Komplexe Systeme: Computerphysik I

Ludger Santen und Heiko Rieger

Vorlesungsmanuskript

Saarbrücken, WS 2003/2004

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Warum Computerphysik?

Physikalische Fragestellungen führen auf mathematische Probleme, die man in der Mehrzahl nicht analytisch behandeln kann. In der klassischen Mechanik, Elektrodynamik und Quantenmechanik geht es in erster Linie um die Behandlung von gewöhnlichen oder partiellen Differentialgleichungen, während man in der statistischen Mechanik typischerweise mit der Lösung von Integralen und Reihensummen mit vielen Freiheitsgraden konfrontiert wird.

Zur Behandlung solcher analytisch nicht zugänglichen Probleme greift man häufig auf Standardmethoden der numerischen Mathematik zurück. Daneben sind aber auch spezialisierte Methoden entwickelt worden, die zumindest ursprünglich zur Behandlung physikalischer Fragestellungen entwickelt worden sind. Hier sind vor allem sog. Monte Carlo Methoden zu nennen, die eine stochastische Berechnung von Reihensummen und Integralen erlaubt. Die Einführung von verschiedenen Monte Carlo Algorithmen und die Anwendung auf physikalische Fragestellungen soll einen Schwerpunkt dieser Vorlesung bilden.

Eine weitere Methode, die im Rahmen dieser Vorlesung besprochen werden soll, ist die Methode der Molekulardynamik Simulation. Bei der Molekulardynamik Simulation werden die Bewegungsgleichungen der Teil-

chen numerisch integriert. Diese Methode ist im Gegensatz zur Monte Carlo Methode deterministisch.

Neben diesen Schwerpunkten sollen auch noch weitere numerische Methoden vor dem Hintergrund ihrer physikalischen Anwendungen vorgestellt werden. Die Themen lauten im Einzelnen:

- Erzeugung von Zufallszahlen
- Monte Carlo Simulation klassischer Vielteilchensysteme
- Einführung in die Molekulardynamik Simulation
- Monte Carlo Simulation von Quantensystemen
- Methoden zur Diagonalisierung großer Matrizen

1.2 Elemente der statistischen Mechanik des Gleichgewichts

In diesem Abschnitt sollen einige Grundtatsachen der statistischen Mechanik angesprochen werden, da die meisten Algorithmen in Bezug auf Anwendungen in der statistischen Physik diskutiert werden.

In dieser ersten Diskussion betrachten wir nur solche Systeme, die sich im thermischen Gleichgewicht befinden, da man für diese Systeme den Zusammenhang zwischen dem mikroskopischen Zuständen und dem Makrozustand formulieren kann. Ein Beispiel für ein klassisches (d.h. nicht quantenmechanisches) Vielteilchensystem sind Teilchen, die in einer Box eingesperrt und einem Wärmebad ausgesetzt sind. Bei einem solchen System wird der Mikrozustand durch den Ort und Impuls der Teilchen bestimmt. Bei einem quantenmechanischen System entspricht ein Mikrozustand einem Eigenzustand des Hamiltonoperators.

Nur in sehr wenigen Fällen ist es möglich, das reale System exakt zu behandeln. Daher begnügt man sich häufig mit Modellen, die die für die Fragestellung wesentlichen Eigenschaften des realen Systems beinhalten.

Die Beziehung zwischen der mikroskopischen Konfiguration des (Modell-) Systems und den makroskopischen Größen wird durch das folgende fundamentale Ergebnis der statistischen Mechanik hergestellt:

Es sei E_α die Energie des Mikrozustandes α. Wenn das System nun mit einem Wärmebad der Temperatur T in Kontakt steht, ist die Wahrscheinlichkeit P_α, dass das System sich im Zustand α befindet, proportional zu

$$c^{-E_\alpha/k_BT} = e^{-\beta E_\alpha}$$

mit der Boltzmannkonstanten $k_B = 1.38 \times 10^{-23}\, J/k$.

Die Proportionalitätskonstante muss natürlich so gewählt sein, dass die Wahrscheinlichkeitsverteilung normiert ist. Es sei nun

$$P_\alpha = \frac{1}{Z}\, e^{-\beta E_\alpha}$$

mit der Normierungskonstanten $Z = \sum_\alpha e^{-\beta E_\alpha}$, wobei die Summation über alle möglichen Mikrozustände geht (bei kontinuierlichen Freiheitsgraden muss man die entsprechenden Integrale auswerten). Die Normierungskonstante wird als *Zustandssumme* bezeichnet. Wir können mit dieser Verteilung beliebige Mittelwerte $\langle X \rangle$ von Messgrößen X berechnen, wenn wir

- alle Mikrozustände α und die zugehörigen Energien E_α angeben können.
- den Wert (X_α) einer Messgröße X in einem beliebigen Mikrozustand α kennen.
- die Summe

  $$\langle X \rangle = \sum_\alpha X_\alpha P_\alpha = \frac{1}{Z} \sum_\alpha X_\alpha\, e^{-\beta E_\alpha}$$

  berechnen können.

Die Berechnung der Zustandssumme, die im Allgemeinen sehr schwierig ist, erlaubt uns die Berechnung von thermodynamischen Größen, also die Verknüpfung von thermodynamischen und mikroskopischen Eigenschaften des Systems.

Die thermodynamischen Eigenschaften des Systems können auch von verschiedenen Zwangskräften abhängen, die auf das System wirken. Beispiele für solche Zwangskräfte sind ein äußeres Magnetfeld oder einfach eine Box, in die die Teilchen eingesperrt sind. Die Zwangskräfte müssen natürlich auch in die mikroskopische Beschreibung einfließen. So berücksichtigt man die Beschränkung der Teilchen auf ein vorgegebenes Volumen dadurch, dass man die potentielle Energie $U_{pot}(\underline{x}_t) = 0$ innerhalb und $U_{pot}(\underline{x}_i) = \infty$ außerhalb der Box wählt, wobei durch $\underline{x}_i$ die Position des $i$ten Teilchen bezeichnet wird.

Wir wollen nun konkret den Zusammenhang zu thermodynamischen Größen herstellen. Man sieht dabei, dass das funktionelle Verhalten von Z eine entscheidende Rolle spielt. Wir betrachten zunächst die mittlere Energie $U = \langle E \rangle$ eines Systems im thermischen Gleichgewicht:

$$U = \langle E \rangle = \frac{1}{Z} \sum E_\alpha \, e^{-\beta E_\alpha}$$

Wenn man nun die Zustandssumme kennt, kann man diesen Ausdruck einfach berechnen, denn

$$U = -\left(\frac{\partial \log z}{\partial \beta} \right)_{\{V\}}$$

führt auf den obigen Mittelwert. Der Index $\{V\}$ soll andeuten, dass die Zwangskräfte, die auf das System ausgeübt werden, unverändert gelassen werden.

Auch die spezifische Wärme können wir leicht berechnen, da gilt

$$C_{\{V\}} \equiv \left(\frac{\partial U}{\partial T} \right)_{\{V\}} = k_B \beta^2 \left(\frac{\partial^2 \log Z}{\partial \beta} \right)_{\{V\}} .$$

Man kann ebenfalls zeigen, dass die freie Energie

$$F = U - TS$$

die einfache Form

$$F = -\frac{1}{\beta} \log\, Z$$

hat.

1.3 Numerische Berechnung von thermodynamischen Mittelwerten

Die direkte Berechnung von thermodynamischen Mittelwerten erfordert die Bestimmung der Zustandssumme, was für realistische Systeme wegen der hohen Zahl der Freiheitsgrade unmöglich ist.

Eine Möglichkeit dieses Problem zu umgehen, besteht darin, Konfigurationen zufällig zu generieren und somit einen **Schätzwert** für die gesuchte Größe zu berechnen:

$$x_B = \frac{\sum_{k=1}^{B} x_k\, e^{-\beta E_k}}{\sum_{k=1}^{B} e^{-\beta E_k}} \,.$$

Diese einfache Methode liefert korrekte Werte für die thermischen Mittelwerte unter der Voraussetzung, dass man eine genügend große Zahl von Konfigurationen generiert. Letzteres ist jedoch insbesondere bei niedrigen Energien schwierig, da die Gewichte der Konfigurationen extrem unterschiedlich sind. Es ist zudem sehr unwahrscheinlich, dass man Konfigurationen zufällig generiert, die ein großes Gewicht in der Zustandssumme besitzen.

Abhilfe schafft das sogenannte "Importance Sampling", die Grundlage aller *Monte Carlo*-Algorithmen. Der Begriff des Importance Samplings beinhaltet, dass die Wahrscheinlichkeit eine Konfiguration zu generieren, **automatisch** dem Gewicht in der Zustandssumme entspricht. Dies wird dadurch realisiert, dass die Konfigurationen nicht unabhängig generiert werden, sondern auseinander hervorgehen. Die Übergangswahrscheinlichkeiten

zwischen zwei Konfigurationen sind dabei so gewählt, dass die Konfigurationen tatsächlich mit ihrem Gewicht in der Zustandssumme erzeugt werden.

Kapitel 2

Erzeugung von (Pseudo-)Zufallszahlen

In der Einleitung wurde deutlich, dass man bei der Simulation von Vielteilchensystemen die Zustandssumme und thermodynamischen Mittelwerte dadurch abschätzt, dass man in zufälliger Art und Weise Konfigurationen des Systems erzeugt. Dies führt uns direkt auf das Problem der Generierung von (Pseudo-)Zufallszahlen auf dem Computer.

2.1 Echte- und Pseudozufallszahlen

Zufallszahlen werden dadurch charakterisiert, dass ihr Wert nicht vorhergesagt werden kann. Präziser: Wenn wir eine Sequenz von Zufallszahlen generieren, ist die Wahrscheinlichkeitsverteilung der neuen Zufallszahl unabhängig von dem Wert der bisherigen Sequenz. *Bsp.:* Beim Roulette ist die Wahrscheinlichkeit, dass die "18" fällt, **unabhängig** von den bisher gezogenen Zahlen.

Echte Zufallszahlen können nur experimentell generiert werden. Als Beispiel sei hier der radioaktive Zerfall genannt, der als quantenmechanischer Prozess **echt** zufällig ist. Eine Möglichkeit eine Sequenz von Zufallszahlen zu erzeugen, besteht darin, die Anzahl der emittierten α-Teilchen in einem vorgebenenen Zeitintervall zu messen und der Sequenz eine 0 hinzu-

zufügen, wenn eine gerade Zahl von α-Teilchen detektiert wird und eine 1 bei einer ungeraden Zahl. Die so erzeugte Sequenz von Nullen und Einsen ist dann eine echte Zufallssequenz. Wenn man sie zur Nutzung in Computerprogrammen heranziehen will, muss man aber noch berücksichtigen, dass die Wahrscheinlichkeiten für "0" und "1" unter Umständen nicht gleich sind.

In der Praxis ist der Einsatz von echten Zufallszahlen jedoch zu umständlich. Man benutzt vielmehr Algorithmen zur Generierung von Pseudo-Zufallszahlen, die zwar deterministisch erzeugt werden, aber trotzdem viele Eigenschaften echter Zufallszahlen besitzen.

Pseudo-Zufallszahlen werden meist durch Operationen auf Integer-Zahlen generiert. Die Integer-Zahlen werden typischerweise aus der Menge $\{0, 1, 2, \ldots, m-1\}$ erzeugt und dann durch Division durch m auf das Intervall $[0, 1[$ abgebildet. Die einfachste Methode zur Generierung von Zufallszahlen i_n ist:

$$i_n = f(i_{n-1}) \ , \tag{2.1}$$

d.h. die n-te "Zufallszahl" ist einfach eine Funktion der $(n-1)$-ten "Zufallszahl".

Anforderungen an $f(i_{n-1})$:

- $f(i)$ sollte nur Integer-Operationen enthalten $(+, -, \cdot, \div, XOR, \ldots)$.
- die Abbildung sollte möglichst Zahlenfolgen generieren, die die gleichen statistischen Eigenschaften (in Bezug auf die jeweilige Anwendung) wie echte Zufallszahlen haben.

Man initialisiert den Zufallszahlengenerator mit dem sog. "**seed**", d.h. i_0 wird vorgegeben. Durch die Wahl von $f(i)$ und i_0 wird die Sequenz vollständig bestimmt, was man für Testzwecke ausnutzen kann.

Eine wichtige Eigenschaft **aller** Zufallszahlengeneratoren ist, dass sie Zyklen erzeugen, die die Maximallänge m haben. Wenn bei einem Zufallszahlengenerator ein Zyklus der Länge k auftritt, bedeutet das, dass $i_n = i_{n-k}$ für ein beliebiges $n > n_0$ gilt.

Wir suchen natürlich solche Zufallszahlengeneratoren mit möglichst großer Zykluslänge k. Die Zykluslänge lässt sich durch Funktionen

$$i_n = f(i_{n-1}, i_{n-2}, \cdots, i_{n-l}) \ , \tag{2.2}$$

die von mehreren Zufallszahlen abhängen, vergrößern: im günstigsten Fall auf die Länge m^l, was der Zahl der möglichen l-Tupel entspricht. Zur Initialisierung benötigt man dann die Startwerte $i_0, i, , \ldots, i_{l-1}$.

2.2 Lineare-Kongruenz-Generatoren

Lineare-Kongruenz-Generatoren verwenden die Relation:

$$i_n = (a\, i_{n-1} + c) \mod m \tag{2.3}$$

wobei $p \mod q$ den Rest ergibt, wenn p durch q geteilt wird (Bsp.: $9 \mod 4 = 1$). Das Ergebnis dieser Operation liegt also zwischen 0 und $m - 1$.

Eine einfache Methode besteht darin, die Modulo-Operation wegzulassen und den Overflow zu ignorieren (entspricht $m_{max} = 2^{32}$) [1]. Dies führt aber dazu, dass die niedrigen Bits nicht sehr zufällig sind: das $k$te Bit hat höchstens die Periode 2^k. Das heißt insbesondere, dass alle Zahlen entweder immer gerade oder immer ungerade sind oder zwischen gerade und ungerade alternieren.

Die Prozedur sollte auch keinen Overflow produzieren, da sonst nicht gewährleistet ist, dass man die volle Periode erhält. Daher muss m wesentlich kleiner als m_{max} sein.

Die richtige Wahl von a, c, m ist subtil. Knuth [1] hat folgende Werte vorgeschlagen:

[1] Es ist heute Standard, dass Integer-Variablen auf 32 Bits definiert sind. Wenn man `signed int` benutzt, bestimmt das führende Bit das Vorzeichen, so dass $2^{31} - 1$ die größte darstellbare Zahl ist. Der Maximalwert eines `unsigned int` ist dementsprechend $2^{32} - 1$.

$$\begin{aligned} \text{(unsigned int)} \quad & a = 2416, c = 374441, m = 1771875 \\ \text{(signed int)} \quad & a = 9301, c = 49297, m = 233280 \end{aligned}$$

Die Zykluslänge des Generators lässt sich durch den *Trick von Schrage* vergrößern, der den Overflow verhindert. Es sei q der Quotient und r der Rest bei der Division, genauer

$$q = \lfloor m/a \rfloor \, ; \, r = m \mod a \tag{2.4}$$

($\lfloor \ldots \rfloor$ ist die größte ganze Zahl, die kleiner oder gleich m/a ist.)

Wir können mit q, r die Zahl m folgendermaßen zerlegen:

$$m = aq + r \tag{2.5}$$

Es gilt dann:

$$\begin{aligned} a\, i_n \mod m &= (a\, i_n - \lfloor i_n/q \rfloor m) \mod m \\ &= (a\, i_n - \lfloor i_n/q \rfloor (a\, q + r)) \mod m \\ &= (a\, (i_n - \lfloor i_n/q \rfloor q) - \lfloor i_n/q \rfloor r) \mod m \end{aligned}$$

Der erste Term in der Klammer lässt sich folgendermaßen abschätzen:

$$a\, (i_n - \lfloor i_n/q \rfloor q) = a \underbrace{(i_n \mod q)}_{\in \{0,1,\ldots,q-1\}} < aq < m$$

Falls zusätzlich $r < q$ gilt, erhält man für den zweiten Term:

$$\lfloor i_n/q \rfloor \, r \leq i_n(r/q) < i_n < m$$

Damit ergibt die Operation $a(i_n \mod q) - \lfloor i_n/q \rfloor r$ eine Zahl aus $[-m, +m]$. Es wird also der Overflow vermieden, allerdings muss man auf signed Integer zurückgreifen.

Insgesamt müssen wir also die Operation

$$a\, i_n \mod m \tag{2.6}$$

durch:

$$\begin{aligned} i_{n+1} &= a\,(i_n \mod q) - \lfloor i_n/q \rfloor r \\ if(i_{n+1} < 0) & \\ & i_{n+1} = i_{n+1} + m \end{aligned}$$

ersetzen. Eine gute Wahl (Lewis '69), die gründlich getestet wurde und breit eingesetzt wird, ist:

$$\begin{aligned} a &= 16807 \\ m &= 2^{31} - 1 = 2147483647 \\ c &= 0 \end{aligned}$$

Damit ist $q = 127773$ und $r = 2896$ und die Bedingung $q > r$ erfüllt. Die Wahl $c = 0$, die von Knuth für alle Zufallszahlengeneratoren mit $m = 2^n \pm 1$ empfohlen wird, verbietet den Seed $i_0 = 0$. Damit ist die maximale Zykluslänge $m - 1$. Die Zykluslänge des Generators ist $\sim 2 \cdot 10^9$. Die erzeugten Zufallszahlen sind allerdings schon für Sequenzlängen $> 10^7$ nicht mehr statistisch unabhängig. Diese effektive Länge reicht aus für kleine Monte Carlo Simulationen.

Neben der relativ kleinen Zykluslänge haben die obigen Generatoren noch ein weiteres Problem: Wenn man k-komponentige Vektoren erzeugt, deren Elemente die generierten Zufallszahlen sind, füllen diese den Raum nicht homogen aus. Die Vektoren liegen vielmehr auf $k-1$ dimensionalen Hyperebenen. Die Maximalzahl der Hyperebenen ist auf $m^{1/k}$ beschränkt.

Bei der Benutzung von linear-kongruenten Generatoren ist zudem zu beachten, dass man die hohen Bits benutzt, da sie weniger Korrelationen aufweisen. Benötigt man z.B. eine ganze Zufallszahl zwischen 1 und 10, sollte man auf folgende Methode zurückgreifen

$$c = 1 + \lfloor (10.0 \star ran()/(m_{max} + 1.0)) \rfloor \ ,$$

wobei $ran()$ einen beliebigen Zufallszahlengenerator bezeichnet, der Zahlen $\in \{0, \ldots, m_{max}\}$ erzeugt. Vermeiden sollte man dagegen Funktionen wie

$$j = 1 + (ran() \mod 10) \ ,$$

da dann nur auf die niedrigsten Bits zugegriffen wird.

2.3 Shuffling-Schema

Das Problem der Korrelationen zwischen aufeinander folgenden Zufallszahlen kann man durch das Shuffling Schema von Bays und Durham beheben.

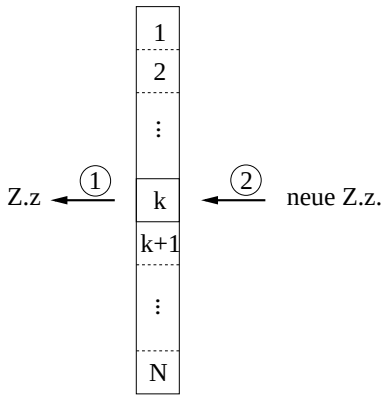


Abbildung 2.1: Shuffling: Man wählt den $k$ten Platz eines Feldes j zufällig aus. Die Zufallszahlen $j[k]$ wird ausgegeben und dann mit einer neuen Zufallszahlen belegt.

Der Zufallszahlengenerator hat folgende Struktur:

Initialisierung

- Belegung eines Feldes mit N Elementen z.B. durch einen linear-kongruenten Zufallszahlengenerator.

- Generierung eine weiteren Zufallszahl iy

Iteration

(1) Durch die neue Zufallszahl iy wird ein Element k des Feldes j zufällig bestimmt.

(2) Element k wird ausgegeben und durch iy ersetzt.

(3) Generierung einer neuen Zufallszahl $iy \rightarrow zu(1)$.

Eine Implementierung dieses Zufallszahlengenerators findet sich in der Numerical Recipes Bibliothek (die Funktion $ran1()$). Der dort realisierte Generator hat eine *nutzbare* Periode von 10^8. Durch geschickte Kombination zweier oder mehrere Zufallszahlengeneratoren lässt sich die Zykluslänge weiter vergrößern (realisiert in Numerical Recipes Bibliothek als Funktion $ran2()$).

2.4 Weitere Generatoren

2.4.1 Shift-Register Zufallszahlengeneratoren

Diese Generatoren, die von Tausworthe vorgeschlagen wurden, basieren auf Bit-Shift-Operationen: Es sei $R^s(L^t)$ ein rechts-(links)-Shift um $s(t)$ Plätze.
Bsp. :

$$R^4\ 1011\overbrace{1010}^{s=4} \rightarrow 00001011$$

Die Shift-Operation wird mit der XOR-Operation kombiniert.

$$\begin{aligned} i'_{n-1} &= i_{n-1} \ \oplus\ R^s i_{n-1} \\ i_n &= i'_{n-1} \ \oplus\ L^t i'_{n-1} \end{aligned} \tag{2.7}$$

Bei 31-Bit Generatoren wählt man z.B. $s = 18, t = 13$ oder $s = 28, t = 3$. Bei 32-Bit Generatoren hat sich $s = 15, t = 17$ als günstige Wahl erwiesen. Die Operationen, die dem Zufallszahlengenerator zu Grunde liegen, sind in Abbildung 2.2 dargestellt. Dieser Generator ist zusammen mit einem Shuffling Schema eine gute Wahl.

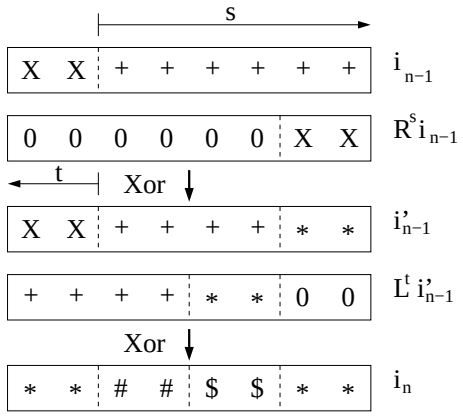


Abbildung 2.2: Illustration der Shift-Operationen im Shift-Register-RNG. Man sieht, dass alle Bits der neuen Zufallszahl manipuliert werden. Um dies zu gewährleisten, muss $s + t = 32$ gelten.

2.5 Lagged-Fibonacci-Generatoren

Diese Generatoren basieren auf der Operation

$$i_n = (i_{n-r} \underbrace{\circ}_{+,-,\times,\oplus} i_{n-s}) \mod m \tag{2.8}$$

Lewis und Payne haben die Operation XOR vorgeschlagen, die die Modulo-Operation überflüssig macht.

Eine häufige Wahl für r, s ist $r = 147$, $s = 1279$. Eine andere Variante, die ebenfalls sehr häufig eingesetzt wird, ist der `r250` -RNG von Kirkpatrick & Stoll bei dem $r = 147$, $s = 250$ gesetzt sind. Die Periode des `r250`-RNG ist $2^{250} - 1$ und somit auch ausreichend für großskalige Simulationen.

Für **kleinere** Werte von r, s sind Addition und Multiplikation besser. Beispiel ist der Generator $ran3()$ in Numerical Recipes. Er benutzt die Operation "$-$" und $s = 55, r = 24$.

Die Benutzung beider Generatoren setzt voraus, dass man ein Feld mit $w = Max(r, s)$ Elementen anlegt (durch linear kongruenten Generator).

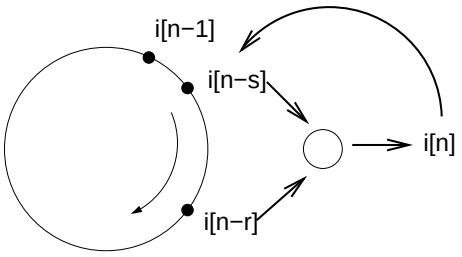


Abbildung 2.3: Struktur des Liste von gespeicherten Zufallszahlen im Circular-FIFO-Buffer.

Die neu generierte Zufallszahl i_n ersetzt dann einfach das Element i_{n-w} im Array.

Bemerkung: Die Lagged-Fibonacci Generatoren sind sehr schnell und haben große Zyklen. Der Nachteil ist, dass die Theorie nicht so gut ausgearbeitet ist.

2.6 Erzeugung nicht-gleichverteilter Zufallszahlen

2.6.1 Invertierungsmethode

Ziel ist es, mit Hilfe eines Zufallszahlengenerators $ran()$, der gleichverteilte Zufallszahlen zwischen $]0,1[$ generiert, Zufallszahlen zu erzeugen, die gemäß $p(z)$ verteilt sind.
Es sei nun

$$P(z) = \text{Prob}\,(z' < z) = \int_{-\infty}^{z} p(z')dz' \tag{2.9}$$

Die Verteilung $p(z)$ zu samplen bedeutet, dass man die gleichverteilten Zufallszahlen x_i als Funktionswerte von $P(z_i)$ an der Stelle z_i auffasst. Ausgegeben wird dann der Wert z_i, den wir durch Invertierung von $P(z)$ erhalten. Durch dieses Verfahren generieren wir Zufallszahlen z_i,

die gemäß $p(z)$ verteil sind.

Bsp.:
(a) Exponentialverteilung:

$$\begin{aligned} p(z) &= \lambda e^{-\lambda z}; \quad z \in [0, \infty[\\ \rightarrow P(z) &= \int_0^z \lambda e^{-\lambda z'}\, dz' = e^{-\lambda z'}|_0^z = 1 - e^{-\lambda z} \\ \Rightarrow z &= \frac{-1}{\lambda} \ln(1 - P(z)) = \frac{-1}{\lambda} \ln(1 - ran()) \end{aligned} \tag{2.10}$$

Die Abbildung der gleichverteilten Zufallszahlen $ran()$ auf exponentiell verteilte Zufallszahlen erfolgt also durch die Funktion $-\ln(ran())/\lambda^2$.
(b) Die Lorentz-Verteilung:

$$\begin{aligned} p(z) &= \frac{1}{\pi} \frac{\Gamma}{\Gamma^2 + z^2} \\ \rightsquigarrow P(z) &= \frac{1}{\pi} \int_{-\infty}^{z} dz' \frac{\Gamma}{\Gamma^2 + (z')^2} = \frac{1}{\pi} \int_{-\infty}^{z} \frac{dz'}{\Gamma} \frac{1}{1 + (z'/\Gamma)^2} \\ &= \frac{1}{\pi} \int_{-\infty}^{z/\Gamma} dn \frac{1}{1 + n^2} = \frac{1}{\pi} \left[\arctan[z/\Gamma] + \frac{\pi}{2} \right] \\ &= \frac{1}{2} + \frac{1}{\pi} \arctan[z/\Gamma] = ran() \\ z &= \Gamma \tan \left[\pi (ran() - \frac{1}{2}) \right] \end{aligned} \tag{2.11}$$

2.6.2 Rejektionsmethode

Die Invertierungsmethode kann man natürlich nur auf Verteilungen anwenden, die invertierbar sind. Allgemeiner anwendbar ist die Rejektions-

[2] Da durch `ran()` gleichverteilte Zufallszahlen $\in]0, 1[$ erzeugt werden, sind die Funktionen $\ln(1 - ran())$ und $\ln(ran())$ äquivalent.

methode, die man am leichtesten für Verteilungen anwenden kann, die auf kompakten Intervallen definiert sind.

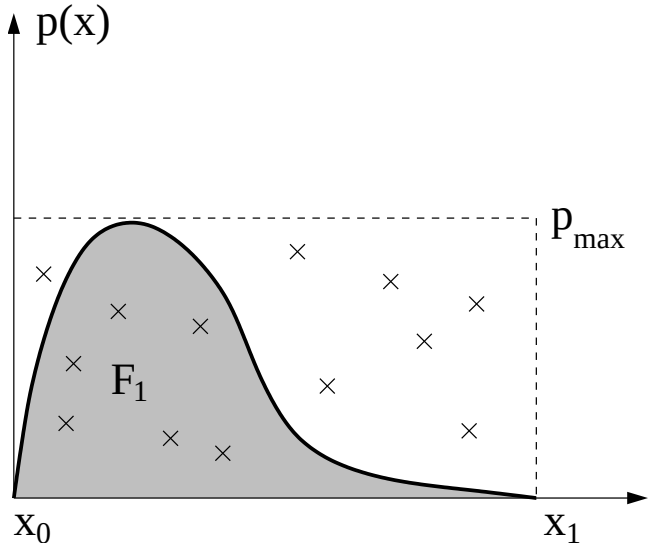


Abbildung 2.4: Rejektionsmethode: Man generiert zwei zweidimensionale Zufallsvektoren, die das oben skizzierte Rechteck gleichmäßig abdecken. Die x-Komponente dieses Vektors wird akzeptiert, wenn die zugehörige y-Komponente unterhalb des Funktionswertes $p(x)$ liegt.

Der Algorithmus basiert auf folgender Idee:

Man erzeugt ein zufälliges Zahlenpaar (xy) von gleichverteilten Zufallszahlen. Die x-Koordinate sei zwischen $[x_0, x_1]$ verteilt und die y-Koordinate zwischen $[0, p_{\max}]$. Es werden dann nur solche x ausgegeben für die $y < p(x)$ gilt.

algorithm rejection
begin
notfound = FALSE;
while (found = FALSE)**do**

```
        u1 = ran();
        z = z0 + (z1-z0)*u1;
        u2 = ran();
        y = pmax * u2;
        if (y <= p(z)) found = TRUE;
    end while
    return(z);
end
```

Die Methode ist offensichtlich korrekt, denn

$$\left.\begin{aligned} p_{\text{gen}}(x) &= \frac{dx}{x_1 - x_0} \\ p_{\text{accept}}(x) &= \frac{p(x)}{p_{\max}} \end{aligned}\right\} \quad \Rightarrow \quad p(x)\,dx = p_{\text{gen}}(x)\,p_{\text{accept}}(x) \tag{2.12}$$

$$= \quad \frac{p(x)\,dx}{p_{\max}(x_1 - x_0)} \propto p(x)\,dx \tag{2.13}$$

Die Effektivität der Methode hängt vom Verhältnis von $F_1/p_{\max}(x_1 - x_0)$ ab (mit $F_1 = \int_{x_0}^{x_1} dx p(x)$). Für eine auf dem Intervall $[-6, 6]$ definierte Gaußverteilung

$$p(x) = \frac{1}{\sqrt{2\pi}}\, e^{-x^2/2}$$

ist das Verhältnis

$$\frac{2p_{\max}(x_1 - x_0)}{\int_{x_0}^{x_1} p(x)\,dx} = 9.57$$

Man muss also im Mittel 9.57 gleichverteilte Zufallszahlen generieren, um eine gaußverteilte Zufallszahl zu generieren.

Eine Verbesserung der Effizienz ist durch die Kombination von Invertierungs- und Rejektionsmethode möglich. Dazu benutzt man eine zweite Verteilungsfunktion $g(x)$, die invertierbar ist und die Eigenschaft hat $p(x) < g(x) \quad \forall x$ besitzt.

Graphisch

1. Schritt: Die x-Koordinate wird aus der Invertierung von $g(x)$ bestimmt.
2. Schritt: Die x-Koordinate wird mit der Wahrscheinlichkeit $p(x)/g(x)$ angenommen.
Bedingung für die Anwendbarkeit der Methode ist, dass $g(x) > p(x)\ \forall\, x \in \mathbb{R}$ gilt. Die Effizienz wird durch das Verhältnis von $\int_{-\infty}^{\infty} dx g(x) \,/\, \int_{-\infty}^{\infty} dx p(x)$ bestimmt.

2.6.3 Generierung von gaußverteilten Zufallszahlen

Für die Gauß-Verteilung:

$$p_g(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-m)^2}{2\sigma^2}\right)$$

kann man bekanntermaßen keine Stammfunktion angeben, so dass die Inversionsmethode nicht anwendbar ist. Dennoch lassen sich gaußverteilte Zufallszahlen effizient generieren.

Die **einfachste Methode** beruht auf dem **Zentralen Grenzwertsatz**: Es seien x_i, $i = 1, 2, \ldots, n$ unabhängige Zufallsvariablen, die durch die Wahrscheinlichkeitsdichten $f_i(x)$ beschrieben werden. Die $f_i(x)$ haben die Mittelwerte μ_i und die Varianz σ_i^2. Dann hat die Zufallsvariable

$$z = \frac{\sum_{i=1}^{n} x_i}{n}$$

die folgenden Eigenschaften

(i) Der Erwartungswert von z ist $E(z) = \left(\sum_{i=1}^{n} \mu_i\right)/n$

(ii) Die Varianz der Verteilung ist $V(z) = \left(\sum_{i=1}^{n} \sigma_i^2\right)/n^2$

(iii) Im Limes $n \to \infty$ ist z gaußverteilt

Wenn alle x_i gleichverteilt sind zwischen $[0, 1]$, dann ist $z = \sum_{i=1}^{12} x_i$ annähernd gaußverteilt. Der Mittelwert der x_i ist $\frac{1}{2}$, so dass $E(z') = 12 \cdot \frac{1}{2} = 6$. Entsprechend ist die Varianz der Einzelwerte $\sigma_i = \frac{1}{12}$, so dass $V(z') = 1$. Normalverteilte $(m = 0, \sigma^2 = 1)$ Zufallszahlen lassen sich also durch

$$z = \sum_{i=1}^{12} x_i - 6$$

generieren.

Wesentlich effizienter ist die **Box-Müller-Methode**, die eine exakte Methode zur Generierung von gaußverteilten Zufallszahlen ist.

Zur Begründung der Methode betrachten wir zunächst Verteilungsfunktionen von mehreren Variablen: Es seien $x_1, x_2, \ldots, x_n$ gemäß $p(x_1, \ldots, x_n)$ verteilt. Ferner seien $y_1, y_2, \ldots, y_n$ Funktionen der Variablen $x_1, \ldots, x_n$. Durch die Transformation der Variablen muss die Wahrscheinlichkeit erhalten bleiben, was in einer Dimension der Bedingung $p(x)dx = p(y)dy$ entspricht. Die entsprechende Verallgemeinerung führt in höheren Dimension auf:

$$p(y_1, \ldots, y_n)\, d_y, \ldots, dy_n = p(x_1, \ldots, x_n) \left| \frac{\partial(x_1, \ldots, x_n)}{\partial(y_1, \ldots, y_n)} \right| dy, \ldots, dy_n \ .$$

Wir betrachten nun die Variablen:

$$\begin{aligned} y_1 &= \sqrt{-2 \ln x_2}\, \cos(2\pi x_1) \\ y_2 &= \sqrt{-2 \ln x_2}\, \sin(2\pi x_1) \end{aligned}$$

$$\Rightarrow \tan(2\pi x_1) = \frac{y_2}{y_1} \quad \text{bzw.} \quad x_1 = \frac{1}{2\pi} \arctan(y_2/y_1)$$

und

$$\begin{aligned} y_1^2 + y_2^2 &= -2 \ln x_2 (\cos^2(2\pi x_1) + \sin^2(2\pi x_1)) \\ \text{bzw.} \quad x_2 &= \exp(-\frac{1}{2}\,(y_1^2 + y_2^2)) \ , \end{aligned}$$

wobei die Zufallsvariablen x_1, x_2 gleichverteilt im Intervall $]0,1[$ sind. Die Jakobideterminante lautet

$$\frac{\partial(x_1,x_2)}{\partial(y_1,y_2)} = \begin{vmatrix} \partial x_1/\partial y_1 & \partial x_1/\partial y_2 \\ \partial x_2/\partial y_1 & \partial x_2/\partial y_2 \end{vmatrix} \tag{2.14}$$

$$= \begin{vmatrix} -\frac{1}{2\pi}\frac{y_2}{y_1^2+y_2^2} & \frac{1}{2\pi} - \frac{y_1}{y_1^2+y_2^2} \\ -y_1 x_2 & -y_2 x_2 \end{vmatrix} \tag{2.15}$$

$$= \frac{1}{2\pi}\, x_2 = \left[\frac{1}{\sqrt{2\pi}}\, e^{-y_1^2/2}\right] \times \left[\frac{1}{\sqrt{2\pi}}\, e^{-y_2^2/2}\right] , \tag{2.16}$$

d.h. die Zufallsvariablen y_1, y_2 sind tatsächlich unabhängig voneinander gaußverteilt[3].

2.7 Statistische Tests für RNG

2.7.1 Test auf Gleichförmigkeit der Zufallszahlensequenz

Man erzeugt Zufallszahlen auf dem Intervall $]0,1[$. Die Zufallszahl wird dann durch die Operation

$$k = \lfloor ran() \cdot \nu \rfloor$$

auf ν-Werte abgebildet. Die Werte $k \in \{0, 1, \ldots, \nu - 1\}$ müssen im Mittel mit der gleichen Häufigkeit $n_k = n$ auftreten. Die Frage, ob die Abweichung der tatsächlichen Häufigkeit N_k mit der der Wert k in einer Sequenz auftritt vom Erwartungswert $\langle N_k \rangle = n$ systematisch oder von statistischer Natur sind, lässt sich durch den sog. χ^2-Test entscheiden.

Beim χ^2-Test misst man die relative quadratische Abweichung der N_k von dem jeweiligen Erwartungswert:

$$\chi^2 = \sum_{k=0}^{\nu-1} \frac{(N_k - n_k)^2}{n_k} , \tag{2.17}$$

[3]Da x_1, x_2 gleichverteilte Zufallszahlen $\in]0,1[$ sind, gilt $p(x1, x2) = p(x_1)p(x_2) = 1$. Die statistische Unabhängigkeit von y_1, y_2, die sich durch $p(y_1, y_2) = p(y_1)p(y_2)$ ausdrückt, ist durch die geschickte Wahl der Transformation bedingt und nicht einfach Resultat der statistischen Unabhängigkeit der x_i.

wobei in unserem Fall $n_k = n = N/\nu$ (mit $N = \sum_{k=0}^{\nu-1} N_k$) gilt.

Ein großer Wert von χ^2 signalisiert, dass die Hypothese der Gleichförmigkeit nicht erfüllt ist. Man kann dies weiter quantifizieren, wenn die Summanden in (2.17) Quadrate normalverteilter Zufallszahlen $(N_k - n_k)$ sind (für große Werte von ν ist die Annahme erfüllt). Für diesen Fall gibt die Funktion $Q(\chi^2|\nu)$ die Wahrscheinlichkeit an, dass χ^2 bei einer wahren Hypothese größer ist, als der gefundene Wert. Es gilt:

$$Q(\chi^2|\nu) = Q\left(\frac{\nu}{2}, \frac{\chi^2}{2}\right) \tag{2.18}$$

mit

$$\begin{aligned} Q(\alpha, x) &= \frac{1}{\Gamma(\alpha)} \int_x^\infty e^{-t} t^{\alpha-1} dt \qquad (\alpha > 0) \qquad \text{und} \\ \Gamma(\alpha) &= \int_0^\infty t^{\alpha-1} e^{-t} dt \end{aligned} \tag{2.19}$$

Es gilt $Q(\alpha, 0) = 1$ und $Q(\alpha, \infty) = 0$. Die Funktion $Q(\alpha, x)$ ist in der Numerical Recipes Bibliothek als `gammaq()` aufgeführt.

2.7.2 Serielle Korrelationen

Neben der Gleichförmigkeit ist die Abwesenheit von seriellen Korrelationen eine wichtige Eigenschaft von Pseudo-Zufallszahlen. Diese kann man dadurch bestimmen, dass man d-Tupel von Zufallszahlen generiert (z.B. $d = 2$: Fasse Paare von Zufallszahlen (x_{2i}, x_{2i+1}) zusammen). Jeder Koordinate des d-Tupels ordnet man dann wie beim Gleichförmigkeitstest eines von ν-Intervallen zu, so dass die d-Tupel auf ν^d Boxen verteilt werden. In Abwesenheit von seriellen Korrelationen sollten die erzeugten d-Tupel gleichmäßig auf die Boxen verteilt sein. Wir erhalten also:

$$\chi^2 = \sum_{k=1}^{\nu^d} \frac{(N_k - n)^2}{n} \qquad \text{mit} \qquad n = \nu^{-d} \cdot N \ , \tag{2.20}$$

wobei N die Zahl der zufällig generierten d-Tupel bezeichnet.

2.7.3 Gap-Test

Der Gap-Test ist ebenfalls ein Test auf Gleichförmigkeit. Bei diesem Test misst man, wieviele Zufallszahlen generiert werden müssen, bis eine neu generierte Zufallszahl zum zweiten Mal in das Intervall $[\alpha, \beta]$ fällt.
Wie üblich kann man das Ergebnis der Messung mit der Vorhersage

$$P(j) = \overline{p}p^j \tag{2.21}$$

vergleichen, wobei $p = 1 - \overline{p} = \frac{\nu-1}{\nu}$ ist und ν die Anzahl der Intervalle bezeichnet.

2.7.4 Der Maximum-von t-Test

Bei diesem Test generiert man ν Untersequenzen von Zufallszahlen x_i aus dem Intervall $x_i \in [0, 1[$. Die Untersequenzen haben die Länge t. Aus den Untersequenzen wird dann das jeweilige Maximum bestimmt.

Die theoretische Erwartung kann man leicht angeben, wenn man die Wahrscheinlichkeit $P(x)$ berechnet, dass das Maximum kleiner als ein vorgegebener Wert x ist. Wegen der Unabhängigkeit der Variablen gilt $P(x) = x^t$, da die Wahrscheinlichkeit für jede einzelne Zufallsvariable, einen Wert kleiner als x anzunehmen, x ist. Die korrespondierende Dichtefunktion ist dann

$$p(x) = tx^{t-1} \ . \tag{2.22}$$

Wir können die Gleichförmigkeit der Zufallszahlen durch die Einordnung der x_i in Teilintervalle und anschließenden χ^2-Test ermitteln. Die Diskretisierung der x_i ist aber überflüssig, wenn man den **Kolmogorov-Smirnov(KS)-Test** anwendet.

Beim KS-Test wird die kumulative Verteilung zum Vergleich zwischen theoretischer Erwartung und den erzeugten Daten herangezogen.

Die Signifikanz einer theoretischen Erwartung bestimmt sich dann aus dem Wert von $D_N = \max_{-\infty<x<\infty} |S_N(x) - P(x)|$, wobei $S_N(x)$ den Anteil der N Datenpunkte bezeichnet, die kleiner als x sind, und $P(x) =$

$\int_{-\infty}^{x'} dx'\, p(x')$ die kumulative Verteilung der theoretisch erwarteten Wahrscheinlichkeitsdichte darstellt. Die Signifikanz der Hypothese wird dann durch die Summe

$$Q_{KS}(\lambda) = 2\sum_{j=1}^{\infty}(-1)^{j-1}\, e^{-2j^2\lambda^2} \tag{2.23}$$

bestimmt, die die Eigenschaft

$$Q_{KS}(0) = 1 \qquad \text{und} \qquad Q_{KS}(\infty) = 0 \tag{2.24}$$

hat. Die Wahrscheinlichkeit, dass der Abstand D größer als der beobachtete Wert ist, ist gegeben durch

$$\text{Prob}(D > D_N) = Q_{KS}\left([\sqrt{N} + 0.12 + 0.11/\sqrt{N}]D_N\right)\ . \tag{2.25}$$

Die Funktion $Q_{KS}()$ findet sich unter `probks()` in der Numerical Recipes Bibliothek.
Weitere Tests, die in [1, 3] ausführlicher besprochen werden, sind

- **der Kollisionstest**: Es werden N Zufallszahlen erzeugt und in M Intervalle eingeteilt. Es gelte $N \ll M$. Ausgewertet wird die theoretische und empirische Wahrscheinlichkeit, dass zwei oder mehrere Zufallszahlen in das gleiche Intervall fallen (Test auf Gleichförmigkeit).

- **der Run-Test**: Analyse der Zahl der auf- und absteigenden Untersequenzen von Zufallszahlen der Länge $0 \leq i \leq l$ in der Sequenz von $x_1, \ldots, x_n$.

2.7.5 Bit-Level Tests

Die obigen Testverfahren haben die Eigenschaften des Wertes der Zufallszahlen getestet. Für einige Anwendungen ist allerdings auch die Generierung von möglichst zufälligen Bits wichtig (Bsp.: Simulation von einfachen stochastischen Cellular Automaten). In diesen Fällen müssen die

Eigenschaften der einzelnen Bits gesondert getestet werden. Man spricht dann von *Bit-Level Tests*.
Zwei dieser Tests sollen hier kurz vorgestellt werden:

d-Tupel Test

Beim d-Tupel Test werden die vom Zufallszahlengenerator erzeugten s Bits der Integer Variablen x_i benutzt. Wir stellen also die Integer-Zahl folgendermaßen dar:

$$\begin{aligned} I_1 &= b_{1,1} \dots b_{1s} \\ I_2 &= b_{2,1} \dots b_{2s} \\ \vdots &\quad \vdots \\ I_n &= b_{n,1} \dots b_{n,s} \end{aligned}$$

Jede dieser Binärsequenzen wird in Untersequenzen I_i' der Länge l aufgeteilt, die dann neue Binärsequenzen bilden. Dann fasst man d solcher Untersequenzen zu einer neuen Sequenz $\overline{I}$ in folgender Weise zusammen:

$$\begin{aligned} \overline{I}_1 &= b_{1,1} \dots b_{1,l}\, b_{2,1} \dots b_{2,l} \dots b_{d,1} \dots b_{d,l} \\ \overline{I}_2 &= b_{2,1} \dots b_{2,l} \dots\dots b_{d+1,1} \dots b_{d+2,2} \\ \vdots &\quad \vdots \\ \overline{I}_k &= b_{k,1} \dots b_{k,l} \dots b_{d+k,1} \dots b_{d+k,l} \end{aligned}$$

Die $\overline{I}_k$ sind aus der Menge $\overline{I}_k \in \{0, \dots, 2^{dl} - 1\}$. Es muss dann die Verteilung der generierten Werte von $\overline{I}_k$ analysiert werden (siehe Referenz [2] in Vattalainen et al).

Der Rang-Test

Aus den Zufallszahlen konstruiert man $(v \times w)$-Binärmatrizen, deren Rang analysiert wird. Die Verteilung für den Rang solcher zufälligen Binärmatrizen kann man berechnen und mit der empirischen Verteilung vergleichen.

2.7.6 Weitere Tests

(1) Spektraltests: Beschreiben die Eigenschaften der Hyperebenen von Linear-Kongruenz Generatoren (Diskussion in [1]).

(2) Visuelle Tests: Die visuellen Tests erlauben eine schnelle qualitative Analyse von Zufallszahlengeneratoren.
Beispiele:

(i) Man plottet Paare von Zufallszahlen und sucht nach geordneten Strukturen in der Abbildung.

(ii) Binärsequenzen können als Zustände des zweidimensionalen Ising-Modells bei der Temperatur $T = \infty$ verstanden werden. Die Eigenschaften können dann mit denen des korrespondierenden Ising Modells verglichen werden.

(i) Es werden n-Zufallszahlen erzeugt und die Abstände $d_{ij} = |x_i - x_j|$ zwischen allen Zahlen berechnet. Der Wert von d_{ij} wird in eine Graustufe übersetzt, so dass die d_{ij} ein zweidimensionales Muster erzeugen. Flächen mit gleicher Graustufe deuten auf Korrelationen hin. In gleicher Weise kann man auch den Gap-Test visualisieren.

Literatur zu diesem Kapitel

[1] D.E. Knuth, The Art of Computer Programming (3 ed.), Addison-Wesley, Longman (1998), Bd. 2, Kap. 3

[2] W.H. Press et al; Numerical Recipes in C, Cambridge University Press, Cambridge (1992), Kap. 7

[3] I. Vattalainen et al; Preprint, hep-lat/9304008

Kapitel 3

Der Random Walk

Der **Random Walk** (dt.: "Irrweg", RW) ist ein einfacher **stochastischer Prozeß**, der die zufällige Bewegung eines Teilchens beschreibt. Aus physikalischer Sicht ist der RW u.a. als Modell für einen Diffusionsprozess von Bedeutung. Den RW kann man im zeitlichen und räumlichen Kontinuum formulieren, aber auch als Gittermodell in diskreter Zeit. Letztere Variante eignet sich naturgemäß am besten für Computersimulationen. Beim diskreten RW in einer Dimension wird die Dynamik des Teilchens einfach dadurch realisiert, dass das Teilchen mit gleicher Wahrscheinlichkeit auf eines seiner beiden Nachbarplätze springt (siehe Abb. 3.1).

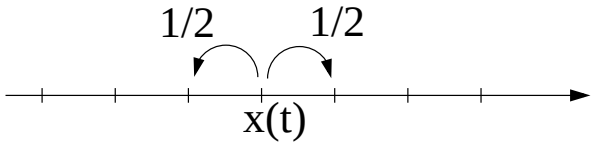


Abbildung 3.1: Dynamik des "Random Walkers": Zum Zeitpunkt t befinde sich der RWer am Platz $X(t)$. Daher wird der RWer nächsten Zeitschritt jeweils mit der Wahrscheinlichkeit $1/2$ auf dem Platz $X(t)+1$ oder $X(t)-1$ anzutreffen sein.

algorithm Random Walk in $d = 1$

```
begin
   pos = 0;
   for nsteps do
      if ran() < p then
         position = position +1;
         else position = position -1;
   end do
end
```

Der obige Algorithmus beschreibt einen eindimensionalen RWer in einer Dimension. Der Walker geht von der Startposition $x = 0$ mit der Wahrscheinlichkeit p in positive x-Richtung und entsprechend mit der Wahrscheinlichkeit $1 - p$ in negative x-Richtung.

Durch die stochastische Natur des RWs können sinnvolle Aussagen über Messgrößen nur im statistischen Mittel sinnvoll getroffen werden. Die Mittelung setzt aber die Kenntnis der die Größe $P(x,t)$ voraus, also der Wahrscheinlichkeit, dass der Walker zur Zeit t am Ort x zu finden ist.

Die Zeitentwicklung von $P(x,t)$ wird durch die sogenannte *Mastergleichung* bestimmt. Sie lautet in diskreter Zeit ganz allgemein:

$$P(x,t+1) = -\sum_{x'} w(x \to x',t)P(x,t) + \sum_{x'} w(x' \to x,t)P(x',t), \quad (3.1)$$

und stellt auf der rechten Seite die Bilanz des Wahrscheinlichkeitsflusses zwischen Übergängen, die von x wegführen, und Übergängen, die zu x hinführen, dar. Durch $w(x \to x',t)$ wird die Wahrscheinlichkeit bezeichnet zur Zeit t von der Konfiguration x zur Konfiguration x' üeberzugehen. Die Summation erstreckt sich über alle Zustände des jeweiligen stochastischen Prozesses. Speziell für den RW ergibt sich

$$w(x \to x',t) = p\delta_{x,x+1} + (1-p)\delta_{x,x-1} \quad (3.2)$$

und damit im speziellen Fall $p = 1/2$ die Mastergleichung[1]

$$P(x, t+1) = \frac{1}{2}\left(P(x+1, t) + P(x-1, t)\right) \ . \tag{3.3}$$

Offenbar ist Position $X(t)$ des Walkers nach t Schritten eine stochastische Variable, die eine Summe von t **unabhängigen** Zufallszahlen $S_{t'} \in \{-1, +1\}$ ist:

$$X(t) = \sum_{t'=1}^{t} S_{t'}, \quad \text{Prob}(S_{t'} = \pm 1) = 1/2 \ .$$

Die Wahrscheinlichkeitsverteilung für diese Summe ist die Binomialverteilung:

$$\Rightarrow P(x, t) = \frac{1}{2^t} \begin{pmatrix} t \\ (t-x)/2 \end{pmatrix} \ .$$

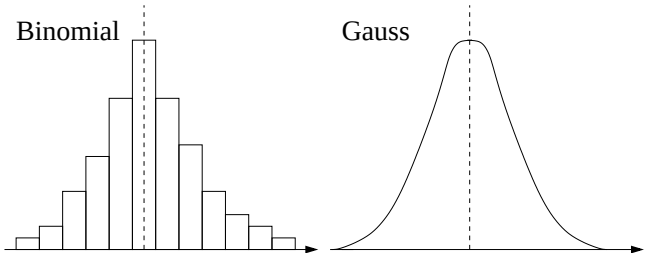


Abbildung 3.2: Die Gaußverteilung als Kontinuumslimes der Binomialverteilung

Im Limes großer Zeiten $t \to \infty$ kann man die Binomialverteilung als Gaußverteilung nähern, wie man formal mit Hilfe der Stirling-Formel für die in den Binomialkoeffizienten auftretenden Fakultäten zeigt. Man kann aber auch einfach den **zentralen Grenzwertsatz** anwenden, da $\langle S_{t'} \rangle =$

[1]Beim diskreten RW mit dem Startpunkt $x = 0$ zur Zeit $t = 0$ sind bei geraden Zeitschritten $t = 2, 4, 6, ..$ nur gerade Plätze besetzt. Ensprechendes sind für ungerade Zeitschritte nur ungerade Plätze. Es treten folglich keine Verlustterme in der Mastergleichung auf, da die entsprechenden Wahrscheinlichkeiten $P(x, t)$ verschwinden.

0 und $\langle S_{t'}^2 \rangle = 1$ gilt. Wir erhalten also für die gesuchte Wahrscheinlichkeit $P(x,t)$ im Limes $t \to \infty$:

$$\lim_{t\to\infty} \tilde{P}(y = x/\sqrt{t}, t) = \frac{1}{\sqrt{2\pi}} e^{-y^2/2}$$

Die typische *Längenskala* des RWs wird durch $\sqrt{t}$ beschrieben, da wir durch Reskalierung von x durch $\sqrt{t}$ eine invariante Form von $P(x,t)$ erhalten. Gleichzeitig ist $\sqrt{t}$ die typische Entfernung des Walkers vom Startpunkt nach t Schritten.

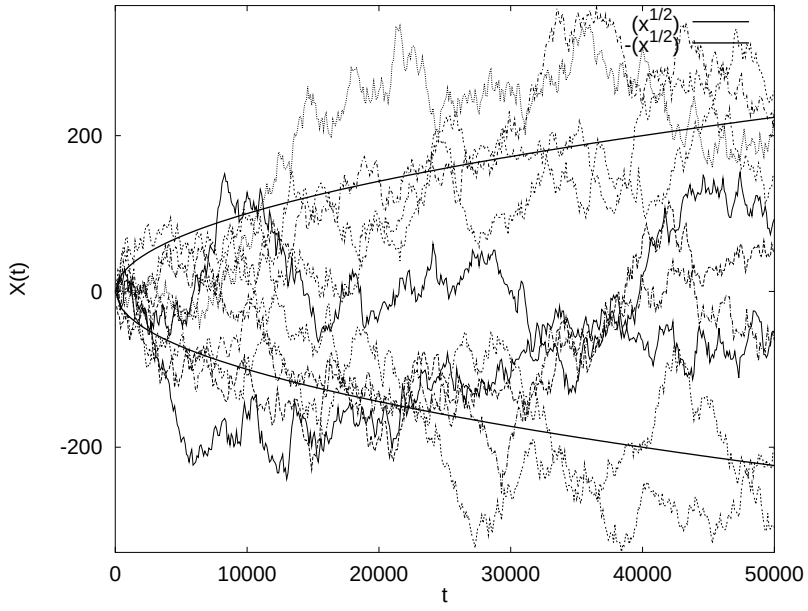


Abbildung 3.3: $X(t)$ für einige Realisierungen des RW im Vergleich mit $\pm\sqrt{t}$.

Wenn man die Verteilungsfunktion $P(x,t)$ durch eine Computersimulation bestimmen möchte, muss man zunächst viele Realisierungen $X(t)$ des RW generieren, um dann aus der Mittelung $P(x,t)$ zu bestimmen (siehe Abb. 3.3).

Das Modell kann auch leicht verallgemeinert werden. Zunächst einmal betrachten wir den **Kontinuumslimes**, durch Einführung einer Schrittweite a und Zeitintervalls τ. Damit ergibt sich für die Master-Gleichung:

$$P(x,t+\tau) = \frac{1}{2}\{P(x-a,t) + P(x+a,t)\}\,.$$

Die Taylorentwicklung der linken (Entwicklung um t) und rechten (um x) liefert in führender Ordnung ($\tau \ll 1$, $a \ll 1$).

$$\begin{aligned} P(x,t) + \tau\frac{\partial P}{\partial t} + \ldots &= \frac{1}{2}(P(x,t) - a\frac{\partial P(x,t)}{\partial x} + \frac{1}{2}a^2\frac{\partial^2 P(x,t)}{\partial x^2} \\ &+ P(x,t) + a\frac{\partial P(x,t)}{\partial x} + \frac{1}{2}a^2\frac{\partial^2 P(x,t)}{\partial x^2} + \ldots) \\ &= P(x,t) + \frac{1}{2}a^2\frac{\partial^2 P(x,t)}{\partial x^2} + \Theta(a^4) \end{aligned}$$

Setzen wir nun: $\lim_{a,\tau\to 0}\frac{a^2}{2\tau} = D$, erhalten wir die Diffusionsgleichung

$$\frac{\partial P(x,t)}{\partial t} = D\frac{\partial^2 P(x,t)}{\partial x^2},$$

die die Lösung

$$P(x,t) = \frac{1}{\sqrt{4\pi D t}}\,\exp\left(-\frac{x^2}{4Dt}\right)$$

besitzt. Wir sehen also, dass wieder die Längenskala $\sqrt{t}$ auftaucht.

Die Beobachtung, dass die Einführung eines Gitters die Eigenschaften auf großen Skalen nicht ändert, gilt auch für andere Modelle. Man kann also häufig bei der Simulation auf das entsprechende Gittermodell zurückgreifen.

Die Verallgemeinerung des Prozesses auf d-Dimensionen ist ebenfalls leicht zu bewerkstelligen. Wenn man den Prozess auf einem kubischen Gitter definiert, wählt man einfach einen der $2d$ nächsten Nachbarplätze aus. Die typische Längenskala des Prozesses bleibt unverändert, die Diffusionskonstante verallgemeinert sich allerdings zu $\lim_{a,\tau\to 0}\frac{a^2}{2d\tau} = D$.

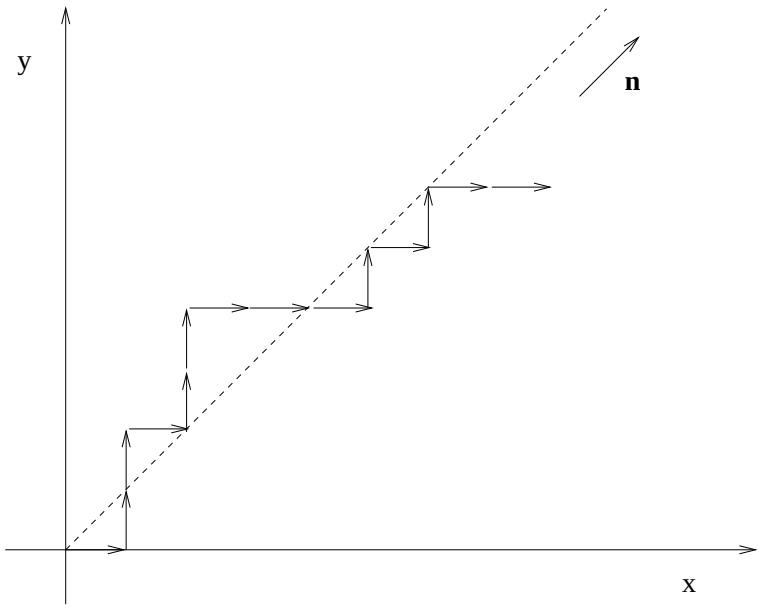


Abbildung 3.4: Gerichteter Walk in 2 Dimensionen $\underline{n} = (1, 1)$. Es sind nur Schritte in positive x-Richtung bzw. positive y-Richtung erlaubt.

Der **gerichtete** RW ist eine Variante des RWs, bei der nur Schritte in eine Richtung erlaubt sind, die eine positive Projektion auf eine Vorzugsrichtung $\underline{n}$ besitzen:

Der gerichtete RW kann z.B. als einfaches Modell für ein Polymer in einer Strömung aufgefasst werden oder aber als Modell für eine Grenzschicht zwischen verschiedenen Domänen eines Ferromagneten.

Die Master-Gleichung des gerichteten RW kann einfach durch die Zerlegung von $\underline{x}(t)$ in einen Anteil $\underline{x_{||}}(t)$ parallel zur Vorzugsrichtung und in einen Anteil $\underline{x_{\perp}}(t)$ senkrecht zu $\underline{n}$ gelöst werden. Parallel zu $\underline{n}$ bewegt sich der Walker mit konstanter Geschwindigkeit. Senkrecht zu $\underline{n}$ entspricht die Dynamik der eines gewöhnlichen $d-1$ dimensionalen RWs.

Für die mittlere quadratische Entfernung nach N Schritten ergibt sich dementsprechend

$$\langle x^2_{||,N} \rangle \sim N^{2\nu_{||}},$$

mit $\nu_{||} = 1$ und

$$\langle x^2_{\perp,N} \rangle \sim N^{2\nu_{\perp}}$$

mit $\nu_{\perp} = \frac{1}{2}$ unabhängig von der Dimension.

Nichttriviale Ergebnisse erhält man für gerichtete RWs in ungeordneten Medien, bei denen die Übergangswahrscheinlichkeiten zwischen zwei Gitterplätzen zufällig gewählt wurden.

3.1 Einfache Polymermodelle und der Self-Avoiding-Walk

Polymere sind lange Kettenmoleküle, die aus einer großen Zahl von konstituierenden Einheiten bestehen, den sog. Monomeren. Auf der Skala von einigen Monomeren werden die Eigenschaften der Polymere wesentlich von den chemischen Eigenschaften der Monomere bestimmt. Auf großen Skalen aber, d.h. für lange Polymerketten, spielen Details der chemischen Bindung keine Rolle mehr. Man operiert dann mit einfachen Modellen, die nur die Eigenschaften von Polymeren beschreiben, die das Verhalten auf großen Längenskalen beeinflussen. Das einfachste Modell eines Polymers ist der RW. Die Verbindung zum Polymer wird offensichtlicher, wenn man die sog. frei verbundene Kette betrachtet. Bei diesem Modell wird eine feste Schrittweite gewählt, die Richtung des Walkers ist aber beliebig.

Eine Variante des freely joined chains ist die, dass man den Winkel ϑ festhält, wie es der Natur der chemischen Bindung entspricht. Dies erzeugt Korrelationen auf kurzen Skalen, die aber das asymptotische Verhalten von $\langle R^2 \rangle$ nicht beeinflussen.

Wir halten also fest, dass die Eigenschaften dieser einfachen Polymermodelle auf großen Skalen denen des einfachen RWs entsprechen.

Eine wichtige Eigenschaft von Polymeren, die bislang vernachlässigt wurde, ist, dass sich Polymere nicht durchdringen können. Dies bedeutet für das RW Modell der Polymere, dass sich die Trajektorien nicht mit sich selbst schneiden dürfen. Die Selbstabstoßung der Polymere wird durch das Lösungsmittel sogar noch verstärkt, da es bei einem guten Lösungsmittel für die Monomere energetisch günstiger ist, sich mit Molekülen des Lösungsmittels zu umgeben. Dies führt zu einer effektiven Abstoßung der Monomere untereinander.

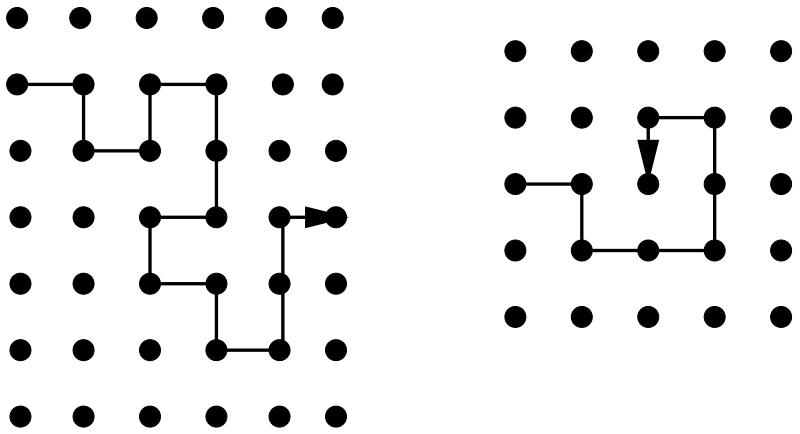


Abbildung 3.5: Realisierungen des SAW auf einem zweidimensionalen Quadratgitter. Jeder Gitterplatz kann nur einmal besucht werden. Dies kann zum Abbruch des Walks führen, wenn kein freier Nachbarplatz zur Verfügung steht.

Ein einfaches Polymermodell, dass den Effekt der Selbstabstoßung berücksichtigt ist der sog. Self-Avoiding Walk (SAW). Der SAW wird auf einem Gitter definiert. Der Volumenausschluss wird dadurch berücksichtigt, dass der Walker jeden Gitterplätz nur einmal besuchen kann.
Zur Simulation des SAW muss man also ein Gitter definieren, auf dem die Plätze, die bereits besucht worden sind, abgespeichert werden. Die Zahl der möglichen Schritte des Walkers bei einer gegebenen Position kann daher reduziert werden. Falls an der aktuellen Position des Walkers

kein Nachbarplatz frei ist, muss der SAW abgebrochen werden (vergl. Abb. 3.5).

Der SAW kann mit folgendem einfachen Algorithmus simuliert werden:

```
algorithm self-avoiding walk
begin
   Initialize
   for ( i< N) latt[i]=0;
   pos = 0;
   while ( t < nsteps ) do
      pos = select_neighbour(pos);
      if ( pos > 0 ) latt[pos] = 1;
      else break;
      t=t+1;
   end while
end


procedure select_neighbour(pos)
begin
   nn = count= 0;
   while ( nn < neighbors ) do
      if ( !latt[neigbor[pos][nn]] )
         newsite[count] = latt[neigbor[pos][nn]];
         count = count+1;
      end if
   end while
   if ( count ) select = ran()*count;
      return newsite[select];
   end if
   else return -1;
end
```

Die Selbstwechselwirkung ändert die universellen Eigenschaften des RWs. So wird beispielsweise der Exponent ν, der das Verhalten von $\langle R^2 \rangle \sim N^{2\nu}$

beschreibt, dimensionsabhängig und entspricht nur für $d \geq 4$ dem des einfachen RWs. Aus einer Skalenanalyse ergibt sich die Form

$$\begin{aligned} \nu &= \frac{3}{d+2} \quad \text{, d.h.} \\ \nu &= 1, \frac{3}{4}, \frac{3}{5}, \frac{1}{2} \end{aligned}$$

die in $d = 1, 2, 4$ mit dem exakten Ergebnis übereinstimmt und für $d = 3$ zumindest eine gute Näherung darstellt. Da der Exponent für $2 < d < 4$ analytisch nicht exakt berechnet werden kann, muss der Wert aus der Skalenanalyse numerisch überprüft werden. Problematisch ist aber dabei die Generierung großer Walks, da die Zahl der SAWs im Verhältnis zu der Anzahl der möglichen RWs exponentiell veschwindet. In jüngster Zeit sind aber sehr effiziente Algorithmen entwickelt worden, die große Kettenlängen generieren können. Diese Techniken werden in einem späteren Abschnitts erläutert.

Kapitel 4

Der Perkolationsübergang

4.1 Perkolationmodelle

Bei der Perkolation werden ähnlich wie beim Random Walk in einfacher Weise zufällige Konfigurationen erzeugt. Eine Art der Perkolation, die sog. Site-Perkolation, besteht darin, dass man auf einem vorgegebenen Gitter Plätze (Sites) mit einer vorgegebenen Wahrscheinlichkeit p besetzt. Entsprechend bleiben Plätze mit der Wahrscheinlichkeit $1 - p$ unbesetzt. Die beiden Gitterzustände (besetzt/unbesetzt) können für vielfältige Eigenschaften stehen. Ein Beispiel ist, die besetzten Sites als leitend und die unbesetzten Sites als isolierend zu betrachten. Legt man an ein solches System eine Spannung an, kann ein Strom nur dann fließen, wenn benachbarte Plätze besetzt sind. Bei einer hohen Besetzungswahrscheinlichkeit p erwarten wir, dass das Gesamtsystem leitend ist, weil man einen oder mehrere Pfade zwischen nächsten Nachbarn findet, auf denen man das System durchqueren kann. Umgekehrt wird man bei sehr kleinen Werten von p solche Pfade nicht finden und das System verhält sich isolierend.
Die Existenz von isolierender und leitender Phase impliziert, dass es einen Wert p_c gibt, für den der Strom zum ersten Mal das System durchdringen kann. Diesen Wert p_c bezeichnet man auch als **Perkolationsschwelle**.
Neben der Site-Perkolation betrachtet man auch die *Bond-Perkolation*, bei der die Bindungen zwischen benachbarten Gitterplätzen zufällig ge-

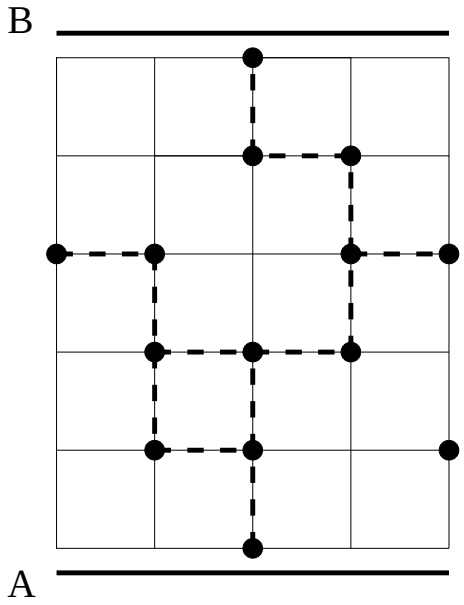


Abbildung 4.1: Zweidimensionales Quadratgitter mit zufällig besetzten Sites. Knoten, die sich auf benachbarten (nächste Nachbarn!) Gitterplätzen befinden, gehören zum gleichen Cluster. Wenn sich ein Cluster von A nach B erstreckt, spricht man von einem perkolierenden Cluster.

setzt werden. Beide Arten der Perkolation haben zahlreiche Anwendungen. Die Bond-Perkolation kann noch natürlicher als zufälliges Netzwerk von elektrischen Leitern verstanden werden. Eine weitere Anwendung stammt aus der Chemie, der Sol-Gel Übergang. Der Sol-Gel Übergang ist der Übergang von einer flüssigen Phase mit kleinen sich verzweigenden Molekülen (Sol) zu einem makroskopischen Netzwerk (Gel) durch Aktivierung von Bindungen zwischen den Molekülen. Das Gel besitzt dann mechanische Eigenschaften analog zu denen eines festen Körpers. Ein Beispiel für den Sol-Gel Übergang ist das Eierkochen.

Im Rahmen eines Perkolationsmodells wird die Zahl der aktiven Bindungen durch die Wahrscheinlichkeit p parametrisiert, eine Bindung zwischen benachbarten Plätzen zu setzen.

- $p > p_c$: makroskopisches Netzwerk von Bindungen (Gel)

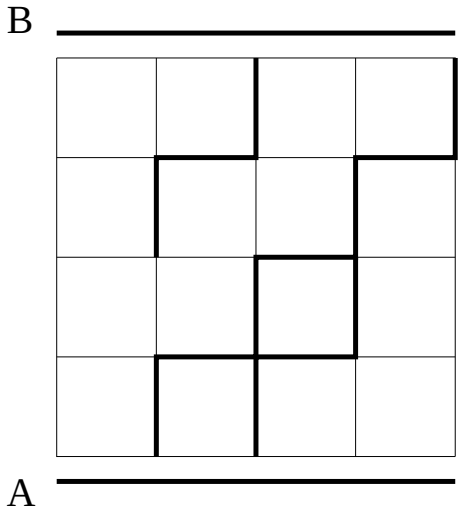


Abbildung 4.2: Bond-Perkolation: Quadratgitter mit zufällig gesetzten Kanten (fett).

- $p < p_c$: Bindungen nur auf endlichen Skalen (Sol)
- $p = p_c$: Sol-Gel Übergang

Die Struktur des Problems macht auch eine Verknüpfung von Bond und Site-Perkolation sinnvoll, da man dann auch die zufällige Anordnung des Materials berücksichtigt.

Eine weitere wichtige Variante ist die Besetzung der Gitterpunkte gemäß einer gewählten stochastischen Dynamik. In Modellen zur Ausbrütung von Epidemien (besetzt $\hat{=}$ krank / unbesetzt $\hat{=}$ gesund) oder Waldbränden (besetzt $\hat{=}$ brennend) werden die **Nachbarsites** mit einer vorgegebenen Wahrscheinlichkeit besetzt.

Schließlich kann man Perkolation auch im Kontinuum formulieren (Schweizer-Käse Modell). Bei dieser Variante bohrt man an zufälligen Positionen Löcher einer vorgegebenen Größe in das Material und analysiert den Anteil des verbleibenden Materials p_c an den Punkt bei dem das Restmaterial gerade noch perkoliert. In $d = 2$ ist $p_c = 0.312$. Der Perkolationsübergang wird durch die geometrischen Eigenschaften des großen

Cluster in der Nähe von p_c charakterisiert.
Aus theoretischer Sicht wird er dadurch interessant, dass er als kritisches Phänomen verstanden werden kann.

4.2 Perkolation als kritisches Phänomen

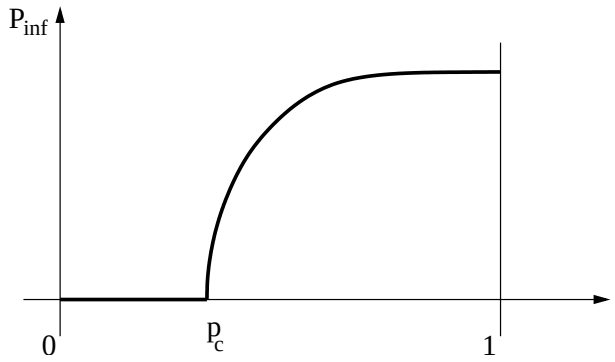


Abbildung 4.3: Typischer Verlauf von P_∞ im unendlichen System.

Die Eigenschaften von Perkolationsmodellen entsprechen zumindest teilweise denen thermodynamischer Systeme am kritischen Punkt. Wie bei thermodynamischen Phasenübergängen kann man einen **Ordnungsparameter** finden, der in einer Phase ($p < p_c$) verschwindet und in der anderen Phase ($p > p_c$) einen endlichen Wert annimmt (vergl. Abb. 4.3).
Bei der Perkolation ist der Ordnungsparameter die Wahrscheinlichkeit P_∞, dass ein Bond/Site zum unendlichen Cluster gehört.
Wir haben also folgenden Verlauf von P_∞:

$$\begin{aligned} p < p_c &: \quad P_\infty = 0 \\ p > p_c &: \quad P_\infty \propto (p - p_c)^\beta \end{aligned}$$

An der Perkolationsschwelle betrachtet man also ein algebraisches Anwachsen von P_∞.
Der Wert von p_c hängt, wie man Tabelle 4.2 entnehmen kann, von der Gitterstruktur, der Dimension und der Art der Perkolation ab.

		p_c	
d	Gitter	Site-Perkolation	Bond-Perkolation
2	Dreieck	1/2	$2\sin(\pi/18)$
	Quadrat	0.5927460	1/2
	Honigwaben	0.6962	$1-2\sin(\pi/18)$
3	fcc	0.198	0.119
	bcc	0.245	0.1803
	sc	0.31161	0.248814

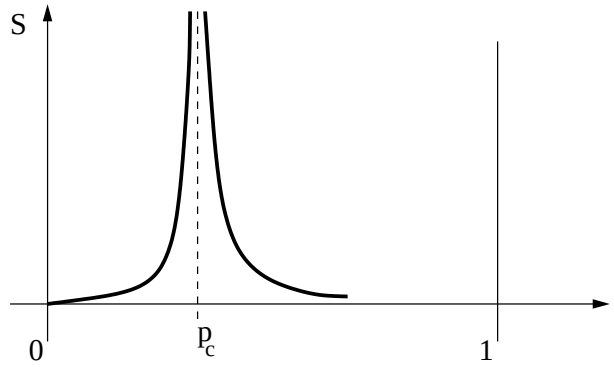


Abbildung 4.4: Typischer Verlauf von ξ im unendlichen System.

Eine weitere Charakterisierung des Phasenübergangs wird durch die Definition der **Korrelationslänge** ξ erreicht. Die Korrelationslänge beschreibt die lineare Ausdehnung der *endlichen* Cluster. Das Verhalten von ξ wird durch

$$\xi \sim |p - p_c|^{-\nu}$$

beschrieben (siehe Abb. 4.4). Der Exponent, der die Divergenz beschreibt, ist für beide Phasen derselbe. Gemessen wird die Korrelationslänge durch die Bestimmung des mittleren Abstandes zweier Punkte im gleichen Cluster (für $p > p_c$ werden die perkolierenden Cluster nicht betrachtet).

Ein weiterer Exponent wird durch die mittlere Zahl der Plätze bzw. die *Masse der endlichen Cluster* S festgelegt. In der Nähe von p_c beobachtet

man:

$$S \propto |p - p_c|^{-\gamma}$$

(Der Exponent γ hat ebenfalls für $p > p_c$ und $p < p_c$ den gleichen Wert.)

Die Werte von β, ν, γ sind **universell**, d.h. sie hängen ausschließlich von der Dimension, nicht aber von den Details der Gitterstruktur ($fcc, sc, \ldots$) oder von der Art der Perkolation (Site/Bond) ab.

Man hat folgende Werte für die Exponenten gefunden

	$d = 2$	$d = 3$	$d \geq 6$
β	$5/36$	0.417 ± 0.003	1
ν	$4/3$	0.875 ± 0.008	$\frac{1}{2}$
γ	$43/18$	1.795 ± 0.003	1

Die Werte für $d = 2$ wurden durch eine exakte analytische Rechnung bestimmt. Ab der Dimension 6 wird die Mean-Field Näherung exakt. In den übrigen Dimensionen müssen die Exponenten numerisch bestimmt werden.

4.3 Algorithmen zur Cluster-Identifizierung

Die numerische Untersuchung des Perkolationsübergangs setzt zunächst die Generierung von zufälligen Konfigurationen mit der vorgegebenen Besetzungswahrscheinlichkeit voraus. Dies ist aber trivial. Die anspruchsvollere Aufgabe besteht darin, effizient Cluster zu identifizieren und zu analysieren. In diesem Abschnitt werden drei Algorithmen vorgestellt zur Identifizierung von Clustern vorgestellt. Die ersten beiden Algorithmen sind Suchalgorithmen, die auf allgemeine Graphen angewendet werden können, während beim dritten Algorithmus die Gitterstruktur ausgenutzt wird.

4.3.1 Depth-First-Search

Der *Depth-First-Search Algorithmus* eignet sich zur Bestimmung von Clustern auf allgemeinen Graphen $G(V, E)$, wobei V für Knoten (Vertices,

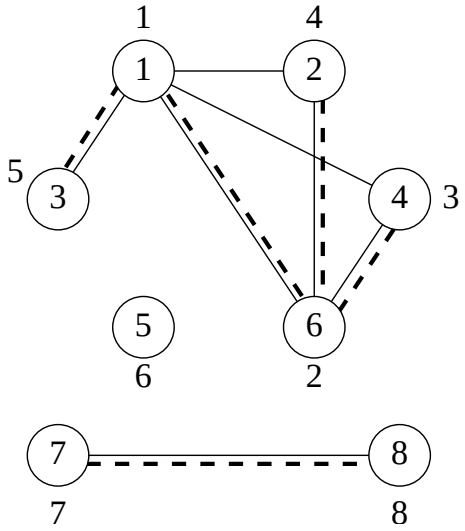


Abbildung 4.5: Veranschaulichung des Depth-First-Search Algorithmus: Nachdem alle Knoten des Graphen, als nicht zu einem Cluster gehörend initialisiert worden sind, startet man die Suche mit dem Knoten 1. Dann geht man zu einem Nachbarn, der noch nicht zum Cluster gehört (Knoten 6). Die Suche wird in die Tiefe fortgesetzt, d.h. man kehrt nicht zu 1 zurück, sondern besucht zunächst die Nachbarn des neuen Knotens, die noch nicht im Cluster sind. Die Reihenfolge der besuchten Knoten ist durch die Zahlen ausserhalb des Kreises angezeigt.

Gitterpunkte) und E für Kanten (Edges, Bindungen) steht. Die Wirkungsweise des Algorithmus ist in Abb. 4.5 dargestellt. Kennzeichnend für den Algorithmus ist, dass man soweit wie möglich in die Tiefe geht, bevor die Nachbarn des Startknotens i besucht werden.
Jeder Knoten eines Clusters wird genau einmal besucht. Es wird ein Baumstruktur erzeugt, die den Cluster aufspannt (Spanning Tree). Für den Gesamtgraphen erzeugt man einen Spanning Forest.

Der Algorithmus hat die folgende Struktur:

algorithm components (G)
begin

```
    Initialize
    for (i ∈ V ) comp[i] := 0;
    t :=1;
    while(there is a node i with comp[i]=0) do
        depth_first(G,i,comp,t);
        t = t+1;
    end while
end


procedure depth_first(G,i,comp,t)
begin
    comp[i] := t;
    for all neighbors j of i do
        if (comp[j]=0)
            depth_first(G,j,comp,t);
        end if
    end for
end
```

4.3.2 Breadth-First-Search

Eine weitere Möglichkeit zur Bestimmung von Clustern ist der Breadth-First-Search (BFS) Algorithmus, der zunächst alle Nachbarknoten abarbeitet bevor er zum nächsten Knoten übergeht.
Der Algorithmus identifiziert alle Elemente eines Clusters und bestimmt zusätzlich den Abstand vom Startknoten. Ferner wird der kürzeste Pfad von einem beliebigen Punkt auf dem Graphen zu seinem Startpunkt gespeichert (durch pred ()).
Zur vollständigen Bestimmung der Cluster einer Konfiguration muss man den BFS Algorithmus mehrfach mit verschiedenen Startknoten aufrufen.

Der Algorithmus kann in folgender Art und Weise implementiert werden:

procedure breadth-first-search(G,s)

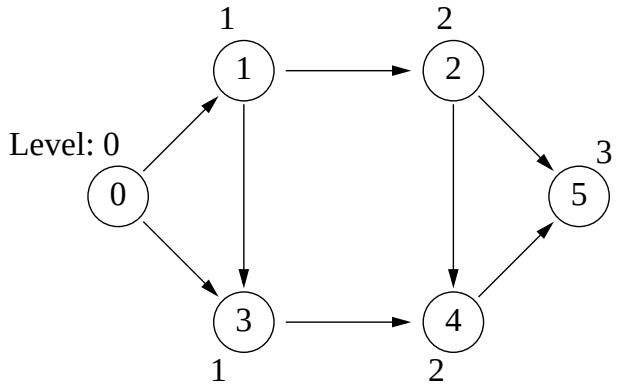


Abbildung 4.6: Suchschema beim BFS Algorithmus. Ausgehend vom 0ten Knoten werden *alle* Nachbarknoten besucht und ihnen der Level 1 zugeordnet. Dann werden alle Nachbarsites des nächsthöheren Levels abgearbeitet, bis schließlich alle Sites abgearbeitet sind. Jede Kante des Graphen wird **zweimal** berührt.

```
begin
    Initialize
    queue Q:= s;
    level(s):= 0;
    for  (all other vertices) level(i) = -1;
    queue pred(0):= -1;
    while (Q not empty) do
        Remove first vertex i of Q;
        for (all neighbors j of i) do
            if (level(j) = -1 ) do
                set level(j) := level(i) + 1;
                set pred(j) := i ;
                add j at the end of Q ;
            end if
        end for
    end while
end
```

4.3.3 Der Hoshen-Kopelman Algorithmus

Der Hoshen-Kopelman Algorithmus (HKA) stellt eine sehr effiziente Methode zur Analyse von Clustern auf *regulären* Gittern dar. Die Einschränkung bezüglich der Gitterstruktur wird im Vergleich zu den beiden Suchalgorithmen dadurch kompensiert, dass der HKA sich effektiv parallelisieren lässt und somit die Simulation von sehr großen Gittern zulässt.
Die Grundidee des HKA besteht darin, den verschiedenen Clustern jeweils genau ein Label zuzuordnen. Die Zuordnung des Labels erfolgt für jeden Gitterplatz durch eine lokale Operation. Durch den HKA kann die Größenverteilung der Cluster in einem Durchlauf bestimmt werden.

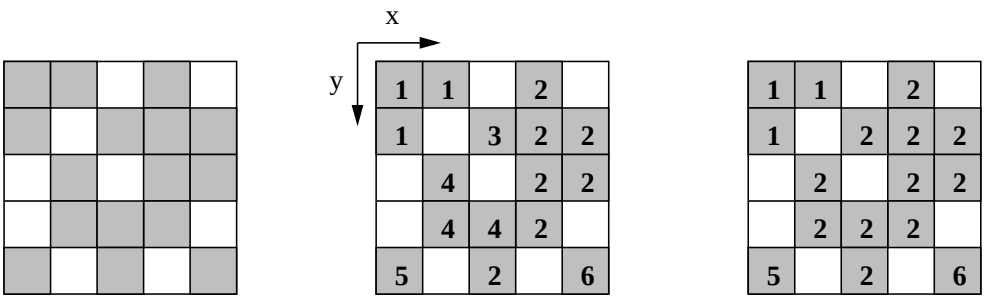


Abbildung 4.7: Iterationsstufen im Hoshen-Kopelman Algorithmus: Links: Beispielkonfiguration auf dem $2d$-Quadratgitter mit offenen Randbedingungen. Mitte: Indizierung der Cluster mit den aktuellen "wahren" Labeln der Nachbarn. Rechts: Korrektur der Label in einem zweiten Durchlauf.

Wir wollen die Arbeitsweise des Algorithmus am Beispiel des zweidimensionalen Quadratgitters erläutern:
Im ersten Durchlauf wird jedem besetzten Site das *Minimum der wahren Nachbarlabel* zugordnet. Da man sich bei der Zuordnung der Label auf den oberen und linken Nachbarn beschränkt ist gelegentlich eine Korrektur des Labels erforderlich. Entscheidende Idee des Algorithmus ist es nun, die Korrektur des Labels nicht tatsächlich auszuführen, sondern nur zu speichern, in welches Label das temporär zugeordnete Label übergegangen ist.

In unserem Beispiel tritt ein solcher Fall bei der Betrachtung des Platzes $(4, 2)$ auf, der zum Cluster "2" gehört, ebenso wie der Platz $(3, 2)$, der mit dem temporären Label "3" belegt wurde. Die Vereinigung der Cluster wird dadurch berücksichtig, dass man ein Feld $N(m)$ führt, dass die Anzahl der Gitterplätze im Cluster m speichert. Für den Fall unterschiedlicher Label der Nachbarn kann addiert man die Zahl der Gitterplätze der verschiedenen Cluster und ordnet sie dem Minimum der Label zu. In unserem Fall ergibt sich $N(2) = N(2) + N(3) + 1 = 1 + 1 + 1 = 3$. Zur späteren Korrektur des Index speichert man dann mit welchem Cluster der Cluster "3" zusammenhängt. Dies geschieht einfach durch die Zuordnung $N(3) = -2$. Durch diese Art des Speicherns kann man auf die Korrektur der Label verzichten.

Damit die Clustergröße direkt nach einem Durchlauf ausgegeben werden kann, muss man immer das wahre Label berücksichtigen. Dies bedeutet z.B. für den Platz $(3, 5)$, dass wegen $N(4) = -2$ und $N(2) = 10$ dem Platz der Index "$2''$ zugeordnet wird. (Im allgemeinen Fall ist es auch möglich, dass man mehrere Schritte braucht, bis man auf ein positives $N(a)$ und damit auf den wahren Index stößt.) Im Anschluss erhöht man die Zahl der Elemente im Cluster "2" auf $N(2) = N(2) + 1$.
In einem zweiten Durchlauf wird dann die Korrektur der Indizes durchgeführt, d.h. man ordnet allen Gitterplätzen mit einem Index m, für den $N(m) < 0$ gilt, den wahren Index zu.
Der HKA für ein offenes 2d-Quadratgitter hat die folgende Struktur:

```
algorithm Hoshen-Kopelman
begin
   m=1;
   for (i < N ) do
      if ( occ[i] ) do
         lx = ly = -1;
         if ( occ[nx[i]] ) lx = get_label ( nx[i] );
         if ( occ[ny[i]] ) ly = get_label ( ny[i] );
         if ( (lx>0) & (ly>0)) label[i] = merge (nx[i], ny[i]);
         else if (lx>0) do
            label[i] = lx;
```

```
            n[lx] = n[lx]+1;
        end else if
        else if (ly>0)do
            label[i] = ly;
            n[ly] = n[ly]+1;
        end else if
        else  do
            label[i] = m;
            n[m] = 1;
            m = m+1;
        end else
      end if
   end for
   for (i ∈ 1...N ) do
      if ( occ[i] )
         label[i] = get_label ( i );
   end for
end


procedure get_label ( i )
begin
   s = label[i];
   while ( n[s] < 0 ) do s = -n[s];
   return return s;
end


procedure  merge(lx,ly)
begin
   c1 = min{ lx,ly } ;
   c2 = max{ lx,ly } ;
   n[c1] = n[c1] + n[c2] + 1 ;
   n[c2]= -c1;
   return c1;
end
```

4.4 Der Perkolationsübergang: Skalentheorie und Finite-Size-Scaling

Der Perkolationsübergang bei $p = p_c$ ist ein Phasenübergang zweiter Ordnung. Er genügt mathematischen Gesetzmäßigkeiten, die allgemeiner Natur sind, d.h. wir werden sie bei anderen Phasenübergängen zweiter Ordnung wieder antreffen. Diese Gesetzmäßigkeiten werden in Form von sogenannten **Skalengesetzen** wiedergegeben.
Die erste Größe, die wir betrachten ist $n_s(p)$ ($s\hat{=}$ "size"), die Konzentration von Clustern der *endlichen* Größe s im unendlichen System mit Besetzungswahrscheinlichkeit p. Demnach ist $n_s(p)$ die Wahrscheinlichkeit, daß ein beliebig herausgegriffener Gitterplatz zu einem Cluster der Größe s gehört, dividiert durch die Größe des Clusters s!

Es stellt sich heraus, daß $n_s(p_c)$ **algebraisch** mit s abfällt:

$$n_s(p_c) \propto s^{-\tau} \tag{4.1}$$

wobei τ ein sogenannter kritischer Exponent ist ($\tau < 3$), der von der z.B. von der Dimension des Gitters abhängt.

In der Nähe von p_c genügt $n_s(p_c)$ einer **Skalenform** bzw. einem Skalengesetz:

$$n_s(p) = s^{-\tau} f\left((p - p_c)^{1/\sigma}\, s\right) \tag{4.2}$$

wobei σ ein anderer kritischer Exponent ist und $f(x)$ eine sogenannte **Skalenfunktion**. Skalenfunktionen hängen immer von dimensionslosen

Skalenvariablen ab, z.B. dem Verhältnis zweier Längenskalen. In unserem Fall ist die Skalenvariable das Verhältnis der linearen Clustergröße s^{1/d_f}[1] und der Korrelationslänge

$$\xi = |p - p_c|^{-\nu}. \tag{4.3}$$

Wenn die Skalenannahme (4.2) erfüllt ist, so sollte für σ gelten

$$s(p - p_c)^{1/\sigma} \sim \frac{\xi}{\xi^{d_f}} \sim s(p - p_c)^{\nu d_f},$$

d.h. man erwartet die Skalenrelation:

$$\boxed{\frac{1}{\sigma} = \nu d_f} \tag{4.4}$$

Bemerkung: (1) Wir haben stillschweigend die **fraktale Dimension** d_f eingeführt, die in der Nähe von p_c die Beziehung zwischen linearer Ausdehnung R_s eines Clusters und seiner Geamtmasse S beschreibt:

$$\boxed{S \propto R_S^{d_f}} \tag{4.5}$$

(2)

$$\text{Es ist } f\left(\frac{S}{\xi^{d_f}}\right) = \tilde{f}\left(\frac{S^{1/d_f}}{\xi}\right), \quad \text{wobei } \tilde{f}(x) = f(x^{1/d_f})$$

[1] Perkolationsclustern

eine andere Skalenfunktion ist, die ebenso von einer dimensionslosen Variablen abhängt. Wir können also das Argument x einer Skalenfunktion $f(x)$ beliebig manipulieren.

Wesentlich für die Gültigkeit einer Skalentheorie wie oben ist die Tatsache, daß das System **asymptotisch** (d.h. für $S \to \infty$, $\delta = |p - p_c| \to 0$) nur **eine** relevante Längenskala ξ besitzt, die für $\delta \to 0$ divergiert. Auf kleinen Skalen können durchaus signifikante Abweichungen von den Skalengesetzen auftreten (müssen auch, denn sonst wären alle Gitter auch mikroskopisch gleich!).
Aus (2) ergibt sich für die mittlere Masse S der Cluster:

$$S = \sum_{s=1}^{\infty}{}' \, s \underbrace{\left(\frac{s\, n_s}{\sum_{s=1}^{\infty} s\, n_s} \right)}_{= p}$$

wobei $s\, n_s / \sum_{s=1}^{\infty} s\, n_s$ die Wahrscheinlichkeit dafür ist, daß ein besetzter Platz zu einem Cluster mit s "Sites" gehört.

$$\begin{aligned} S \;\; &= \frac{1}{p} \sum_{s=1}^{\infty} s^2 n_s \;\; \sim \frac{1}{p} \underbrace{\sum_{s=1}^{\infty} s^{2-\tau} f(\delta^{1/\sigma}\, s)}_{\text{divergiert wegen } 2<\tau<3} \\ &\sim \int_1^{\infty} ds\, s^{2-\tau} f(cs) \quad c = \delta^{1/\sigma} \\ &\sim c^{3-\tau} \int_*^{\infty} dx\, x^{2-\tau} f(x) \\ &\delta^{(3-\tau)/\sigma} = |p - p_c|^{(3-\tau)/\sigma} \end{aligned}$$

$$\boxed{\Rightarrow \gamma = \frac{3-\tau}{\sigma}} \tag{4.6}$$

Betrachten wir nun die Wahrscheinlichkeit P_∞, daß ein Platz zum unendlichen Cluster gehört:

$$P_\infty = 1 - \frac{1}{p} \underbrace{\sum_{s=1}^{\infty}{}' s n_s}_{= p \text{ für } p < p_c \text{ d.h. } P_\infty = 0}$$

Denn jeder Platz ist entweder:
a) leer mit Wahrscheinlichkeit $1 - p$
b) besetzt **und** im unendlichen Cluster mit Wahrscheinlichkeit pP_∞
c) besetzt, aber **nicht** im unendlichen Cluster

Da $\sum_s s n_s$ konvergent ist, können wir zur Bestimmung des singulären Anteils nicht einfach die Summe durch ein Integral ersetzen. Wir wenden folgenden Trick an:

$$P_\infty = 1 - \frac{1}{p} \sum_{s=1}^{\infty} s^{1-\tau} f(cs)$$

$$\begin{aligned} \Rightarrow \frac{dP_\infty}{dc} &= \frac{1}{p} \sum_{s=1}^{\infty} s^{2-\tau} f'(cs) \\ &\sim \int_1^{\infty} ds\, s^{2-\tau} f'(cs) \\ &\propto c^{\tau-3} \end{aligned}$$

$$\Rightarrow P_{\infty,\text{ sing.}} \propto c^{\tau-2} = \delta^{(\tau-2)/\sigma} \quad (2 < \tau < 3\ !)$$

$$\boxed{\Rightarrow \beta = \frac{\tau - 2}{\sigma}} \tag{4.7}$$

Umgekehrt ergeben sich aus $\gamma = (3-\tau)/\sigma$ und $\beta = (\tau-2)/\sigma$:

$$\boxed{\sigma = \frac{1}{\beta+\gamma}, \quad \tau = 2 + \frac{\beta}{\beta+\gamma}} \tag{4.8}$$

Bemerkung:
$M_0 = \sum_s n_s$ ist die totale Anzahl von Clustern.
$M_0 \propto c^{\tau-1}$ (mit obigem Trick), d.h. $M_0 \propto \delta^{(\tau-1)/\sigma} = \delta^{2-\alpha}$
mit

$$\boxed{2 - \alpha = \frac{\tau-1}{\sigma} = 2\beta + \gamma} \tag{4.9}$$

Die genaue Definition der **linearen Ausdehnung** R_s eines Clusters der Größe s (der "mittlere quadratische Abstand" zwischen **allen** Paaren eines Clusters) ist:

$$R_s^2 = \frac{1}{s(s-1)} \sum_{i,j=1}^{s} (\underline{r}_i - \underline{r}_j)^2 \propto s^{2/d_f}$$

Jeder Gitterplatz gehört mit Wahrscheinlichkeit sn_s zu einem Cluster der Größe s und ist darin mit s (genau $s-1$) anderen Plätzen verbunden. Der Wert R_s^2 wird daher mit der Wahrscheinlichkeit $s^2 n_s / (\sum_{s=1}^{\infty} s^2 n_s)$ angenommen. (s^2 taucht wegen Gleichgewichtung jedes **Paares** auf).
Im Mittel ist daher R_S^2 gleich ξ^2, der Korrelationslänge zum Quadrat.

$$\xi^2 = \frac{\sum_{s=1}^{\infty} R_S^2 \, s^2 \, n_s}{\sum_{s=1}^{\infty} s^2 \, n_s}$$

Also:

$$\xi^2 \sim \frac{\sum_{s=1}^{\infty} s^{2/d_f+2-\tau}\, f((p-p_c)^{1/\sigma}\, s)}{\sum_{s=1}^{\infty} s^{2-\tau}\, f((p-p_c)^{1/\sigma}\, s)}$$

Wir überführen die Summen in Integrale, um die Asymptotik abzuschätzen: $\delta := p - p_c$

$$\sum_{s=1}^{\infty} s^{\alpha} f(\delta^{1/\sigma}\, s) \sim \int_1^{\infty} ds\, s^{\alpha} f(\underbrace{\delta^{1/\sigma}\, s}_{=x}) = \delta^{-(1+\alpha)/\sigma} \int_*^{\infty} dx\, x^{\alpha} f(x)$$

$$\Rightarrow \xi^2 \sim \frac{\delta^{-(2/d_f+3-\tau)/\sigma}}{\delta^{-(3-\tau)/\sigma}} \sim \delta^{-2/\sigma d_f} = |p-p_c|^{-2/\sigma d_f}$$

Dies ergibt wieder die schon oben erwähnte Relation:

$$\boxed{\nu = \frac{1}{\sigma d_f}}$$

Die **Fraktale Dimension** des perkolierenden Clusters:

Da

$$P_{\infty} = \frac{\text{Masse des perkolierenden Clusters}}{\text{Gesamtmasse}}$$

In einem endlichen System der Länge L mit $L \approx \xi = (p-p_c)^{-\nu} \quad (p \geq p_c)$ ist $P_{\infty} = \xi^{d_f}/\xi^d$, andererseits $P_{\infty} = \delta^{\beta} = \xi^{-\beta/\nu} \Rightarrow$ (Hyperskalenrelation)

$$\boxed{\frac{\beta}{\nu} = d - d_f} \tag{4.10}$$

Bis hierhin haben wir über die in einem **unendlichen** System auftretenden Singularitäten und die sie charakterisierenden kritischen Exponenten gesprochen. In einer Computer-Simulation können wir aber nur **endliche** Systeme untersuchen – wie können wir aus den Daten für P_{∞}, S, ξ, usw. für ein endliches System (mit linearer Ausdehnung L) Erkenntnisse,

d.h. insbesondere Schätzwerte und deren Fehlerbalken, für die gesuchten kritischen Exponenten $\beta, \gamma, \nu, \sigma, \tau, \ldots$ erhalten? Der naive (aber hin und wieder beschrittene) Weg besteht darin, "sehr große" Systeme zu untersuchen und an die Daten irgendwie die Funktionen mit den entsprechenden algebraischen Singularitäten anzufitten. Dies ist recht unzuverlässig. Der "bessere", weil systematischere und zuverlässigere, Weg besteht in einer **Finite-Size-Scaling** (FSS) Analyse.

Betrachte ein System der Größe L, z.B. ein $L \times L$-Quadratgitter. Im unendlichen System gibt es für $p < p_c$ keinen **unendlichen** Cluster, für $p > p_c$ gibt es immer einen. Für $p = p_c$ gibt es manchmal einen, sagen wir mit Wahrscheinlichkeit $1/2$. Nennen wir $P_{\text{perc}}(p)$ die Wahrscheinlichkeit dafür, daß ein unendlicher Cluster auftritt, so ist dies eine Stufenfunktion.

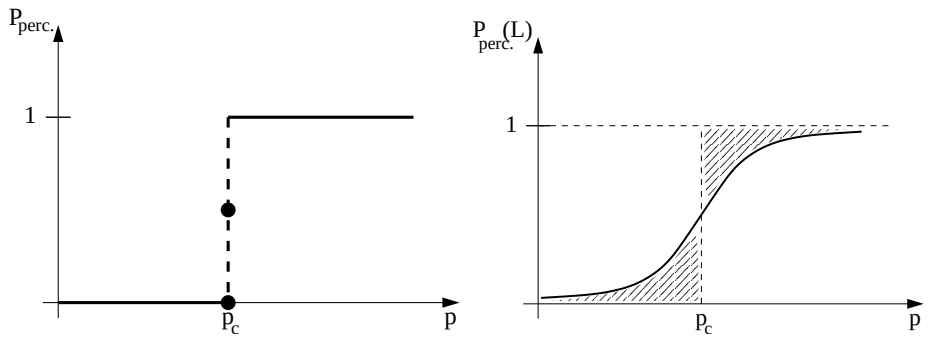


Abbildung 4.8: Abhängigkeit der Perkolationswahrscheinlichkeit von p im unendlichen (links) und im endlichen (rechts) System.

Im endlichen System wird der Übergang unscharf, d.h. es gibt eine endliche Wahrscheinlichkeit bereits für $p < p_c$ einen perkolierenden Cluster im System zu finden bzw. für $p > p_c$ keinen. $P_{\text{perc}}(L, p)$ ist eine analytische Funktion. Die Steigung von $P_{\text{perc}}(L, p)$ bei $p = p_c$ wird mit zunehmendem L immer größer wird und die schraffierten Flächen werden immer kleiner.

Neben L existiert eine weitere charakteristische Längenskala im System, die Korrelationslänge ξ. Die FSS-Form für $P_{\text{perc}}(L)$ ist demnach:

$$\boxed{P_{\text{perc}}(L,p) \sim \tilde{P}(L/\xi) = \tilde{P}(L\,\delta^{\nu})} \tag{4.11}$$

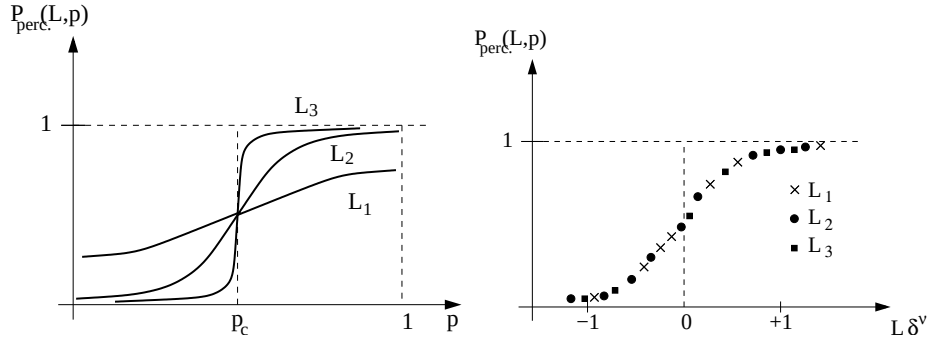


Abbildung 4.9: Links: Plot von $P_{\text{perc}}(L,p)$ für verschiedene L als Funktion von p ($L_1 < L_2 < L_3$). Rechts: Skalenplot von $P_{\text{perc}}(L,p)$ für verschiedene L aufgetragen gegen die Skalenvariable $L/\xi = L\delta^{\nu}$.

Bei $p = p_c$ wird gemäß (4.11) $P_{\text{perc}}(L,p)$ unabhängig von L (das Argument der Skalenfunktion ist für alle Werte von L Null, da die Korrelationslänge divergiert), alle Kurven $P_{\text{perc}}(L,p)$ als Funktion von p (für verschiedene L) schneiden sich in einem Punkt $P_{\text{perc}}(L, p = p_c)$ − der Wahrscheinlichkeit dafür bei p_c einen perkolierenden Cluster zu finden.

Damit haben wir ein probates, d.h. sehr zuverlässiges, Mittel an der Hand, p_c zu bestimmen!

Der Skalenplot oben rechts ermöglicht eine zuverlässige, erste Bestimmung von ν: Durch Auftragung von $P_{\text{perc}}(L,p)$ als Funktion von $\delta^{\nu}\,L = (p - p_c)^{\nu}\,L$ (p_c wie oben bestimmt) müssen gemäß (**??**) alle Daten für verschiedene L auf **eine** Kurve fallen, der Skalenfunktion $\tilde{P}(x)$. Da wir ν i.a. nicht a priori kennen, und wir wissen, daß der sog. Datenkollaps nur für die richtige Wahl von ν eintritt, ist ν ein Fit-Parameter. Wir variieren ihn so lange, bis durch bloßes Hinsehen die Daten möglichst gut

"kollabieren" (d.h. zusammenfallen). Dies ist sicherlich für ein ganzes Intervall $[\nu - \Delta\nu, \nu + \Delta\nu]$ akzeptabel, z.B. $\Delta\nu = 0.01$. Wir erhalten so einen groben Schätzwert für den Fehler des so bestimmten Wertes von ν. Systematischer würde man versuchen, eine stetige Kurve mittels Least-Square-Fit (z.B. Splines) durch die Daten zu legen und den Fehler durch einen χ^2-Test bestimmen.

Als nächstes betrachten wir $P_\infty(L,p)$: Am kritischen Punkt ist $\xi = \infty$ und die lineare Ausdehnung $R_s^{\text{perc. cl.}}$ des größten Clusters (largest cluster,l.c.) ist identisch mit der Systemgröße L (denn er perkoliert). Es gilt also: $p = p_c \Rightarrow \xi = \infty \Rightarrow R_s^{\text{perc. cl.}} = L$. Damit ist die Wahrscheinlichkeit, daß ein Bond zum perkolierenden Cluster gehört, einfach die Masse des perkolierenden Clusters dividiert durch das Gittervolumen:

$$\boxed{\Rightarrow P_\infty(L, p = p_c) = \frac{L^{d_f}}{L^d} = L^{-\beta/\nu}} \tag{4.12}$$

Durch doppelt-logarithmische Auftragung von $P_\infty(L, p = p_c)$ verschiedener L sollten wir also eine Gerade mit der Steigung $-\beta/\nu$ erhalten. Das ergibt, durch Least-Square-Fit einen Schätzwert (incl. Fehler) für β/ν.

Für $p \neq p_c$ gibt es zwei Grenzfälle:

$L \gg \xi$	$\Rightarrow$	P_∞ verhält sich fast wie im unendl. System	$\Rightarrow$	$P_\infty \sim (p - p_c)^\beta$
$L \ll \xi$	$\Rightarrow$	P_∞ verhält sich fast so wie eines mit $\xi = \infty$, d.h. bei $p = p_c$	$\Rightarrow$	$P_\infty \sim L^{-\beta/\nu}$

Da asymptotisch nur das Verhältnis der beiden Längenskalen L/ξ in $P_\infty(L,p)$ eingeht, ist die entsprechende FSS-Form

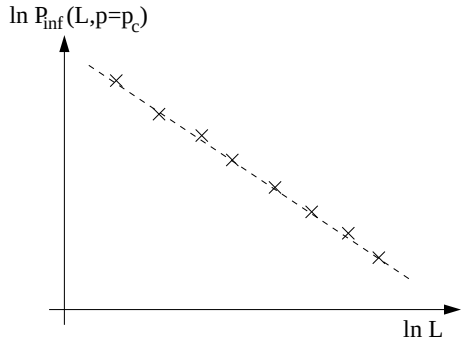


Abbildung 4.10: Die doppelt-logarithmische Auftragung von $P_\infty(L, p = p_c)$ erlaubt die Bestimmung von β/ν. Der Quotient ist gemäß Gl.(**??**) die Steigung der sich ergebenden Geraden.

$$\boxed{P_\infty(L,p) \sim L^{-\beta/\nu}\, \tilde{g}(L/\xi)} \tag{4.13}$$

$$\text{mit } \tilde{g}(x) \sim \begin{cases} x^{\beta/\nu} & \text{f. } x \ll 1 \\ \text{const.} & \text{f. } x \gg 1 \end{cases}$$

(denn für $L/\xi \ll 1$ ist dann $L^{-\beta/\nu}\, g(L/\xi) \sim L^{-\beta/\nu}(L/\xi)^{\beta/\nu} = \xi^{-\beta/\nu} = \delta^\beta$)

Analog verläuft die Behandlung von $S(L,p)$, der mittleren Masse der Cluster

$$\boxed{S(L, p = p_c) \propto L^{\gamma/\nu}} \tag{4.14}$$

$$L \ll \xi \Rightarrow S \sim (p - p_c)^{-\gamma}$$
$$L \gg \xi \Rightarrow S \sim L^{\gamma/\nu}$$

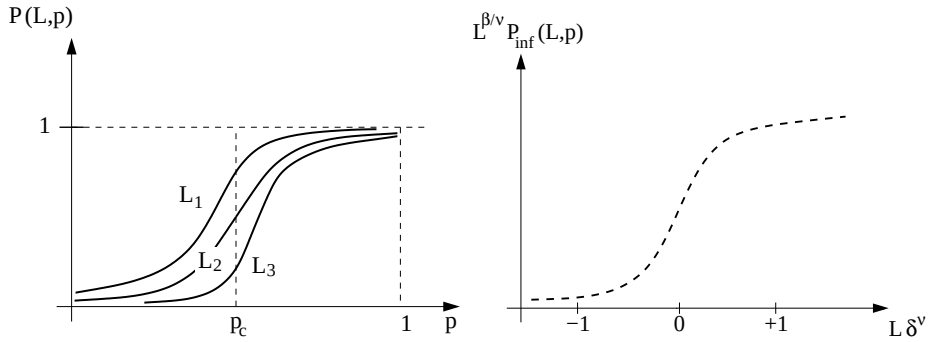


Abbildung 4.11: Links: P_∞ für verschiedene L. Rechts: Skalen-Plot, durch den man einen zweiten Schätzwert für ν erhält.

$$\boxed{\rightsquigarrow S(L,p) \sim L^{\gamma/\nu}\, \tilde{h}(L/\xi)} \tag{4.15}$$

$$\tilde{h}(L/\xi) \sim \begin{cases} x^{-\gamma/\nu} & \text{f. } x \ll 1 \\ \text{const.} & \text{f. } x \gg 1 \end{cases}$$

Die **fraktale Dimension** d_f erhält man einfach durch Analyse der Masse des perkolierenden Clusters im endlichen System

$$S^{\text{perc. cl.}}(p = p_c) \propto L^{d_f}$$

($\rightarrow$ log-log Plot von $S^{\text{p.c.}}$ vs. L bei $p = p_c$)
Eine unabhängige Bestimmung von ν erhält man durch Berechnung von ξ über R_s:

τ und σ, für die man Schätzwerte aus (4.8) erhält, wenn β und ν schon berechnet sind, kann man anhand von Skalenplots gemäß (4.2) für $S \ll L^{d_f}$ überprüfen: Für jeweils feste s sollte ein Plot $n_s(p) \cdot s^\tau$ versus $\delta^{1/\sigma} s$ einen Datenkollaps für verschiedene s erzeugen. Bei $p = p_c$ andererseits kann man n_s im log-log Plot (vs. s) betrachten, das ergibt einen Schätzwert für τ.

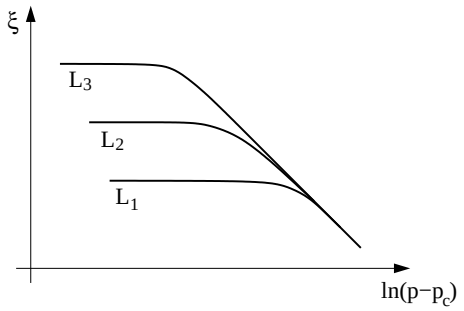


Abbildung 4.12: Man kann über den Wert von R_S die Korrelationslänge berechnen. Dies erlaubt die Bestimmung von ν im log-log Plot direkt aus der definierenden Gleichung (4.3).

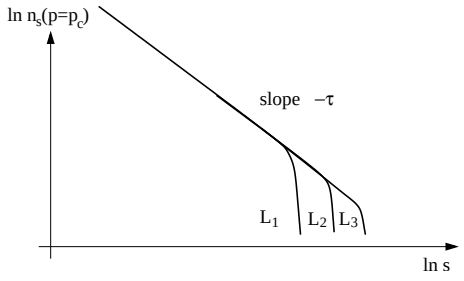


Abbildung 4.13: Unabhängige Bestimmung von τ: Am kritischen Punkt kann man im log-log Plot von n_s versus s den Exponenten τ aus der Geradensteigung bestimmen.

4.5 Verwandte Perkolations-Probleme

4.5.1 Epidemieen und Waldbrände

Der Prozess ist auf einem Quadratgitter definiert, jeder Gitterplatz entspricht einem Individuum. Mit Wahrscheinlichkeit p sei das Individuum infizierbar, mit Wahrscheinlichkeit $1 - p$ immun.

$t = 0$: Individuum auf zentralem Gitterplatz sei infiziert.
$t \to t + 1$: Infiziertes Individuum infiziert **alle** nicht-immunen Nachbarn.

Das Problem entspricht der Site-Perkolation:

$p < p_c$: Nur endliche Cluster $\rightsquigarrow$ Epidemie stoppt nach einer gewissen Zeit
$p > p_c$: Epidemie breitet sich über das gesamte System aus.

$$M(t) \quad = \quad \text{Anzahl infizierter Individuen} \propto t^{d_l} \tag{4.16}$$

$d_l \hat{=}$ Graph-Dimension, beschreibt wie die Cluster-Masse mit der **chemischen Distanz** l skalt. Die chemische Distanz l zwischen zwei Punkten entspricht der Länge des kürzesten Weges *auf dem Cluster*. Es gilt

$$l \sim r^{d_{\min}}, \tag{4.17}$$

wobei r den euklidischen Abstand zwischen den ausgewählten Punkten bezeichnet. (Für Cluster, die eine reguläre euklidische Gitterstruktur haben, z.B. Quadratgitter, gilt: $d_l = d_f = d$ und $d_{\min} = 1$)

$$R(t) \quad = \quad \text{Ausbreitungs} - \text{Radius} \propto M(t)^{1/d_f} \tag{4.18}$$

$$(??) \Rightarrow d_l = \frac{d_f}{d_{\min}}$$

Einige Werte für die verschiedenen Dimensionen, für $p \leq p_c$:

	$d = 2$	$d = 3$
$d_f \sim$	91/48	2.524
$d_l \sim$	1.678	1.84
$d_{\min} \sim$	1.13	1.374

Für $p > p_c$ und $R \gg \xi$ wird der Cluster kompakt, d.h. $d_l = d_f = 2$. Oberhalb von p_c ist daher in erster Linie die Ausbreitung der Front von Interesse. Diese wird durch die **Ausbreitungsgeschwindigkeit** quantifiziert. Sie ist gegeben durch $v = dR/dt \sim t^{d_l/d_f - 1}$. Im allgemeinen gilt $d_l \leq d_f$ (da der Radius des Clusters höchstens linear in t wächst). Ausserdem haben wir die Randbedingung $v \to 0$ für $p \to p_c$, da sich die Epidemie nur für $p > p_c$ für alle Zeiten weiter ausbreitet.

Für die Ausbreitung unterscheiden wir zwei Zeitskalen:

$$R(t) \propto t^{d_l/d_f}\, f(t/t_\xi) \propto \begin{cases} t^{d_l/d_f} & \text{für } t \ll t_\xi \\ t & \text{für } t \gg t_\xi \end{cases}$$

$$\Rightarrow f(x) \propto x^{1-d_l/d_f} \text{ für } x \gg 1$$

t_ξ ist die charakteristische Zeit um Cluster der Größe ξ zu generieren, d.h.:

$$t_\xi \propto \xi^{d_f/dl}$$

Die Bedeutung von t_ξ ergibt liegt in der unterschiedlichen Struktur des Clusters zu kleinen und großen Zeiten: Solange der Cluster kleiner als die Korrelationslänge ist, hat er eine fraktale Struktur, wenn er größer als ξ wird, ist er kompakt.
Insgesamt wächst die Front also wie:

$$\Rightarrow R(t) \propto t_\xi^{d_l/d_f-1} t \propto t\dot{\xi}^{1-d_f/d_l} \text{ für } t \gg t_\xi$$

und die Ausbreitungsgeschwindigkeit ist gegeben durch:

$$v \sim (p-p_c)^{-\nu(1-d_f/d_l)} \sim (p-p_c)^{\nu(d_{min}-1)}. \tag{4.19}$$

In obigem Modell stecken die infizierten Individuen **alle** nicht-immunen Nachbarn an.

Modifikation:
Es wird eine zusätzliche Wahrscheinlichkeit q eingeführt, daß ein infiziertes Individuum einen **nicht**-immunen Nachbarn ansteckt. Für $(p,q) < 1$ entspricht dies der Site-Bond-Perkolation, für $p = 1$ hat man reine Bond-Perkolation.
Kleine qs unterhalb der Perkolations-Schwelle können die Ausbreitung einer Epidemie verhindern. Als Beispiel sei hier die Immunschwächekrankheit AIDS genannt, deren Ausbreitung durch geeignete Schutzmaßnahmen zumindest eingedämmt werden konnte.

Eine andere Variante besteht darin, bei der Ausbreitung nicht mehr gleichzeitig alle nicht-immunen Nachbarn zu identifizieren, sondern einen

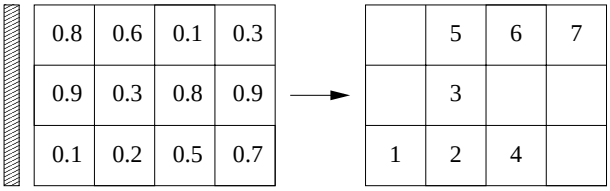
Nachbarn zufällig auszuwählen. Man kann dann auch "Lebensdauern" für angesteckte Sites einführen und erhält so für $p \geq p_c$ kritische Lebensdauern $\to \tau_c(p)$ für eine perkolierende Epidemie.

4.5.2 Invasions-Perkolation

Betrachte das Eindringen von Wasser in einem mit Öl gefüllten porösen Medium (z.B. ein Felsen). Da Wasser und Öl inkompressibel und nicht mischbar sind, muß das eindringende Wasser das Öl verdrängen. Das poröse Medium kann als Netzwerk von Poren angesehen werden, die durch enge Mündungen miteinander verbunden sind.

Modell: Ein anfangs unbesetztes $L \times L$ Quadratgitter repräsentiert das mit dem Öl ("Defender") besetzte poröse Medium. Das Wasser ("Invader") wird anfangs entlang einer Kante plaziert. Der Widerstand der Mündungen wird durch Zufallszahlen zwischen 0 und 1 modelliert.

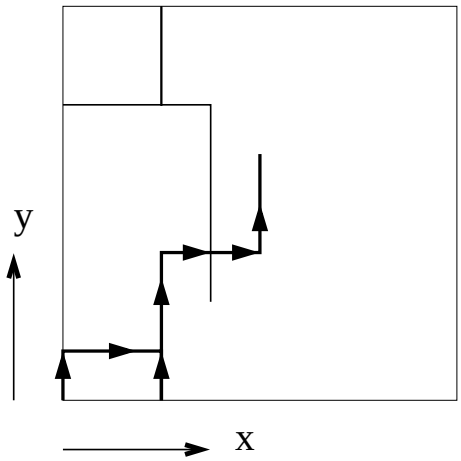
Das Eindringen des Wasser folgt dem Pfad des geringsten Widerstandes: In jedem Zeitschritt wird der Platz an der Oberfläche des eindringenden Wassers besetzt, der den kleinsten Widerstand hat. Wegen der Unmischbarkeit und Inkompressiblität der beiden Flüssigkeiten kann von Wasser umschlossenes Öl nicht mehr verdrängt werden.
Z.B.:



$\rightsquigarrow d_f \approx 1.82, d_l = 1.40$

Gerichtete Perkolation

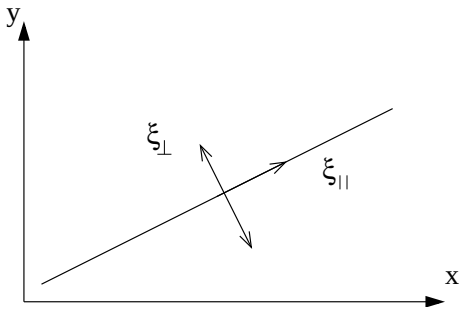
Gerichtete Perkolation kann z.B. als Modell für Waldbrand mit starkem Wind aus einer Richtung angesehen werden. Sie wird realisiert als Bond-Perkolation auf Quadratgitter mit Konzentration p. Zusätzlich wird jedem besetzten Bond eine Richtung zugeordnet:



Horizontale Bonds zeigen nach rechts, vertikale Bonds nach links. Illustration: Interpretiere besetzte Bonds als Leiter, die aber den Strom nur in **eine** Richtung durchlassen. $\rightsquigarrow p < p_c$ Isolator; $p > p_c$ Leiter.

$p_c =$ 0.644701 für Quadratgitter (größer als bei normaler Perkolation)
0.479 für ein Dreiecksgitter

Die Cluster-Struktur ist stark anisotrop, man hat daher zwei relevante Längenskalen $\xi_\perp$, $\xi_\parallel$. Die Cluster sind nicht selbstähnlich wie bei der gewöhnlichen Perkolation, sondern selbstaffine geometrische Objekte.



Für die Längenskalen hat man auch zwei unabhängige kritische Expo-

nenten:

$$\xi_\perp \sim |p - p_c|^{-\nu_\perp}$$
$$\xi_\parallel \sim |p - p_c|^{-\nu_\parallel}$$
$$\nu_\perp = 1.0972, \qquad \nu_\parallel = 1.7334 \text{ in 2d}$$

4.5.3 Die fraktale Dimension

Die Struktur des unendlichen Clusters bei $p = p_c$ entspricht der eines Fraktals. Ein Beispiel eines mathematischen Fraktals ist das sog. Sierpinsky Gasket.

Bildungsgesetz

Die fraktale Dimension des Sierpinsky Gasket erschließt sich aus der definierenden Relation

$$M(l) \sim l^{dj}$$

wobei $M(l)$ die Masse des fraktalen Objekts auf der Längenskala l bezeichnet. Wir ordnen nun jedem vollen Dreieck die Masse "1" zu, so dass wir

$$\begin{aligned} M(2) &= \text{const}\, 2^{dj} = 3 \\ \text{und}\, M(4) &= \text{const}\, 4^{dj} = 9 \end{aligned}$$

erhalten. Damit folgt sofort

$$\boxed{dj = \frac{\ln 3}{\ln 2} = 1.585}$$

Die obige Beziehung gilt insbesondere für $r = a\xi$ mit $a < 1$.

$$\Rightarrow P_\infty \sim \frac{\xi^{dj}}{\xi^d}$$

aus $P_\infty \propto (p-p_c)^\beta$ und $\xi \propto (p-p_c)^{-\nu}$ erhalten wir sofort die Beziehung

$$(p-p_c)^\beta \propto (p-p_c)^{\nu(d-dj)}$$

$$\Rightarrow \boxed{dj = d - \frac{\beta}{\nu}}$$

zwischen fraktaler Dimension und kritischen Exponenten.

Die fraktale Dimension beschreibt die Eigenschaften des unendlichen Clusters nicht vollständig. Ähnlich wichtig sind kürzeste Pfade auf dem Cluster.

Die Länge eines solchen Pfades skaliert mit dem euklidischen Abstand wie:

$$\boxed{l \sim r^{d_{\min}}}$$

Ähnlich wie beim Sierpinsky Gasket handelt es sich auch beim perkolierenden Cluster um ein fraktales Objekt. Wir hatten bereits gesehen, dass es bei $p = p_c$ Löcher auf allen Längenskalen gibt. Zudem ist der Cluster selbstähnlich für alle Längenskalen größer als die Gitterlänge. Die Masse des perkolierenden Clusters verhält sich also wie

$$\boxed{M(r) \propto r^{dj}}$$

Für $p > p_c$ gelten die perkolierenden Eigenschaften nur für $r < \xi$, so dass wir insgesamt

$$M(r) \sim \begin{cases} r^{dj} & r \ll \xi \\ r^d & r \gg \xi \end{cases}$$

erhalten. Man kann die fraktale Dimension in Beziehung zu den kritischen Exponenten setzen. Es gilt offensichtlich

$$P_\infty \sim \frac{r^{d_j}}{r^d} \quad r < \xi$$

(für $p = p_c$ entspricht das der Definition; für $r < \xi$ hat der unendliche Cluster die gleichen Eigenschaften wie der perkolierende Cluster bei $p = p_c$.)

Kapitel 5

Die Monte Carlo Simulation

Die stochastische oder Monte Carlo Simulation kann nicht nur für solche Systeme eingesetzt werden, die von Beginn an als stochastische Prozesse definiert sind, sondern auch zur approximativen Berechnung von hochdimensionalen Integralen und Reihensummen. Diese Eigenschaft der Monte Carlo Simulation macht sie für die Anwendungen in der Physik von Vielteilchensystemen interessant.

5.1 Die Monte Carlo Integration

Die numerische Integration wird gewöhnlich durch Auswertung des Integranden $f(x)$ an verschiedenen Stützstellen x_i durchgeführt. Ein Beispiel für diese Technik ist die Regel von Simpson: Es sei $I = \int_a^b dx\, f(x)$ das zu berechnende Integral. Den Wert des Integrals kann man durch die Reihe

$$I \approx \frac{h}{3}\left[f(a) + 4f(a+h) + 2f(a+2h) + \cdots + 4f(b-h) + f(b)\right]$$

approximieren. Die Ungenauigkeit dieser Methode ist von der Ordnung $\Theta(h^4)$ mit $h = \frac{a-b}{N}$.

Alternativ lässt sich das Integral auch stochastisch berechnen. Dazu erzeugt man zufällig Stützstellen, die homogen im Intervall $[a, b]$ verteilt

sind, so dass man das Integral durch

$$I_{MC} = \frac{(b-a)}{N} \sum_{i=1}^{N} f(x_i)$$

abschätzen kann. Für große Werte von N konvergiert der Schätzwert I_{MC} gegen den wahren Wert von I. Der Fehler von I_{MC} lässt sich durch die Varianz:

$$\sigma^2 = \left\langle \left(\frac{(a-b)}{N} \sum_{i=1}^{N} f(x_i) \right)^2 \right\rangle - \left(\left\langle \frac{a-b}{N} \sum_{i=1}^{N} f(x_i) \right\rangle \right)^2$$

abschätzen, wobei die spitzen Klammern $\langle \cdots \rangle$ die Mittelung über verschiedene Sequenzen der Zufallszahlen bezeichnet. Nach dem zentralen Grenzwertsatz erwarten wir $\sigma \sim 1/\sqrt{N}$. Dies ist im Vergleich zu Standardmethoden der numerischen Integration, die einen Fehler $\sim N^{-k}$ mit $k > 1$ (Simpson-Regel $k = 4$) aufweisen, unakzeptabel.

Interessant wird die Methode aber in höheren Dimensionen. Wenn das Integral über einen d-dimensionalen Hyperkubus der Kantenlänge L definiert ist, muss man die Funktion bei dem üblichen numerischen Verfahren an $N = (L/h)^d$ Stützstellen auswerten. Wenn nun der Fehler der Methode von der Ordnung $\Theta(h^k)$ ist, konvergiert die Methode insgesamt wie $N^{-k/d}$. Das Ergebnis $\sigma \sim N^{-1/2}$ bei der Monte Carlo Integration gilt aber unabhängig von der Dimension, so dass sie schneller konvergiert, falls $d > 2k$!

Dieses überraschende Resultat liegt darin begründet, dass durch die zufällige Wahl der Stützpunkte das Integrationsvolumen homogener ausgefüllt wird als durch ein reguläres Gitter. Wenn wir z.B. eine Dichte ρ von Stützpunkten $\underline{x}_i$ vorgeben, so wächst bei einem regulären Gitter die Zahl der Stützpunkte immer dann sprunghaft an, wenn eine Kantenlänge das Integrationsvolumen um ein Vielfaches von h überschreitet. Bei der zufälligen Wahl der Stützpunkte sind solche Sprünge für große N sehr unwahrscheinlich.

5.2 Importance Sampling

Hauptursache für die statistischen Fluktuationen bei der stochastischen Berechnung von Integralen ist die Tatsache, dass die Funktion homogen gesampled wird, auch wenn das größte Gewicht der Funktion in einem engen Teilintervall liegt. Dieses Problem kann vermieden werden, wenn man eine Funktion $\rho(x)$ findet für die gilt:

- $\int_a^b dx\rho(x) = 1$
- $f(x)/\rho(x) \approx$ const im Intervall $[a, b]$

Das Integral kann dann berechnet werden, indem man

$$\int_a^b dx\, f(x) = \int_a^b dx\, \rho(x) \left[\frac{f(x)}{\rho(x)}\right]$$

auswertet.

Man generiert dann die Punkte nicht gleichverteilt, sondern gemäß der Wahrscheinlichkeitsdichte $\rho(x)$. Durch dieses Verfahren werden dann mit hoher Wahrscheinlichkeit dort Punkte generiert, wo die Funktion den größten Absolutwert erreicht. Dieses Verfahren reduziert die Schwankungen erheblich.

algorithm (importance sampling)
begin
  Initialize
    I=0;
  for (i< N) **do**
    I=I+f(x)/ ρ(x);
    end db= (b-a) * I/N;
    print I;
end

5.3 Importance Sampling und Markov Prozesse

Wir betrachten zunächst das kanonische bzw. NVT Ensemble (d.h. wir halten die Teilchenzahl N, das Volumen V und die Temperatur T während der Simulation fest).

Die Wahrscheinlichkeit, eine bestimmte Konfiguration x zu finden, ist proportional zu

$$\rho(x) \propto \exp\left(-\beta E(x)\right)$$

Eine Möglichkeit zur Simulation besteht darin, dass man zufällig Konfiurationen generiert und diese mit der Wahrscheinlichkeit $\exp\left(-(x)\right)$ akzeptiert (wir nehmen an, dass wir insbesondere bei niedrigen Temperaturen extrem niedrige Akzeptanzraten haben.

Ausweg: Wir generieren neue Konfigurationen nicht mehr unabhängig, sondern mit einer Wahrscheinlichkeitsverteilung, die von der aktuellen Konfiguration abhängt. Die bedeutet, dass die Wahrscheinlichkeit $P_N(x_1, x_2, \ldots, x_N)$ die Konfigurationen $x_1, \ldots, x_N$ zu erzeugen, durch

$$P_N(x_1, x_2, \ldots, x_N) = P_1(x_1)\, T(x_1 \to x_2) \ldots T(x_{N-1} \to x_N)$$

gegeben ist. Die Übergangswahrscheinlichkeiten sind normiert, d.h.

$$\sum_{x'} T(x \to x') = 1.$$

Ein *Beispiel* für einen solchen **Markov-Prozess** ist der Random Walk.
Die Markov-Eigenschaft des Prozesses besagt, dass die Übergangswahrscheinlichkeit nur von der aktuellen Konfiguration abhängt (Gegenbeispiel: SAW: Die Übergangswahrscheinlichkeiten hängen von der gesamten Historie ab.)

Ziel: Generierung einer Markovkette, so dass

- die Konfigurationen x mit einer Wahrscheinlichkeit $\propto \exp\left(-\beta E(x)\right)$ erzeugt werden

- die Wahrscheinlichkeitsverteilung unabhängig von der Anfangskonfiguration ist

Die *Unabhängigkeit* von den Anfangsbedingungen wird dadurch erreicht, dass:

- jede Konfiguration nach einer endlichen Zahl von Schritten von jeder beliebigen Konfiguration erreicht werden kann

- keine Perioden auftreten (Zufallszahlengenerator)

Nun müssen wir noch sicherstellen, dass die Konfigurationen tatsächlich gemäß der Boltzmann-Verteilung generiert werden.

Es sei $\rho(x,t)$ die Wahrscheinlichkeit, die Konfiguration x nach t Schritten zu generieren. Dann gilt

$$\rho(x,t+1) - \rho(x,t) = -\sum_{x'} T(x \to x')\,\rho(x,t) + \sum_{x'} T(x' \to x)\,\rho(x',t)$$

(Mastergleichung)

Wenn nun die stationäre Verteilung $\rho(x)$ erreicht ist, gilt:

$$\sum_{x'} T(x \to x')\,\rho(x) = \sum_{x'} T(x' \to x)\,\rho(x'). \tag{5.1}$$

Diese Gleichung ist im Allgemeinen schwer zu lösen. Eine spezielle Lösung kann man aber sofort angeben

$$T(x \to x')\,\rho(x) = T(x' \to x)\,\rho(x'), \tag{5.2}$$

d.h. die Stationaritätsbedingung (5.1) ist sogar paarweise erfüllt. (Die Bedingung 5.2 nennt man **detailliertes Gleichgewicht / detailed balance**).

Es sei nun

$$T(x \to x') = \omega_{xx'}\,A_{xx'},$$

wobei $\omega_{xx'} = \omega_{x'x}$, $0 \leq \omega_{xx'} \leq 1$ und $\sum_{x'} \omega_{xx'} = 1$.

Damit ergibt sich für A aus der Bedingung (5.1):

$$\frac{A_{xx'}}{A_{x'x}} = \frac{\rho(x')}{\rho(x)}$$

Wir identifizieren die symmetrische Wahrscheinlichkeit $\omega_{xx'}$ als Wahrscheinlichkeit, eine Änderung der Konfiguration $x \to x'$ zu erzeugen und $A_{xx'}$ als Wahrscheinlichkeit, die neue Konfiguration auch anzunehmen.

Die Wahl von **Metropolis** für $A_{xx'}$ ist:

$$A_{xx'} = \min\,(1, \rho(x')/\rho(x))$$

Die Simulation mit Hilfe der Markovkette impliziert, dass die Konfigurationen korreliert sind. Im Limes großer Zeiten (die aber praktisch unerreichbar sein können) ist die statistische Unabhängigkeit garantiert.

Wenn wir nun den Mittelwert $\langle A \rangle$ einer Observablen A bestimmen wollen, so gilt:

$$\langle A \rangle = \frac{1}{n - n_0} \sum_{\nu > n_0}^{n} A_\nu$$

wobei n_0 die Zahl der MC-Schritte bis zum Erreichen des Gleichgewichts, A_ν den Wert von A beim Zeitschritt ν und n die Länge der Markovkette bezeichnet.

Wichtig: Die Messung von A erfolgt nach jedem Zeitschritt und nicht nur dann, wenn sich die Konfiguration ändert, da die Konfigurationen gemäß ihrem statistischen Gewicht erzeugt werden.

Bemerkungen:
1) Es ist auch möglich $\omega_{xx'}$ unsymmetrisch zu wählen; dann gilt

$$A_{xx'} \quad = \quad \min\,(1, q_{xx'}) \quad \text{wobei } q_{xx'} = \frac{\omega_{x'x}\rho(x')}{\omega_{xx'}\rho(x)}$$

2) Neben dem Metropolis-Algorithmus gibt es auch noch andere Akzeptanzwahrscheinlichkeiten, die die detaillierte Bilanz erfüllen. Ein Beispiel

ist der *Heat-Bath Algorithmus*. Beim Heat-Bath Algorithmus werden nur wenige Freiheitsgrade geändert. Es sei $E = E(x)$ und $E' = E(x')$. Dann erfüllt die Akzeptanzwahrscheinlichkeit

$$P(x \to x') \propto \exp\left(-\beta(E' - E)/2\right)$$

für symmetrisches $\omega_{xx'}$ die detaillierte Bilanz.

Beim Heat-Bath Algorithmus wird ein Spin zufällig ausgewählt und unabhängig von seinem aktuellen Wert wird die neue Orientierung zufällig bestimmt. Wenn der Spin i (bzw. der lokale Zustand) q verschiedene Zustände annehmen kann, lautet die Wahrscheinlichkeit einen beliebigen Zustand n aus den q Möglichkeiten zu wählen

$$p_n = \frac{e^{-\beta E_n}}{\sum_{m=1}^{q} e^{-\beta E_m}},$$

wobei E_n die Energie des Systems ist, wenn der i-te Spin den Wert der Orientierung n annimmt.

Man kann leicht sehen, dass die Akzeptanzraten Detailed Balance erfüllen, denn:

$$\frac{P(n \to n')}{P(n' \to n)} = \frac{p_{n'}}{p_n} = \frac{e^{-\beta E_{n'}}}{\sum_{m=1}^{q} e^{-\beta E_m}} \frac{\sum_{m=1}^{q} e^{-\beta E_m}}{e^{-\beta E_n}} = e^{-\beta(E_{n'} - E_n)}$$

5.4 Einfache Anwendungen der Monte Carlo Methode

5.4.1 Das Ising-Modell

Wir betrachten das Ising-Modell auf einem zweidimensionalen Quadratgitter. Auf jedem Gitterplatz befinde sich ein Spin s_i mit $s_i \in \{-1, 1\}$. Die Hamiltonfunktion des Systems lautet

$$\mathcal{H}(\{s_i\}) = -J \sum_{\langle i,j \rangle} s_i s_j - h \sum_i s_i$$

wobei durch $\sum_{\langle i,j \rangle}$ die Summation über nächste Nachbarn bezeichnet wird.

Die Monte Carlo Simulation des Ising-Modells ist denkbar einfach. Man wählt einen Spin aus und ändert die Orientierung. Dann bezeichnet man die Energieänderung ΔE. Der Spinflip wird dann mit der Metropolis- oder Heat-Bath-Wahrscheinlichkeit angenommen. Für das Ising-Modell lauten die Auswahlwahrscheinlichkeiten des Heat-Bath Algorithmus:

$$p_{up} = \frac{e^{-\beta E_{up}}}{e^{-\beta E_{up}} + e^{-\beta E_{\text{down}}}} \; ; \; p_{\text{down}} = \frac{e^{-\beta E_{\text{down}}}}{e^{-\beta E_{up} + e^{-\beta E_{\text{down}}}}}$$

Die Summation über alle Bonds können wir durch Ausklammern von $e^{\frac{1}{2}\beta(E_{up}+E_{\text{down}})}$ vermeiden. Wir suchen nun die Wahrscheinlichkeit für einen Spinflip von $\downarrow$ nach $\uparrow$, die durch $A = p_{up}$ gegeben ist. Mit $\Delta E = E_{up} - E_{\text{down}}$

$$\begin{aligned} A &= \frac{e^{-\beta E_{up}}}{e^{-\beta E_{up}} + e^{-\beta E_{\text{down}}}} \\ &= \frac{e^{-\beta(E_{up} - \frac{1}{2}E_{up} - \frac{1}{2}E_{\text{down}})}}{e^{-\frac{1}{2}\beta\Delta E} + e^{\frac{1}{2}\beta\Delta E}} = \frac{e^{-\frac{1}{2}\beta\Delta E}}{e^{-\frac{1}{2}\beta\Delta E} + e^{\frac{1}{2}\beta\Delta E}} \end{aligned}$$

mit $\Delta E = E_{up} - E_{\text{down}}$. Die Akzeptanzrate für einen Spinflip von $\uparrow \rightarrow \downarrow$ ist ebenfalls A, wobei $\Delta E = E_{\text{down}} - E_{up}$.

Hinweise für die effiziente Simulation

(1) Das effektive Feld

$$h_i = -J \sum_j s_j - h$$

sollte abgespeichert werden, wobei $\sum_j$ die Summe über die nächsten Nachbarn bezeichnet. Dies erleichtert die Berechnung der Energiedifferenz ($\Delta E(s_i \rightarrow -s_i) = -2h_i s_i$). Wird der Spinflip angenommen, müssen die effektiven Felder der Nachbarn korrigiert werden ($h_j \rightarrow h_j + 2J\, s_i$).

(2) Man kann die Übergangswahrscheinlichkeiten für die verschiedenen Werte von h_i tabellieren.

5.4.2 Monte Carlo Simulation eines monoatomaren Gases

Bei der Berechnung der Zustandssumme Z haben wir gesehen, dass der nichttriviale Anteil von Z durch den Potentialterm bestimmt wird. Damit können zur Bestimmung der Gleichgewichtseigenschaften die Geschwindigkeiten der Teilchen vernachlässigt werden und wir können uns auf die Generierung von verschiedenen Konfigurationen konzentrieren.

Für ein Wechselwirkungspotential $U(\underline{r}^N)$ ($\underline{r}^N$ steht für die Gesamtheit der Teilchenpositionen) ist die Struktur des Algorithmus folgendermaßen gegeben:

(i) Wähle zufällig ein Teilchen i.

(ii) Bestimme einen zufälligen Verschiebungsvektor $\Delta\underline{r}$, so dass $\underline{r}_i \to \underline{r}_i + \Delta\underline{r}$ und berechne die Energie der geänderten Konfiguration $\underline{r}'^N$.

(iii) Akzeptiere die neue Konfiguration mit der Wahrscheinlichkeit $\min\,(1, \exp[-\beta(U(\underline{r}'^N) - U(\underline{r}^N))])$.

Bemerkung:
1) Die Heat-Bath Wahrscheinlichkeit ist nicht praktikabel für Systeme mit kontinuierlichen Freiheitsgraden.
2) Die Schrittweite sollte so gewählt werden, dass die Akzeptanzrate $\in [0.4, 0.6]$ liegt.

Wir wollen nun einige Details der Implementierung diskutieren:

Wahl der Randbedingungen

Ziel der Simulationen ist es, die makroskopischen Eigenschaften des Systems zu bestimmen. In der Praxis ist es aber nur möglich, Systeme in der Größenordnung von einigen 1000 Teilchen zu simulieren, so dass man Randeffekte nicht vernachlässigen kann. Man wendet daher häufig periodische Randbedingungen an, so dass es in dem System keine Randplätze mehr gibt.

Bild einführen

Die Einführung periodischer Randbedingungen ist dann von besonderem Nutzen, wenn die Wechselwirkung kurzreichweitig sind. Bei langreichweitigen Wechselwirkungen kann es zu Selbstwechselwirkungen kommen, die die Berechnung der Akzeptanzraten erschweren. In der Praxis löst man dieses Problem durch geeignetes Abschneiden der Randbedingungen.
Auch bei einer periodischen Box haben Größe und Form Einfluss auf das Simulationsergebnis. Die Position der Box dagegen nicht.

5.5 Nachbarschaftslisten

Im Gegensatz zum Ising-Modell, bei dem sich die Änderung der Energie durch den Spinflip aus der Orientierung der nächsten Nachbarn ergibt, muss bei kontinuierlichen Systemen im Prinzip über alle Teilchenpositionen summiert werden.

Die Laufzeit des Algorithmus kann aber für Systeme, bei denen die Wechselwirkung auf einen endlichen Radius r_c beschränkt ist, durch Einführung von Nachbarschaftslisten erheblich reduziert werden.

Dies geschieht am einfachsten durch Einführung eines Gitters mit Gitterkonstanten $a > r_c$. Man ordnet dann jeder Teilchenposition eine Box des Gitters zu. Damit kann die Änderung der potentiellen Energien durch Summation über alle Teilchen in der gleichen bzw. in benachbarten Boxen durchgeführt werden.

Bild fehlt

5.6 Monte Carlo Simulationen in verschiedenen Ensemblen

Das kanonische Ensemble ist für die meisten Problemstellungen die natürliche Wahl. Es kann aber für verschiedene Fragestellungen vorteilhaft sein, zu einem anderen Ensemble überzugehen. Wir wollen einige dieser Möglichkeiten diskutieren.

5.6.1 Das (NPT)-Ensemble

Das NPT-Ensemble entspricht einer Situation, die experimentell häufig realisiert wird, nämlich den Druck statt des Volumens festzuhalten.

Der Mittelwert einer Messgröße A im NPT-Ensemble ist gegeben durch:

$$\langle A\rangle_{\mathrm{NPT}} = \frac{\int_0^\infty dV\, e^{-\beta PV} \int dR\, A(R)\, e^{-\beta U(R)}}{Q(N,P,T)}$$

wobei $Q = \int dV\, e^{-\beta PV}\, Z(N,V,T)$.

Die Realisierung des Druckensembles bedeutet, dass man das Volumen der Simulationsbox variieren muss. Die Variation des Volumens kann aber nicht unabhängig von den Teilchenpositionen erfolgen, da es entsprechend unwahrscheinlich ist, für das neue Volumen gültige Konfigurationen zu erzeugen.

Praktikabler ist es, die Teilchenpositionen entsprechend der Volumenänderung zu reskalieren. Wir betrachten nun speziell eine kubische Simulationsbox mit $V = L^3$. Die Teilchenkoordinaten seien nun durch $\underline{T}_i = \underline{S}_i L$ parametrisiert.
Damit gilt:

$$\langle A\rangle_{(\mathrm{NPT})} = \frac{\int_0^\infty dV\, e^{-\beta PV} V^N \int ds\, A(LS)\, e^{-\beta U(LS)}}{Q(N,P,T)}$$

wobei S die Positionen $\underline{s}_1, \ldots, \underline{s}_N$ bezeichnet, die bei der Volumenänderung nicht verändert wurden. Das Boltzmanngewicht im NPT-Ensemble lautet daher:

$$\rho(V,S) = e^{-\beta PV} V^N\, e^{-\beta U(LS)},$$

so dass die Akzeptanzrate für eine Volumenänderung durch

$$\exp[-\beta P\,(V_{\mathrm{new}} - V_{\mathrm{old}})] \left(\frac{V_{\mathrm{new}}}{V_{\mathrm{old}}}\right)^N \exp[-\beta(U(L_{\mathrm{new}}S) - U(L_{\mathrm{old}}S))]$$

gegeben ist.

Die Bestimmung der Differenz der potentiellen Energie ist im allgemeinen Fall rechenzeitintensiv, da man über alle Teilchenpositionen summieren kann. Für ein Wechselwirkungspotential aber, das als eine Linearkombination von Potenzen von $(\sigma/r)^k$ darstellbar ist, ist sie aber einfach, da gilt

$$\sigma^k \sum_{i<j} (L_{\text{after}} S_{ij})^{-k} = \sigma^k \left[\sum_{i<j} (L_{\text{before}} S_{ij})^{-k} \right] (\text{Ł}_{\text{after}}/L_{\text{before}})^{-k}, \qquad (5.3)$$

d.h. man muss nur wenige Summanden reskalieren.

Bsp. Das Lennard-Jones-Potenzial $U(r) = 4[r^{-12} - r^{-6}]$. Man speichert die Werte von $U_1 = 4 \sum_{i<j} (LS_{ij})^{-12}$ und $U_2 = 4 \sum_{i<j} (LS_{ij})^{-6}$ separat ab und reskaliert gemäß (5.3).

5.6.2 Das großkanonische Ensemble

Im **großkanonischen** Ensemble ist der Mittelwert einer Messgröße A durch den Ausdruck

$$\langle A \rangle_{\mu VT} = \frac{\sum_{N=0}^{\infty} \frac{1}{N!} e^{\beta\mu N} \Lambda^{-3N} \int dR_N \, A(R_N) \, e^{-\beta U(R_N)}}{Z_g(\mu, V, T)}$$

mit der großkanonischen Zustandssumme

$$\begin{aligned} Z_g(\mu, V, T) &= \sum_{N=0}^{\infty} \frac{1}{N!} e^{\beta\mu N} \int dR_N \, A(R_N) e^{-\beta U(R_N)} \\ \Lambda &= \frac{h}{\sqrt{2\pi m \, k_B T}} \end{aligned}$$

Das chemische Potential bezeichnet die mittlere Energie pro Teilchen. Der Wert von μ bestimmt die mittlere Anzahl der Teilchen im System, ähnlich wie der Druck das mittlere Volumen. Der Faktor $N!$ stammt aus

der Quantenmechanik und resultiert aus der Ununterscheidbarkeit der Teilchen.

Damit ist das relative Gewicht einer Konfiguration gegeben durch:

$$\rho(N, R_N) = e^{-\beta U(R_N)} \Lambda^{-3N} / N!\, e^{\beta \mu N}$$

Der Metropolis-Algorithmus hat die folgende Struktur

(1) Bestimmen Sie zufällig, welche der Schritte

- Teilchenerzeugung
- Teilchenvernichtung
- Teilchenbewegung

erfolgen soll.

(2) **Teilchenerzeugung**

Es wird zufällig eine Position des neuen Teilchens ausgewählt und die Potentialenergiedifferenz

$$\Delta U^+ = U(R_{N+1}) - U(R_N)$$

bestimmt. Da die Wahrscheinlichkeit ein Teilchen in einem Volumenelement zu plazieren durch $\frac{d^3 T_{N+1}}{V}$ gegeben ist, akzeptieren wir die Teilchenerzeugung mit der Wahrscheinlichkeit

$$\min\left(\frac{V}{\Lambda^3 (N+1)}\, e^{-\beta \Delta U^+} e^{\beta \mu}\, ;\, 1\right)$$

Teilchenvernichtung

- Zufällige Auswahl eines Teilchens
- Berechnung der Potentialenergiedifferenz

$$\Delta U^- = U(R_{N-1}) - U(R_N)$$

- Akzeptanz mit

$$\min\left(1, e^{-\beta\Delta U^-}\Lambda^3 N/V e^{-\beta\mu}\right)$$

Teilchenbewegung: Die Teilchenbewegung erfolgt wie im (N, V, T)-Ensemble.

5.6.3 Das Gibbs-Ensemble

Die Untersuchung von Phasengleichgewichten ist mit der bisherigen Methode schwierig, da die Teilchen in den verschiedenen Phasen die gleiche Temperatur, den gleichen Druck und das gleiche chemische Potential besitzen müssen. Man kann aber nicht nur die intensiven Variablen fixieren, da sonst das Volumen und die Teilchenzahl unbeschränkt sind.

Eine Methode diese Schwierigkeiten zu überwinden, ist die Simulation im Gibbs-Ensemble bei der die Teilchen in zwei Subsysteme aufgeteilt werden, so dass sie keine Barriere in der freien Energie zu überwinden haben. Die Ideee der **Gibbs-Ensemble** Methode ist die folgende: Man teilt das System in zwei Subvoluminen V_1, V_2 auf, zwischen denen Teilchen ausgetauscht werden können. Das Gesamtsystem ist aber im (N, V, T)-Ensemble.

Skizze fehlt

Weiterhin halten wir fest, dass es keine Wechselwirkung zwischen den Teilvolumen gibt, d.h. die Akzeptanzraten für den Teilchenaustausch hängt nur von den Energieänderungen in den jeweiligen Teilchenvolumina selbst ab. Neben dem Teilchenaustausch werden auch noch die üblichen MC-Moves und Änderungen der Teilvoluminen durchgeführt.

Wie im NPT-Ensemble werden die Teilchenpositionen reskaliert, so dass $R_k = L_k S_k$ (mit $k = 1, 2$) und $V_k = L_k^3$. Das Gewicht einer Konfiguration mit $(V_1, V_2, N_1, N_2, S_1, S_2)$ ist

$$\rho\left(V_1, N_1, S_1, V_2, N_2, S_2\right) = \frac{V_1^{N_1}}{N_1!}\,\frac{V_2^{N_2}}{N_2!}\,e^{-\beta k(L_1 S_1)} e^{-\beta k(L_2 S_2)}$$

Durch dieses Boltzmanngewicht werden die Akzeptanzraten für die MC-Moves bestimmt:

Für die Simulation muss man zunächst die apriori Wahrscheinlichkeiten für

- Teilchenbewegung in den Subsystemen
- Teilchenaustausch **zwischen** den Teilsystemen (jeweils ein Teilchen, gleiche Wahrscheinlichkeit für alle Teilchen)
- relative Volumenänderungen

setzen. Die **Akzeptanzraten** in den einzelnen Schritten lauten dann

(i) **Teilchenbewegung**

$$\min\left(1, \exp\{-\beta[U(LS_k^{\text{new}}) - U(LS_k^{\text{old}})]\}\right)$$

(ii) **Volumenänderungen**

$$\min\left(1, \prod_{k=1,2} \exp\{-\beta[U(L_k^{\text{new}} S_k) - U(L_k^{\text{old}} S_k)]\} \left(\frac{V_k^{\text{new}}}{V_k^{\text{old}}}\right)^N\right)$$

(iii) **Teilchenaustausch** (Transfer von $1 \rightarrow 2$)

$$\min\left(1, \exp[-\beta(\Delta U_2^+ - \Delta U_1^-)] \frac{N_1}{N_2 + 1} \frac{V_2}{V_1}\right)$$

mit $\Delta U_2^+ =$ Energieänderung im Teilsystem 2 durch das Hinzufügen eines Teilchens; U_1^- entsprechend.

Bemerkung: Das Problem dieser Methode ist das gleiche wie bei der großkanonischen Simulation, nämlich die Schwierigkeit bei hohen Dichten Teilchen einzufügen. Damit eignet sich die Methode nicht sehr gut, um flüssig-fest Phasengleichgewichte zu simulieren.

5.7 Cluster-Algorithmen

Die Simulationen des Ising-Modells mit der Single-Spin-Flip-Dynamik werden durch das sogenannte Critical-Slowing-Down erschwert, d.h. die Korrelationszeiten wachsen gemäß

$$\xi^z$$

an, wobei z den sogenannten dynamischen Exponenten bezeichnet. Für die meisten Systeme und Algorithmen ist $z \approx 2$. Für die effiziente Analyse ist es natürlich erstrebenswert, den dynamischen Exponenten z zu verkleinern. Dies wird durch die Einführung von Cluster-Update erreicht.

5.7.1 Der Swendsen-Wang Algorithmus

1987 haben Swendsen und Wang einen Algorithmus vorgestellt, der für das Ising-Modell einen dynamischen Exponenten $z \in [0,2; 0.27]$ hat und damit signifikant kleiner als der Wert $z = 2.125$ des Standard Algorithmus ist.

Der Swendsen-Wang Algorithmus hat die folgende **Struktur**:

1. Zwischen benachbarten Spins, die eine unterschiedliche Orientierung haben, wird kein Link gesetzt.

2. Wenn die Spins die gleiche Orientierung besitzen, wird mit der Wahrscheinlichkeit $p = 1 - e^{-2\beta J}$ ein Link gesetzt.

Diese Prozedur wird solange fortgesetzt, bis alle nächsten Nachbarpaare aktualisiert worden sind. Durch die Links werden Cluster erzeugt, denen eine zufällige, aber gemeinsame Orientierung zugeordnet wird.

Die Ergodizität des Algorithmus ist offensichtlich, da Übergänge zwischen beliebigen Konfigurationen mit endlicher Wahrscheinlichkeit stattfinden.

Wir müssen daher zeigen, dass der Algorithmus die Bedingung der detaillierten Bilanz erfüllt. Dazu betrachten wir zunächst das Setzen **eines** Links. Wenn ein Link gesetzt ist, bedeutet dies, dass die Wechselwirkungsenergie V_{ij} für den Bond (ij) $V_{ij} = \infty$ ist, im anderen Fall ist

$V_{ij} = 0$. Damit ergibt sich für den Hamiltonian

$$\mathcal{H} \to \mathcal{H}_0 + V_{ij},$$

wobei $\mathcal{H}_0$ den ursprünglichen Hamiltonian ohne den Bond ij bezeichnet und $\mathcal{H}$ den ursprünglichen Hamiltonian.

Wir betrachten nun die detaillierte Bilanz für den Übergang von $S \to S'$

$$\frac{T(S \to S')}{T(S' \to S)} = e^{-\beta[\mathcal{H}(S')-\mathcal{H}(S)]} = \frac{T_0(S \to S')}{T_0(S' \to S)} e^{\beta J(S'_j S'_i - S_j S_i)}$$

wobei wir die Aufspaltung so gewählt haben, dass wir die Analyse des SWA für einen Link ausführen können.

Die Übergangswahrscheinlichkeit:

$$T(S \to S')$$

kann in folgender Weise zerlegt werden:

$$T(S \to S') = T_f(S \to S')P_f(S) + T_d(S \to S')P_d(S'),$$

wobei "f" für das Setzen ("freeze") und "d" für das Löschen eines Links steht.

Für einen gelöschten Bond ist $T_d(S \to S') = T_0(S \to S')$. Für die Übergangswahrscheinlichkeit $T_f(S \to S')$ gilt:

$$T_f(S \to S') = \begin{cases} T_0(S \to S') & \text{falls } S'_i = S'_j \\ 0 & \text{falls } S'_i \neq S'_j \end{cases}$$

Bem.: $S_i = S_j$ muss erfüllt sein, da sonst der Bond nicht eingefroren wäre.

Wir können nun die detaillierte Bilanz in der Form

$$\begin{aligned} \frac{T(S \to S')}{T(S' \to S)} &= \frac{T_f(S \to S')P_f(S) + T_d(S \to S')P_d(S)}{T_f(S' \to S)P_f(S') + T_d(S' \to S)P_d(S')} \\ &= e^{-\beta[\mathcal{H}(S')-\mathcal{H}(S)]} \end{aligned}$$

darstellen. Wir betrachen nun die verschiedenen Möglichkeiten für die Spineinstellungen:
1. Fall: $S_i = S_j\,;\, S_i' = S_j'$

$$\begin{aligned} &\Rightarrow \quad P_d = \exp[-2\beta J] \text{ und } P_f = 1 - \exp[-2\beta J] \\ &\Rightarrow \quad \frac{T(S \to S')}{T(S' \to S)} = \frac{T_0(S \to S')[1 - \exp(-2\beta J)] + T_0(S \to S')\exp[-2\beta J]}{T_0(S' \to S)[1 - \exp(-2\beta J)] + T_0(S' \to S)\exp[-2\beta J]} \\ &= \quad \frac{T_0(S \to S')}{T_0(S' \to S)} \end{aligned}$$

so dass, wegen $e^{-\beta J(S_i'S_j' - S_iS_j)} = 1$ die detaillierte Bilanz erfüllt ist.
2. Fall:$S_i \neq S_j\,;\, S_i' \neq S_j'$
Damit ergibt sich

$$\begin{aligned} & P_f(S) = P_f(S') = 0\,;\, P_d(S) = P_d(S') = 1 \\ \Rightarrow \quad & \frac{T(S \to S')}{T(S' \to S)} = \frac{T_0(S \to S')}{T_0(S' \to S)} \text{ und damit ist wieder wg.} \\ & e^{-\beta J(S_i'S_j' - S_iS_j)} = 1 \text{ die detaillierte Bilanz erfüllt} \end{aligned}$$

3. Fall: $S_i = S_j\,;\, S_i' \neq S_j'$

$$\Rightarrow \quad P_d(S) = \exp[-2\beta] = 1 - P_f(S)\,;\, P_d(S') = 1$$

Die Übergangswahrscheinlichkeit $T_f(S \to S') = 0$, da die Spins in S' eine unterschiedliche Orientierung haben, was mit dem "Einfrieren" der Bindung unverträglich ist, so dass wegen

$$\frac{T(S \to S')}{T(S' \to S)} = \frac{T_0(S \to S')\exp[-2\beta J]}{T_0(S' \to S)}$$

die detaillierte Bilanz erfüllt ist. Die analoge Argumentation gilt für den **4. Fall** $S_i \neq S_j$ und $S_i' \neq S_j'$. Damit ist der Beweis vollständig, da das Setzen der Links eine rein lokale Operation ist und unabhängig von den übrigen Spins ist.

Der nichttriviale Anteil des Algorithmus besteht darin, effizient Cluster zu identifizieren. Hier können wir auf die Algorithmen, die wir im Perkolationskapitel besprochen haben, zurückgreifen.

5.7.2 Bestimmung des dynamischen Exponenten z

Durch die zufällige Orientierung der Cluster nach einem Update kann man zur Bestimmung der Korrelationszeit **nicht** die Autokorrelationsfunktion $C(t) = \langle \mathcal{M}(t_0)\mathcal{M}(t_0+t)\rangle \propto e^{-t/\tau}$ heranziehen, da sie immer auf Korrelationszeiten $\mathcal{O}(1)$ führt ($\langle \ldots \rangle = \frac{1}{(T-t)} \sum_{t_0=1}^{T-t}$). Wir betrachten stattdessen die Suszeptibilität pro Gitterplatz

$$\mathcal{X} = \frac{1}{L^{2d}} \left\langle \left(\sum_{i=1}^{L^d} S_i \right)^2 \right\rangle$$

die auf die Korrelationsfunktion

$$C_{\mathcal{X}}(t) = \frac{\sum_{n=1}^{T-t} \mathcal{X}(n+t)\mathcal{X}(n)}{\sum_{n=1}^{T-t} \mathcal{X}^2(n)}$$

wobei die "Zeit" t in MC-Sweeps gemessen wird.

Die Suszeptibilität kann direkt aus der Konfiguration bestimmt werden. Alternativ kann man aber auch sog. "improved estimators" benutzen: Wir können in der Form

$$\chi = L^{-2d} \left\langle \left(\sum_c N_c S_c \right)^2 \right\rangle$$

darstellen, wobei N_c die Zahl der Spins im Cluster c bezeichnet und s_c die Orientierung des entsprechenden Clusters.

Dies kann umgeschrieben werden in

$$\chi = \frac{1}{L^{2d}} \left\langle \sum_c N_c S_c \sum_{c'} N_c S_{c'} \right\rangle,$$

wobei wir über alle möglichen Spinorientierungen für eine gegebene Verteilung von Clustern mitteln müssen, so dass sich die Terme mit unterschiedlichen Vorzeichen wegheben und wir schließlich

$$\chi = \frac{1}{L^{2d}} \left\langle \sum_c N_c^2 \right\rangle$$

erhalten. Diese Art der Bestimmung von gibt einen verbesserten Schätzwert, da er für eine gegebene Clusterverteilung bereits die Mittelung über die mögliche Spineinstellung beinhaltet. beinhaltet.

5.7.3 Der Wolff-Algorithmus

Wolff hat eine andere Cluster-Methode vorgeschlagen, bei der bei jedem Schritt nur ein Cluster erzeugt wird.

Struktur des Algorithmus:

```
algorithm
begin
Initialize
        spin=(int) (N* ran());
        nlist = 1;
        for (i< N) clust[i]=0;

        new[0] = spin;
        nspin=1; clust[spin]=1
        while(nspin> 0) do
            cl= new[nspin-1];
            for(k< k neighb) do
                nn= neighb[cl][k];
                if((clust[nn]==0) AND (s[nn]=orient))do
                    if(ran()<(1 − e^{−2J}))do
                        new[nspin]=nn;
                        nspin++;
                        clust[nn]=1;
```

```
                    s[nn]=-s[nn];
                end if
            end if
        end for
        new[0] = new[nspin-1];
        nspin - -;
    end while;
end
```

Wir beginnen also den Algorithmus mit der zufälligen Auswahl eines Spins i. In den Cluster werden benachbarte Spins mit gleicher Orientierung mit der Wahrscheinlichkeit $1-e^{-2\beta J}$ aufgenommen und in eine Liste eingetragen. Die Nachbarn der Listenelemente werden abgearbeitet und die Konstruktion des Clusters ist beendet, wenn die Liste keine Elemente mehr enthält. Alle Spins im Cluster werden geflippt.

Die Auswahlwahrscheinlichkeit der Spins entspricht der des Swendsen-Wang Algorithmus, so dass die detaillierte Bilanz erfüllt ist. Beim Wolff Algorithmus entfällt die Bestimmung der zusammenhängenden Cluster. Außerdem ist die Korrelationszeit reduziert, da die Wahrscheinlichkeit, einen Cluster zu flippen, proportional zur Anzahl seiner Elemente ist. Daher werden typischerweise **große** Cluster geflippt, was tendenziell zu kurzen Korrelationszeiten führt.

Bei der Bestimmung der Korrelationszeiten muss man berücksichtigen, dass nur ein einziger Cluster mit variabler Größe geflippt wird. Wenn man die Korrelationszeit τ_W in Einheiten von Updates pro Spin bestimmen will, muss man die Korrelationszeit $\overline{\tau}_W$ in Clusterupdates in Beziehung zur mittleren Clustergröße setzen:

$$\tau_W = \overline{\tau}_W \frac{\langle N_{SC} \rangle}{L^d}$$

wobei $\langle N_{SC} \rangle$ die mittlere Größe eines einzelnen Clusters ist. $\langle N_{SC} \rangle$ ist zugleich "improved estimator" für die Suszeptibilität, denn

$$\langle N_{SC} \rangle = \left\langle \frac{N_{SWC}}{L^d} N_{SWC} \right\rangle = \chi \,;$$

da die Wahrscheinlichkeit einen "SW-Cluster" im Wolff-Algorithmus zu finden durch N_{SWC}/L^d gegeben ist.

5.7.4 Kontinuierliche Spins

Neben dem Ising-Modell betrachtet man auch Modelle mit N oder sogar kontinuierlichen Spineinstellungen. Die Wechselwirkung zwischen benachbarten Spins wird dann durch das Skalarprodukt

$$\underline{S}_i \cdot \underline{S}_j$$

parametrisiert. Für die Konstruktion des Algorithmus ist es wichtig, dass in diesen Modellen die Spineinstellungen langsam variieren, abrupte Anwendungen der Spinorientierungen sind selten. Dieser physikalischen Eigenschaft des Systems sollte auch im Algorithmus Rechnung getragen werden. Man muss also die möglichen Änderungen der Orientierung beschränken. Dies führt allerdings zu langen Korrelationszeiten.

Wolff hat daher vorgeschlagen, die Änderungen abhängig von der Orientierung zu machen. Wir wählen dazu zunächst einen Einheitsvektor mit zufälliger Orientierung $\underline{u}$. Jeder Spin wird dann in einen Anteil parallel und senkrecht zu $\underline{u}$ zerlegt:

$$\begin{aligned} \underline{s}_i'' &= (\underline{s} \cdot \underline{u})\underline{u} \\ \underline{s}_i^{\perp} &= \underline{s}_i - \underline{s}_i'' \end{aligned}$$

Beim Spin-Flip wird $|s''|$ und $|s^{\perp}|$ konstant gelassen und nur die parallele Komponente geflippt.

$$\underline{s}_i = \underline{s}_i^{\perp} + \epsilon_i \, |s_i''| \underline{u} \text{ mit } \epsilon_i$$

Wenn sich die Orientierung ϵ_i ändert, entspricht dies einem Spinflip um die Ebene senkrecht zu $\underline{u}$.
Figur fehlt!!

Zur Konstruktion des Clusters schreiben wir den Hamiltonian in der Form

$$\begin{aligned} \mathcal{H}[\epsilon_i] &= \sum_{\langle ij \rangle} J_{ij}\, \epsilon_i \epsilon_j \\ \text{mit } J_{ij} &= J|s_i''||s_j''| \end{aligned}$$

Dieses "Ising-Modell' mit variablen Kopplungsstärken wird dann mit dem gewöhnlichen Cluster-Algorithmus simuliert.

Die Effizienz des Algorithmus liegt darin begründet, dass man für jede Orientierung von $\underline{u}$ einen geeigneten Rand des Clusters findet, bei dem die Spins (fast) senkrecht zu $\underline{u}$ stehen und damit $J_{ij} \ll 1$. Das bedeutet, dass die Clustergrenze typischerweise da verläuft, wo die Spins ihre Orientierung durch den Spinflip kaum ändern.

Die Implementierung des Algorithmus erfolgt analog zum Ising-Modell:

- Wir wählen einen zufälligen Spin $\underline{s}_i$ aus

- Bestimmung eines Zufallsvektors $\underline{u}$ mit $|\underline{u}| = 1$ (in 2D: Wir bestimmen zufällig den Winkel zur x-Achse)

- Ein Nachbar von i wird mit der Wahrscheinlichkeit

  $$P_f = 1 - \min\{1, \exp[2\beta \underline{s}_i \cdot \underline{s}_j]\}$$

  in den Cluster genommen und geflippt (Beachte: s_i ist schon die Orientierung nach dem Spinflip)

Die kontinuierlichen Spineinstellungen verhindern, dass man z.B. die Wahrscheinlichkeiten P_f tabellieren kann. Dies ist aber für das n-Zustands Clock-Modell möglich, bei dem die Spins die Werte $2\pi j/n$ mit $j = 0, \ldots, n-1$ annehmen können. Für groše Werte von n (n¿20) entsprechen die Eigenschaften des Modells denen des XY-Modells. Beim XY-Modell in 2D beobachtet man den sog. Kosterlitz-Thouless Phasenübergang.

Wir wollen an dieser Stelle kurz das Verhalten des Modells diskutieren:
Das Modell hat zunächst einmal kontinuierliche Anregungen, d.h. Spinwellen bei denen sich die Spineinstellungen nur langsam ändern.
Daneben gibt es aber auch noch Vortex ("Wirbel")-Anregungen.
Bild
Diesen Anregungen ordnet man eine Wirbelstärke (vorticity) zu, die misst, um welchen Wirbel sich die Spins entlang eines geschlossenen Pfades um den Wirbel ändern ($\pm 2\pi$, bei hohen Temperaturen auch: $\pm 4\pi, \ldots$).

In einem großen System sind einzelne Wirbel nicht möglich, da die Energie einer solchen Anregung logarithmisch divergiert. Vortex-Paare können sich aber bilden. Zwischen den Vortexpaaren gibt es eine effektive Wechselwirkung $\alpha \ln R$ (entspricht der Coulomb-WW in 2D).
Das System hat einen Phasenübergang bei dem man das Entkoppeln der Vortexpaare beobachtet. Unterhalb der kritischen Temperatur beobachtet man langreichweitige Kopplung der Spineinstellungen gemäß

$$\langle \Theta_i - \Theta_j \rangle \sim \frac{1}{|\underline{r}_i - \underline{r}_j|^{x_T}}$$

Der Exponent x_T ist temperaturabhängig.

Eine Größe, die den Phasenübergang gut charakterisiert, ist die Steifigkeit der Spins ("spin-ware stiffness") Γ. Γ misst die freie Energie, die man braucht, um bei niedrigen Temperaturen und fixierten Randbedingungen die Spins um einen Wirbel δ zu drehen.

Die Energiedifferenz skaliert wie

$$\Delta \sim \Gamma \delta^2.$$

Am Phasenübergang verschwindet $\Gamma(T)$. Im periodischen System kann man Γ gemäß

$$\begin{aligned}\Gamma &= \frac{J}{2L^2}\left\{\left\langle \sum_{\langle ij\rangle} \cos(\Theta_i - \Theta_j)\right\rangle - \frac{J}{2L^2 k_B T}\left\langle \left[\sum_i \sin(\Theta_i - \Theta_{i+e_x})\right]^2\right\rangle\right. \\ &- \left.\frac{J}{2L^2 k_B T}\left\langle \left[\sum_i \sin(\Theta_i - \Theta_{i+e_y})\right]^2\right\rangle\right.\end{aligned}$$

(5.5)

messen. Bei T_{KT} nimmt Γ den Wert $2k_BT_{KT}/\pi$ an.
Bild

Kapitel 6

Molekulardynamik-Simulationen

Die Molekulardynamik-Methode stellt den Versuch dar, die experimentelle Situation nachzustellen: Man präpariert das System in einem Zustand und löst dann die Newtonschen Bewegungsgleichungen. Das System wird dann solange seiner Dynamik überlassen bis die Messgrößen sich nicht mehr ändern.

Wir wollen nun mit der Struktur eines Molekulardynamikprogramms beginnen und dann die einzelnen Elemente erläutern.

6.1 Struktur des Programms

1. Einlesen der Parameter, die die Bedingungen der Simulation steuern (z.B. Temperatur, Teilchendichte, Zeitschritt usw.).

2. Initialisierung des Systems, d.h. die Teilchenpositionen und Geschwindigkeiten werden festgelegt.

3. Berechnung aller Kräfte, die auf ein Teilchen wirken (wird iterisiert).

4. Integration der Newton-Gleichungen (wird iterisiert).

5. Mittelung über die Messgrößen und Ausgabe

6.1.1 Initialisierung

Die **Anfangspositionen** sollten so gewählt werden, dass sie mit der Wechselwirkung des Systems kompatibel sind. Dies erreicht man häufig dadurch, dass man die Teilchen auf ein Gitter setzt.

Dann muss man den Teilchen noch **Geschwindigkeiten** zuordnen. Eine einfache (und nicht optimale) Möglichkeit besteht darin, den Teilchen zufällige Geschwindigkeitskomponenten zuzuordnen, d.h. $v_k(i) \in [-0.5, 0.5]$ ($v_k(i)$ ist die k-te Komponente der Geschwindigkeit des i-ten Teilchens). Nach der zufälligen Initialisierung muss man noch gewährleisten, dass der Gesamtimpuls verschwindet. Dazu setzen wir $\underline{v}(i) = \underline{v}(i) - \frac{\underline{v}_s}{N}$; $\underline{v}_s = \sum_{i=1}^{N} \underline{v}(i)$, N = Teilchenzahl, alle Teilchen haben die gleiche Masse. Nun müssen wir die Geschwindigkeit reskalieren, um die Temperatur einzustellen.

Im thermischen Gleichgewicht gilt:

$$\langle v_k^2 \rangle = \frac{k_B T}{m}$$

Entsprechend erhalten wir:

$$k_B T(t) = \sum_{i=1}^{N} \frac{m \underline{v}_i^2(f)}{N_f}$$

wobei N_f die Zahl der Freiheitsgrade des Systems bezeichnet. Eine **Annäherung** an die gewünschte Temperatur T erhalten wir durch Reskalieren der Geschwindigkeiten mit $(T/T(t))^{1/2}$.
Die Temperatur wird aber während der Simulation fluktuieren.

Bei der Integration der Bewegungsgleichungen braucht man aber nicht die Geschwindigkeiten, sondern die Teilchenpositionen zu den Zeiten t und $t-1$. Daher dient die Anfangsgeschwindigkeit dazu, die Teilchenposition zur Zeit $-dt$ zu bestimmen:

$$\underline{x}n(i) = \underline{x}(i) - \underline{v}(i)\, dt$$

```
procedure init
begin

    for (k< D) sum v [k] = 0.;
    unleserlich
    sum v2 = 0.;

    for (k<D) do
        for (i<N) do
            x [k] [i] = latt-pos (k,i);
            v [k] [i] = ran() -0.5;
                sum v[k] = sum v [k] + v [k] [i];
                sum v2 [k] = sum v2 [k] + v [k] [i] * v [k] [i];
            end for
            sum v [k] = sum v [k]/N;
        end for
            sum v = sum v2 /(D · N);
        v scale = sqrt (temp/sum v2);
        for (k<D) do
            for (i< N) do
            v [k] [i] = (v [k] [i] - sum v [k]) + v scale;
            x m[k] [i] = x [i] - v [i] * dt;
            end do
    end do
end
```

6.1.2 Berechnung der Kräfte

Dieser Teil einer Molekulardynamik Simulation ist gewöhnlich der zeitaufwändigste. In einer naiven Implementation müßten hier $N \times (N-1)/2$ Distanzen betrachtetet und bei periodischen Systemen und langreichweitigen Wechselwirkungen eventuelle Selbstwechselwirkungen berücksichtigt werden. Man kann die Bestimmung der Kräfte effizienter gestalten, indem man zunächst einen Cut-off einführt. Dies wollen wir am Beispiel des

Lennard-Jones Potentials illustrieren. Wir wählen eine Distanz $r_c < L/2$ wobei L die Kantenlängen der Box bezeichnet. Damit sind Selbstwechselwirkungen ausgeschlossen. Im nächsten Schritt müssen wir den kürzesten Abstand in der periodischen Box berechnen. Dies geschieht (für jede Komponente) einfach durch

$$\begin{aligned} xr &= x[k][i] - x[k][j]; \\ xr &= xr - \text{box}\, x \lfloor xr/L \rfloor; \end{aligned}$$

Die Distanz r_{ij}^2 muss dann mit r_c verglichen werden. Falls $r_{ij}^2 < r_c$ gilt, müssen wir die Kraft auf das Teilchen bestimmen. Für die x-Komponente gilt

$$f_x(r) = -\frac{x}{r}\left(\frac{\partial n}{\partial r}\right)$$

so dass wir für das Lennard-Jones Potential ($n(r) = 4\,(\frac{1}{r_c^{12}} - \frac{1}{r_c^6})$

$$f_x(r) = \frac{48x}{r^2}\left(\frac{1}{r^{12}} - 0.5\,\frac{1}{r^6}\right)$$

erhalten.
Wir führen schließlich einen Shift des Potentials ein, damit $n(r)$ bei r_c stetig ist, also $n(r) = 4\,(\frac{1}{r_c^{12}} - \frac{1}{r_c^6}) - e_c$ wobei $e_c = 4\,(\frac{1}{r_c^{12}} - \frac{1}{r_c^6})$.

procedure force (f,en)
begin

```
en =0:
for (k < D) do
    for (i < N) do
        f [k] [i] = 0;
    end do
end do
for (i = i, ..., N - 1) do
    for (j = i + 1, ..., N) do
        rsqr = 0;
```

```
            for (k < D) do
                xr [k] = x [k] [i] - , x [k] [j];
                xr [k] = x r [k] - L * L x * [k]/⌊ ⌋;
                rsqr = rsqr - xr [k] *, x r [k];
            end for
            if (rsqr < rc2) do
                r2i = 1 / rsqr ;
                r6i = r2i * r2i *r2i ;
                ff = 48 * r2i *r6i * (r6i - 0.5) ;
            for (k < D) do
                f[k] [i] = f[k] [i] + ff * xr [k];
                f[k] [i] = f[k] [j] - ff * xr [k];
                en = en + 4 * r6i * (r6i - 1) - c cut ;
            end for
        end for
    end for
    return en ;
end
```

Das letzte Element einer Molekulardynamik Simulation bildet die Integration der Bewegungsgleichungen. Wir benutzen hier den sog. Verlet-Algorithmus, der eine von mehreren möglichen Varianten darstellt.

Zur Herleitung betrachten wir zunächst die Taylorentwicklung

$$r(t+\Delta t) = r(t) + v(t)\Delta t + \frac{f(t)}{2m}(\Delta t)^2 + \frac{(\Delta t)^3}{3!}\overset{\cdots}{r} + \Theta((\Delta t)^4)$$

und

$$r(t-\Delta t) = r(t) - v(t)\Delta t + \frac{f(t)}{2m}(\Delta t)^2 - \frac{(\Delta t)^3}{3!}\overset{\cdots}{r} + \Theta((\Delta t)^4)$$

$\Rightarrow$

$$r(r+\Delta t) = 2r(t) - r(t-\Delta t) + \frac{f(t)}{m}(\Delta t)^2 + \Theta((\Delta t)^4)$$

Die Geschwindigkeit der Teilchen wird dann einfach durch

$$v(t) = \frac{r(t + \Delta t) - r(t - \Delta t)}{2\Delta t} + \Theta((\Delta t)^2)$$

abgeschätzt.

(1. Komponente, die übrigen analog:)

procedure move
begin

```
sum v = 0 ;
sum v2 = 0 ;
dsqr = dt * dt ;
for (i < N) do
    xx = 2 * x [i] - xm [i] + dsqr * f [i] ; Verlet Algorithmus
    vi = (xx - xm [i]) / 2 * dt ;
    sum v2 = sum v2 + v i * v i ;
    xm [i] = x[i] ;
    x [i] = xx ;
end for
temp = sum v2 / 3* n part ;
e t○t = (en + 0.5 * sum v2) / n part ;
```

end

6.1.3 Bemerkungen zur Integration der Bewegungsgleichungen

Der aufwendigste Schritt bei der Molekulardynamik-Simulation ist die Berechnung der Kräfte, so dass man prinzipiell daran interessiert sein sollte, dass der Fehler von möglichst hoher Ordnung in dt ist. Dies würde dann relativ große Schrittweiten für dt erlauben, also für ein gegebenes Zeitintervall die Zahl der Kraftberechnungen erniedrigen.

Das Problem von Algorithmen, die Ableitungen von hoher Ordnung berücksichtigen, ist, dass die Energieerhaltung lokal zwar in sehr guter Näherung erfüllt

ist, so dass sich für große Zeiten eine systematische Drift einstellt. Dies steht im Gegensatz zum Verlet Algorithmus, dessen Eigenschaften bzgl. der Energieerhaltung auf großen Zeitskalen günstiger sind.

Den Algorithmen ist gemein, dass sie die Trajektorien schlecht voraussagen. Dies ist in dem Sinne zu verstehen, dass Trajektorien von benachbarten Punkten im Phasenraum sich exponentiell voneinander entfernen. Man spricht dann von einer sog. Lyapanov Instabilität. Hierzu muss man aber betonen, dass es auch nicht das Anliegen einer MD-Simulation ist, Trajektorien detailgetreu zu reproduzieren, sondern vielmehr für das System typische Trajektorien zu generieren.

Einige Integrationsschemata sind äquivalent zum Verlet-Algorithmus. Das erste Beispiel ist das sog. Leap-Frog Verfahren. Ausgehend vom Verlet-Algorithmus kann man den Leap-Frog Algorithmus durch Einführung von Geschwindigkeit an Halbschritten herleiten: