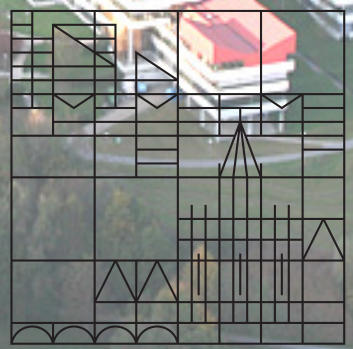
Computerphysik II

Teil 3 - Stochastische Methoden

S. Gerlach

WiSe 2022/23

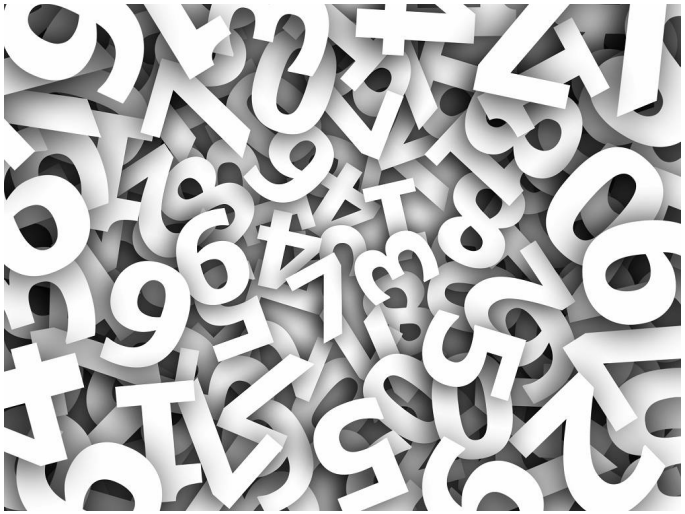
Universität
Konstanz



Teil 3 - Stochastische Methoden

INHALT

1. Zufallszahlen und -generatoren
2. Nicht-gleichverteilte Zufallszahlen
3. Monte-Carlo Integration
4. Random-Walk
5. Zelluläre Automaten
6. Perkolation, etc.



Evaluation - CoPhy II WS 22/23

Bis 30.1.2023:

`https://evasys.uni-konstanz.de/evasys/online.php?pswd=ZATLQ`

Zufallszahlen

Zufallszahlen für Kryptographie:

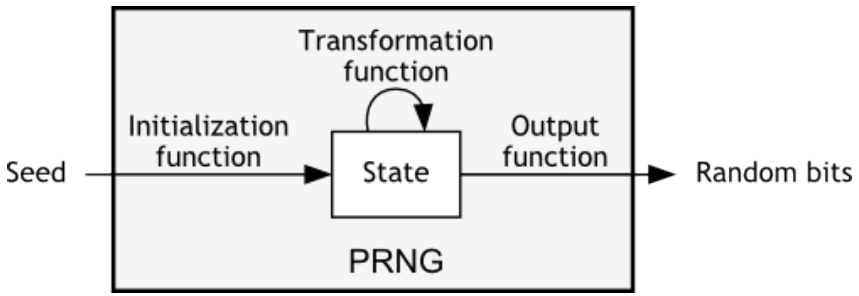
- Wenige Ziffern (typisch: 128 - 4096 bit)
- Anwendung: Verschlüsselung, Authentifizierung, Integrität
- "kryptographische" Qualität
- `/dev/random`, `/dev/urandom`, CPUs (RDRAND)

Zufallszahlen für Computerexperimente:

- Anwendung: Stichprobe, Randomisierung, Simulationen
- Oft sehr große Anzahl
- Statistische Qualität wichtig

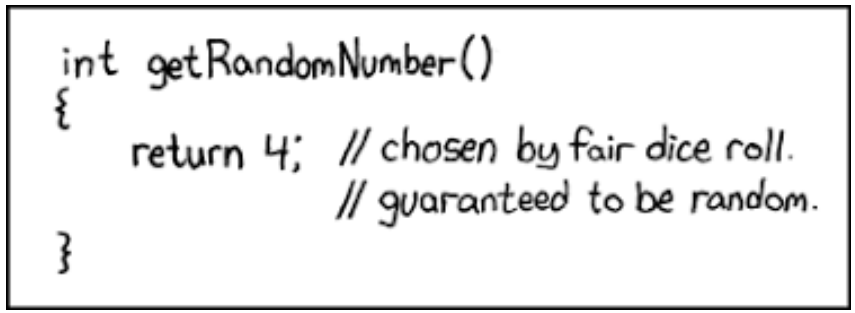
Zufallszahlen

- **Echte Zufallszahlen** (Münze, Würfeln, Radioaktiver Zerfall, elektr. Rauschen, Quanteneffekte etc.) sind aufwendig
- Verwende sog. **Pseudozufallszahlen**, die mittels determ. Algorithmen berechnet werden. Diese Algorithmen nennt man **Pseudozufallszahlengeneratoren** (PRNG).
- PRNGs erzeugen eine Folge von Zufallszahlen (ganzzahlig) ausgehend von Startwert (*seed*)
- Wichtige Eigenschaften von PRNGs: (stat.) **Qualität**, **Periode**, **Geschwindigkeit**
- **Qualität** lässt sich mit stat. Tests prüfen (Unabhängigkeit, Verteilung, Korrelationen)
- **Periode** durch den Alg. festgelegt, **Geschwindigkeit** abh. von Implementierung
- Wichtige PRNGs: Arithmetische Berechnungen, Kongruenzgeneratoren, Mersenne-Twister



Zufallszahlengenerator

PRNG: Iterative Berechung einer Folge von Zufallszahlen mit Startwert (*seed*). Ergebnis: gleichverteilte Zufallszahlen
Hybrider Generator: Initialisierung mit echter Zufallszahl (Hardware/Nutzer Infos)



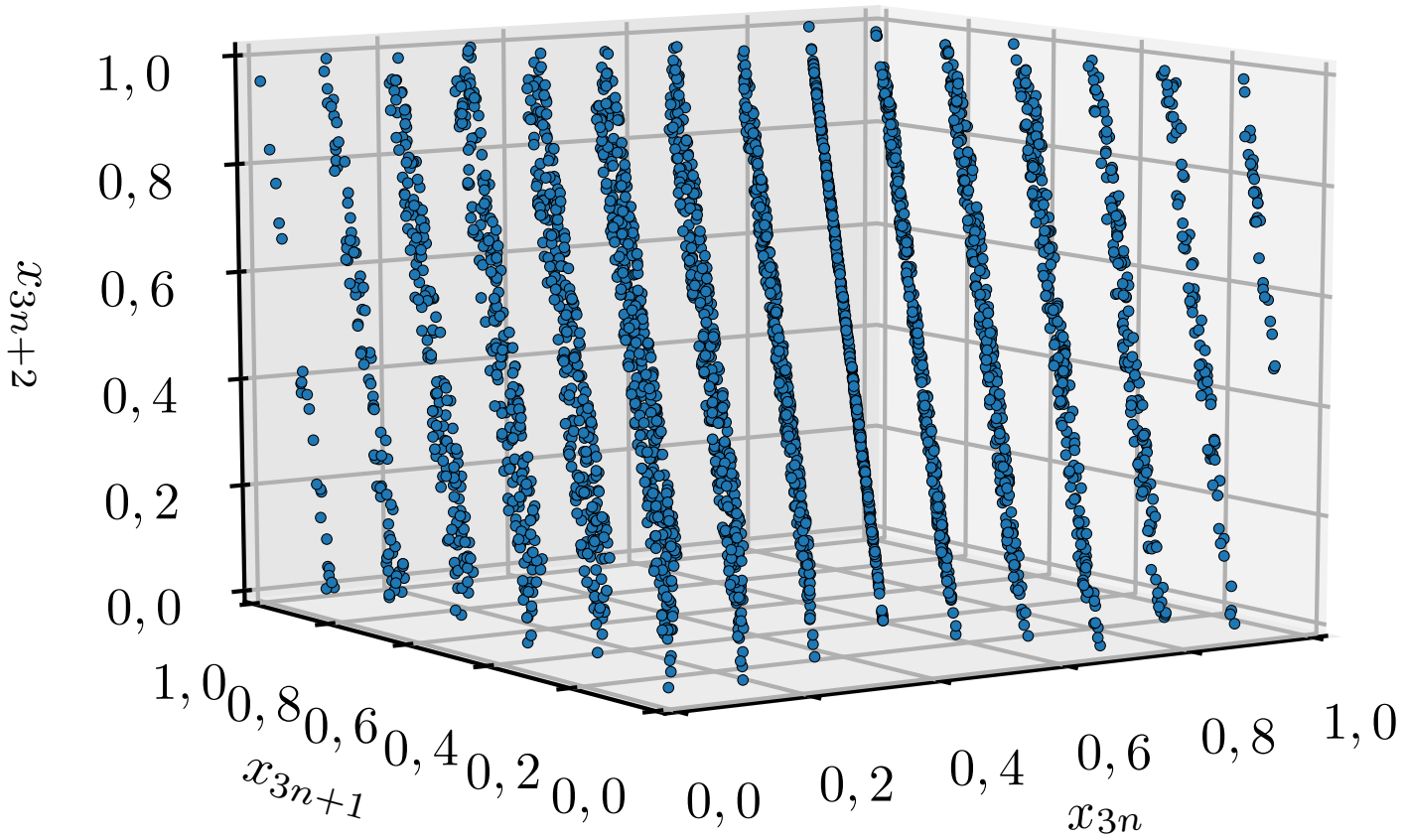
Beispiel **Linearer Kongruenzgenerator**(LKG):

$$x_{n+1} = (ax_n + b) \mod m$$

Parameter: a, b, m
x_0 - Startwert
m bestimmt die Periode und je nach Wahl a und b schwankt die Qualität.
LKG: geringe Qualität, da z. B. Hyperebenenverhalten.

Hyperebenenverhalten

3D Spektraltest des LKG für $a = 65539$, $b = 0$ und $m = 2^{31}$:



PRNG - Mersenne-Twister

- Sehr gute stat. Qualität
- Extrem lange Periode (**MT19937**: $2^{19937} - 1 \approx 10^{6001}$)
- Oft in Monte-Carlo-Simulationen (MC) und Molekulardynamik (MD) genutzt

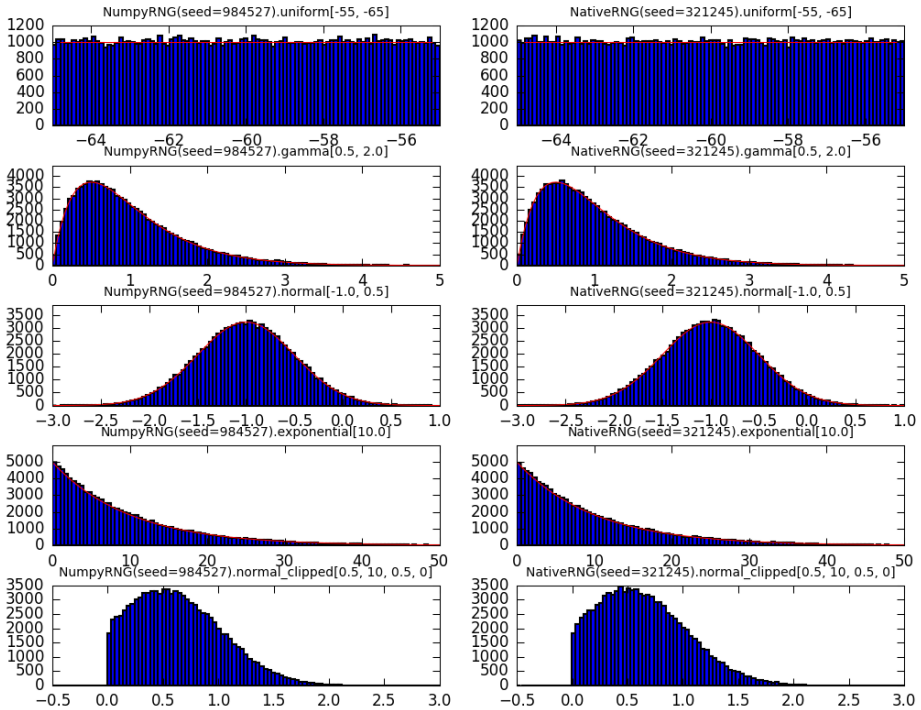
Python verwendet MT19937 (Numpy: wählbar, default: PCG64)

`random.seed(S)`: Startwert S setzen
`random.random()`: Zufallszahlen im Bereich $[0, 1)$
`random.uniform(a, b)`: Zufallszahlen im Bereich $[a, b)$
`random.randint(a, b)`: ganzzahlige Zufallszahlen im Bereich $[a, b)$

Zufallszahlenverteilungen

PRNGs liefern gleichverteilte Zufallszahlen. Viele Anwendungen:
Zufallszahlen mit einer bestimmten **Wahrscheinlichkeitsverteilung**
Wichtige **Methoden**:

- **Inversionmethode** (Bsp. Exponentialvert., Lorentz-vert.)
- **Verwerfungsmethode**
- Normalverteilung: Box-Muller, Polarmethode, Ziggurat-Methode



Inversionsmethode

Transformation der Verteilung (von Gleichverteilung auf gewünschte Verteilung):
Sei u eine gleichverteilte Zufallsvariable, d. h.

$$F_u(x) = P(u \leq x) = \int_0^x \mathrm{d}x' = x, \tag{1}$$

Ausgehend von der normierten Wahrscheinlichkeitsverteilung $\rho(x)$ ist die kumulative Verteilungsfunktion einer Zufallsvariable z

$$F_z(x) = P(z_\rho \leq x) = \int_{x_0}^x \rho(x')\,\mathrm{d}x'$$

dann erhält man eine ρ-verteilte Zufallsvariable z durch

$$\boxed{z = F_z^{-1}(u)}.$$

denn

$$F_z(x) = P(z \leq x) \overset{(1)}{=} F_u(F_z(x)) = P(u \leq F_z(x)) = P(F_z^{-1}(u) \leq x)$$

Inversionsmethode - Beispiele

Exponentialverteilte Zufallszahlen($\rho(x) = a\mathrm{e}^{-ax}$ $(x \in [0, \infty))$):

$$F(z) = \int_0^z \rho(x')\,\mathrm{d}x' = 1 - \mathrm{e}^{-az} = u,$$

wobei u eine gleichverteilte Zufallsvariable ist. Damit $z = -\frac{1}{a}\ln(1-u)$. Also

$$\boxed{z_i = -\frac{1}{a}\ln(1-u_i)} \quad (i = 1, ..., n).$$

Lorentz-verteilte Zufallszahlen($\rho(x) = \frac{\gamma}{\pi(\gamma^2+x^2)}$ $(x \in (-\infty, \infty))$):

$$F(z) = \int_{-\infty}^x \rho(x')\,\mathrm{d}x' = \frac{1}{2} + \frac{1}{\pi}\,\mathrm{atan}\left(\frac{x}{\gamma}\right) = u.$$

Invertieren: $z = \gamma\tan\left(\pi\left(u - \frac{1}{2}\right)\right)$, d. h.

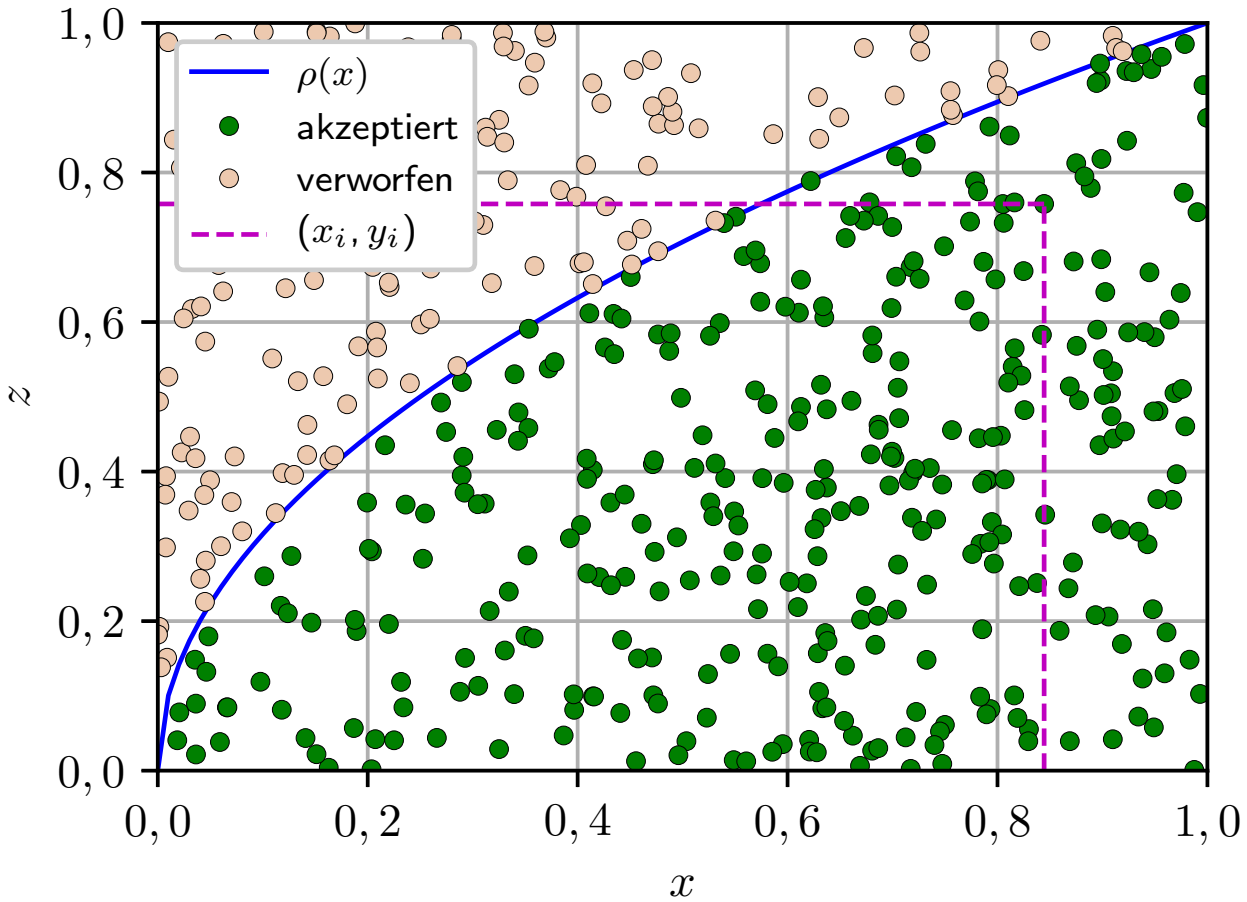
$$\boxed{z_i = \gamma\tan\left(\pi\left(u_i - \frac{1}{2}\right)\right)} \quad (i = 1, ..., n).$$

Verwerfungsmethode

z. B. Erzeuge Zufallszahlen mit einer Wurzelverteilung

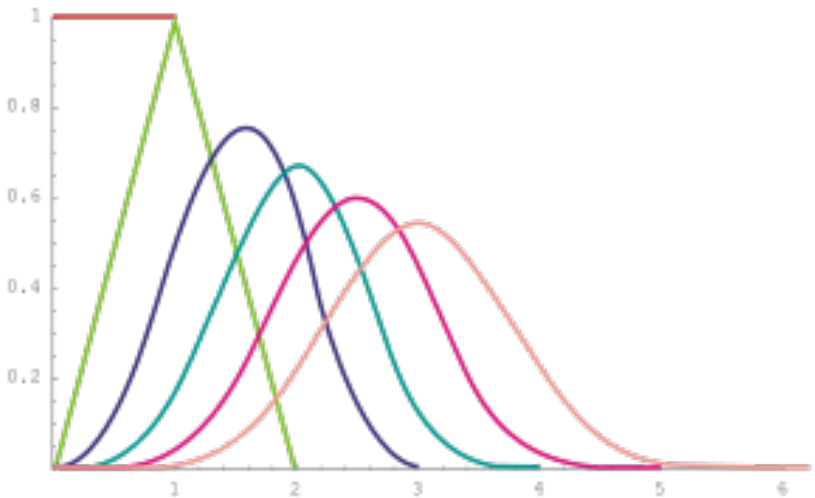
Effizienz:

$$\frac{\int_a^b \rho(x)\,\mathrm{d}x}{(b-a)\Delta y}$$



Normalverteilte Zufallszahlen - Zwölfelregel

Betrachte **Summe von n gleichverteilten Zufallszahlen**. Die Summe hat dann eine Irwin-Hall-Verteilung, die für große n sehr schnell zur Normalverteilung konvergiert (zentrale Grenzwertsatz).



Zwölferregel: 12 gleichverteilten Zufallszahlen im Interval $[0:1]$ + Normierung $((s - n/2)\sigma + \mu)$
Vorteil: sehr geringer Rechenaufwand (Summe + Normierung)
Nachteil: Unabhängigkeit der Zufallszahlen wichtig (Spektraltest)
$\rightarrow$ Andere Methoden (Polarmethode oder Ziggurat) heutzutage besser

Normalverteilte Zufallszahlen - Box-Muller Methode (1958)

$$\rho(x,y) = \frac{1}{2\pi} e^{-\frac{x^2+y^2}{2}}$$

bzw. in Polarkoordinaten

$$\rho(r,\varphi) = \frac{r}{2\pi} e^{-\frac{r^2}{2}}.$$

Nun kann für r die Inversionsmethode angewendet werden:

$$F(r) = \int_0^r e^{-\frac{r'^2}{2}} r' \, dr' = 1 - e^{-\frac{r^2}{2}} = 1 - u_1.$$

Hier kann man $1 - u_1$ statt u_1 verwenden, da u_1 gleichverteilt ist in dem Intervall $[0,1]$ und damit auch $1 - u_1$.
Das Ergebnis ist daher

$$r = F^{-1}(1 - u_1) = \sqrt{-2 \ln u_1},$$

bzw. wieder in kartesischen Koordinaten ($\varphi =: 2\pi u_2$ ist gleichverteilt in $[0, 2\pi]$)

$$\begin{aligned} x &= r\cos\varphi = \sqrt{-2\ln u_1}\cos(2\pi u_2), \\ y &= r\sin\varphi = \sqrt{-2\ln u_1}\sin(2\pi u_2). \end{aligned}$$

Mit dieser Vorschrift erhält man also zu je zwei gleichverteilten Zufallszahlen $u_1, u_2 \in [0,1]$, zwei normalverteilte Zufallszahlen x, y.

Normalverteilte Zufallszahlen - Polarmethode (Marsaglia)

Aufwendige trigonom. Funktionen bei Box-Muller-Methode vermeiden: transformiere auf Polarkoordinaten

Polarmethode:

Für zwei gleichverteilte Zufallszahlen $u_1, u_2 \in [-1, 1]$ berechnet man $q = u_1^2 + u_2^2$ so lange, bis $q \leq 1$. Ist diese Bedingung erfüllt, erhält man mit

$$\boxed{p = \sqrt{-2\frac{\ln q}{q}}}$$

zwei normalverteilte Zufallszahlen durch

$$\begin{aligned} x &= u_1 p, \\ y &= u_2 p. \end{aligned}$$

Normalverteilte Zufallszahlen - Ziggurat Methode

Schneller ist die **Ziggurat Methode**:

- Überdeckung der Normalverteilung mit Rechtecken $\rightarrow$ Zigguratform (Tempelform in Mesopotamien)
- Für jedes Rechteck: Verwerfungsmethode (nur in wenigen Prozent der Fälle verworfen: dann Berechnung aufwendiger)
- $\rightarrow$ sehr schnell, aber auch aufwendiger. Wird von NumPy verwendet und z. B. in der GSL.

Nicht-gleichverteilte Zufallszahlen

In **Python**:
(verwendet 256-Schritt Zigguratmethode)

```
1  from numpy.random import default_rng
2
3  rng = default_rng()
4
5  vals = rng.standard_normal(100)
6  # oder
7  vals = rng.normal(mu, sigma, 100)
```

oder

```
1  from numpy import random
2
3  vals = random.standard_normal(100)
4  # oder
5  vals = random.normal(mu, sigma, 100)
```

oder

```
1  from random import gauss
2
3  gauss(mu, sigma)
```

Monte-Carlo-Integration (MC-Integration)

Bisher: Numerische Integration durch Verwendung von äqudistanten Stützstellen.
Jetzt: verwende **zufällige Stützstellen** x_i und summiere die Funktionswerte, d. h.

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{N}\sum_{i=1}^{N} f(x_i).$$

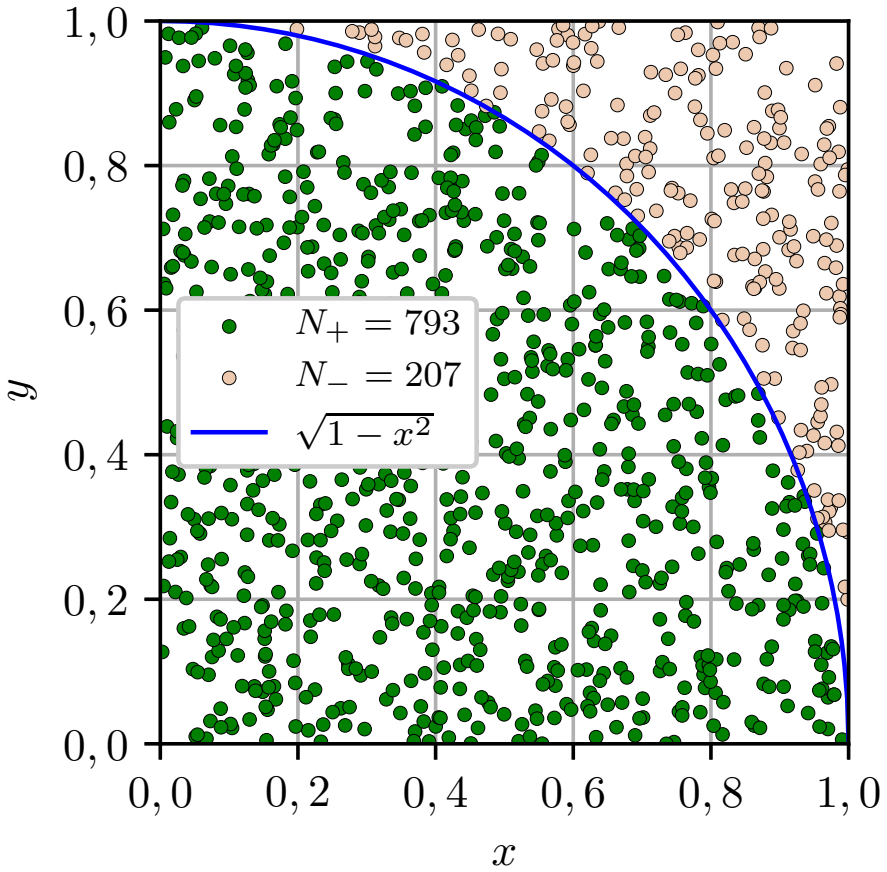
Gesetz der gr. Zahlen: Nähert sich dem Integralwert mit steigendem N.

MC-Integration

Bsp.: MC-Integration als **Steinwurfmethode** für den Viertelkreis

$$\boxed{f(x) = \sqrt{1 - x^2} \quad (x \in [0, 1])}$$

Ergebnis: $\pi = 4A \approx 4\frac{N_+}{N} = 4\frac{793}{1000} = 3,172.$

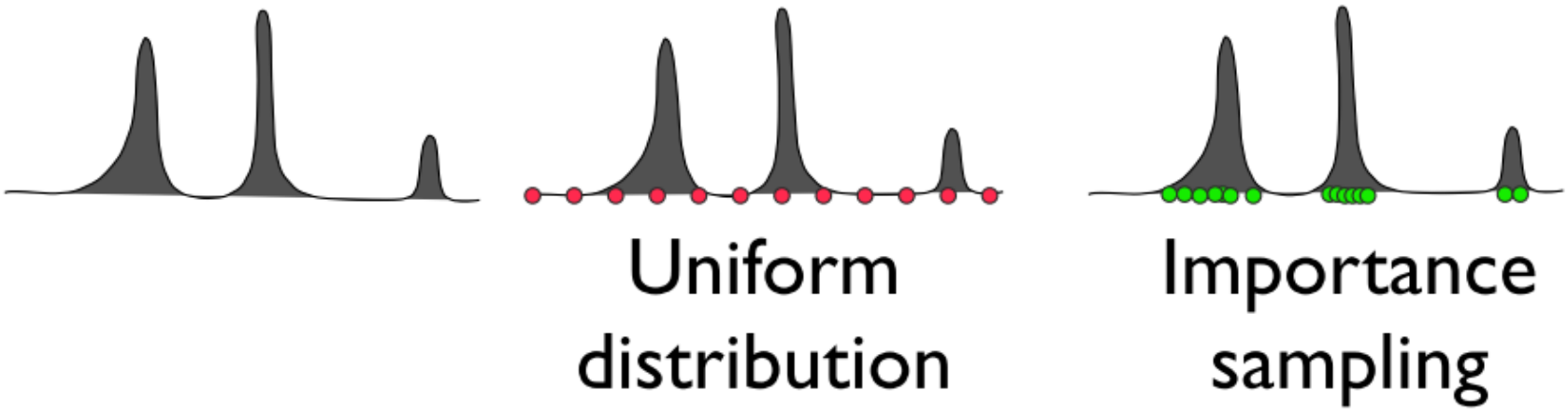


MC-Integration

Aufwand der numerischen Integration hängt von **Dimension** ab, nicht aber bei MC-Integration!
Bei hochdimensionalen Integralen (z.B. Phasenraum): MC-Integration.
Außerdem: Gleichverteilung der Stützstellen schlecht, insbesondere bei lokalisierten Funktionen.

$\rightarrow$ **Importance Sampling**:

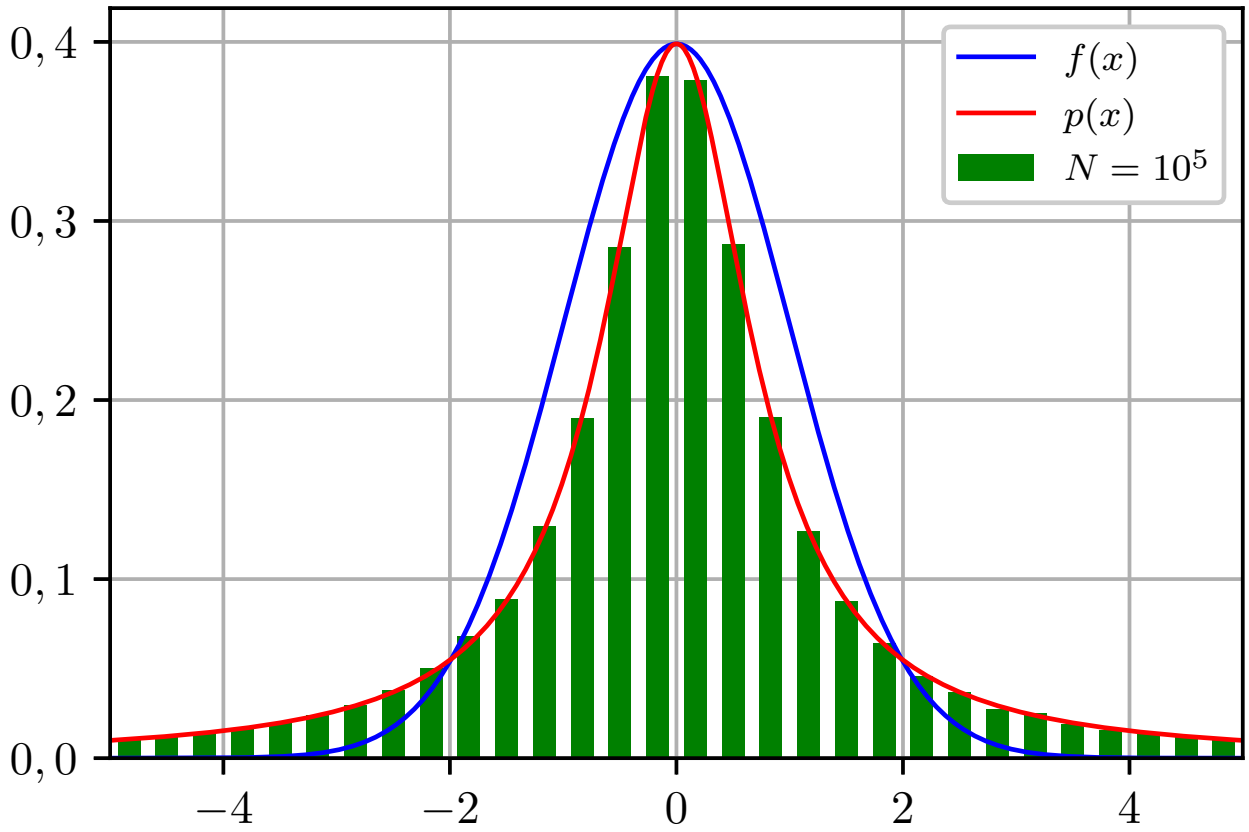
Verwende Zufallszahlen, deren Verteilung etwa der zu integrierenden Funktion entsprechen. Sonst werden Zufallszahlen an Stellen erzeugt, wo die Funktion praktisch null ist.



Importance Sampling

$$\int_a^b f(x)\,\mathrm{d}x = \int_a^b \frac{f(x)}{p(x)} p(x)\,\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N} \frac{f(z_i)}{p(z_i)}.$$

Beispiel: Gauss-Integral mithilfe von Lorentz-vert. Zufallszahlen:



Importance Sampling - Vergleich

Bestimmung von π mit **gleichverteilten Zufallszahlen**:

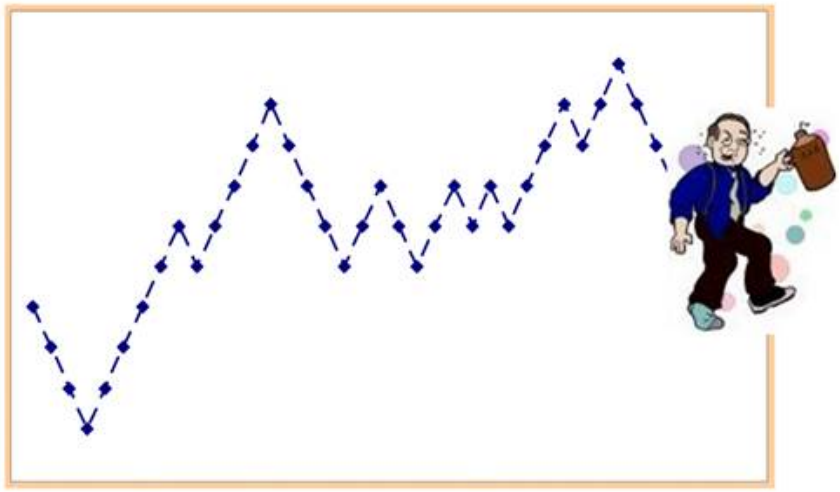
Anzahl Stützstellen	**Ergebnis**	**Mittlerer rel. Fehler**
N	$\frac{4}{Z}\sum_{k=1}^{Z} A_k$	$\frac{1}{Z\pi}\sum_{k=1}^{Z} \lvert\pi - 4A_k\rvert$
10	$3,172$	40 %
100	$3,1748$	11 %
10^3	$3,14312$	4 %
10^4	$3,14540$	$1,4$ %
10^5	$3,14148$	$0,4$ %
10^6	$3,14166$	$0,13$ %

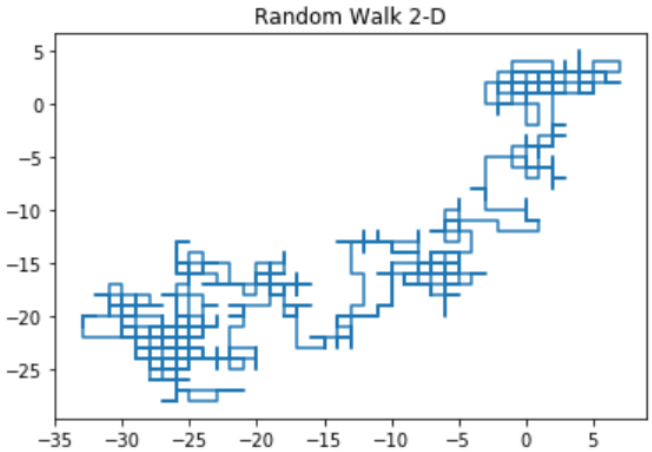
Gaussintegral mit **Lorentz-verteilten Zufallszahlen**:

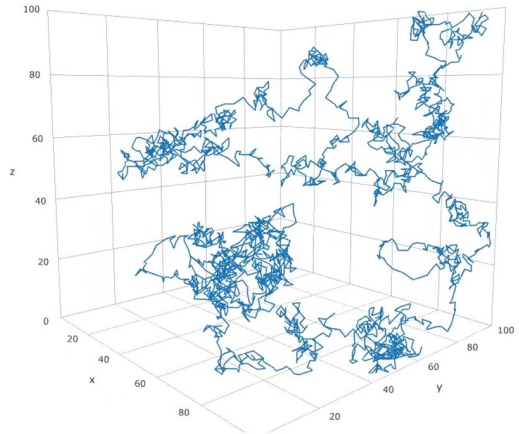
Anzahl Stützstellen	**Ergebnis**	**Mittlerer Fehler**
N	$\frac{1}{Z}\sum_{k=1}^{Z} I_k$	$\frac{1}{Z}\sum_{k=1}^{Z} \lvert 1,0 - I_k\rvert$
10	$0,9985516$	13 %
100	$0,9946688$	4 %
10^3	$1,0000843$	$1,3$ %
10^4	$0,9999832$	$0,4$ %
10^5	$0,9998997$	$0,11$ %
10^6	$1,0000215$	$0,04$ %

Random-Walk

$\rightarrow$ Zufallsbewegung (nicht deterministisch = stochastisch, "drunken sailor") auf ein- oder mehrdimensionalem Gitter bzw. Zeitschritt





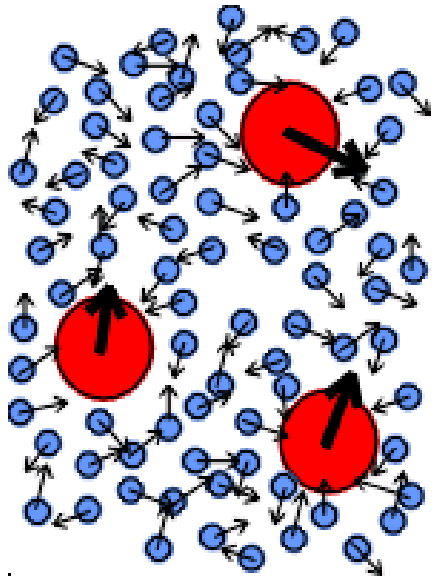


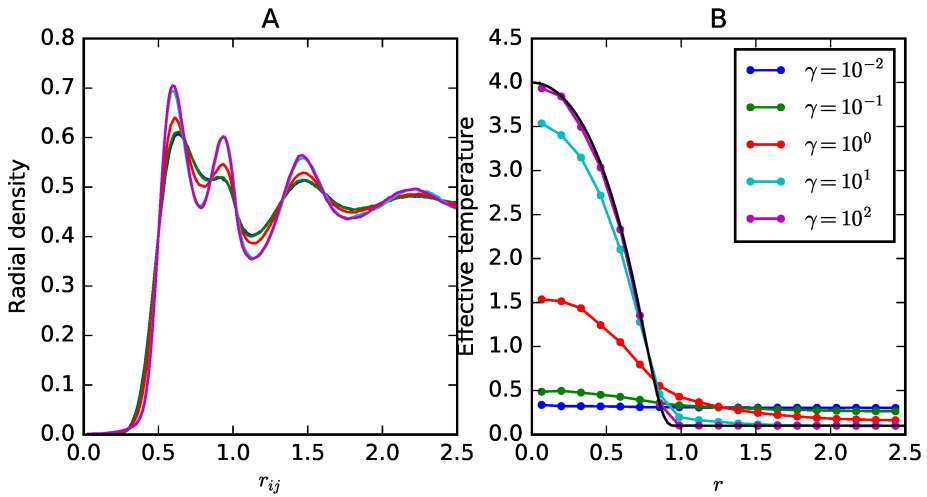
Random-Walk - Beispiele

- **Brownsche Bewegung** (Wiener Prozess)
- **Langevin-Dynamik** $\rightarrow$ Bewegungsgleichung mit stoch. Anteil, z. B.

$$m\ddot{\mathbf{x}}(t) = -\alpha\dot{\mathbf{x}}(t) + \boldsymbol{\eta}(t)$$

- typisch: **"weißes Rauschen"** mit $\langle\boldsymbol{\eta}\rangle = \mathbf{0}$ und $\langle\eta_i(t)\eta_j(t')\rangle = 2\alpha kT\delta_{i,j}\delta(t-t')$
- Bei multiplikativem Rauschen: zwei verschiedene Schema (**Ito und Stratonovich**)

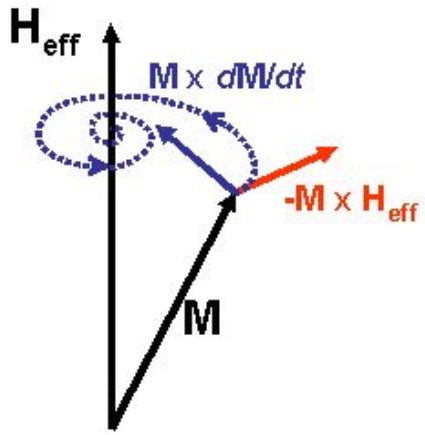




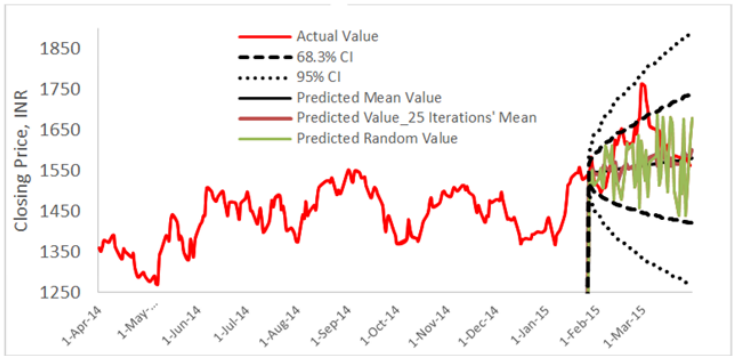
Random-Walk - Mehr Beispiele

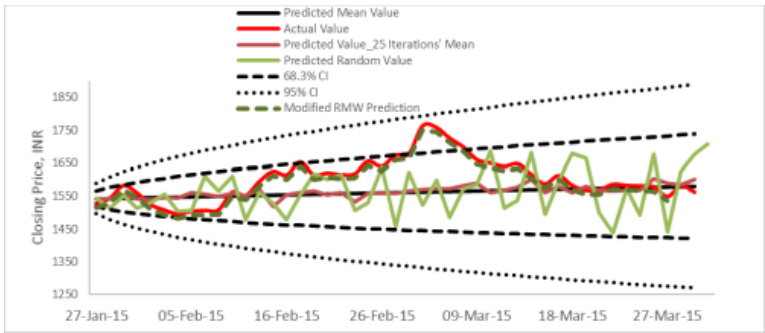
- Spindynamik: stoch. Landau-Lifschitz (LL) Gleichung

$$\dot{\mathbf{m}}(t) = -\gamma \mathbf{m} \times [\mathbf{H}_{\text{eff}} + \boldsymbol{\eta}(t)] - \lambda \mathbf{m} \times (\mathbf{m} \times [\mathbf{H}_{\text{eff}} + \boldsymbol{\eta}(\mathbf{t})])$$



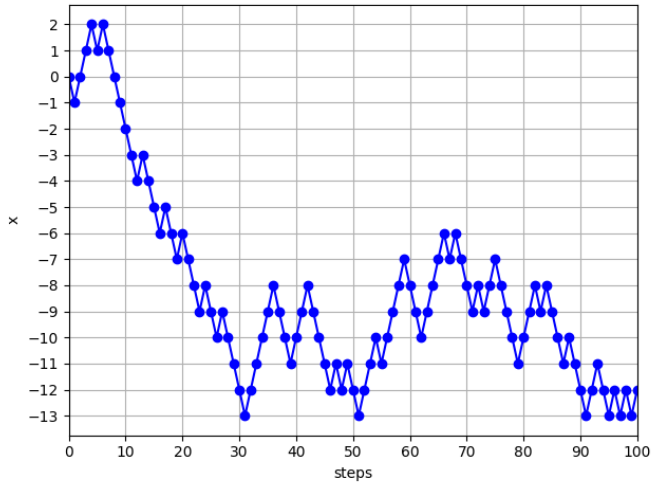
- Aktienkurse (Drift + period. Anteil + Rauschanteil), Black-Scholes-Modell

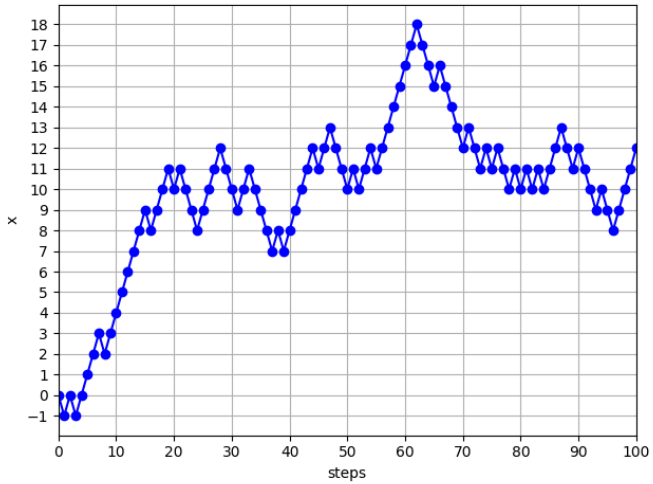


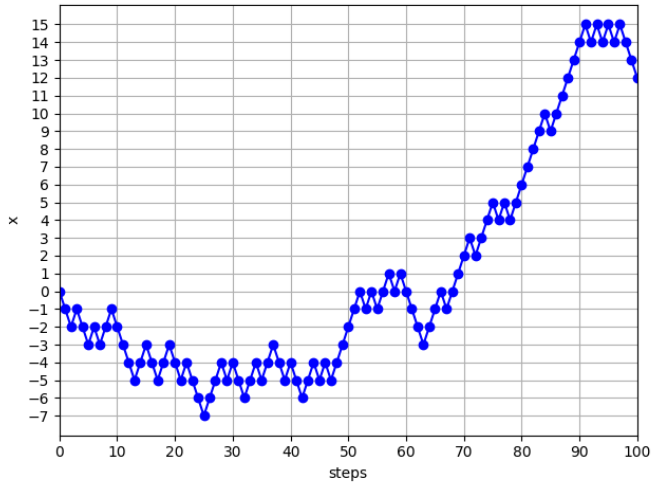


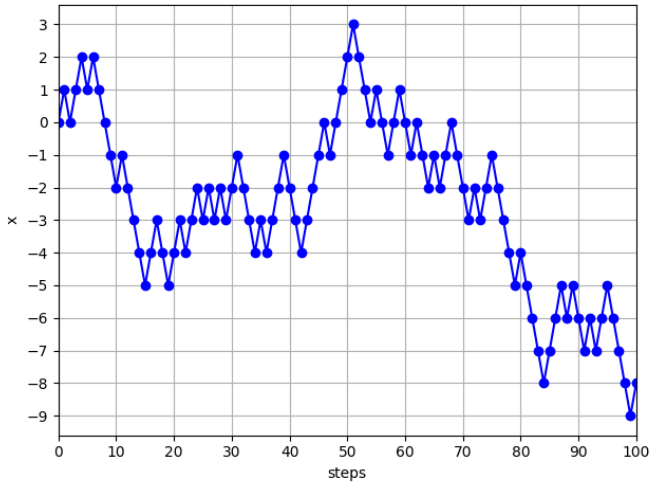
Random-Walk - simuliert

Simuliert: eindimensional, feste Schrittweite, symmetrisch (50-50)



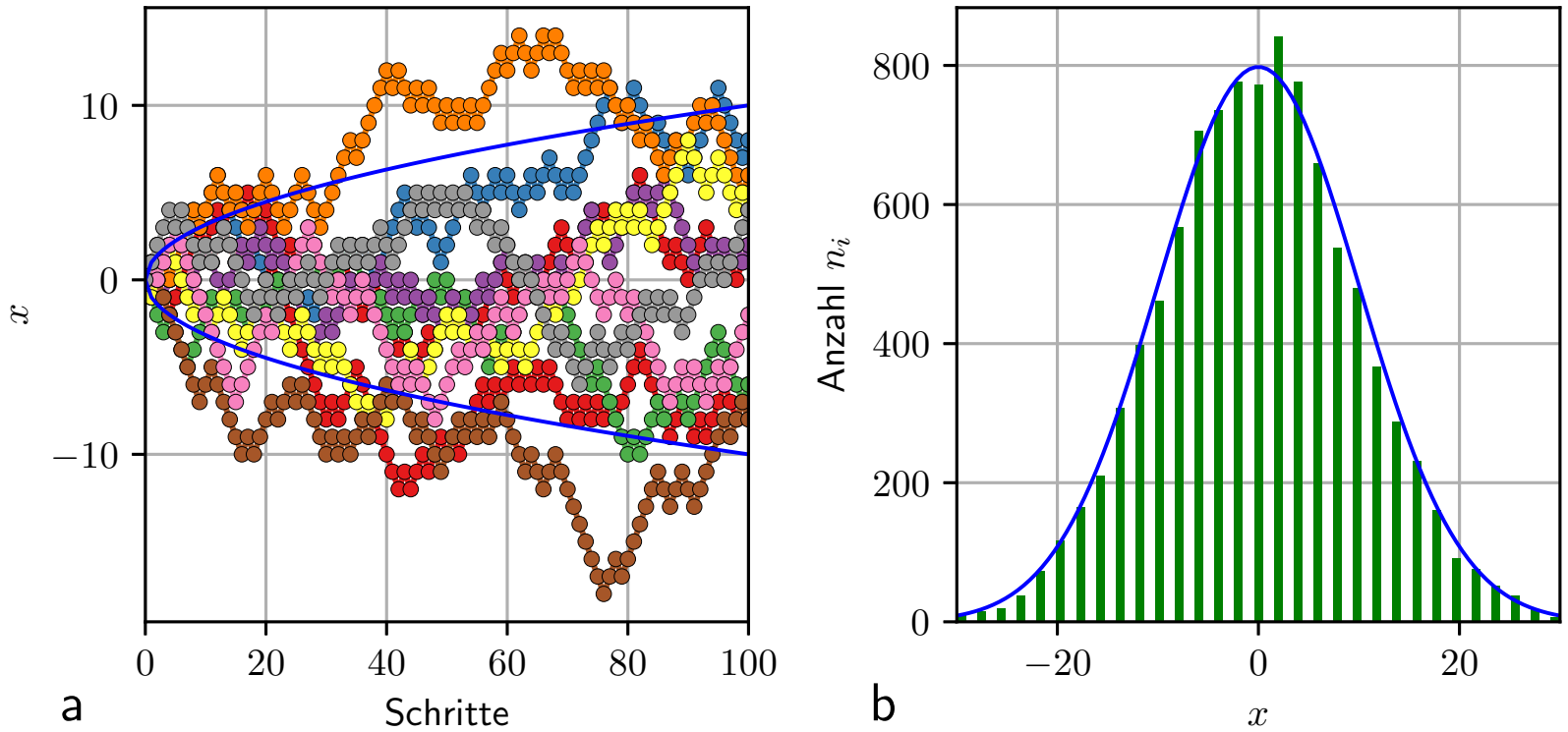






Random-Walk - Statistik

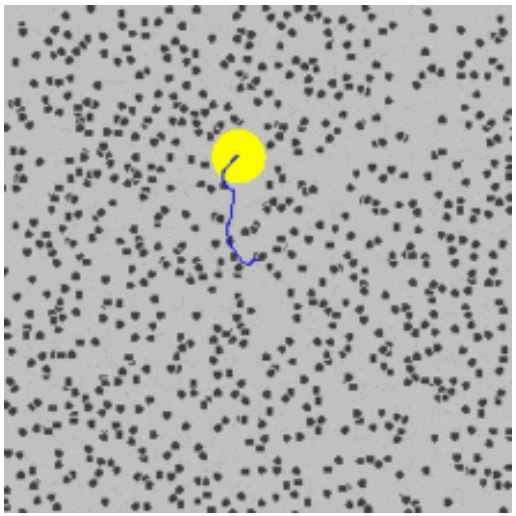
(**a**) Eindimensionaler *Random Walk* für zehn verschiedene Beispiele
(**b**) Verteilung der Endposition nach 100 Schritten für 10.000 Durchläufe:

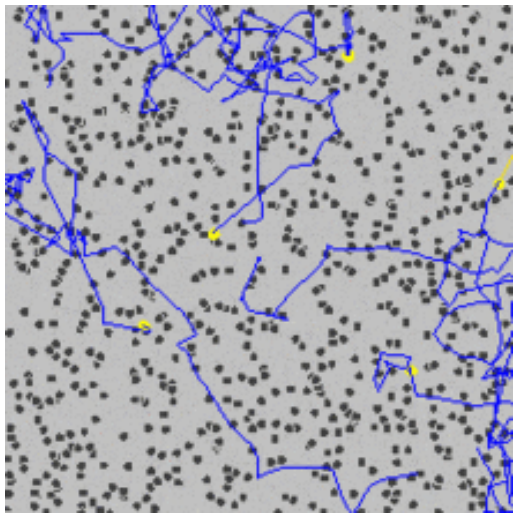


Binomialstatistik mit $\langle x_n \rangle = n(2p-1)$ und $\sigma^2 = 4np(1-p)$.
Für $p = 0,5$ und $n \to \infty$: normalverteilt mit $\mu = 0$ und $\sigma = \sqrt{n}$

Stochastische Simulationen

Molekulardynamik (MD) Simulation: Berechne mit diskreten Zeitschritten die Zeitentw. der (stoch.) Bewegungsgl. und damit Dynamik eines Systems
Bsp. **Brownsche Bewegung**





Monte-Carlo (MC) Simulation: Zufallsexperimente und statistische Auswertung der Ergebnisse
Bsp. **Ising-Modell**

MC Simulation - Ising Modell

- Simulation von wechselwirkenden Spins S_i auf Gitter
- $S_i = \pm 1$ (Spin up - Spin down)
- Energie = Wechselwirkung untereinander + ext. Magnetfeld:

$$E_i = -J \sum_{NN} S_i S_j - \mu_B S_i B$$

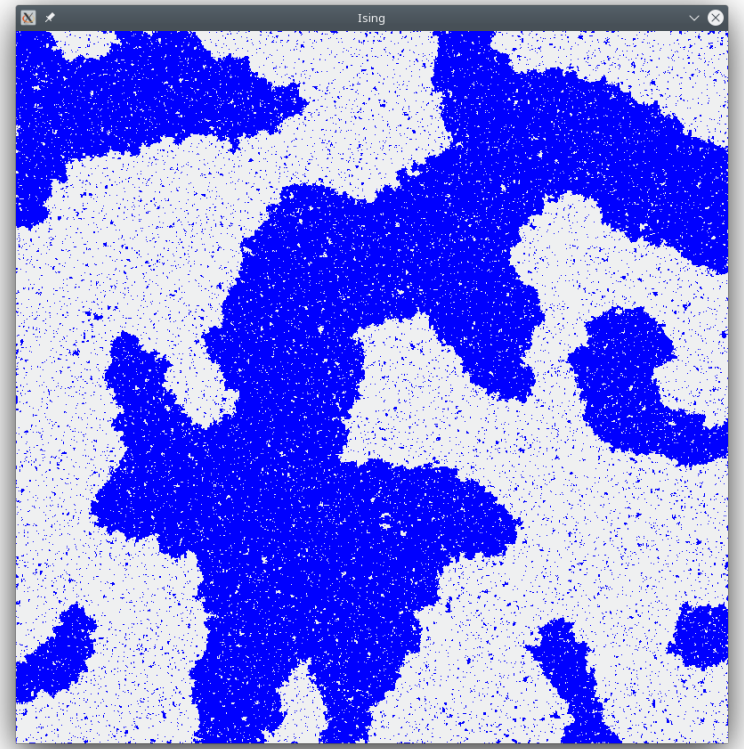
- Wahrscheinlichkeit für Flip hängt von Nachbarn ab (und Kopplung)

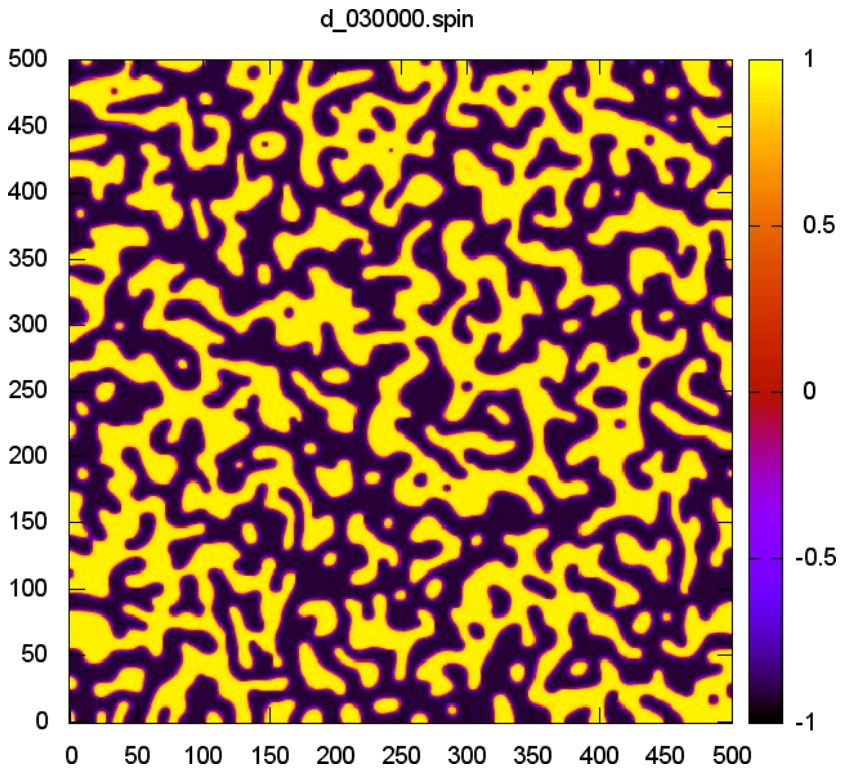
$$P = \exp(-\beta E) = \exp(-\frac{1}{k_B T} S_i (J \sum_{NN} S_j + \mu_B B))$$

$\rightarrow$ GUI-Simulation

MC Simulation - Ising Modell

Ergebnisse:





Zelluläre Automaten

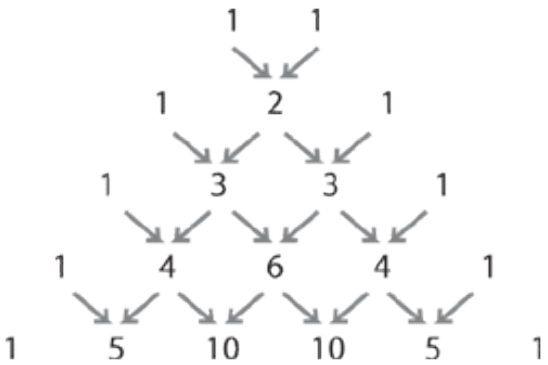
Simuliere diskrete dynamische Systeme "auf einem Gitter". **Einfache Regeln** beschreiben **iterative Entwicklung** von Zellen. Verhalten kann trotzdem komplex werden (vgl. nicht-lineare Dynamik).
Anwendungen: (Selbst-)Organisation, Evolution (Bio), Strukturbildung (Chemie), ...
Beispiele:

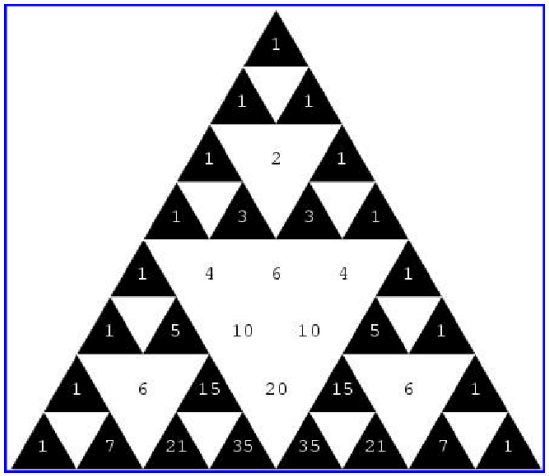
- Pascalsches Dreieck
- Wolfram elementary celluar automata (`https://de.wikipedia.org/wiki/Zellul%C3%A4rer_Automat`)
- Langton-Schleife (`https://de.wikipedia.org/wiki/Langton-Schleife`)
- Game of Life (Convay) (`https://de.wikipedia.org/wiki/Conways_Spiel_des_Lebens`)
- Nagel-Schreckenberg-Modell (`https://de.wikipedia.org/wiki/Nagel-Schreckenberg-Modell`)
- HPP/FHP-Modell (Lattice Gas)

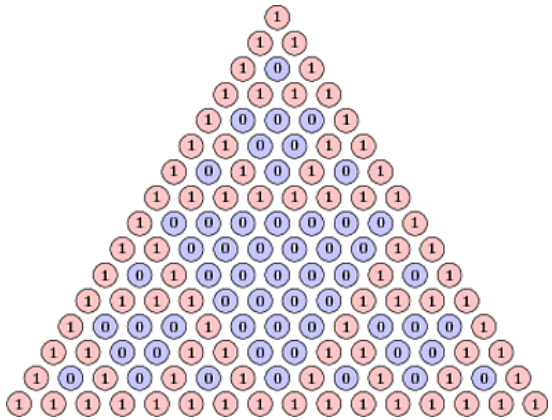
Zelluläre Automaten - Pascalsches Dreieck

- Neue Zeile ist Summe von zwei benachbarten Werten
- Startbedingung: ...000001000000...

...0 0 0 1 0 0 0...
... 0 0 1 1 0 0...
...0 0 1 2 1 0 0...
... 0 1 3 3 1 0...
...0 1 4 6 4 1 0...







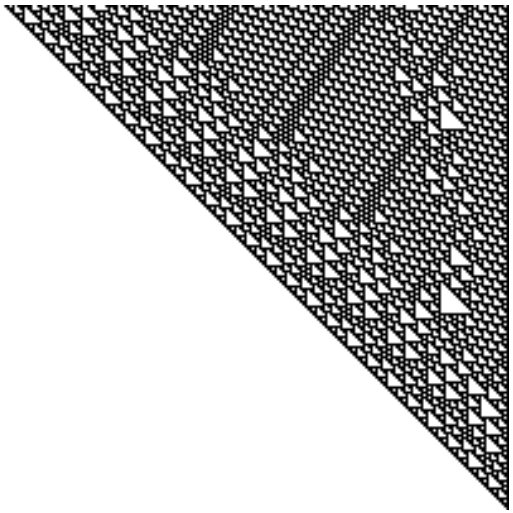
mod 2: Sierpinski-Dreieck

Zelluläre Automaten - Wolfram's Elementarer Zellulärer Automat

Betrachte 3 Zellen mit den Werten 0 oder 1 $\rightarrow$ 8 mögliche Zustände.
Damit $2^8 = 255$ versch. Regeln, wie neue Zustände entstehen:

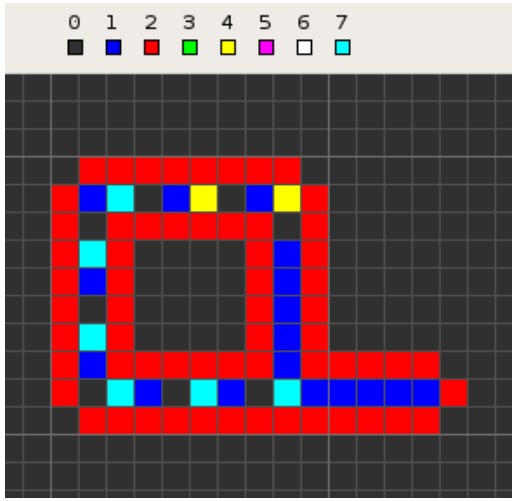
Regel	000	001	010	011	100	101	110	111
1	0	0	0	0	0	0	0	0
..								
30	0	0	0	1	1	1	1	0
..								
110	0	1	1	0	1	1	1	0
..								

Interessante Regeln: 18,90 (Sierpinski) 110, 30

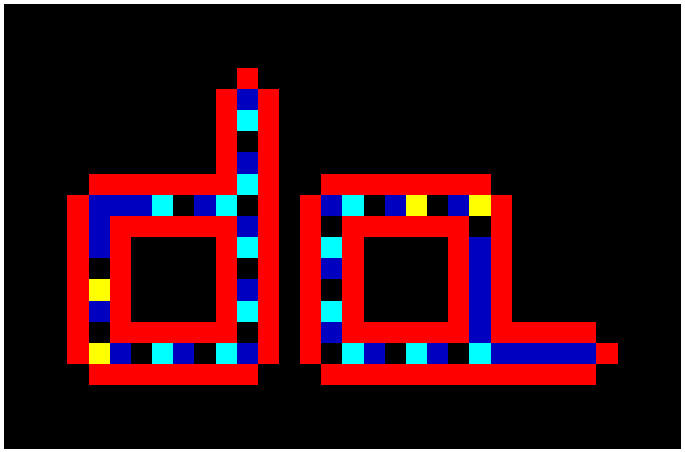


Zelluläre Automaten - Langton-Schleife

Startbedingung (genetische Information im Ring):



Einfachstes, selbstreproduzierendes System $\rightarrow$ Movie

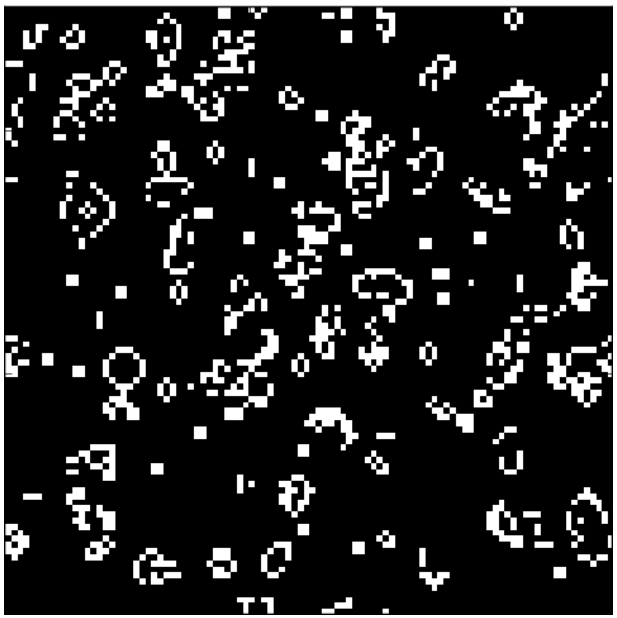


Korallenartig (Reproduktion nur an Rändern)

Zelluläre Automaten - Game of Life (Convay - 1970)

Startbed.: zufällig (50-50)
Regeln:

- tote Zelle mit genau 3 Nachbarn $\rightarrow$ lebt,
- lebende Zelle mit < 2 oder > 3 Nachbarn: tot

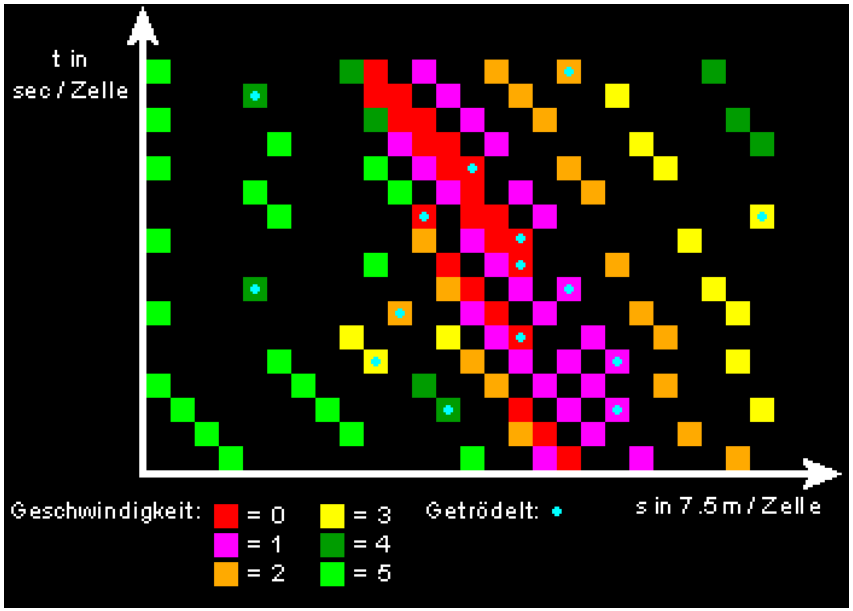


$\rightarrow$ Animation
Ergebnis: Stabile, instabile, period., Raumschiff- und reprod. Strukturen
Modellierung einer Turing-Machine möglich: Turing vollständig!

Zelluläre Automaten - Nagel-Schreckenberg-Modell (NaSch-Modell)

Simulation von Straßenverkehr
Bremsen/Beschleunigen je nach freier Strecke + "Trödelparameter"

Stau aus dem Nichts, Verkehrssimulation in Echtzeit, Verkehrsprognose.
Erweiterung: KI

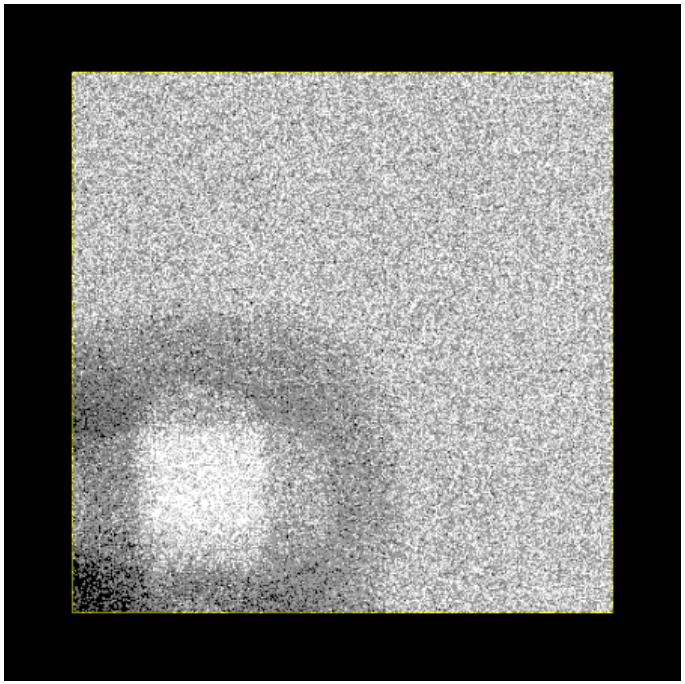


Zelluläre Automaten - Gitter-Gas-Modelle

HPP - orthogonales Gitter (vierzählige Symmetrie)
FHP - Dreiecksgitter (sechszählige Symmetrie)

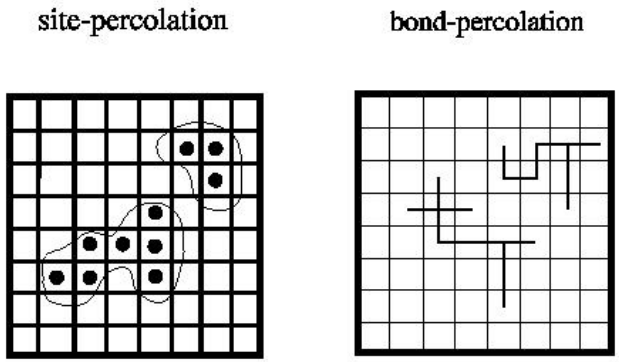
- Teilchen auf Gitterpunkten mit Richtung (Geschw.)
- je Gitterpunkt max. 1 Teilchen (FHP: max. 6)
- Streuung an Gitterpunkten mit Impulserhaltung (det. vs. stoch.)

Grundlage für Lattice-Boltzmann-Methode, Grenzfall von Navier-Stokes-Gleichung

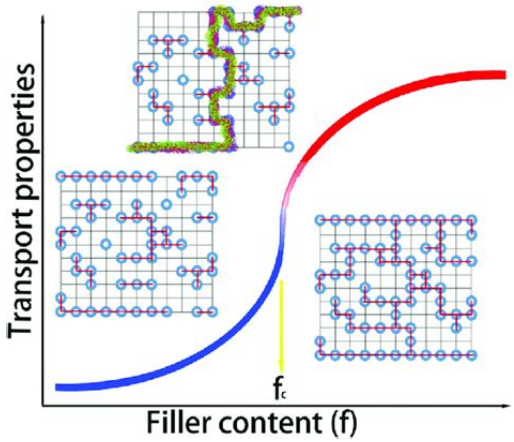


Perkolationstheorie

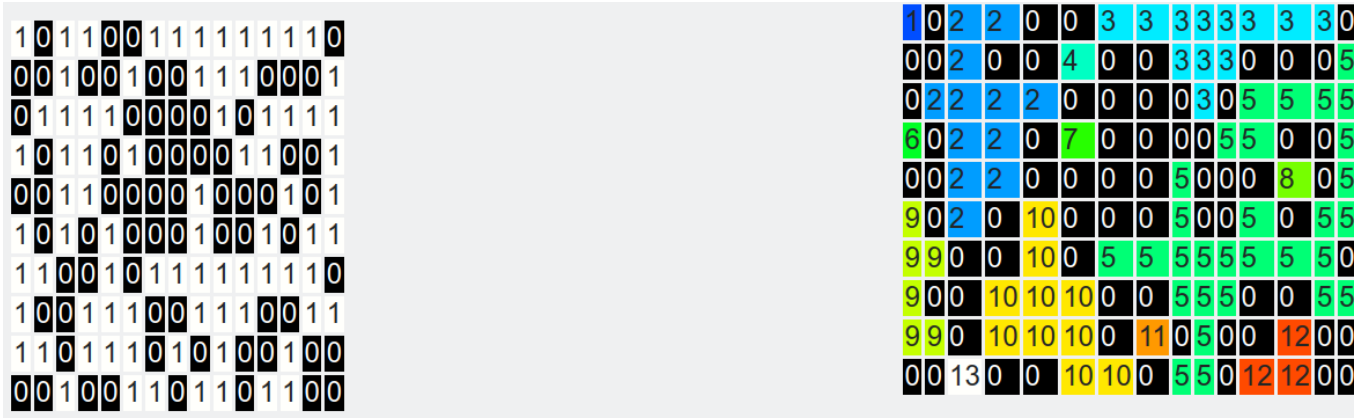
Bildung von zusammenhängenden Strukturen (Cluster) bei zufälliger Besetzung von Gitterplätzen: Knotenperkolation/Kantenperkolation



Ergebnis: Verständnis von **Phasenübergängen** (kritische Werte für Leitfähigkeit, Waldbrände, Wachstumsmodelle)

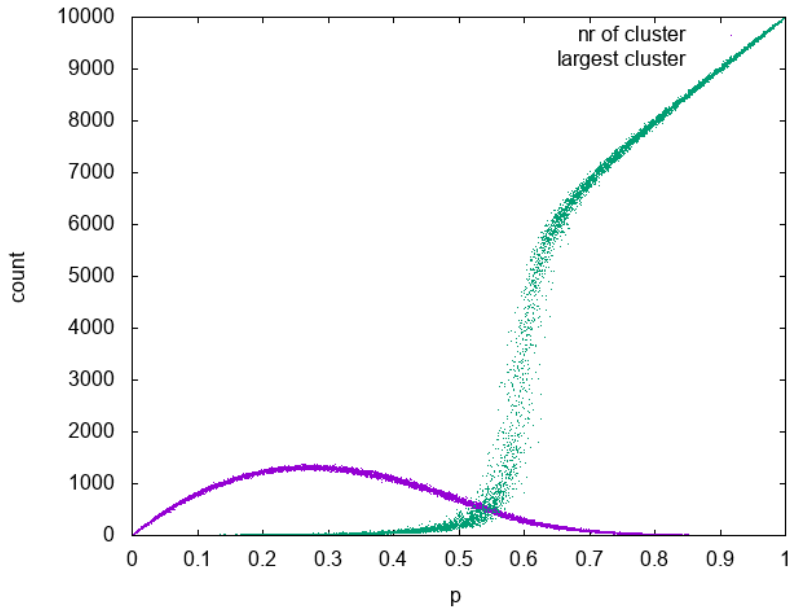


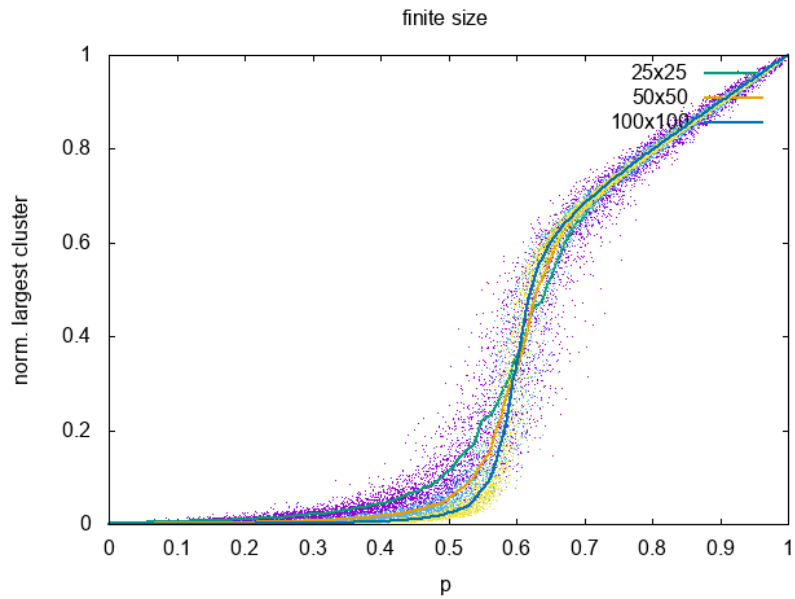
Perkolationstheorie - Labeln: Hoshen-Kopelman (1976)



```
1  nr_label = 0;
2  for x in 0 to n_columns {
3   for y in 0 to n_rows {
4     if occupied[x,y] then
5       left = occupied[x-1,y];
6       above = occupied[x,y-1];
7
8       if (left == 0) and (above == 0) then    // Neither a label above nor to the left.
9         nr_label++;                           // Make a new, as-yet-unused cluster label.
10        label[x,y] = nr_label;
11      else if (left != 0) and (above == 0):   // One neighbor, to the left.
12        label[x,y] = find(left);
13      else if (left == 0) and (above != 0):   // One neighbor, above.
14        label[x,y] = find(above);
15      else                                    // Neighbors BOTH to the left and above.
16        union(left,above);                    // Link the left and above clusters.
17        label[x,y] = find(left);
18    }
19 }
```

Perkolationstheorie - Ergebnisse (2D Quadratgitter)



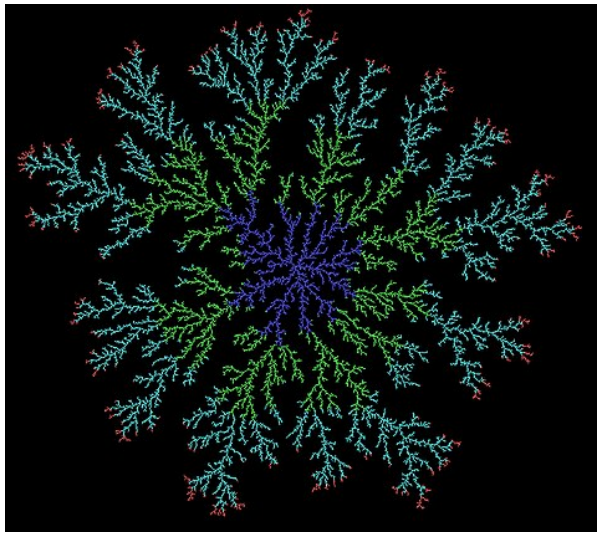
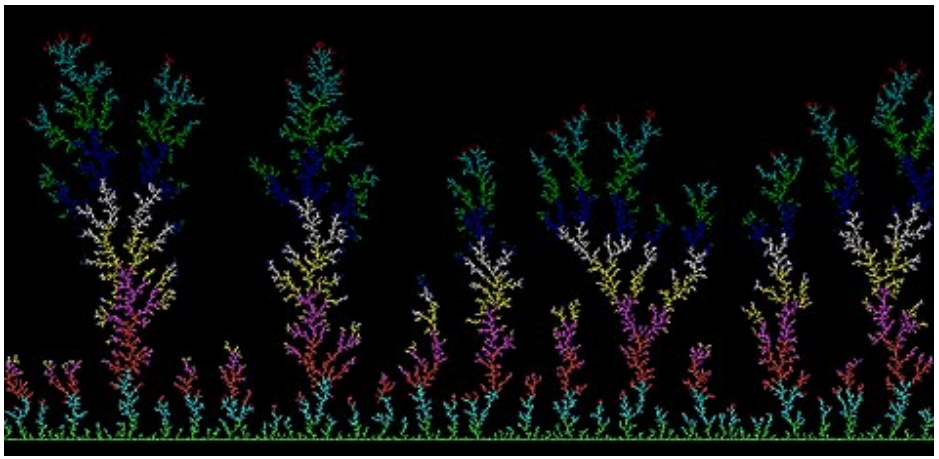
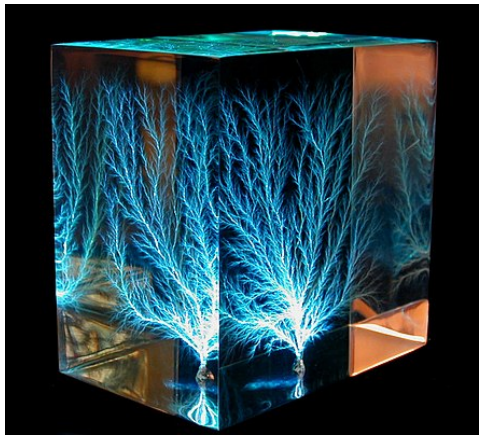
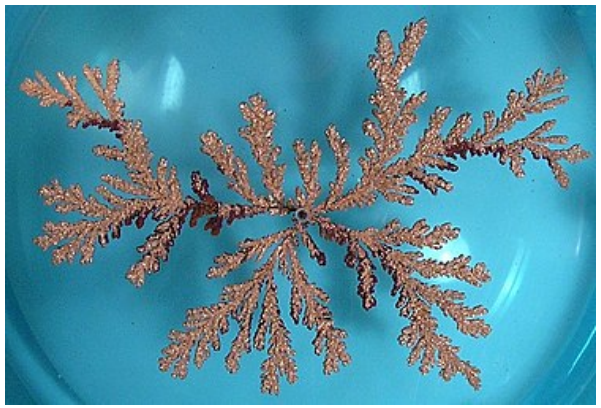


- Perkolation: Ein Cluster verbindet Ränder: Sprung in Leitfähigkeit
- Phasenübergang bei $p \approx 0,59$
- Finite-Size-Effekte (rechts)

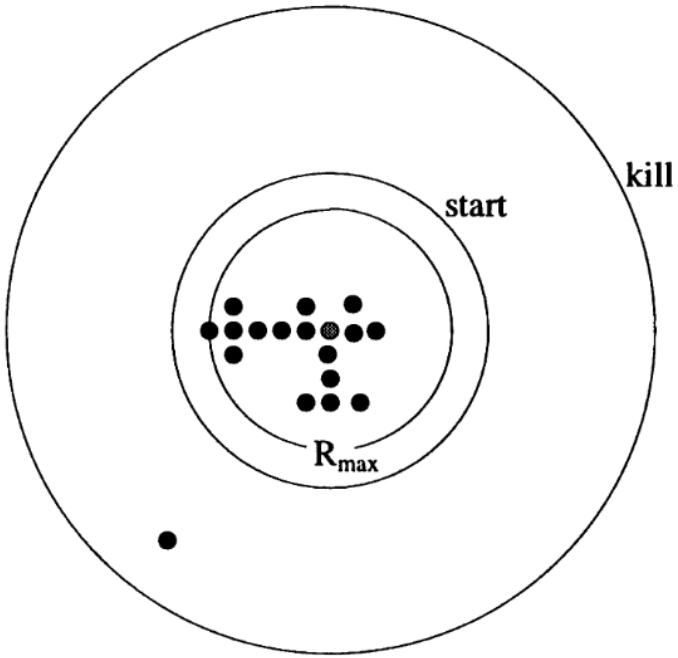
	lattice	bond	site
1D	any	1.0	1.0
2D	square	1/2	.592746
	triangular	.34729	1/2
	honeycomb	.65271	.6962
3D	simple cubic	.2488	.3166
	BCC	.1803	.246
	FCC	0.1992365(10)	0.1201635(10)
	HCP	0.1992555(10)	0.1201640(10)
	tetrahedral(ice)	0.388(10)	0.433(11)

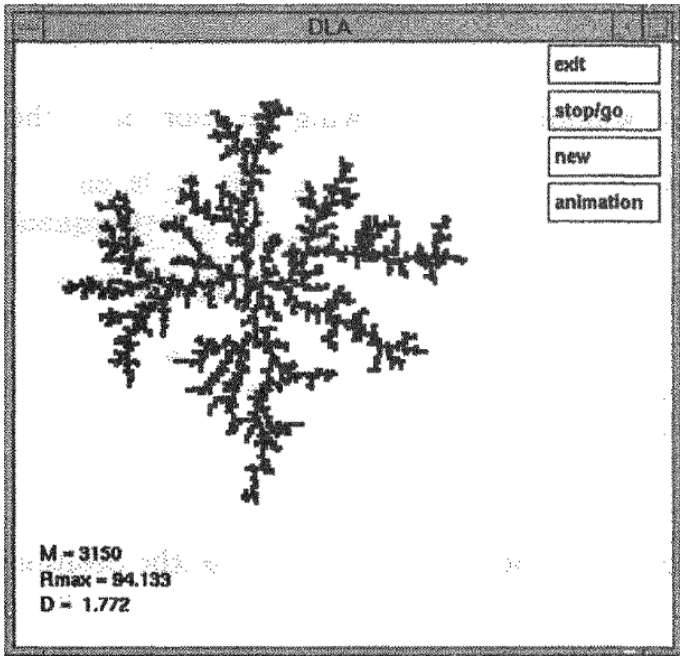
Diffusion Limited Aggregation (DLA)

Random Walk mit Anlagerung an Cluster (Witten & Sander, 1981)
Diffusionsdominierter Transport mit Anlagerung
Ergebnis: Brownsche Bäume
unten: Lichtenberg-Figur, Elektrodeposition, Schneeflocke


Diffusion Limited Aggregation (DLA) - Simulation

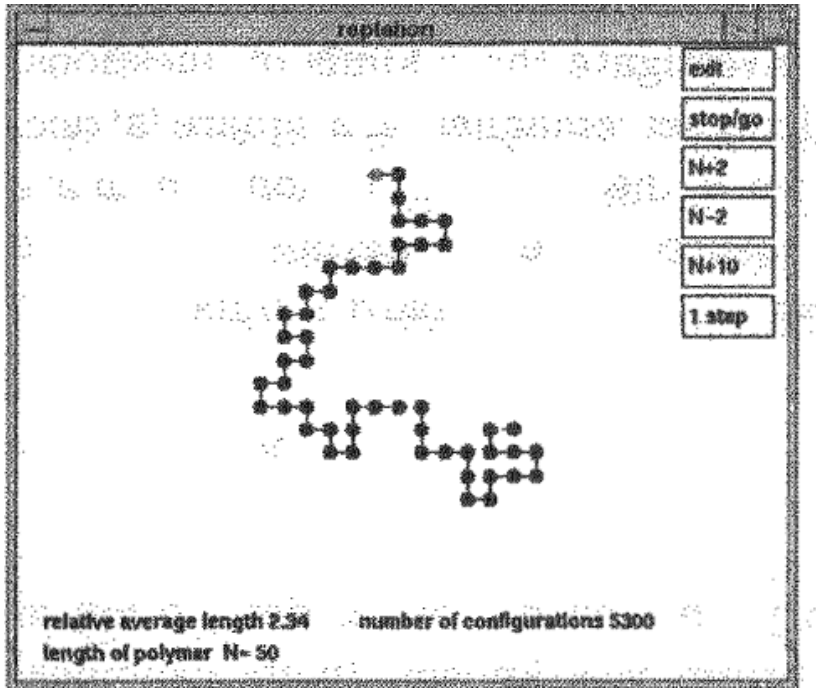


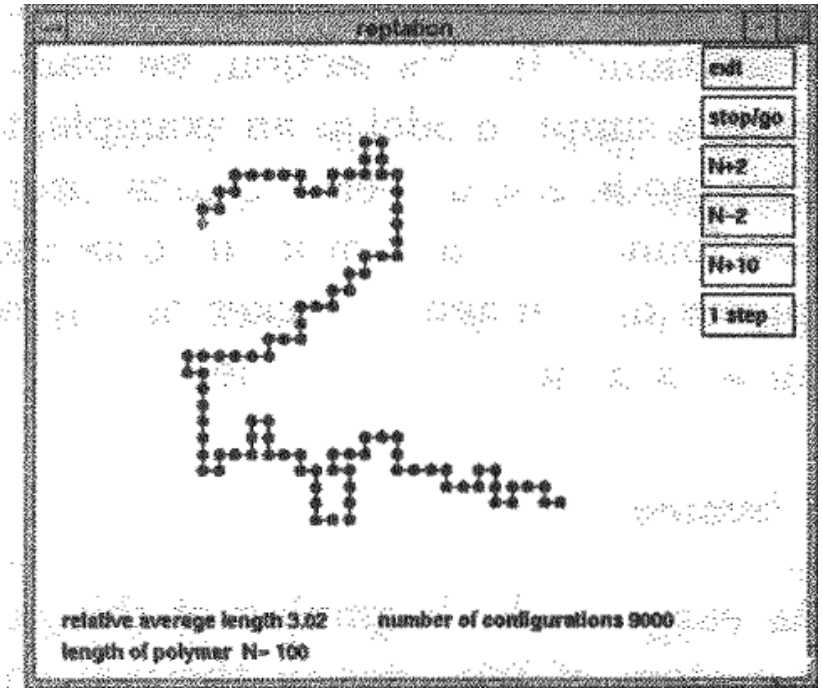


Fraktale Dimension: $D = \ln N / \ln R_{\max}$
(Achtung: Wachtum nicht beendet, d.h. D(r))

Polymerketten

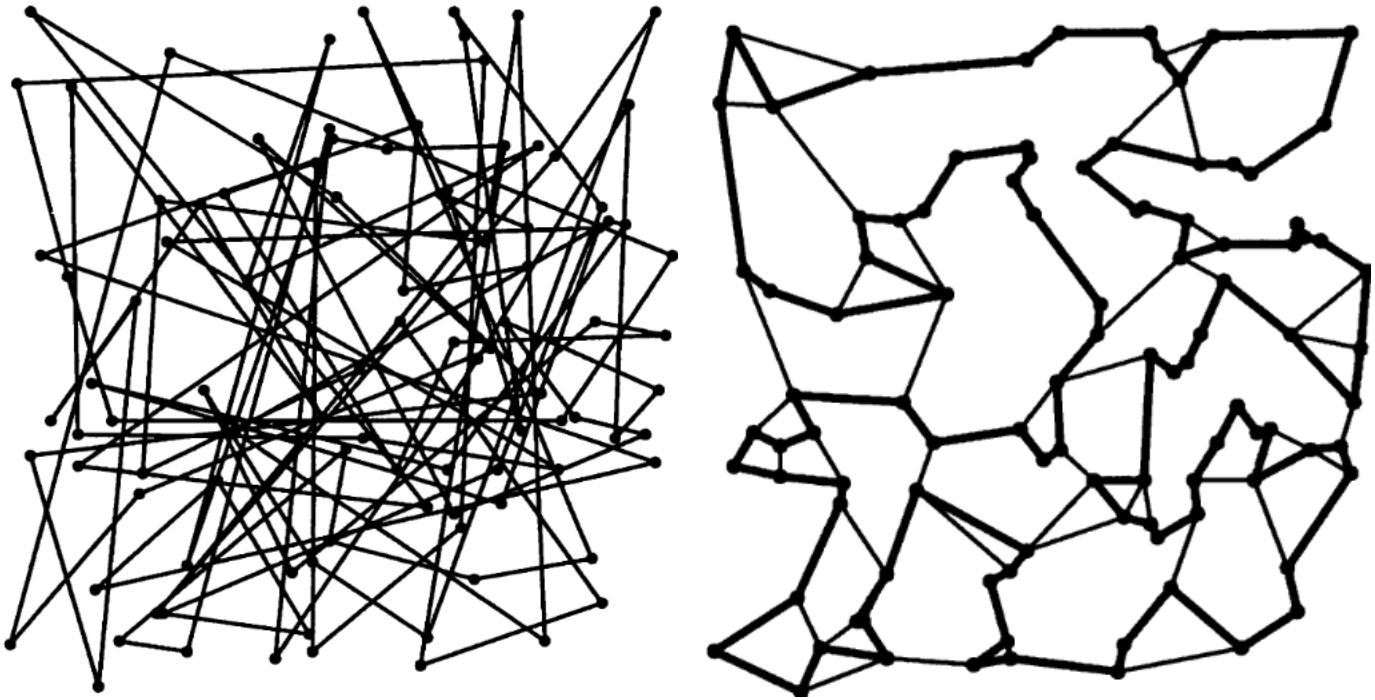
Ketten von Molekülen fester Länge. Simulation mit Hilfe von *self-avoiding random walk* (SAW)





Stochastische Optimierung

$\rightarrow$ Optimierung mit Hilfe von Zufallszahlen
Z.B. *Traveling Salesman*: Finde kürzesten Weg um alle Städte zu bereisen
Algorithmen: Exakt, Hill-Climbing, Simulated Annealing, Ameisen-Alg., evolutionäre Alg., Neuronale Netze
Start: zufällig (hier: $l = 4.8$), Ende: $l \approx 0.85$



Ähnliche Probleme in Schach, Go, etc.: MC Tree Search (MCTS), d.h. zufälliges Rastern der Auswahlmöglichkeiten