchenbündel im Detektor



MC-Daten bilden
Schmittstelle
Theorie ↔ Experiment

Bildquelle: CMS

Monte Carlo-Anwendungen

Monte Carlo Methoden werden in praktisch allen Gebieten der Physik und auch in anderen Forschungsfeldern angewandt.

Als Integrationsmethode in numerischen Werkzeugen,
beim Ersatz fehlender Detailinformation durch statistische Modelle,
in allen Gebieten der Quantenphysik,
in Wettervorhersage und Klimamodellierung,
in den Wirtschaftswissenschaften, ...

Formelsammlung

Transformationsmethode

Beispiele Transformationsmethode

Dreieck-Verteilung

$$\begin{aligned}g(t) &= 2t, 0 \leq t \leq 1 \\t(r) &= \sqrt{r}\end{aligned}$$

$$\begin{aligned}g(t) &= (n+1) t^n, 0 \leq t \leq 1, n > -1 \\t(r) &= r^{1/(n+1)}\end{aligned}$$

Exponentialverteilung

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

Log-Weibull -Verteilung

$$\begin{aligned}f(x) &= e^{-x-e^{-x}} \\x &= -\ln(-\ln r)\end{aligned}$$

Breit-Wigner-Verteilung

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

Paar von Gauß-verteilten Zahlen

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

Anm.:

1-r kann durch
r ersetzt werden

Vorbereitung VL04 Parameterschätzung

- Vorbereitendes Material anschauen:
[V04a_ParameterSchätzung](#) (Folien und Kurzvideos)