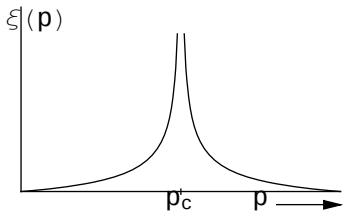
Computational Physics: *Stochastic Simulation*

P.M.A. Sloot

Section Computational Science
University of Amsterdam,
January 2002

sloot@science.uva.nl

http://carol.science.uva.nl/~sloot




```
ShowDepositio n [Molecular Deposition [200, 10 000 ]];
```

Preface Lecture notes on Stochastic Simulation

The real purpose of modelling and simulation is to discover that nature hasn't misled you into thinking you know something, you don't actually know.

Adapted from R. Pirsig

In the ever growing field of computer simulation stochastic simulation plays a more and more prominent role. This is mainly due to the enormous potential of nowadays computer systems. The type of systems that need to be studied through stochastic simulation are systems of which the behaviour can only be predicted by explicit simulation, rather than analytical calculation. We classify these systems by "computationally irreducible complex systems". Examples are: The motion of molecules in a classical liquid, light beams in ray-tracing optics, spreading phenomena (of diseases, forest fires etc.), growth phenomena (of crystals or macroscopic biological entities), or more general, all problems from physics that can be described in terms of N-body interactions.

This text is meant to be background undergraduate material for self study in the field of stochastic simulation of problems in physics. The text is accompanied by addenda with exercises and training material: there is no way you can understand computer simulation without actually doing it!

The course can be considered as a good preparation for courses on more advanced topics of simulation such as Computational Physics II: "Understanding Molecular Simulations", "Parallel Scientific Computing and Simulation" and "Simulation of Complex Systems".

This text started with the preparation of a lecture on modelling and simulation at the 1994 CERN summer school in Hungary. Large parts of the original idea's and text are adapted from, and motivated by, the literature listed at the end of this document.

This text is clearly a compromise where I have chosen stimulating examples from computational physics rather than rigorous treatment of selected topics. Since I plan to update the text each year I would appreciate to receive your comments, suggestions, additions, and corrections.

Computer simulation is fun, but we have to stay critical and keep a close watch on the figures and data we produce and ask ourselves if it is physics we are looking at or rather a poor mimic of it. In the words of E. Wigner (after seeing a complicated computer simulation): "*I think the computer understands it, now **I** want to understand it!"*

Peter Sloot

Amsterdam, January 2002

Email: sloot@science.uva.nl

URL: http://www.science.uva.nl/~sloot

Contents

1 Monte Carlo Methods & Variance Reduction

Monte Carlo method [Origin: after Count Montgomery de Carlo, Italian gambler and random-number generator (1792-1838).] A method of jazzing up the action in certain statistical and number-analytic environments by setting up a book and inviting bets on the outcome of a computation.

-S. Kelly-Bootle

The Devil's DP Dictionary

1.1 Introduction

In this chapter, one of the most interesting probabilistic simulation methods, called "Monte Carlo" (MC), is introduced. It can be applied to study all kinds of systems. E.g. statistical physics problems, crystallisation processes, phase transitions in systems with large degrees of freedom, evaluation of integrals of high dimensionality, etc. The MC technique will be compared with a standard numerical integration technique: Simpson's rule. In principle MC simulations are static, stochastic and discrete of nature. Furthermore several variance reduction techniques for probabilistic simulations will be introduced. Such techniques can aid in arriving faster at a result that falls within certain given error bounds.

A working knowledge of statistics is expected. Please read Appendix A for an overview of the relevant aspects of Probability theory and statistics.

1.2 Monte Carlo Integration

■ 1.2.1 Motivation

In many particle systems, thermodynamical observables are described by integrals such as:

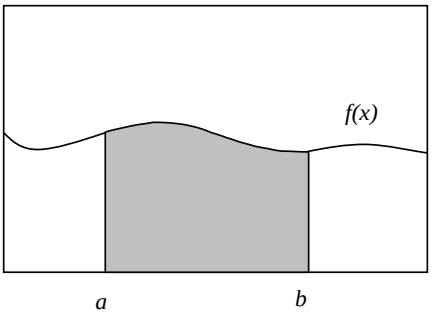
$$Z = \int \dots \int d\,\bar{r}_1 \dots d\,\bar{r}_n \, e^{-\beta \sum_{i<j} v(r_{i,j})} \tag{1.1}$$

In quadrature approximation in 10×10×10 box and with N=20 particles we have $10^{3N} = 10^{60}$ points to evaluate, without taking any symmetries into account. On a machine with 100 MFlop/s it would take 10^{52} seconds, what is 10^{37} times more than the age of Universe.

■ 1.2.2 Introduction to Numerical Integration

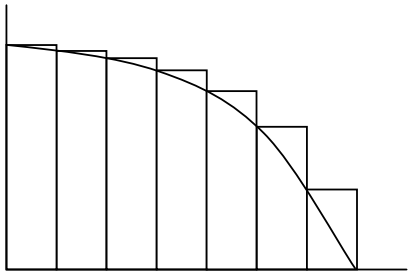
We first briefly discuss "classical" methods of numerical integration. Consider a one-dimensional definite integral of the form:

$$\theta = \int_a^b f(x)\,\mathrm{dx}. \tag{1.2}$$



For certain choices of the integrand $f(x)$, this integration can be done analytically. However there are many common functions whose integrals are intractable and which must be evaluated numerically.

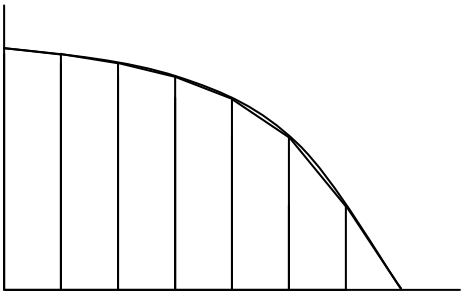
The simpliest numerical integration method is the rectangular approximation. We divide the interval $[a, b]$ in n equal parts of length $\Delta x = (b - a)/n$.



The estimate of the integral is the sum of the rectangles:

$$\theta_n = \sum_{i=0}^{n-1} f(x_i)\,\Delta\mathrm{x} \tag{1.3}$$

The next method uses trapezoids instead of rectangles.



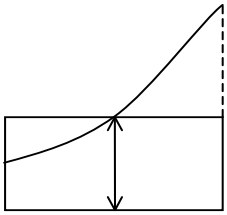
The area of each column is equal to:

$$\tfrac{1}{2}[f(x_{i+1}) + f(x_i)]\,\Delta x \tag{1.4}$$

what gives the total value:

$$\theta_n = \left[\tfrac{1}{2}\, f(x_0) + \textstyle\sum_{i=1}^{n-1} f(x_i) + \tfrac{1}{2}\, f(x_n)\right] \tag{1.5}$$

The midpoint rule is another method to estimate an integral on the small interval.



$a+h/2$

$$\int_a^{a+h} f(x)\,\mathrm{dx} \simeq h\left(a + \frac{h}{2}\right) \tag{1.6}$$

It can be shown that errors of trapezoidal approximation (T_{error}) and of midpoint rule (M_{error})are related in such a way that $T_{\text{error}} \simeq -\frac{1}{2} M_{\text{error}}$.

$$\begin{aligned} T &= \int_a^b f(x)\,\mathrm{dx} + \text{Error} \\ M &= \int_a^b f(x)\,\mathrm{dx} - \frac{1}{2}\,\text{Error} \end{aligned} \tag{1.7}$$

We can cancel these errors by combining both approximations:

$$\frac{1}{3}\,T + \frac{2}{3}\,M \simeq \int_a^b f(x)\,\mathrm{dx} \tag{1.8}$$

It gives the Simpson's rule:

$$S = h\left[\frac{1}{6}\, f(a) + \frac{2}{3}\, f\left(a + \frac{h}{2}\right) + \frac{1}{6}\, f(a+h)\right] \tag{1.9}$$

The Simpson's rule applied to the whole interval divided in n parts of length Δx is given by :

$$\theta_n = \frac{1}{3}\left[f(x_0) + 4\,f(x_1) + 2\,f(x_2) + 4\,f(x_3) + \cdots 2\,f(x_{n-2}) + 4\,f(x_{n-1}) + f(x_n)\right]\Delta x. \qquad (1.10)$$

Simpson's rule requires that n is even. It can be shown that the error in this estimation is proportional to n^{-4}. More general one can show for numerical integration techniques that if the error is proportional to n^{-a} in one dimension, then the error in d dimensions is proportional to $n^{-a/d}$.

Question: Show qualitatively why the prefactor in (1.10) is 1/3.

Question: Show qualitatively why the exponent in the error for d-dimensions equals -a/d.

■ 1.2.3 Integral Estimation using Hit-Miss.

A method that can be seen as a MC method is the so-called "Hit-Miss" technique. The principle is as follows:

Let $g(x)$ be a bounded function over a finite interval, for example $0 \le g(x) \le b$ with x in the interval [0, 1]. We are again interested in estimating:

$$\theta = \int_0^1 g(x)\,dx. \qquad (1.11)$$

The hit-miss method for accomplishing this is to generate a pair of independent random numbers $U_1$and U_2. Next set $X=U_1$and $Y=bU_2$. In this way random points (X, Y) are generated, that are uniformly distributed in a rectangle of length 1 and height b. Now set

$$I = I(X, Y) = \begin{cases} 1 & \text{if } Y < g(X) \\ 0 & \text{otherwise,} \end{cases} \qquad (1.12)$$

then

$$E[I] = \frac{1}{b}\int_0^1 g(x)\,dx, \qquad (1.13)$$

$$\theta = E[I]\,b. \qquad (1.14)$$

That is fun! We observe that by just "throwing darts" in a confined space and calculating the ratio of the number of "hits" under the integral and the total number of "hits and misses" (Equation 1.14), we get an approximation of the numerical value of the Integral under study.

Note that we have used $E[X]$ to denote the Expectation of variable X, this is equal to the average of X. Often we use the notation $\langle X \rangle$ for the same operation (see Appendix A).

Question: Give the Hit and Miss algorithm for Integration with arbitrary boundaries.

1.2.4 Integral Estimation using Monte Carlo

The principle behind the Monte Carlo method can be explained by means of a simple example. Assume we have a function $g(x)$ of which we wish to calculate the following:

$$\theta = \int_0^1 g(x)\,dx. \tag{1.15}$$

If we take a uniformly distributed random variable, say U , then the following holds:

$$\theta = E\big[g(U)\big]. \tag{1.16}$$

Question: explain why this is true (see Appendix A).

If we have a row of independent random variables $U_1, \ldots, U_k$ that are uniformly distributed over the interval [0,1] the random variables $g(U_1), \ldots, g(U_k)$ are independent and identically distributed random variables having a mean θ. Therefore by the strong law of large numbers, see equation (A.23), it follows that:

$$\frac{1}{k}\sum_{i=1}^{k} g(U_i) \longrightarrow \theta, \quad \text{for } k \longrightarrow \infty. \tag{1.17}$$

So θ may be approximated by generating a large number of random numbers $U_i$and taking as our approximation the average value of $g(U_i)$. The utility of using random numbers to approximate integrals becomes more apparent in the case of multidimensional integrals. g can be a function with an n-dimensional argument where we want to compute:

$$\theta = \int_0^1 \int_0^1 \ldots \int_0^1 g(x_1, \ldots, x_n)\,dx_1 \ldots dx_n. \tag{1.18}$$

Similar to the one dimensional case we can now estimate the value of θ using the following:

$$\theta = E[g(U_1, \ldots, U_n)], \tag{1.19}$$

where $U_1, \ldots, U_n$ are independent uniform [0,1] random variables. If we now generate k independent sets, each consisting of n independent uniform [0,1] random variables:

$$\begin{array}{ccc} U_1^1, & \ldots & U_n^1 \\ U_1^2, & \ldots & U_n^2 \\ \vdots & \vdots & \vdots \\ U_1^k, & \ldots & U_n^k \end{array} \tag{1.20}$$

then, since the random variables $g\left(\ U_1^i, \ldots, U_n^i\ \right)$ for i=1, …, k are all independent and identically distributed random variables with mean θ, we can estimate θ by

$$\theta = \frac{1}{k}\sum_{i=1}^{k} g\left(U_1^i, \ldots, U_n^i\right) \text{ for } k \longrightarrow \infty. \tag{1.21}$$

■ 1.2.5 Generalization of MC Sample Mean Integration

We would like to use Monte Carlo integration on the interval from a to b:

$$\theta = \int_a^b g(x)\,dx = G(b) - G(a). \tag{1.22}$$

Introducing new variable $y = \frac{x-a}{b-a}$ we get :

$$y = \begin{cases} 0 & x = a \\ 1 & x = b \end{cases} \text{ and } \mathrm{dx} = (b-a)\,\mathrm{dy}. \tag{1.23}$$

Inserting this into our integral with $x = (b-a)\,y + a$ we obtain

$$\theta = \int_0^1 g((b-a)\,y + a)\,(b-a)\,\mathrm{dy} = \int_0^1 h(y)\,\mathrm{dy} \tag{1.24}$$

To extend our interval to infinity, for example:

$$\theta = \int_0^\infty g(x)\,dx \tag{1.25}$$

we do this by substituting $y = \frac{1}{x+1}$ what gives :

$$y = \begin{cases} 1 & x = 0 \\ 0 & x = \infty \end{cases} \quad \text{and} \quad \mathrm{dx} = -\frac{1}{y^2}\,\mathrm{dy}. \tag{1.26}$$

We modified the integral to the standard form $\theta = \int_0^1 h(y)\,\mathrm{dy}$, with :

$$h(y) = -\frac{g\left(\frac{1}{y} - 1\right)}{y^2}\,\mathrm{dy}. \tag{1.27}$$

■ 1.2.6 Error Analysis of Monte Carlo Integration

To estimate the error of sample mean integration we compute the mean square error of

$$\theta_n = \frac{1}{k}\sum_{i=1}^{k} g(U_i) \tag{1.28}$$

Since we know that $E[\theta_n] = \theta$ we can put

$$\begin{aligned} E\left[(\theta_n - \theta)^2\right] &= \mathrm{Var}(\theta_n) \\ &= \mathrm{Var}\left(\frac{1}{k}\sum_{i=1}^{k} g(U_i)\right) \\ &= \frac{1}{k^2}\sum_{i=1}^{k} \mathrm{Var}(g(U_i)) \\ &= \frac{\sigma_g^2}{k} \end{aligned} \tag{1.29}$$

where $\sigma_g^2$is the variance in the observed values $g(U_i)$. We see that the uncertainty in the estimate of the integral decreases as $1/\sqrt{k}$ and that the precision increases for smaller σ_g^2, which implies a smoother function g. These observations are independent of the dimensionality of the problem, whereas for conventional methods, such as Simpson's rule or the trapezoidal rule, the error tends to increase with dimensionality, as we saw above. This independence of the dimensionality is one of the important characteristics of MC, explaining its widespread use.

■ 1.2.7 Example 1: Monte Carlo Integration

Assume we want to estimate

$$\theta = \int_0^1 e^{-x}\,\mathrm{dx}. \tag{1.30}$$

We then simply generate k random numbers U_i, which are drawn from a uniform [0,1] distribution. With equation (1.17) we then can estimate the integral. The error in the estimation can be approximated using equation (1.29). Say $\theta(k)$ is the estimate of the integral after k "trials" and σ_k is the estimate of $\sigma_g$after k "trials". Typical results could look like:

$\theta(k)$	σ_k	$\sigma_k / \sqrt{k}$	k
0.62906	0.18002	−0.00306	3600
0.63305	0.18201	0.00092	7200

We can observe in this "experiment" that for $k > 3000$ the variance in the output remains unchanged, which is logical since the uncertainty in an individual trial is constant, whereas the error in the estimation may still reduce for larger k. This is an experimental indication of a well known theoretical result that states that the error in the estimation resembles:

$$\sigma_\theta = \frac{\sigma_k}{\sqrt{k}}. \tag{1.31}$$

We note that we can reduce the error in MC integration by:

- increasing the number of trials
- using more efficient trials (i.e. sampling in the neighbourhood of fast fluctuating values of the function $g(x)$. This brings us to the topic of "variance reduction." The following example gives an idea of the amount of work that may arise when simply applying MC simulation, and not considering the possibility of variance reduction.

■ 1.2.8 Example 2: Necessity of Variance Reduction

Consider a stochastic approximation where the observable X_i is determined (with all X's independent and random). We model, for the sake of simplicity, the probabilistic behaviour by:

$$P\{X_i = 1\} = 1 - P\{X_i = 0\} = p, \text{ where } p = 10^{-6}. \tag{1.32}$$

Now in order to allow for a reliable approximation we want to construct a 95% confidence interval of "p" with a length of say: 2×10^{-6}, or equivalently:

$$\overline{X} \pm Z_{0.025} \times \frac{s}{\sqrt{n}}, \tag{1.33}$$

where n equals the number of trials. Then from statistics we know that the length of the confidence interval around p, is given by

$$2 \times Z_{0.025} \times \frac{s}{\sqrt{n}} = 2 \times 10^{-6}. \tag{1.34}$$

With unit normal $Z_{0.025}$= 1.96 and $s = \sqrt{\mathrm{Var}(X_i)} = \sqrt{p(1-p)}$ for the used random variables

X_i, we find that

$$\sqrt{n} = \frac{2 \times 1.96 \times \sqrt{p(1-p)}}{2 \times 10^{-6}}. \tag{1.35}$$

Therefore $n = 3.8\times10^6$, that is more than a *million* simulation runs to obtain the required accuracy! It is clear that a method that can reduce the variance of individual "trials" may be very helpful in obtaining an estimate that doesn't need as many trials as "normal" MC but still provides the same accuracy.

1.3 Variance Reduction Techniques

■ 1.3.1 Introduction

In a typical simulation study, one is interested in determining θ, a parameter connected with some stochastic model. To estimate θ , the model is simulated to obtain, among other things, the output datum X which is such that $\theta = E[X]$. Repeated simulation runs, the ith one yielding the output variable X_i , are performed. The simulation study is then terminated when n runs have been performed and the estimate of θ is given by:

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i. \tag{1.36}$$

Because this results in an unbiased estimate of θ (the estimator of a parameter is an unbiased estimator of that parameter if its expected value is equal to the parameter), it follows that its mean square error (MSE) is equal to its variance. That is:

$$\text{MSE} = E\left[\left(\overline{X} - \theta\right)^2\right] = \text{Var}(\overline{X}) = \frac{\text{Var}(X)}{n}. \tag{1.37}$$

Hence if we obtain a different unbiased estimate of θ having a smaller variance than does $\overline{X}$, we would obtain an improved estimator, this is called *variance reduction*.

The most relevant *variance reduction* techniques are:

- The use of Control Variates
- Conditioning
- Stratified Sampling
- The use of Antithetic Variables
- Importance Sampling

In this text we will focus on Antithetic Variables and Importance Sampling techniques. For a more extensive treatment the reader is referred to for example [18].

■ 1.3.2 The use of Antithetic Variables.

Suppose we are interested in using simulation to estimate $\theta = E[X]$. and suppose we have generated $X_1$and X_2, which are identically distributed random variables having mean θ. Then from equation (A.21) it follows that

$$\mathrm{Var}\left(\frac{X_1 + X_2}{2}\right) = \frac{1}{4}\left[\mathrm{Var}(X_1) + \mathrm{Var}(X_2) + 2\,\mathrm{Cov}(X_1, X_2)\right]. \tag{1.38}$$

Hence it would be advantageous (in the sense that the variance would be reduced) if $X_1$and X_2were negatively or anti-correlated instead of independent. To see how we might arrange for X_1 and X_2 to be negatively correlated, suppose that X_1 is a function of m random numbers:

$$X_1 = h(U_1, U_2, \ldots, U_m), \tag{1.39}$$

where $U_1, U_2, \ldots, U_m$are m independent random numbers. Now if U is a random number uniform [0,1]) then so is 1-U. Hence the random variable

$$X_2 = h(1 - U_1, 1 - U_2, \ldots, 1 - U_m), \tag{1.40}$$

has the same distribution as X_1. In addition, since 1-U is clearly negatively correlated with U, we might hope that X_2 might be negatively correlated with X_1. Indeed one can prove in the special case where h is a monotone function of each of its coordinates that X_1 and X_2 are negatively correlated.

Hence, after we have generated $U_1, U_2, \ldots, U_m$ to compute X_1, we can use the set $1 - U_1, 1 - U_2, \ldots, 1 - U_m$ to compute X_2. In addition, it should be noted that we obtain a double benefit: not only does our resulting estimator have smaller variance, but we also save the time of generating a second set of random numbers.

Example: The use of Antithetic Variables

Suppose we are interested in using simulation to estimate

$$\theta = E\left[e^U\right] = \int_0^1 e^x \,\mathrm{dx} = e - 1. \tag{1.41}$$

Since the function $h(U)=e^U$is clearly a monotone function, the antithetic variable approach is expected to lead to a variance reduction. Note that Equation (A.20) gives

$$\begin{aligned}\mathrm{Cov}\left(e^U, e^{1-U}\right) &= E\left[e^U e^{1-U}\right] - E\left[e^U\right] E\left[e^{1-U}\right] \\ &= e - (e-1)^2 = -0.2342,\end{aligned} \tag{1.42}$$

and

$$\begin{aligned}\mathrm{Var}\left(e^U\right) &= E\left[e^{2U}\right] - \left(E\left[e^U\right]\right)^2 \\ &= \int_0^1 e^{2x}\,dx - (e-1)^2\end{aligned} \tag{1.43}$$

$$= \frac{\left[e^2 - 1\right]}{2} - (e-1)^2 = 0.2420$$
$$= \mathrm{Var}\left(e^{1-U}\right).$$

We see that the use of independent random numbers results in a variance of

$$\mathrm{Var}\left(\frac{e^{U_1} + e^{U_2}}{2}\right) = \frac{1}{4}\left[\mathrm{Var}\left(e^{U_1}\right) + \mathrm{Var}\left(e^{U_2}\right) + 0\right] = \frac{\mathrm{Var}\left(e^{U}\right)}{2} = 0.1210, \tag{1.44}$$

whereas the use of the antithetic variables U and 1-U gives a variance of

$$\mathrm{Var}\left(\frac{e^{U} + e^{1-U}}{2}\right) = \frac{\mathrm{Var}\left(e^{U}\right)}{2} + \frac{\mathrm{Cov}\left(e^{U}, e^{1-U}\right)}{2} = 0.0039, \tag{1.45}$$

which is a variance reduction of **96.7 %**.

■ 1.3.3 Importance Sampling

Let $\vec{Y} = (Y_1, \ldots, Y_m)$ denote a vector of random variables having joint probability density function $f(\vec{Y})$, and suppose we are interested estimating

$$\theta = E\left[h(\vec{Y})\right] = \int h(\vec{y})\, f(\vec{y})\, dy_1 \ldots dy_m, \tag{1.46}$$

where the above is meant to be an m-dimensional integral on the unit hypercube.

Suppose that a direct simulation of the random vector $\vec{Y}$ so as to compute values of $h(\vec{Y})$ is inefficient, possibly because it is difficult to generate the random vector $\vec{Y}$, or the variance of $h(\vec{Y})$ is large, or a combination of both.

Another way in which we can use simulation to estimate θ is to note that for any other random vector $\vec{W} = (W_1, \ldots, W_m)$, which takes values in the same region as does $\vec{Y}$ and has joint density function $g(\vec{W})$, θ can be expressed as:

$$\begin{aligned}\theta &= \int \frac{h(\vec{w})\, f(\vec{w})}{g(\vec{w})}\, g(\vec{w})\, dw_1 \ldots dw_m \\ &= E\left[\frac{h(\vec{W})\, f(\vec{W})}{g(\vec{W})}\right].\end{aligned} \tag{1.47}$$

Therefore we can estimate θ by successively generating values of the random vector $\vec{W}$ and then using as the estimator the resulting average of the values of $h(\vec{W})\, f(\vec{W})/g(\vec{W})$. If the density function $g(\vec{W})$ can be chosen to be similar in shape to the function $h(\vec{W})\, f(\vec{W})$ then their ratio will not vary much and so $h(\vec{W})\, f(\vec{W})/\, g(\vec{W})$ will have a small variance. Unfortunately, however, this is often difficult to

accomplish and usually a small simulation experiment is necessary to see if this approach does indeed result in an acceptably small variance.

Example: Importance Sampling

Say one is interested in estimating the following integral:

$$H = \int_a^b h(x)\,dx. \tag{1.48}$$

We may introduce a positive function $p(x)$ with the characteristic:

$$1 = \int_a^b p(x)\,dx \tag{1.49}$$

and rewrite equation (1.48) as :

$$H = \int_a^b \frac{h(x)}{p(x)}\,p(x)\,dx\,. \tag{1.50}$$

Therefore one can estimate the integral H after n "trials" by

$$H_n = \frac{1}{n}\sum_{i=1}^{n}\frac{h(X_i)}{p(X_i)}, \tag{1.51}$$

where now the values of X_i are not uniformly distributed on the interval $[a, b]$ but are distributed according to $p(x)$. $X_i$follows from $P^{-1}(U)$, the inverse of the cumulative distribution function $P(x) = \int_{-\infty}^{\infty} p(x)\,d\,x$ and U is the uniform distribution. Moreover $p(x)$ should be chosen to mimic $h(x)$, because in that case the integrand will vary slowly, which means that the variance will be small.

How to choose the best $p(x)$? We wish the variance of H to be small and since ${\sigma_H}^2 = {\sigma_{h(x)/p(x)}}^2 \big/ N$ we must reduce variance of $h(x) / p(x)$. It is given by :

$$\sigma^2_{\frac{h(x)}{p(x)}} = \left\langle\left(\frac{h(x)}{p(x)} - \left\langle\frac{h(x)}{p(x)}\right\rangle\right)^2\right\rangle \tag{1.52}$$

and since $\left\langle\dfrac{h(x)}{p(x)}\right\rangle = H$ it is equal to

$$\left\langle\left(\frac{h(x)}{p(x)} - H\right)^2\right\rangle = \int_0^1\left(\frac{h(x)}{p(x)} - H\right)^2 p(x)\,\mathrm{dx} \tag{1.53}$$

The best solution is to take normalized version of $h(x)$:

$$p(x) = \frac{h(x)}{\int_0^1 h(x)\,\mathrm{dx}} \tag{1.54}$$

Then the variance will vanish and the result will be exact.We can say that in that case we can estimate

the value of integral using only one random number! The problem is that we must know the exact value of the integral before.

Of course the computational cost of sampling an arbitrary random variable is higher than that for using uniformly distributed variables. Table 1.1 shows an example where importance sampling is compared with standard Monte Carlo estimation. The results are for estimation of the integral:

$$H = \int_0^1 e^{-x^2}\, dx \;=\; \frac{\sqrt{\pi}}{2}\,\mathrm{Erf}(1) \;=\; 0.746824. \tag{1.55}$$

	$p(x)$ uniform [0, 1]	$p(x) = A\,e^{-x}$
n (number of trials)	2000	1000
H_n	0.7452	0.7482
σ	0.2009	0.0544
$\sigma / \sqrt{n}$	0.0016	0.0017
CPU / trial	0.0077	0.0280
total CPU	154	28

Table 1.1: Importance sampling versus simple Monte Carlo (The value of A is chosen such that $p(x)$ is normalised on the interval [0, 1]).

Question: What would you chose for A? Explain.

It is clearly visible that the variance per trial is reduced significantly when using importance sampling. The CPU consumption per trial is much higher for importance sampling. Still, since the number of trials necessary to obtain results with comparable accuracy is much lower for importance sampling, the total amount of time spent is lower than when just applying Monte Carlo integration.

2 Markov Chains and Metropolis

2.1 Introduction: Markov chains

In classical physics, a basic role is played by the fundamental principle of scientific determinism: from the state of a physical system at the time t_0 , one may deduce its state at a later instant t. As a consequence one obtains a basic method of analysing physical systems: the state of a physical system at a given time t_2 may be deduced from a knowledge of its state at any earlier (later) time t_1 and does not depend on the history of the system before (after) time t_1.

For physical systems, which obey probabilistic laws rather than deterministic laws, one may use an analogous principle: the probability that the physical system will be in a given state at a given time t_2 may be deduced from a knowledge of its state at any earlier time t_1, and does not depend on the history of the system before time t_1. Stochastic processes which represent observations on physical systems satisfying this condition are called Markov processes.

A special kind of Markov process is a Markov chain; it may be defined as a stochastic process whose development may be treated as a series of transitions between certain values (called the "states" of the process) which have the property that the probability law of the future development of the process, once it is in a given state, depends only on the state and not on how the process arrived in that state.

Example: Written language as a Markov chain

The letters of the alphabet may be divided into two categories, vowels and consonants. Let us denote a letter by a 0 if it is a vowel and by a 1 if it is a consonant. A page of written text then appears as a sequence of 0's and 1's. The vowels and consonants form a Markov chain if given any string of letters the probability for the next letter to be a vowel or consonant (0 or 1) is the same as the probability that the next letter will be a vowel or consonant knowing only the nature of the last letter of the string. For most languages this would not be the case, although it does seem to be the case for sufficiently simple languages such as Samoan. In this language a consonant is never followed by a consonant, and a vowel has probability 0.51 of being followed by a vowel.

Example: Children throwing a ball

A group of 4 children playing a game by throwing a ball to one another. At each stage the child with the ball is equally likely to throw it to any of the other 3 children. Let X_0 denote the child who had the ball originally, and for $n \geq 1$ let X_n denote the child who has the ball after it has been tossed exactly n times. The "chain" of outcomes $X_0, \ldots, X_n$ is a Markov chain.

2.2 Definitions

A *Markov process* is a sequence of trials (or observations), where the probability of the outcome of a given trial depends only on the outcome of the previous trial. Let X_k be a stochastic variable denoting the outcome of the k-th trial. The transition probability at trial k for each pair (i, j) of outcomes is defined as:

$$P_{ij}(k) = P[X_k = j \mid X_{k-1} = i \ ]. \tag{2.1}$$

This describes the probability that given the present state is i the next state will be j. The matrix $P(k)$ whose elements are given by (2.1) is called the transition matrix. For example the transition matrix for the ball-throwing children is given by:

$$P = \begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix}. \tag{2.2}$$

So when at a certain point a specific child has the ball it will throw it to one of the other 3 children with equal probability.

A Markov process for which the state space (the set of possible outcomes) is discrete is called a *Markov chain*. For example the state space in case of the children throwing a ball is given by the children's "names": {0, 1, 2, 3}.

A Markov chain is called *finite* if it is defined on a finite set of outcomes. For example the number of ball-throwing children is 4, which is definitely a finite set of possible states.

A Markov chain is called *inhomogeneous* if the associated transition probabilities depend on the trial number k. If the transition probabilities are independent of the trial number, the Markov chain is called *homogeneous*. For example the ball-throwing children. Say the children have thrown the ball in total k times. The transition probability matrix is still the same. The ball is still thrown with equal probability to another child as when started.

A Markov chain with transition matrix P is *irreducible* or *ergodic*, if for each pair of solutions (i, j) there is a larger than zero probability of reaching j from i in a finite number of trials. So given an arbitrary initial state every outcome in the state space is reached with probability > 0, for an infinitely long Markov chain that is irreducible. For the ball-throwing children: every child is likely to get the ball at least once if the game continues forever.

Example: Discrete parameter Markov chain

Consider a physical system which is observed at a discrete set of times. Let the successive observations be denoted by $X_1, X_2, \ldots, X_n, \ldots$. It is assumed that X_n is a random variable. The sequence $\{X_n\}$ is called a chain if it is assumed that there are only a finite or countable infinite number of states in which the system can be. The sequence $\{X_n\}$ is a Markov chain if each random variable X_n is discrete and if the following condition is satisfied: for any integer $m>2$ and any set of m points $n_1< n_2< \cdots <n_m$ the conditional distribution of X_{n_m}, for given values of $X_{n_1}, X_{n_2}, \ldots, X_{n_{m-1}}$ depends only on $X_{n_{m-1}}$, the most recent known value; in particular, for any real numbers $x_1, x_2, \ldots, x_m$ it holds that

$$P\left[X_m = x_m \mid X_1 = x_1, \ \ldots, \ X_{m-1} = x_{m-1}\right] = \ P\left[X_m = x_m \mid X_{m-1} = x_{m-1}\right]. \qquad (2.3)$$

2.3 The stationary distribution of a Markov chain and (a)-periodicity.

Finally, the notion of what can be understood as the "stationary distribution" of a Markov chain will be introduced. If we have a specific Markov chain that runs through a certain (discrete) state space (for example the children) then we denote the stationary distribution of this chain as the set of probabilities that describe that after infinitely many "trials" the Markov chain will be in a certain state. For the case of the children intuitively one may deduce immediately that on the long run every child has equal probability to have the ball.

The stationary distribution of a finite homogeneous Markov chain with transition matrix P is defined as the vector $\vec{q}$, whose ith component is given by :

$$q_i = \lim_{k\longrightarrow\infty} P\left[X(k) = i \mid X(0) = j\right] \quad \forall\ j.$$

In case of the ball-throwing children $q_i=\frac{1}{4}$, $\forall\ i$. So in the long run the ball is with equal probability in the possession of any child, no matter what the starting conditions where.

Finally we'll introduce the notion of a-periodicity of a Markov chain. Given a certain state that a Markov chain is in. If we can return into this same state within one Markov step we call the chain *a-periodic*. Else it is called periodic. For example the ball-throwing children: since a child is not allowed to throw the ball to itself the next state within the Markov chain is never the same as the current state. Therefore the Markov chain generated by the children is periodic with period 2.

2.4 The Metropolis Algorithm

■ 2.4.1 Introduction

Given the idea of Markov chains we can, without proving, discuss the algorithm which has been introduced by Metropolis (see [14]).

It is often very difficult or even impossible to generate random variables with an arbitrary distribution. Using the Metropolis algorithm one can generate a homogeneous Markov chain $\{X_n\}$ that steps through a state space, such that the points X_n are distributed according to the required probability density function $p(x)$.

In the Metropolis algorithm we generate a chain of configurations X_n of the system of interest, such that the number of configurations in the "interval" $[X, X+dX]$ is proportional to $p(x)dx$. We do this as follows: Suppose that after some time the system has arrived in state X_n. We generate a trial configuration X_t from X_n by making a small change to X_n. (What is meant by "a small change" depends on the kind of system that is studied.) We then calculate the ratio:

$$\frac{p(X_t)}{p(X_n)} = r. \tag{2.4}$$

Dependent on the value of r the trial configuration is accepted or rejected. If $r \geq 1$ the trial is accepted. If $r<1$ the trial is always accepted with probability r. We do this by drawing a random uniform [0,1] number ξ. If $r \geq \xi$ we accept X_t, else we reject it. If the trial configuration is accepted we set $X_{n+1}=X_t$, else $X_{n+1}=X_n$. By repeating this Markov process we generate a chain of configurations X_n, n = 1, 2, …, N. If N is large enough, every configuration will occur in the chain with a frequency that is proportional to the probability density $p(x)$. Figure 2.1 shows the mechanism of the Metropolis algorithm graphically for an exponential density function.

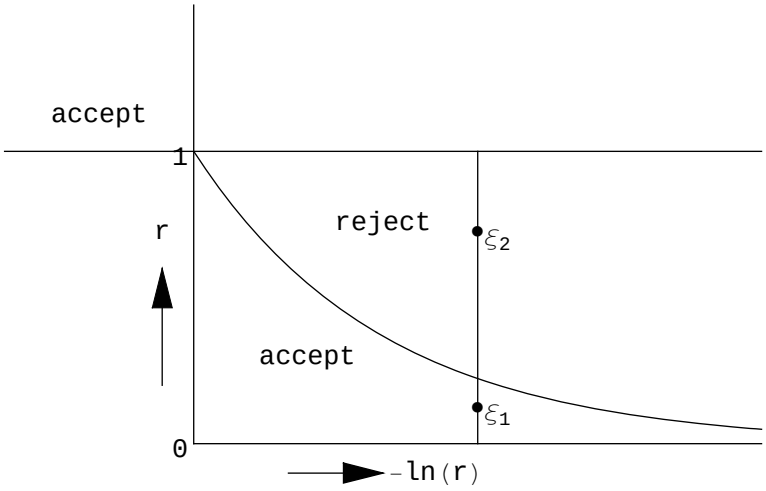


Figure 2.1: If $r \geq 1$ the change is accepted. If $r < 1$ a random number ξ is drawn and the change is accepted if $r \geq \xi$.

■ 2.4.2 Recipe for the Metropolis Algorithm

One starts by defining a "random walk" with a transition probability Γ_{ij} being the probability to get from "state" X_i to X_j such that the distribution of the points (or states) $X_i$converges to $p(x)$. A sufficient condition is the so-called detailed balance condition:

$$p(X_i)\,\Gamma_{ij} = p(X_j)\,\Gamma_{ji} \tag{2.5}$$

Which states that the flow of going from state i to j is equal to the flow of going from state j to i, where the flow of going from state i to j is given by the probability of being in that state i times the probability of going from state i to j. For example:

$$\Gamma_{ij} = \min\left[1, \frac{p(X_j)}{p(X_i)}\right]. \tag{2.6}$$

Given this, one can generate X_{i+1} from X_i by the following pseudo code:

```
Choose trial position X_t=X_i+δ_i with δ_i random over [-δ,δ]

Calculate Γ_ij = min [1, p(X_i+1)/p(X_i)]

if Γ_ij >= 1: accept move and put X_i+1=X_t
if Γ_ij <  1: generate random number R (uniform on [0,1])
              if R <= Γ_ij accept move and put X_i+1=X_t
              else         put X_i+1=X_i
```

It is efficient to take δ such that roughly 50% of the trials are accepted. Furthermore the walk can be started best at a value of X at which $p(X)$ has a maximum. It can be shown that this method guarantees

that, for a large number of steps, "all states" are explored and an equilibrium will be found according to Γ_{ij}.

■ 2.4.3 Variance in the Metropolis-Markov chains

If we apply the method of Metropolis to estimate a specific stochastic parameter θ, we may be interested in an estimate of the variance involved. Due to the fact that now the successive trials (or observables) are statistically correlated it is not allowed to estimate the variance (fluctuations) by the formula:

$$\mathrm{Var}(X) = s^2 = \frac{1}{n}\sum_{i=1}^{n} X_i^2 - \frac{1}{n^2}\left(\sum_{i=1}^{n} X_i\right)^2. \tag{2.7}$$

In order to be able to estimate the variance of the stochastic variable θ correctly, we can only use observables that are separated in time so far that the statistical correlation between them has "died" out. Typically one may write down a so called auto-correlation function for the simulated observables $\{X_n\}$ in the following manner:

$$C_k = \frac{\langle X_i\, X_{i+k}\rangle - \langle X_i\rangle^2}{\left\langle X_i^2\right\rangle - \langle X_i\rangle^2}, \tag{2.8}$$

where

$$\langle X_i\, X_{i+k}\rangle = \frac{1}{n-k}\sum_{i=1}^{n-k} X_i\, X_{i+k}. \tag{2.9}$$

Such a correlation function has the characteristic that it dies out for increasing k. We can calculate a "correlation time" τ by:

$$\tau = \sum_k C_k\,. \tag{2.10}$$

One may then estimate the error by:

$$\sigma_n^2 = \frac{2\,\tau}{n-1}\left[\frac{1}{n}\sum_{k=1}^{n} X_k^2 - \frac{1}{n^2}\left(\sum_{k=1}^{n} X_k\right)^2\right]. \tag{2.11}$$

In practice very often the variance is calculated from block averages. We split up the chain in M blocks of length n/M. We suppose that the mean of every block is independent of the other blocks. Let X_m be the average of X over the mth block. Then

$$\sigma^2_{M\ \mathrm{blocks}} = \frac{1}{M-1}\left[\frac{1}{M}\sum_{m=1}^{M} X_m^2 - \frac{1}{M^2}\left(\sum_{m=1}^{M} X_m\right)^2\right] \tag{2.12}$$

is an estimate of the variance. The block length should be taken at least to be 2τ. Why this is a good value for the block length is not explained here. An additional problem shows up in general when choosing a starting configuration. An arbitrary starting configuration will not be representative of the so called "equilibrium" distribution. In practice therefore one has to equilibrate the system for some time. When the system has reached equilibrium one can start calculating characteristics of the system.

3 MC simulation of the Canonical Ensemble

3.1 The Canonical Ensemble

■ 3.1.1 Introduction

Most physical systems exchange energy with their environment. Since such systems are usually small in comparison to the environment, we can assume that the change in energy of the smaller system does not influence the temperature of the larger system. Hence the larger system acts as a heat reservoir or heat bath at a fixed absolute temperature *T*.

If a small -albeit macroscopic- system is placed in thermal contact with a heat bath, the system reaches thermal equilibrium by exchanging energy with the heat bath until it attains the temperature of the heat bath. In this state the entropy reaches its maximum.

The system can exist in any one of a number of microscopic states that have the same total energy. Since we only know the total energy we have no way to distinguish one microscopic state from the other or to assign probabilities to those various states.

We can imagine the evolution of a system as a flow of points in 6N-dimensional phase space (3 space coordinates and 3 momentum coordinates per particle). If X^N is a point of the phase space, then $H\left(X^N\right)$ is the energy of that configuraton. The phase space can be divided into layers of configurations that have the same energy - such layers are called energy surfaces. If the system is ergodic, what means that averaging over time is equivalent to averaging over the phase space, the probabilities of microstates are based on mechanical properties of the system. Therefore it is equally probable to find the state of the system in different regions of the energy surface if those regions are equal in size. In other words, if the system is ergodic, it will spend equal times in equal areas of the energy surface.

We can define $\Sigma(E)$ to be the area of energy surface. Then, the ergodicity implies that the probability distribution for the energy surface will be:

$$p\left(X^N, E\right) = \begin{cases} \frac{1}{\Sigma(E)} & \text{for } H\left(X^N\right) = E \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

For fixed temperature *T* the probability that the system is in micro state *s* with energy E_s, is given by the Boltzmann distribution:

$$P_s = \frac{1}{Z} e^{-\frac{E_s}{k_B T}}, \tag{3.2}$$

where Z is a normalisation constant. The ensemble defined by (3.2) is known as the *Canonical Ensemble*. Since $\sum_s P_s$=1, Z is given by:

$$Z = \sum_{s=1}^{M} e^{-\frac{E_s}{k_B T}} . \tag{3.3}$$

The summation in (3.3) is over *all M* micro states of the system. The quantity Z is known as the *partition function* of the system.

We can use (3.2) to obtain the ensemble average of the physical quantities of interest [6]. For example, the mean energy is given by:

$$\langle E \rangle = \sum_s E_s P_s = \frac{1}{Z} \sum_s E_s \; e^{-\frac{E_s}{k_B T}} . \tag{3.4}$$

To calculate the mean energy we integrate over all possible energy values (β stands for $\frac{1}{k_B T}$):

$$\langle E_s \rangle = \frac{\int_0^\infty E_s P_s \, dE_s}{\int_0^\infty P_s \, dE_s} = \frac{\int_0^\infty E_s \, e^{-\beta E_s} \, dE_s}{\int_0^\infty e^{-\beta E_s} \, dE_s} \tag{3.5}$$

While integrating the denominator is elementary, to deal with nominator we use integration by parts, namely the formula :

$$\int_a^b u \, \mathrm{dv} = u \, v \, |_a^b - \int_a^b v \, \mathrm{du} \tag{3.6}$$

with $u = E_s$, $d\,v = e^{-\beta E_s} \, d\, E_s$ and $v = -\beta^{-1} \, e^{-\beta E_s}$. Putting that into (3.6) and (3.5) we get both integrals calculated:

$$\langle E_s \rangle = \frac{\beta^{-2}}{\beta^{-1}} = \frac{1}{\beta} = k_B \, T$$

Note that the energy can fluctuate in the canonical ensemble as in contrast to the micro canonical ensemble. We can derive various important relations among equilibrium quantities by exploitation of this fluctuation possibility. For example let's consider the relation of constant volume response to energy fluctuations in the canonical ensemble:

For simplicity we adopt the notation U=$\langle E \rangle$. From the definition of the heat capacity C we have:

$$C \equiv \frac{\partial U}{\partial T} = - \frac{1}{k_B \, T^2} \frac{\partial U}{\partial \beta} . \tag{3.7}$$

From (3.3) we can compute:

$$\frac{\partial}{\partial \beta} \ln Z = \frac{\partial}{\partial \beta} \ln \sum_{s=1}^{M} e^{-\beta \mathrm{E}_s} = \frac{1}{Z} \frac{\partial}{\partial \beta} \sum_{s=1}^{M} e^{-\beta \mathrm{E}_s} = \frac{1}{Z} \sum_{s=1}^{M} \frac{\partial}{\partial \beta} e^{-\beta \mathrm{E}_s} = \frac{1}{Z} \sum_{s=1}^{M} (-E)\, e^{-\beta \mathrm{E}_s} = -\langle E \rangle = -U \tag{3.8}$$

It is a useful formula for calculating the mean energy U from the given partition function Z. Differentiating once again, we have:

$$\frac{\partial U}{\partial \beta} = \frac{1}{Z^2} \left(\frac{\partial Z}{\partial \beta} \right)^2 - \frac{1}{Z} \sum_s E_s^2\, e^{-\beta \mathrm{E}_s} = \langle E \rangle^2 - \left\langle E^2 \right\rangle. \tag{3.9}$$

So from (3.7) and (3.9) we arrive at a very useful description of the heat capacity C as a funcion of the energy dissipation:

$$C = \frac{1}{k_B T^2} \left(\left\langle E^2 \right\rangle - \langle E \rangle^2 \right). \tag{3.10}$$

Note that the heat capacity is derived at constant volume since the partial derivatives were performed with the energy levels E_s kept constant.

The magnetic susceptibility is another example of a response function, since it measures the ability of -for instance- a spin to respond or flip due to a change in an external magnetic field. The zero field isothermal magnetic susceptibility is defined by the thermodynamic derivative:

$$\chi = \lim_{H \longrightarrow 0} \frac{\partial \langle M \rangle}{\partial H}. \tag{3.11}$$

And in close resemblance to the derivation of the heat capacity, we can relate the zero field susceptibility to the magnetisation fluctuations in the system by:

$$\chi = \frac{1}{k_B T} \left(\left\langle M^2 \right\rangle - \langle M \rangle^2 \right). \tag{3.12}$$

In the next paragraphs we will discuss the Ising spin model as an example of a method to study the behaviour of canonical ensembles by means of Monte Carlo techniques.

■ 3.1.2 Simulating the Canonical Ensemble

To estimate the quantity A (e.g. heat capacity, magnetisation) by means of simulation, we should generate a number of microstates and take the mean value of A over that sample. We can write this in the form:

$$A_m = \frac{\sum_{s=1}^{m} A_s\, e^{-\beta \mathrm{E}_s}}{\sum_{s=1}^{m} e^{-\beta \mathrm{E}_s}} \tag{3.13}$$

However, if we generate the microstates in a simple way, many of them may have very small probabilities $\left(e^{-\beta \mathrm{E}_s}\right)$. Such states contribute very little to the mean value and make the simulation ineffective. To do it better, we should make use of *importance sampling* introduced in Chapter 1.3. As in section 1.3.3, we can multiply and divide (3.13) by g_s:

$$A_m = \frac{\sum_{s=1}^{m} (A_s / g_s)\, e^{-\beta \mathrm{E}_s}\, g_s}{\sum_{s=1}^{m} (1 / g_s)\, e^{-\beta \mathrm{E}_s}\, g_s} \tag{3.14}$$

Next, if we generate microstates with probability g_s, then we get:

$$A_m = \frac{\sum_{s=1}^{m} (A_s / g_s)\, e^{-\beta \mathrm{E}_s}}{\sum_{s=1}^{m} (1 / g_s)\, e^{-\beta \mathrm{E}_s}} \tag{3.15}$$

Now we have to choose probability distribution g_s to minimise the variance. Mertopolis suggested that it should be the Boltzmann probability:

$$g_s = \frac{e^{-\beta \mathrm{E}_s}}{\sum_{s=1}^{m} e^{-\beta \mathrm{E}_s}} \tag{3.16}$$

Such choice of $g_s$gives us a simple method of getting A_m. By inserting (3.16) into (3.15), we get:

$$A_m = \frac{\sum_{s=1}^{m} A_s}{m} \tag{3.17}$$

We have shown that we can obtain our estimated result by generating microstates with given probability distribution. How can we get such distribution? The answer is the Metropolis algorithm introduced in Sec. 2.4.2.

According to the Metropolis recipe, we generate a random state (n) that differs a little from the initial (m). Then we compute the energy difference beetween them and the ratio of their probabilities:

$$\Delta E_{nm} = E_n - E_m \text{ and } \frac{P_n}{P_m} = \frac{e^{-\beta E_n}}{e^{-\beta E_m}} = e^{-\beta \Delta E_{nm}} \tag{3.18}$$

Then, if the $\Delta \mathrm{E}_{\mathrm{nm}} \leq 0$ the new state is accepted. If $\Delta \mathrm{E}_{\mathrm{nm}} > 0$ the new state is accepted with probability $e^{-\beta \Delta E_{nm}}$: we generate a random number ξ form [0,1] and accept the state if $\xi < e^{-\beta \Delta E_{nm}}$.

3.2 Simulating the Ising Model

■ 3.2.1 Introduction

As an example of a system that is described by the canonical ensemble we will use Ising Model. The "Ising" system serves as one of the simplest models of interacting bodies in statistical physics. The model has been used to study ferromagnets, anti-ferromagnetism, phase separation in binary alloys, spin glasses and neural networks. It has been suggested [2] that the Ising model might be relevant to imitative behaviour in general, including such disparate systems as flying birds, swimming fish, flashing fireflies, beating heart cells and spreading diseases.

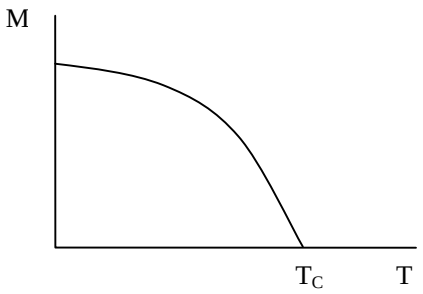
The probabilistic Ising model employs a constant temperature condition (known as the canonical ensemble formulation). Essentially, the (2D) model is comprised of an "*n* by *n"* square lattice in which each lattice site s has associated with it a value of 1 (up spin) or -1 (down spin). Spins on adjacent, nearest-neighbour sites interact in a pair-wise manner with a strength *J* (the exchange energy). When *J* is positive, the energy is lower when spins are in the same direction and when *J* is negative, the energy is lower when spins are in opposite directions. There may also be an external field of strength *H* (the magnetic field). The magnetisation of the system is the difference between the number of up and down spins on the lattice - it reaches the maximum if all spins are oriented in the same direction. The energy of the system is given by:

$$E = -J \sum_{\langle ij \rangle} s_i s_j - \mu_0 H \sum_i s_i , \tag{3.19}$$

with first sum over all pairs of spins which are nearest neighbours and the second term the interaction energy *J* of the magnetic moment μ_0 with the external magnetic field *H*. In spin flipping, lattice sites are selected and either flipped or not, based on the energy change in the system that would result from the flip. The simulation is essentially a Markov Chain simulation.

In the (probabilistic) Ising model we assume a constant temperature condition (the canonical ensemble formulation). A lattice site is selected at random and a decision is made on whether or not to flip the site using the Metropolis Algorithm (see previous section). The Metropolis method allows the system to reach a "global" energy minimum rather than getting stuck in a "local" minimum at low temperature.

With respect to ferromagnetism we are familiar with materials such as iron and nickel which exhibit a spontaneous magnetisation in the absence of an applied magnetic field. This non zero magnetisation occurs only if the temperature is lower than a well-defined temperature known as the critical or Curie temperature T_c. For temperatures $T > T_c$ the magnetisation vanishes. Hence T_c separates the disordered phase for $T > T_c$ from the ferromagnetic phase for $T < T_c$.



Although the origin of ferromagnetism is quantum mechanical in nature, the study of the classical Ising spin model provides much insight into the properties of magnetic systems in the vicinity of a phase transition. Exceptions are systems with $T \ll T_c$ and models of iron and nickel where the individual spin moments are no longer localised.

The physical properties of interest which we want to extract from a Ising spin simulation are: The mean energy $\langle E \rangle$, the mean magnetisation $\langle M \rangle$, the heat capacity C and the magnetic susceptibility χ. We can estimate these properties at the phase transitions by examining the critical exponents (see Chapter 5.3) from computer experiments.

■ 3.2.2 Enumeration of the Ising model on a 2×2 lattice

The number of possible states or configurations of the Ising model increases as 2^N, therefore we can only enumerate the possible configurations for small N. As an example we calculate various quantities of interest for a 2×2 Ising model on a square lattice.

Two different configurations with 2 spins up are shown in Figure 3.1

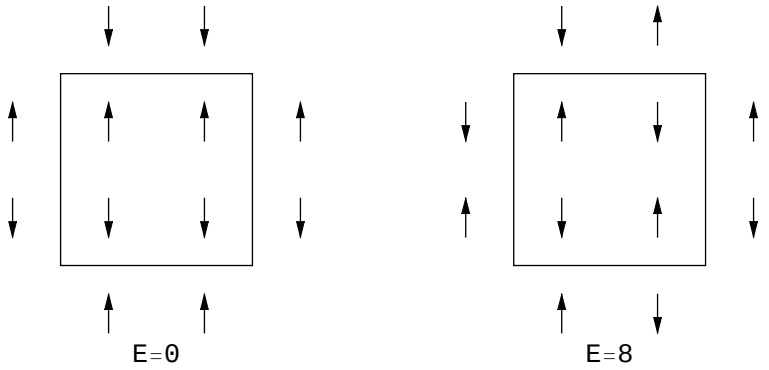


Figure 3.1: Examples of Ising configurations on a 2×2 square lattice, with boundary conditions.

In Table 3.1 we list the states according to their total energy and magnetisation.

# of spins UP	Degeneracy	Energy	Magnetisation
4	1	$-8J$	4
3	4	0	2
2	4	0	0
2	2	$8J$	0
1	4	0	-2
0	1	$-8J$	-4

Table 3.1: Energy and magnetisation of 24 states of the zero-field 2×2 Ising model.

Next we can compute all the quantities of interest using Table 3.1. The partition function is given by:

$$Z = 2e^{8\beta J} + 12 + 2e^{-8\beta J}. \tag{3.20}$$

From this equation and (3.8) we find:

$$U = -\frac{1}{Z}\left[2(8)e^{8\beta J} + 2(-8)e^{-8\beta J}\right]. \tag{3.21}$$

We can find the exact values for $\langle E^2\rangle$, $\langle M\rangle$, $\langle\,|M|\,\rangle$, $\langle M^2\rangle$ and the dependence of C and χ on βJ in the same manner.

In the limit of an infinite large lattice it is also possible to solve the 2D Ising model analytically (see for instance [10] and [13]): The best known value to date for the critical Temperature is $k_B T_c/J = 2.271$ ($k_B T_c/J$ =4.5108 for 3D).

■ 3.2.3 The 2D Ising Spin Algorithm

The 2D Ising Spin Algorithm

1] Create an "n by n" lattice consisting of randomly chosen site values of +1 and -1. The following sequence of steps 2-4 will be executed a number of times (this is described in step 5), first using the initial lattice configuration, and then using the lattice configuration resulting from the previous run-through of the sequence. We will describe the steps in terms of an arbitrary lattice configuration, called *Lattice*.

2] Select a random lattice site in *Lattice*.

3] Determine the energy change involved in "flipping" the spin at the selected lattice site. This is done in a number of steps:

3a] Determine the neighbours to the selected site. When the selected site is in the interior of *Lattice*, the neighbours are the sites north (above), south (below), west (left) and east (right) of the site. When the selected site is along the border of *Lattice*, some neighbours are taken from the opposing side of the lattice. This way of choosing neighbours for border sites is known as the reflecting or periodic boundary condition.

3b] The energy change that would result from flipping the spin of the selected lattice site is determined by the quantity, 2 x (value of selected site) x (B + J(total spin from neighbours)), where B and J are input values.

4] Use the Metropolis method to decide whether to flip the spin of the selected lattice site as follows:

4a] Check if there is a negative energy change as a result of the flip.

4b] If the energy change is non-negative, check if the exponential of (- energy change divided by temperature) is greater than a random number between 0 and 1.

4c] If one of these conditions is satisfied, flip the spin.

5] Execute the sequence of steps 2-4 m times.

6] Create a sublist, monteCarloStepLis, containing every n^2th element from the list of Lattice configurations. (The use of every n^2th element corresponds to giving each lattice site an equal chance to be selected. Each element in monteCarloStepLis is said to correspond to one Monte Carlo step).

7] Calculate for each element in monteCarloStepLis, some global property of the lattice, such as the long range order (the absolute value of the magnetisation of the lattice).

■ 3.2.4 An example run

In the next two simulations we study the natural phenomenon of ferromagnetism as an example [3, 4]. Non-zero magnetism only occurs if the temperature is lower than a well-defined temperature known as the Curie critical temperature T_c. For $T > T_c$ the magnetisation vanishes. T_c separates the disordered phase of the spin system for $T > T_c$ for the ordered ferromagnetic phase for $T < T_c$.

Explore the system for H=0 and $T>T_c$ or $T<T_c$.

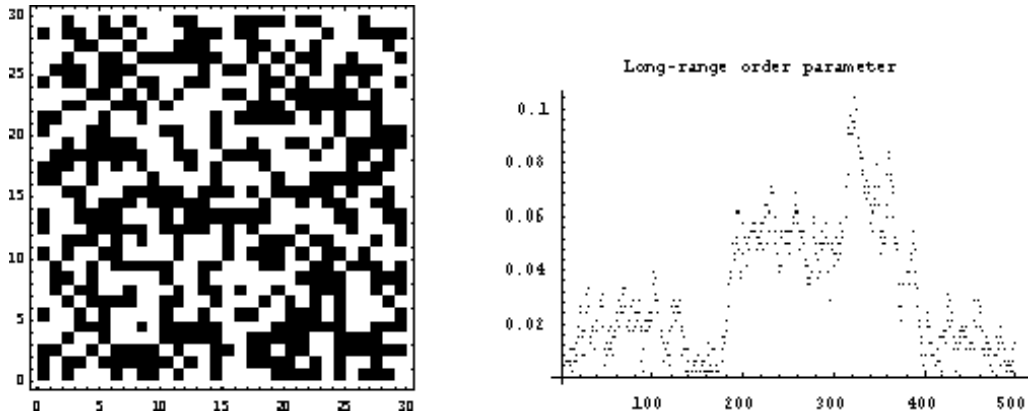


Figure 3.2: a) H=0; $T \gg T_c$, final configuration on a 30x30 lattice, b) (Dis)Order in system versus number of iterations

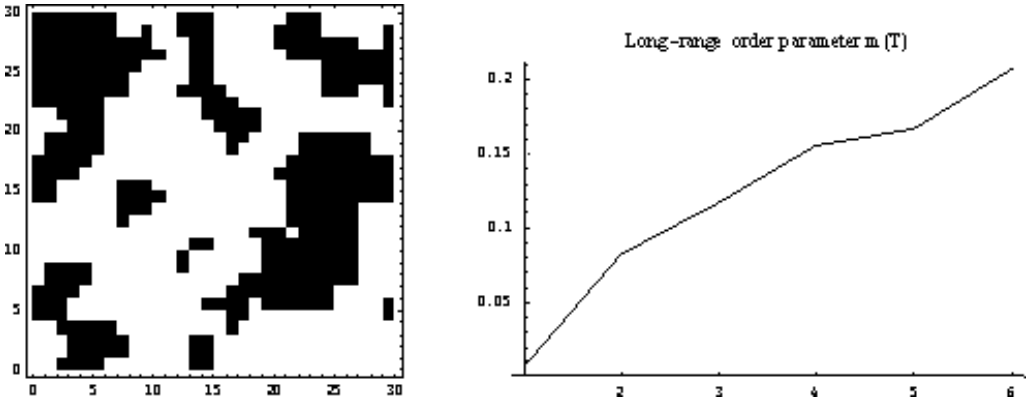


Figure 3.3: a) H=0; T just below T_c, final configuration on a 30x30 lattice, b) Order in system versus number of iterations

In this example we have performed more than 100000 Monte Carlo steps for each cell. To simulate a real system we would of course need 10^{23} cells instead of the 30^2. Fortunately the computer experiments given here already give us some insight in the basic physics of such spin systems. Careful experiments with detailed statistical analyses can indeed reveal very interesting physical phenomena from this simple probabilistic model. Especially the study of phenomena like critical slowing down and phase transitions in classical fluids are fields of active research.

■ 3.2.5 A Lattice gas variant of the Ising model.

The Ising model can describe other systems which might appear to have little in common with ferromagnetism. One variant for example of the Ising model, which has been used to model a lattice gas, replaces the Ising spin flip dynamics with *spin exchange dynamics* in which a pair of nearest neighbour sites is selected, the energy change due to interchanging their spins is determined, and the interchange decision is again made using the Metropolis algorithm. Note that the use of spin exchange dynamics conserves the number of spins in the lattice. In this "lattice gas" model "down spins" for instance represent a lattice site occupied by an atom and "up" spins an empty site. Note the resemblance with the lattice gasses discussed in Chapter 4. The type of Ising lattice gas represents a crude model of the behaviour of a real gas of atoms and is of historical importance as a model for the gas-liquid transition and the critical point. The important difference between a ferromagnet and a lattice gas is that the total number of atoms is fixed, whereas the number of "up" and "down" spins in a ferromagnet can change.

Since the spins correspond to atoms, we can compute the equilibrium single particle diffusion constant of the atoms. We can built a list to record the position of each spin (atom) as a function of time. After equilibrium has been reached, we choose the origin of time and compute $\langle R(t)^2 \rangle$, the mean-square displacement per atom after t units of time. Then - as discussed in Chapter 4 - if the "atoms" undergo a random walk, we can use Equation (4.31) to calculate the self-diffusion constant for different Temperatures and numbers of occupied sites.

■ 3.2.6 Simulation of simple classical fluids

Lattice gas models can also be applied to model fluid flow. Fluid flow is in general difficult, because it is described by the nonlinear Navier-Stokes equation. The main idea of lattice gases is that if the fundamental conservation laws and symmetries associated with fluids are maintained, that this ensures that the correct physics is produced at a macroscopic level after averaging over many particles.

The input of a lattice gas model is the microscopic behaviour. As the name already indicates the particles "live" on a lattice, rather than in free space. Therefore not only the position of the particles is restricted, but also their velocity vectors. In the simplest models only two processes are included, free motion between collisions and the collisions.

A simple example of a lattice gas is the FHP-I model as formulated by Frisch, Hasslacher and Pomeau. It is a 2-dimensional model and formulated on a triangular lattice. All particles will have the same speed and can only move along the links to neighbouring sites, hence there are 6 velocities:

$$\begin{aligned}
\vec{v}_1 &= (1,\ 0)\\
\vec{v}_2 &= \left(1,\ \sqrt{3}\right)\Big/ 2\\
\vec{v}_3 &= \left(-1,\ \sqrt{3}\right)\Big/ 2\\
\vec{v}_4 &= (-1,\ 0)\\
\vec{v}_5 &= \left(-1,\ -\sqrt{3}\right)\Big/ 2\\
\vec{v}_6 &= \left(1,\ -\sqrt{3}\right)\Big/ 2
\end{aligned} \tag{3.22}$$

During a collision we ensure conservation of mass and momentum and only 2- and 3-particles collisions are allowed, as is shown in Figure 3.4. If two particles collide head-on, the outgoing state is changed in one of the others with equal probability. In the case of a 3-body collision there is only a single choice of the outgoing state. If at a given node the configuration is not found in the "collision table" (Figure 3.4), no collision will occur.

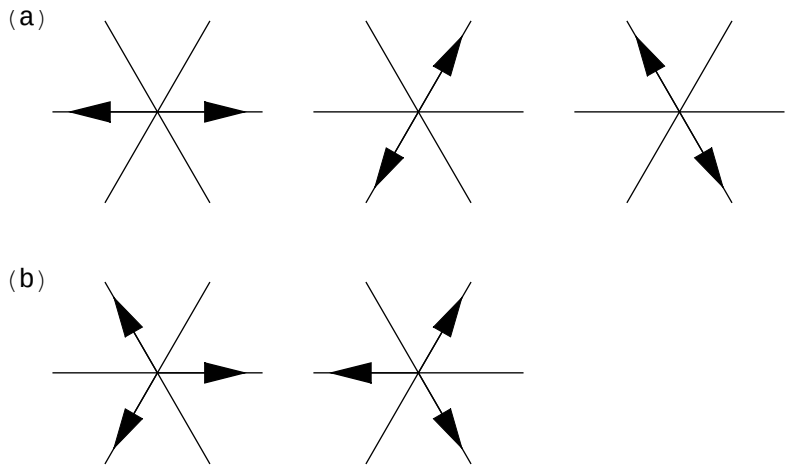


Figure 3.4: The collision rules of a FHP-I model. (a) In a head on collision the outgoing state is randomly selected from the two remaining diagrams. (b) In a 3-particle collision the outgoing state is the remaining one with probability 1.

In Figure 3.5 the two steps in a simulation of a lattice gas are shown. In the first step, the streaming, the particles are moved to the neighbouring lattice site along the direction of their velocity. In the second step it is checked for each lattice site whether the occupancy of the links is found in the collision table. If that is the case, the occupancy of the links is changed accordingly, otherwise nothing is changed.

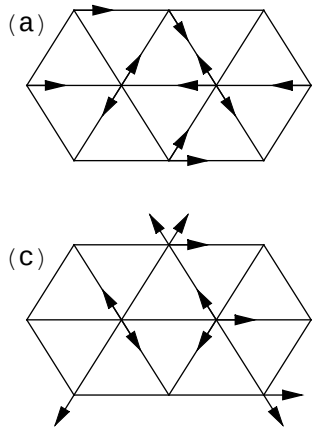


Figure 3.5: (a) An initial configuration of a FHP-I simulation. (b) The result after the streaming step. (c) The result after performing collisions.

Alternative models can be defined, which include more collisions and/or multiple speeds e.g. by including a rest particle. It is also possible to use a square lattice in 2 dimensions, although this will put some extra constraints on the number of particle speeds that should be included. This is caused by the difference in symmetry between the two lattices. Also extensions to 3 and 4 dimensions can be made.

4 Random Walk Methods

4.1 Introduction

In 1906 Pearson posed the "random walk" problem as follows: A drunkard begins at a lamppost and takes N steps of equal length in random directions, how far will the drunkard be from the lamp post? Since this formulation, the random walk model has been extremely useful to scientists in many fields who are studying stochastic (probabilistic) processes: physicists modelling the transport of molecules, biologists modelling the locomotion of organisms and economists modelling the time behaviour of financial markets. General introduction and references can be found in [4, 6, 17, 22]

Chapter 4.2 describes the one- and two-dimensional random walks, some shape characteristics of a walk and the notion of the critical exponent. Chapter 4.3 describes the Self-Avoiding walk and some applications of the method.

4.2 1D and 2D Random walks and Diffusion Process

■ 4.2.1 Simple Random Walk

The simplest Random Walk (RW) model consists of n steps of equal length, back-and-forth along a line. We start at (a lamp post) located at x=0, each step is of equal length l , the direction of each step is independent of the previous one. At each time step the walker has a probability p of a step right and q= 1- p of a step to the left. With np the number of steps to the right en nq the number of steps to the left, the total number of steps N=np+nq. The net displacement after N steps is x =(np-nq)l, with -$Nl \leq x \leq Nl$.

With $P_N(x)$ the probability that after N steps the walker has undergone a net displacement of x, we can compute the mean net displacement $\langle x_N \rangle$ and the variance σ_N^2 of the walker by (the averages are over all possible walks of N steps):

$$\langle x_N \rangle = \sum_{x=-Nl}^{Nl} x\, P_N(x)$$

$$\sigma_N^2 = \left\langle \Delta x_N^2 \right\rangle = \left\langle x_N^2 \right\rangle - \langle x_N \rangle^2, \quad \text{with}$$

$$\left\langle x_N^2 \right\rangle = \sum_{x=-Nl}^{Nl} x^2 P_N(x).$$

By using probability theory we can analyse the random walk analytically (in contrast to Monte Carlo simulation and exact enumeration).

After N steps the displacement x_N is given by:

$$x_N = \sum_{i=1}^{N} s_i \text{ where } s_i = \pm l \tag{4.2}$$

and

$$x^2{}_N = \left(\sum_{i=1}^{N} s_i \right)^2 \tag{4.3}$$

For $p = q = \frac{1}{2}$ we expect $x(N) \to 0$ for $N \to \infty$ (why?) This implies that $\langle x_N \rangle = 0$ for $N \to \infty$.

Therefore:

$$x^2{}_N = \sum_{i=1}^{N} s_i{}^2 + \sum_{i \neq j=1}^{N} s_i \, s_j \tag{4.4}$$

For $i \neq j$ we get

$$s_i \, s_j = \begin{cases} -l \; -l \\ -l \; +l \\ +l \; -l \\ +l \; +l \end{cases} \tag{4.5}$$

We see that on average there are equal chances for $+l^2$ and for $-l^2$. Hence:

$$\sum_{i \neq j=1}^{N} s_i \, s_j = 0 \tag{4.6}$$

For $i = j$ we get $s_i{}^2 = l^2$ so:

$$\sum_{i \neq j=1}^{N} s_i{}^2 = N \, l^2 \tag{4.7}$$

From above, we conclude that

$$\left\langle \Delta x_N^2 \right\rangle = N \, l^2. \tag{4.8}$$

and

$$\mathrm{Var}(x) = \left\langle x_N^2 \right\rangle - \langle x_N \rangle^2 = N\, l^2 \tag{4.9}$$

where Var(x) is called dispersion.

It can be shown [17] that for general case ($p \neq q$)the values for mean displacement and dispersion are given by:

$$\begin{aligned} \langle x_N \rangle &= (p - q)\, N\, l \\ \left\langle \Delta x_N^2 \right\rangle &= 4\, p\, q\, N\, l^2. \end{aligned} \tag{4.10}$$

We can reformulate the RW in terms of the diffusion of a molecule in a dilute gas, a very important interpretation of the RW model (in Appendix C on Bayes theorem we will come back to Diffusion in terms of stochastic processes). Assume a molecule travels a distance l between collisions with other molecules. If the successive displacements undergone by the molecule are "statistically independent", then the motion of the molecule is comporal to the motion of the drunkard. Since the motion of such a molecule can also be described by a diffusion process, we can write one in terms of the other.

■ 4.2.2 RW Example

Let's have a look on a simple random walk example. If we take only 3 steps ($N = 3$)of length $l = 1$ in one dimension ($d = 1$), we can explicitly enumerate all possible ways of evolution of such a system and compute the parameters defined above. From Table 4.1 we can see that there is only one possible way to end at $x = -3$ or $x = 3$ and that there are three possible ways to stop at $x = -1$ or $x = 1$.

$x = -3$	$x = -1$	$x = 1$	$x = 3$
← ← ←	← ← →	→ → ←	→ → →
	← → ←	→ ← →	
	→ ← ←	← → →	
q^3	$3\, p\, q^2$	$3\, q\, p^2$	p^3

Table 4.1 Probabilities for x after N=3 steps

First, we can compute

$$\begin{aligned} \langle x_3 \rangle = \sum_{x=-3}^{3} x\, P_3(x) &= -3\, q^3 - 3\, p\, q^3 + 3\, q\, p^2 + 3\, p^3 = \\ &= 3\left(p^2 - q^2\right)(p + q) = \\ &= 3\,(p + q)\,(p - q) = \\ &= 3\,(p - q) \end{aligned} \tag{4.11}$$

We get the same result as from (4.10). In the similar way we find

$$\left\langle x_3{}^2 \right\rangle = \sum_{x=-3}^{3} x^2\, P_3(x) = [3\,(p-q)]^2\,(p+q) + 12\left(p\,q^2 + q\,p^2\right) =$$

$$= [3\,(p-q)]^2 + 12\,\mathrm{pq}(p+q)$$

From (4.11) and (4.12) we obtain $\left\langle \Delta\, x_3{}^2 \right\rangle = 12\; p\, q$.

■ 4.2.3 The Diffusion Equation

The process of diffusion of a fluid molecule is described by the partial differential equation. We will obtain this equation by inspecting 1D random walk in continuum limit. If there is an equal probability of taking a step to the right or to the left, the random walk can be written in terms of the so-called "master equation":

$$P_N(i) \;=\; \frac{1}{2}\, P_{N-1}(i+1) \;+\; \frac{1}{2}\, P_{N-1}(i-1) \tag{4.13}$$

where $P_N(i)$ is the probability that a walker is at site i after N steps. In order to obtain a differential equation for the probability density $P(x, t)$, we set $t=N\tau$, $x=ia$ and $P_N(i)=aP(x, t)$, where τ is the time between steps and a is the lattice spacing. This notation allows us to rewrite (4.13) in the equivalent form:

$$P(x,\, t) \;=\; \frac{1}{2}\, P(x+a,\, t-\tau) \;+\; \frac{1}{2}\, P(x-a,\, t-\tau)\,. \tag{4.14}$$

Rearranging, subtracting $P(x, t\text{-}\tau)$ from both sides and dividing by τ results in:

$$\frac{1}{\tau}[P(x,\, t) \;-\; P(x,\, t-\tau)] = \frac{a^2}{2\,\tau}[P(x+a,\, t-\tau) \;-2\, P(x,\, t-\tau) + P(x-a,\, t-\tau)\,]\, a^{-2}. \tag{4.15}$$

Next, we can expand $P(x, t\text{-}\tau)$ and $P(x\pm\mathrm{a}, t\text{-}\tau)$ in a Taylor series:

$$\begin{aligned}
P(x,\; t-\tau) &= P(x,\, t) - \tau\, \frac{\partial P(x,\, t)}{\partial t} + \ldots \\
P(x+a,\, t-\tau) &= P(x,\, t) - \tau\, \frac{\partial P(x,\, t)}{\partial t} + a\, \frac{\partial P(x,\, t)}{\partial x} + \frac{a^2}{2}\, \frac{\partial^2 P(x,\, t)}{\partial x^2} + \ldots \\
P(x-a,\, t-\tau) &= P(x,\, t) - \tau\, \frac{\partial P(x,\, t)}{\partial t} - a\, \frac{\partial P(x,\, t)}{\partial x} + \frac{a^2}{2}\, \frac{\partial^2 P(x,\, t)}{\partial x^2} + \ldots
\end{aligned} \tag{4.16}$$

After inserting the above expansions into (4.15) the first order derivatives on the right-hand side are cancelled to 0 by sumation. Then taking the limit $a\to 0$ and $\tau\to 0$ with ratio $D = a^2/2\tau$ finite, we obtain the diffusion equation:

$$\frac{\partial P(x,\, t)}{\partial t} \;=\; D\, \frac{\partial^2 P(x,\, t)}{\partial x^2}, \;\; \text{or in } 3 \text{ Dimensions}:$$

$$\frac{\partial P(\vec{r}, t)}{\partial t} = D \vec{\nabla}^2 P(\vec{r}, t), \text{ with}$$

$$\vec{\nabla}^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \text{ the Laplacian operator.}$$

Equation (4.17) is known as the diffusion or Fokker-Planck equation and is frequently used to describe continuous processes. The proportionality constant D is known as the "self-diffusion constant" of a molecule.

■ 4.2.4 Solution of Diffusion Equation

The solution of the 1-dimensional Fokker-Planck equation in free space can be shown to be a Gaussian whose width increases as $t^{1/2}$:

$$P(x, t) = \frac{1}{\sqrt{4 \pi D t}} e^{-\frac{1}{2}\left(\frac{x^2}{2 D t}\right)}, \tag{4.18}$$

where $P(x,t)$ is the probability of finding the particle at point x at time t if the particle started at x=0 at t=0. To prove that (4.18) is the solution of diffusion equation, we can insert it into (4.17) and after a few differentiations we can see that it works.

The Figures 4.1 and 4.2 show how the probability distribution for 1D and 2D changes with time:

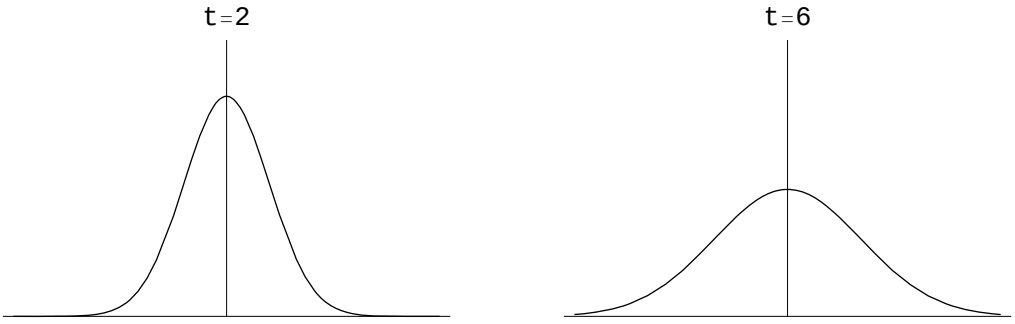


Figure 4.1: $P(x, t)$for two time points

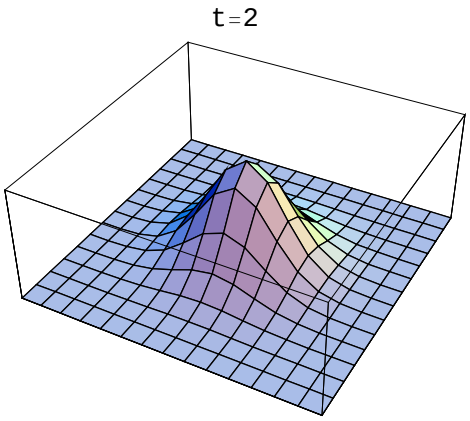


Figure 4.2: $P(x, y, t)$for two time points

The numerical solution of the prototypical parabolic partial differential equation (4.17) is explained in [10] and [16]. An indirect method of analysis of the Fokker-Planck equations (4.17), is to use a Monte Carlo method, that is to replace (4.17) by a corresponding random walk on a lattice with discrete time steps. Since the asymptotic behaviour of the partial differential equation and the random walk model are *equivalent*, this approach can be seen as using the Monte Carlo method as a method of numerical analyses. In contrast, if our goal is to understand a random walk lattice model directly, the Monte Carlo technique is a simulation method, the difference is disappearing. In the course Parallel Scientific Computing and Simulation [19], we tossed the name "Natural Solvers" for this type of techniques and use them to exploit the implicit parallelism in the problem at hand.

■ 4.2.5 Self-diffusion Constant and Random Walk

In this subsection we will find the relation between the self-diffusion constant D and the random walk properties. We will do it by comparing the mean square displacement $\langle \Delta x^2 \rangle$ for both cases.

First, we introduce time scale into random walk: if we take the time between steps τ to be such that $N=t/\tau$ with step lengths l then (4.10) can be rewritten in the form:

$$\langle \Delta x^2 \rangle = 4\, p\, q\, l^2 (t/\tau). \tag{4.19}$$

Next, we will compute $\langle \Delta x^2 \rangle$ for probability distribution $P(x, t)$ satisfying the diffusion equation. As a first step we will find $\langle x(t) \rangle$, which is defined as:

$$\langle x(t) \rangle = \int_{-\infty}^{\infty} x\, P(x, t)\, d\, x \tag{4.20}$$

To find the result, we multiply both sides of diffusion equation by x and integrate:

$$\int_{-\infty}^{\infty} x\, \frac{\partial P(x, t)}{\partial t}\, d\, x = D \int_{-\infty}^{\infty} x\, \frac{\partial^2 P(x, t)}{\partial x^2}\, d\, x \tag{4.21}$$

Left-hand side integral can be simply transformed, since x and t are independent variables:

$$\int_{-\infty}^{\infty} x \frac{\partial P(x,\, t)}{\partial t}\, d\, x = \frac{\partial}{\partial t} \int_{-\infty}^{\infty} x\, P(x,\, t)\, d\, x = \frac{\partial}{\partial t} \langle x\,(t) \rangle \tag{4.22}$$

The right-hand side can be integrated by parts (see Equation 3.6), what leads to:

$$D \int_{-\infty}^{\infty} x \frac{\partial^2 P(x,\, t)}{\partial x^2}\, d\, x = D \Big[x \frac{\partial P(x,\, t)}{\partial x} \Big|_{x=-\infty}^{x=+\infty} - \int_{-\infty}^{\infty} \frac{\partial P(x,\, t)}{\partial x}\, d\, x \Big] \tag{4.23}$$

The first term is equal to zero because $P(x,t)$ and all its spatial derivatives are 0 in infinity (it is a probability distribution function!). The second term also zeroes: $P(x = \infty,\, t) - P(x = -\infty,\, t)=0$. Hence, we find that:

$$\frac{\partial}{\partial t} \langle x\,(t) \rangle = 0 \tag{4.24}$$

what means that $\langle x(t) \rangle$ is constant. Since $x = 0$ at $t = 0$ we can see that $\langle x(t) \rangle=0$ for any t.

To compute $\left\langle \Delta x^2 \right\rangle$ the same procedure is required. Since $\langle x(t) \rangle=0$, we can use the notation $\left\langle \Delta x^2 \right\rangle=\left\langle x^2 \right\rangle$. We start from diffusion equation multiplied by $x^2$and integrate:

$$\int_{-\infty}^{\infty} x^2 \frac{\partial P(x,\, t)}{\partial t}\, d\, x = D \int_{-\infty}^{\infty} x^2 \frac{\partial^2 P(x,\, t)}{\partial x^2}\, d\, x \tag{4.25}$$

The left-hand side gives us $\frac{\partial}{\partial t} \left\langle x^2\,(t) \right\rangle$ in the same way as above. The right-hand side has to be integrated by parts twice, what gives:

$$\begin{aligned} D \int_{-\infty}^{\infty} x \frac{\partial^2 P(x,\, t)}{\partial x^2}\, d\, x &= D \Big[x^2 \frac{\partial P(x,\, t)}{\partial x} \Big|_{x=-\infty}^{x=+\infty} - 2 \int_{-\infty}^{\infty} x \frac{\partial P(x,\, t)}{\partial x}\, d\, x \Big] = \\ &= D \Big[x^2 \frac{\partial P(x,\, t)}{\partial x} \Big|_{x=-\infty}^{x=+\infty} - 2\, x\, P(x) \Big|_{x=-\infty}^{x=+\infty} + 2 \int_{-\infty}^{\infty} P(x,\, t)\, d\, x \Big] \end{aligned} \tag{4.26}$$

The first two terms are equal to zero and the integral in the second term gives 1, since $P(x)$ is normalised probability distribution function. From that we achieved the equation:

$$\frac{\partial}{\partial t} \left\langle x^2\,(t) \right\rangle = 2\, D \tag{4.27}$$

what is equivalent to:

$$\left\langle x^2\,(t) \right\rangle = 2\, D\, t \tag{4.28}$$

In d-dimensional space it will be:

$$\left\langle x^2\,(t) \right\rangle = 2\, d\, D\, t \tag{4.29}$$

Now, we can return back to our random walk result. Comparing (4.19) and (4.29) we find that the diffusion constant of a random walker in one dimension is given by:

$$D = \frac{4\,p\,q\,l^2(t/\tau)}{2\,d\,t} = \frac{l^2}{2\,\tau} \text{ for } p = 1/2. \quad (4.30)$$

The random walk model in higher dimensions is a bit more complicated then the random walk in one dimension. In one dimension, each step of the walk is either 0 degrees (i.e., a forward step) or 180 degrees (i.e., a backwards step) with respect to the preceding step. In higher dimensions, a step can take a range of orientations with respect to previous steps. For 2D we consider a random walk on a lattice. This kind of walk is appropriately referred to as a (two-dimensional square) lattice walk. This walk consists of steps of uniform length, randomly taken in the North, East, South or West direction.

■ 4.2.6 Analyses of the two-dimensional lattice walk

In studying a process that is random in nature, we are often interested in its mean, or average, properties. To obtain the mean value of a quantity, the quantity is computed a number of times, the values that are obtained are summed, and the result is divided by the number of computations. We'll look at two measures of the mean "size" of a two-dimensional lattice walk: "the mean-square end-to-end distance" and "the mean-square radius of gyration".

The square end-to-end distance, r^2, of a two-dimensional lattice walk is given by $(x_f - x_i)^2 + (y_f - y_i)^2$ where (x_i,y_i) and (x_f,y_f) are the initial and final locations of the walk, respectively. Choosing the origin (0, 0) as the starting point of the lattice walk, simplifies the formula to $(x_f^2 + y_f^2)$.

The mean square radius of gyration, $\langle R_g^2 \rangle$, of a random walk is the sum of the squares of the distances of the step locations from the centre-of-mass divided by the number of step locations. The centre of mass is the sum of the step locations divided by the number of step locations. The computations of the centre of mass and the sum of the squares of step distances from the centre of mass are straightforward (see Section 5.3.2).

The critical exponent of a random walk is a measure that has been found experimentally (in the laboratory) and theoretically (on paper) indicating that both $\langle r^2 \rangle$ and $\langle R_g^2 \rangle$ have a power law dependence on the number of steps in the walk, N. The power, ν, in the relationship

$$\left\langle \Delta x_N{}^2 \right\rangle \approx N^{2\nu} \quad (\text{for } N >> 1)$$

is known as the "critical exponent" of the walk, a typical example of an asymptotic scaling law: If the number of steps is doubled, the net mean square displacement of the walk is increased by a factor of 2^{ν}. For one dimensional random walks we find from (4.10) that ν= 1/2. The critical exponent is found to depend on the structure and dimension of the lattice as well as on the nature of the walk (see also Chapter 5.3).

■ 4.2.7 Monte Carlo Simulation of Self-diffusion Constant

We have shown that it is possible to siumulate diffusion process by means of random walk method. Using equation (4.29) we can estimate the self-diffusion constant:

$$D = \frac{\langle x^2(t)\rangle}{2\,d\,t} \tag{4.31}$$

During simulation, if we estimate the error of D, we can observe two effects, shown on Fig. 4.3

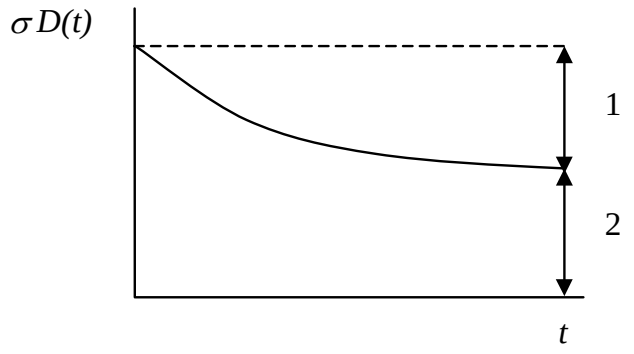


Figure 4.3: Estimated error of self-diffusion constant during simulation

Effect 1: At the beginning, the particle is still located in the neighbourhood of starting point.The simulation has not reached the stable state yet, so $\langle x\rangle \neq 0$.

Effect 2: After longer time we see that $\sigma\, D(t) \rightarrow c \neq 0$

How to estimate the behaviour of Var($D(t)$)? We know from (4.31) that

$$\mathrm{Var}(D) = \mathrm{Var}\left(\frac{\langle x^2(t)\rangle}{2\,d\,t}\right) \sim \frac{1}{t^2}\,\mathrm{Var}\left(x^2\right) \tag{4.32}$$

since $\mathrm{Var}(a\,x) = a^2\,\mathrm{Var}(x)$ (we use the notation $\langle x^2(t)\rangle = x^2$).

Next:

$$\mathrm{Var}\left(x^2\right) = \langle x^4(t)\rangle - \langle x^2(t)\rangle^2 \sim t^2 \tag{4.33}$$

since from (4.19) $\langle \Delta x^2 \rangle \sim \frac{l^2}{\tau}\,t$

Hence, combining (4.32) and (4.33) the time t is reduced. We conclude, that for large t the deviation of $D(t)$is time independent.

4.3 The Self-Avoiding Walk

The steps of the random walk model described, are independent of one another. Incorporating various short-range interactions between the steps, such as placing angular restrictions on successive steps, does not change the model's global properties. For example, the value of $\langle r^2 \rangle$ simply re-scales (changes by a proportionality constant). However, the introduction of long-range interactions, specifically, the property of self-avoidance (excluded volume) which prohibits a walk from returning to a location it has previously visited, fundamentally affects the properties of the walk. For instance, the critical exponent of the walk becomes non-integer.

The model of a random walk with excluded volume, known as the self-avoiding walk (SAW), has several areas of application. The leading area is in polymer physics where an SAW represents a long chain molecule consisting of many small molecules connected together by covalent chemical bonds (a polymer chain can be visualised as a pearl necklace). Actually there are few research areas in which random walk models play as important a role as in polymer physics. The earliest investigations (1934) of polymer configurations were phrased in terms of random walks. Over the past 50-60 years our understanding of the statistical properties of long, flexible polymer chains and of random walks has developed in parallel (typical observed values of the critical exponent equals ν=3/4 with pre-factors depending on the structure of the monomers and on the solvent).

Other systems that make use of the SAW model are: the zero-component ferromagnet, the $N \rightarrow 0$ limit of the N-vector model (a generalisation of the Ising model), and the SAW is relevant to the general subject of critical phenomena.

The SAW is quite difficult to analyse analytically (with paper-and-pencil) and computational experimentation has proven to be an indispensable tool for studying the model.

In studying the average properties of the random walk, a number of n-step walks are examined, each of which is grown by adding one step after another until n steps have been executed. This method of generating a sample of walks is not useful for studying the SAW because the occurrence of a single misstep resulting from a step intersecting a previous step, makes it necessary to discard the walk and start a new one from scratch. The likelihood of such a misstep becomes increasingly likely as the walk proceeds and hence, the attrition rate for generating SAW's by growing them is prohibitively high.

An alternative approach to producing SAW's is to start with a self-avoiding walk and rearrange its shape, counting each rearrangement as another SAW (this process guarantees that the length of the walk is conserved). We will look at a well-known algorithm for accomplishing this rearrangement: the "slithering snake" or "reptation" algorithm (respectively: [4], and [6] or more detailed in [21]).

For simplicity consider a model polymer chain in which all bond angles are 90°. As an example the 5

independent *N*=5 polymer chains are shown in Figure 4.4. Note that other chains differ only by rotation and/or reflection. The method then can be summarised as follows:

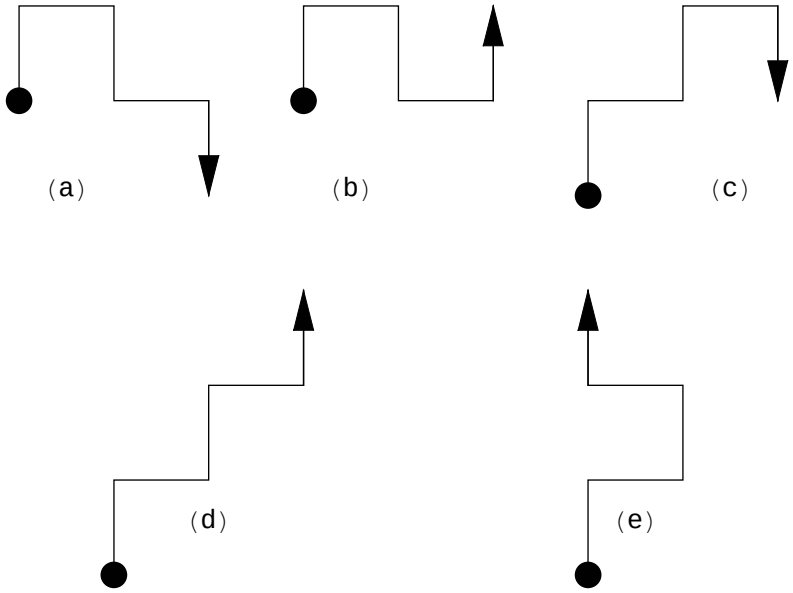


Figure 4.4: The only 5 independent possible walks of *N*=5 polymer on a square lattice with 90° bond angles.

Slithering Snake or Reptation Algorithm

1] Choose a chain at random and remove the tail link.

2] Attempt to add a link to the head of the chain. (There is a maximum of two directions in which the new head link can be added).

3] If the attempt violates the self-intersection constraint, return to the original chain and interchange the head and tail. Include the chain in the statistical sample.

4] Repeat many times and determine the mean-square, end-to-end distance.

Note that from Figure 4.4a we obtain *on average* 0.5c and 0.5d, it can easily be verified that in the end all five chains are equally probable.

There are 2 fundamental drawbacks to using this algorithm. One problem is that it is possible for a SAW to find itself in a "double cul-de-sac" shape from which it cannot extricate itself:

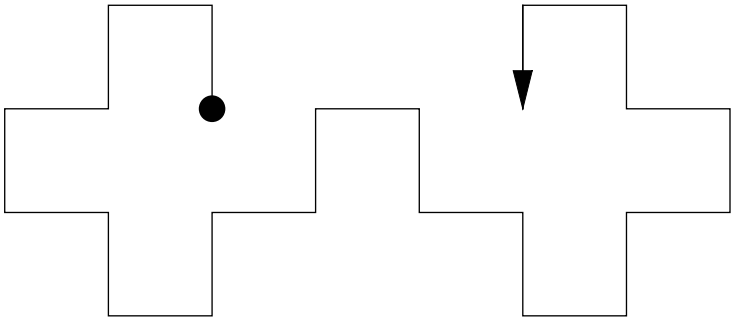


Figure 4.5: Non ergodic Cul-de-Sac SAW.

Because a SAW can become trapped in a configuration from which it can not escape, this SAW method is "non-ergodic" (a process is said to be "ergodic" if any sequence or significant sample is equally representative of the whole). Hence we see that the method introduces a small bias to our statistical sample and the calculated mean end-to-end distance will be slightly larger than if all configurations were considered. The critical exponent found in the slithering snake algorithm however has found to be correct.

Another problem with the slithering snake algorithm is that, because the walk moves one segment at a time, the SAW rearranges itself rather slowly, and very large number of steps must be performed in the simulation to obtain large-scale shape changes. Both these shortcomings can be addressed in the so-called "Pivot algorithm".

The pivot algorithm is a very efficient "dynamic" algorithm for generating *d*-dimensional SAW's in a canonical ensemble (i.e., with a fixed number of steps). It is based on randomly selecting one of the (2^d) $d!$ symmetry (rotation or reflection) operations on a *d*-dimensional lattice and applying the operation to the section of the SAW subsequent to a randomly selected step location, called the pivot. In two dimensions, it is sufficient (for ensuring ergodicity) to consider just three of the eight symmetry operations, specifically rotations of +90, -90 and 180 degrees. To perform these rotations, three two-dimensional rotation matrices, rot90, rot270 and rot180 are defined.

The use of these matrices can be illustrated by rotating the unit vector pointing in the east direction about the origin, {0, 0}. Rotation is accomplished by matrix multiplication. The unit vector in the east direction, is rotated to point in the north, south and west direction by rot90, rot270 and rot180, respectively.

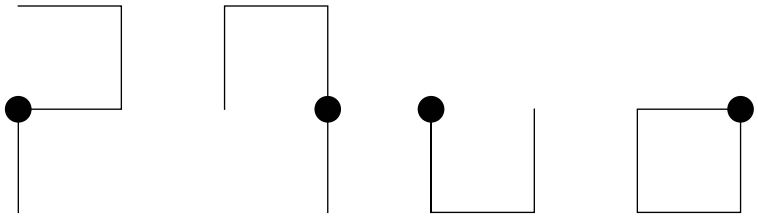


Figure 4.6: SAW and the results after applying the Pivot algorithm for 90, -90 and 180 degrees about the second step location in the chain.

The first picture is the starting chain SAW configuration and the 2nd, 3rd and 4th pictures are the SAW configurations that result from rotations of 90, -90 and 180 degrees about the second step

location in chain. The pictures confirm that the pivot operation has been done correctly.

Building on the example above, we can write down the algorithm for performing the pivoting operation on any SAW:

The Pivoting Self Avoiding Walk in 2D

1a] Create an n-step SAW on a two-dimensional square lattice.

1b] Calculate the square end-to-end distance of the SAW and call it squaredist.

The following sequence of steps 2-5 will be executed a number of times (this is described in step 6), first using the initial SAW configuration given in step 1, and then using the value of the SAW configuration resulting from the previous run-through of the sequence . We will describe the steps in terms of an arbitrary SAW configuration, which we'll call config.

2] Choose at random, a pivot point k (0<k<n) along config and divide config into two parts: one part, consisting of the steps in config prior to and including k, is called fixsec, and the other part, consisting of the steps in config subsequent to k, is called movesec.

3] Choose at random, a symmetry operation of the lattice.

4] Apply the operation to movesec to obtain the rotated chain section, and call it newsec.

5] Check if any of the step locations in newsec and fixsec coincide.

If they do not coincide, create a new SAW configuration, naming it newconfig, by joining newsec and fixsec, calculate its square end-to-end distance and add that to squaredist, and return newconfig.

If they do coincide, calculate the square end-to-end distance of the previous SAW configuration, config, add that to squaredist,and return config.

6] Execute the sequence of steps 2-5, m times, starting with the initial SAW.

7] Determine the mean-square end-to-end distance.

As expected, a single pivot can produce a much bigger change in the SAW shape than can a single "slither", but for a large number of rearrangements, the two algorithms give, on average, the same results.

■ 4.3.1 Restricted random walks and simple lattice gasses

As shown in the previous paragraph, restrictions (and boundaries) do have a serious impact on the random walks.

Consider a one-dimensional lattice which has "trap" sites at x=0 and at x=a (a>0). A RW begins at site x_0 (0<x_0<a) and steps to nearest neighbour sites with equal probability. If we perform a Monte Carlo simulation we find that the mean "first passage time τ" for the particle to be trapped is given by:

$$\langle \tau \rangle = \frac{x_0(a - x_0)}{2D}, \tag{4.34}$$

with D the self-diffusion constant in the absence of the traps and the average is over all possible walks. Such random walk models have played an important role in material sciences. For example, consider the following idealised model of energy transport in solids. The solid is represented as a lattice with two types of sites: Hosts and Traps. An incident photon is absorbed at the host site and excites a molecule. The excitation energy or "exciton" is transferred at random to one of its neighbours and the original excited molecule returns to its ground state. The exciton wanders through the lattice until it reaches a trap site. The exciton is then trapped and a physical process such as a chemical reaction occurs.

One version of this energy transport model is a one-dimensional lattice with traps placed on a periodic sub lattice. Since the traps are placed at regular intervals, we can replace the RW on the infinite lattice by a random walk on a ring:

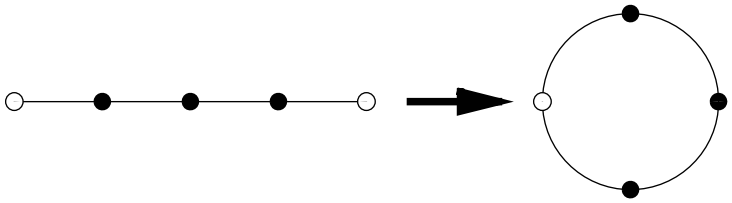


Figure 4.7: Equivalence of a regular lattice of traps with periodicity 4 and a ring of N=3 host sites with one trap site. The trap sites are denoted as open circles.

Another example is a crystalline solid, where the imperfections are a variety of defects. The simplest imperfection is a lattice vacancy, e.g. the absence of an atom from a lattice site and the placement of an additional atom on the surface. At finite temperature, a certain number of lattice vacancies are always present in the otherwise perfect crystal. In many cases the vacancy diffuses by exchanging places at random with neighbouring atoms. Assuming that at time t=0 a vacancy is at the centre of a circle of radius r, we can use Monte Carlo simulations to determine the mean time for a vacancy to reach the surface of the metal at a distance r away, and determine the probability distribution for the "first passage time".

Although the above problems involved random walkers on a lattice, it was not necessary to store the

positions of the lattice sites or the path of the walker. In the following example, we consider a random walk model which requires us to store the lattice positions of a "gas" of random walkers.

Consider a non zero concentration *c* of random walkers (particles) on a square lattice. Each particle moves at random to empty nearest-neighbour sites but double occupancy of sites is excluded; otherwise the particles are non interacting. Such a model is an example of a *Lattice Gas* (we discuss this concept of lattice gasses in more detail in Appendix D). Note that the motion of an individual particle is correlated with the motion of the other particles. The physical motivation of this model arises from metal physics where diffusion is caused by thermal vacancies whose concentration depends on the temperature. The main physical quantity of interest is the self-diffusion constant *D* of an individual (tagged or tracer) particle. The algorithm for a MC simulation of *D* can be described as follows:

```
Monte Carlo Simulation of the self-diffusion constant D

1] Occupy the lattice sites at random with a concentration of c
particles (0<c≤1). Tag each particle and record its initial
position.

2] At each time step choose a particle at random and choose one of
its neighbouring sites. If this site is empty, the particle is
moved to this site; otherwise the particle remains in its current
position.
```

The measure of time in this context is arbitrary. The usual definition which we will use often is that one unit of "time" corresponds to "one Monte Carlo step per particle" each particle attempts one jump on the average. The diffusion constant *D* is obtained as the limit $t \to \infty$ of $D(t)$, where $D(t)$ is given by:

$$D(t) = \frac{1}{2\,d\,t} \left\langle \Delta R(t)^{2} \right\rangle \tag{4.35}$$

and $\langle \Delta R(t)^{2} \rangle$ is the net mean-square displacement per tagged particle after t units of time.

5 Simulation of Percolation

5.1 Introduction

In this chapter we address the notion of geometrical phase transitions by studying percolation methods. Percolation is concerned with connectivity. P.G. de Gennes, winner of the 1991 Nobel Prize in Physics for his profound work on the theoretical physics of disordered materials, has described the percolation transition in the following way: "many phenomena are made of random islands and in certain conditions, among these islands, one macroscopic continent emerges."

Historically percolation theory goes back to Flory and to Stockmayer who used it during World War II to describe how small branching molecules form larger and larger macromolecules if more and more chemical bounds are formed between the original molecules. Percolation phenomena are widespread in nature. One of the most common places to observe percolation is in the kitchen when jello is prepared or when milk curdles. This sol-gel transition, as it is called, occurs in chemical systems, (a polymerisation reaction and the vulcanisation of rubber), biological systems (the immunological antibody-antigen reaction and the coagulation of blood), and in physical systems (in critical phenomena). A nice example is due to Stauffer [20] where the dynamics of forest fires are discussed. We will look more generally at the clustering that occurs in the random site percolation model.

5.2 (Random) Site Percolation

The random site percolation model consists of an m by m random Boolean lattice. This is a lattice whose sites have values 0 and 1, where 0 represents an empty site and 1 represents an occupied site. The probability p, of a site being occupied is independent of its neighbours. A cluster is defined as a group of occupied nearest-neighbour (nn) sites (nn sites are those above, below, left or right of a site). Two occupied sites belong to the same cluster if they are linked by a path of nearest-neighbour connections joining occupied sites. Spanning clusters "span" the lattice from one side to the other, this typically occurs at values $p \sim 1$. In the limit of an infinite lattice there exists a well defined critical probability p_c such that below that value no spanning cluster exists and just above p_c a spanning cluster emerges. This implies a qualitative change in connectedness from one state to the other $\longrightarrow$ a phase transition.

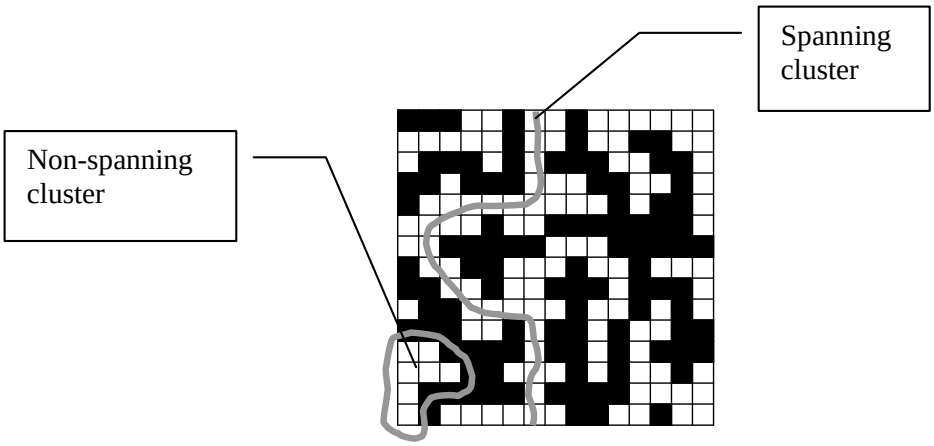


Figure 5.1: Example of spanning cluster.

Examples of percolation phenomena are the electrical conductivity of composite systems made of a mixture of metallic and insulating materials, the spread of disease in population, the behaviour of magnets diluted by non magnetic impurities, and the characterisation of gels.

The standard description of the Site Percolation program

1] Each site on a two-dimensional m by m square lattice is assigned a value of a=Random[].

2] If a≤p, its value is changed to 1; otherwise its value is changed to 0.

Once the clusters are identified (visually or numerically), there are a number of cluster-related quantities that are interesting to look at, such as the spatial characteristics of clusters (e.g., their Fractal dimensions) as a function of p and the percolation threshold, which is the value of p at which a spanning cluster (an uninterrupted path across the lattice) first appears.

Typical parameters characterising site percolation are:

1] The mean cluster-size distribution (the size here denotes the number of sites in the cluster, rather than its spatial extent)

$$n_s(p) = \frac{\text{Average number of clusters of size } s}{\text{Total number of sites in the lattice}}. \tag{5.1}$$

2] The probability that an occupied site chosen at random is part of an s-site cluster:

$$w_s = \frac{s\, n_s}{\sum_s s\, n_s}. \tag{5.2}$$

3] The mean cluster size S:

$$S = \sum_s s\, w_s = \frac{\sum_s s^2\, n_s}{\sum_s s\, n_s}. \tag{5.3}$$

4] The probability $P_\infty(p)$ that an occupied site belongs to the spanning cluster:

$$P_\infty = \frac{\text{Number of sites in the spanning cluster}}{\text{Total number of occupied sites}}. \quad (5.4)$$

For an infinite lattice, $P_\infty(p)=0$ for $p<p_c$ and $P_\infty(p)=1$ for $p=1$.

■ 5.2.1 Cluster labelling

We will discuss a method to perform automatically cluster labelling: the determination of the existence of a connected path and counting the number of clusters (see Figure 5.2). The best known method is the famous Hoshen-Kopelman algorithm [9] (named after its creators), which involves scanning the lattice just once. The difficulty is that assignment of a site to a cluster is a *global* rather than a *local* property of the site.

```
The Hoshen-Kopelman Algorithm

We'll be labelling a nested list r (the lattice), consisting of m
lists, each containing m elements which are either zero's or one's.

1] A list u, is created by adding: (a) a zero to the front of each
of the lists in r and (b) a list of (m + 1) zero's to the front of
r (in a matrix representation of r, this operation takes the m×m
matrix and forms an (m + 1)×(m + 1) matrix by adding a top row of
zero's and a left column of zero's).

2] Two lists ul and ulp, are created with the same size as u and
consisting of all zero's.

3] The list u is scanned in a "typewriter" fashion, starting from
position {2, 2} and proceeding along each row in succession, going
from the second element to the last element in the row. The
elements in ul and ulp are changed during the scan of u according
to the criteria that follow:

Note: In the following step, we will refer to the elements in u and
ul as sites. A site with a value of zero will be said to be empty
and a site with a non-zero value will be said to be occupied.
Additionally, we will refer to the site u[[i, j-1]], lying to the
left of u[[i, j]], as uback, the site u[[i-1, j]], lying above
u[[i, j]], as uup, the site ul[[i, j-1]], lying to the left of
ul[[i, j]], as ulback, and the site ul[[i-1, j]], lying above
ul[[i, j]], as ulup.

    3a] At each site, u[[i, j]], look to see if it is occupied or
    empty and do the following:
```

3a1] If u[[i, j]] is empty, go on to the next site.

3a2] If u[[i, j]] is occupied, look at the nearest neighbour (nn) site in the previous column, uback, and the nn site in the previous row, uup, and do the following:

3a2a] If both uback and uup are empty, set ul[[i,j]] equal to one more than the current maximum value in ul and then add this new maximum value in ul to ulp.

3a2b] If only one of the uback and uup sites is occupied, set ul[[i, j]] equal to the non-zero value of ulup or ulback.

3a2c] If both uback and uup are occupied, set ul[[i, j]] equal to the smaller of ulp[[ulup]] and ulp[[ulback]] and set the value of the position in ulp having the larger of the values of ulp[[ulup]] and ulp[[ulback]] equal to the smaller of these values.

4] After the scan in step 3 is completed, relabel the occupied sites in ul so that connected sites (i.e., occupied sites adjacent to occupied sites) have the same number and change the cluster numbers in ul so that they run sequentially, without any gaps.

1	1		2	2
	1			2
			2	2
	3	3		
4				5

Figure 5.2: A labelled percolation configuration of a square lattice with L=5.

5.3 Critical Exponents and Finite-Size scaling

■ 5.3.1 Introduction

Most substances exhibit a thermodynamical phase transition with respect to temperature and pressure (e.g. from ice to water to vapour etc.) and a critical point beyond which it is no longer possible to distinguish between different phases.

Consider for example a forest fire modelled by percolation. We are interested in the lifetime of the fire (in terms of the number of Monte Carlo sweeps) with respect to the probability p that a tree is expected at a site of the lattice. In Figure 5.3 we have shown the behaviour of the lifetime in the vicinity of critical probabilities p.

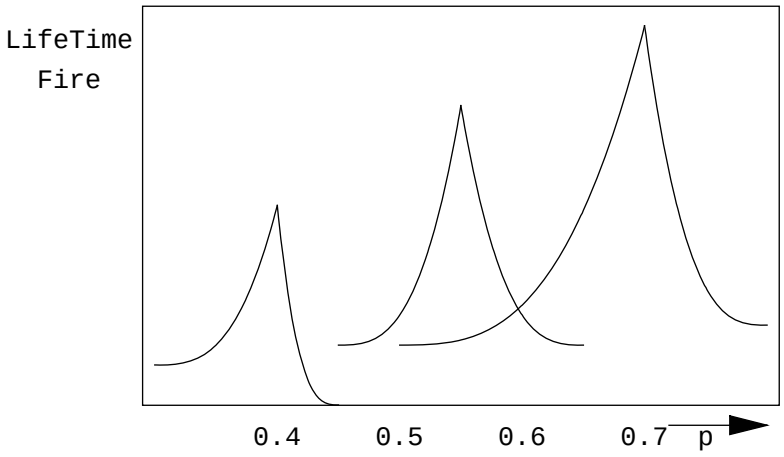


Figure 5.3: Average termination time for forest fires, as simulated on a square lattice. The centre curve corresponds to the simplest case described in the text. The left-hand curve gives data if the fire can spread to both nearest and next-nearest neighbours. For the right-hand curve two burning trees are needed to ignite a nearest or next-nearest neighbour (adapted from [20]).

Note that if we increase p to the percolation threshold p_c (the critical point) then the lifetime seems to grow exponentially, whereas beyond p_c the lifetime decreases again. We can interpret this behaviour by noting that beyond p_c spanning clusters occur (e.g. from the bottom of the lattice to the top) and the fire dies out. Just before p_c however the fire needs a long time to find out that it cannot penetrate the forest, and thus only after many sweeps through the lattice the fire will be extinguished. Therefore the lifetime will become very large if p approaches p_c from below or above, this behaviour is also known as "critical slowing down".

Another familiar but less known example of a critical point occurs in magnetic systems at the so-called "Curie" (or critical) temperature T_c. At low temperatures some substances exhibit ferromagnetism, a spontaneous magnetisation in the absence of an external magnetic field. If we would raise the temperature of such a ferromagnet the spontaneous magnetisation decreases and vanishes continuously at a "critical" temperature T_c. For $T > T_c$ the system is a paramagnet. In Chapter 6 we will investigate this behaviour. Since the understanding of thermodynamic phase transitions requires a strong background in statistical physics, it is of interest to study the percolation phase transition. Although percolation is an unusual phase transition since temperature is not involved, we find that properties of the geometrical transitions in the percolation problem can serve as a simple introduction to thermodynamic phase transitions.

A major conclusion will be that in the vicinity of a phase transition, the qualitative behaviour of the system is governed by the appearance of the long-range correlation. For instance we know that the essential physics near the percolation threshold is associated with the existence of large but finite clusters. For example for $p<p_c$ we know that n_s decays exponentially with s; for $p>p_c$ n_s will decrease more rapidly with s. However for $p=p_c$, the s-dependence of n_s is qualitatively different and n_s decreases much more slowly. This different behaviour of n_s at p_c is due to the presence of all length scales (large and small). A more direct way of observing the effects of the length of clusters is to introduce a characteristic linear dimension or "mean connectedness length" $\xi(p)$. Next, we will explore $\xi(p)$ in more detail.

■ 5.3.2 Parameters in the vicinity of a phase transition

One operational definition of ξ is to identify it with the "radius of gyration" R_s of the largest non-spanning cluster (see Chapter 4.2). The radius of gyration of a single cluster of s particles is given by

$$R_s^2 = \frac{1}{s}\sum_{i=1}^{s}(r_i - \bar{r})^2, \text{ where}$$
$$\bar{r} = \frac{1}{s}\sum_{i=1}^{s} r_i. \qquad (5.5)$$

Here r_i is the position of the ith site in the cluster and $\bar{r}$ is the centre-of-mass of the cluster. For large lattices we find that $\xi(p)$ is an increasing function of p for $p<p_c$, and a decreasing function of p for $p>p_c$:

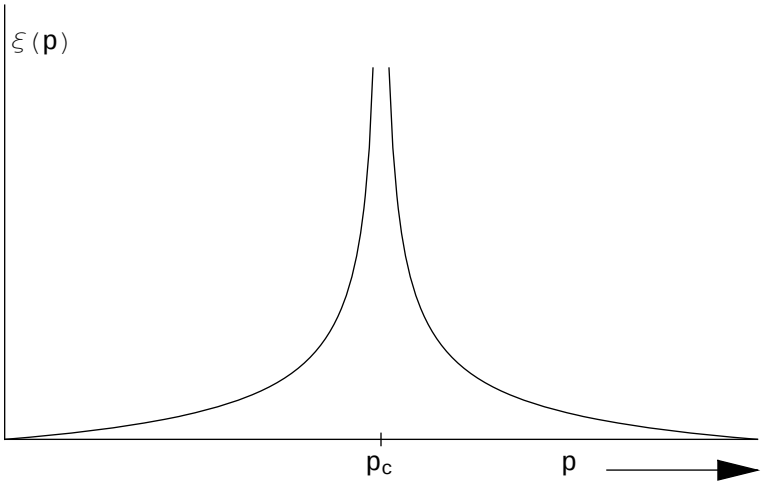


Figure 5.4: Qualitative behaviour of the mean connectedness length ξ.

Furthermore $\xi(p=p_c)$ is approximately equal to the lattice size L and diverges as $L\to\infty$, since as p approaches p_c the probability that two sites are in the same cluster increases. Therefore we expect that for $L\to\infty$, $\xi(p)$ diverges in the critical region, $|p\text{-}p_c| \ll 1$. In Chapter 4.2 we introduced the critical exponent ν, experiments show the following relation hold in the limit $L\to\infty$:

$$\xi(p) \approx |\, p - p_c \,|^{-\nu}$$

and for P_∞(the order parameter)

$$P_\infty \approx (p - p_c)^{\beta} \qquad (5.6)$$

and the mean clustersize

$$S(p) \approx |\, p - p_c \,|^{-\gamma} .$$

Table 5.1 summarises the critical exponents for the percolation and magnetism phase transitions:

Quantity	*Functional form*	*Exponent*	$d = 2$	$d = 3$
Percolation				
Order parameter	$P_\infty \approx (p - p_c)^{\beta}$	β	5/36	0.4
Mean size of finite clusters	$S(p) \approx \|\, p - p_c \,\|^{-\gamma}$	γ	43/18	1.8
Correlation length	$\xi(p) \approx \|\, p - p_c \,\|^{-\nu}$	ν	4/3	0.9
Cluster numbers	$n_s \approx s^{-\tau}$	τ	187/91	2.2
Magnetism				
Order parameter	$M(T) \approx (T_c - T)^{\beta}$	β	1/8	0.32
Susceptibility	$\chi(T) \approx \|\, T - T_c \,\|^{\gamma}$	γ	7/4	1.24
Correlation length	$\Xi(T) \approx \|\, T - T_c \,\|^{-\nu}$	ν	1	0.63

Table 5.1: Critical exponents for percolation and magnetism phase transitions in d=2 and 3 dimensions [6]. Note the scaling law: $2\beta + \gamma \sim \nu\, d$.

Since we can only simulate *finite* lattices a direct fit of the measured quantities of Table 5.1 to Equations (5.6) will not yield the correct estimates. The reason is that finite size effects don't allow a

close approach of p to p_c. (e.g. since $\xi(p)$ is comparable to L for p close to p_c) For $p \ll p_c$ or $p \gg p_c$ the physical quantities are not affected by the finite size of L. We can however by means of *finite size scaling* study the critical exponents.

Notice that after inversion of $\xi(p)$ in Equation (5.6) and substitution of ξ by L (for $p \sim p_c$) results in:

$$\begin{aligned} &\left| p - p_c \right| (\approx L)^{-1/\nu} \\ &P_\infty(p = p_c) \approx L^{-\beta/\nu} \quad \text{for } L \to \infty \end{aligned} \tag{5.7}$$

With this relation we can determine the critical exponents: Suppose that we generate percolation configurations at $p = p_c$ for different values of L and analyse P_∞ as a function of L. Then if L is sufficiently large, we can use the asymptotic relation (5.7) to estimate the ratio β/ν. Similar analyses can be done for the other parameters of Table 5.1.

More generally speaking systems with correlation functions that decay as power laws are said to be scale invariant. This terminology reflects the familiar fact that power laws look the same on all scales. For example, the replacement of $x \to ax$ in the function $f(x) = Ax^{-\eta}$ yields, for any value of the exponent η a function $g(x)$ that is indistinguishable from f(x), except in the amplitude A by the factor $a^{-\eta}$. This invariance does not hold for functions that decay exponentially, since making the same replacement in the function $e^{-x/\xi}$ changes the correlation length ξ (the characteristic scale for decay) by a factor a. An enormous variety of systems in physics, chemistry, and biology seem to exhibit scale invariance in some form or another. The origin of the ubiquitous occurrence of scale invariance in systems with short-ranged interactions remains largely mysterious. This is in part because the systems that have been studied are those in thermodynamic equilibrium. Straightforward arguments show that in the absence of special symmetries or long-range interactions, scale invariance in equilibrium systems can occur *only* at isolated critical points or surfaces in parameter space; for more generic choices of parameters exponential behaviour will be obtained. Therefore one might expect that an explanation for the widespread occurrence of scale invariance is due to systems being driven externally into *non equilibrium*. This is however outside the scope of this course, the interested reader is referred to [7] and [8] and references therein.

5.4 Renormalisation

An important step in the determination of critical exponents, was made by the development of the *renormalisation group* method, for which Kenneth Wilson in 1981 was awarded with a Nobel prize in physics. This method allows one to extend the examination of physical quantities near a critical point on different length scales beyond finite size scaling and will directly yield the critical exponents. Although it was initially used to study thermodynamic phase transitions, it will be illustrated here by the percolation transition.

Consider a lattice where a cell is occupied with a probability $p=p_0<p_c$ and imagine to look at it from an increasing distance. By moving further away from it one is not able to distinguish independent sites. Details such as single site clusters and narrow bridges connecting large "blobs" will become invisible. As a consequence the connectedness ξ will decrease and the less detailed picture would resemble one generated with a probability $p_1<p_0$ and have a connectedness $\xi(p_1)<\xi(p_0)$. Continuation of considering the system at larger distance will eventually lead to the limiting case of a trivial *fixed point* p=0.

A similar type of reasoning for the case $p=p_0>p_c$, shows that at larger distances the unoccupied sites will be absorbed in the clusters, and the overall behaviour would look the same as if it were generated at a probability $p_1>p_0$. In the limiting case this would lead to the trivial *fixed point* p=1. Only at $p=p_c$ the lattice would look the same at each distance, therefore p_c can be considered to be a *nontrivial* fixed point.

In a renormalisation theory one tries to capture the above described effect in a transformation, which tells us how the quantity of interest, in this case percolation, behaves if it is considered on a larger length scale. This is achieved by introducing supersites, which describe the effective behaviour of a group of unit sites. In the case of a square lattice, a natural supersite is defined by 2×2 unit sites, although also larger, preferably square-like groups can be used.

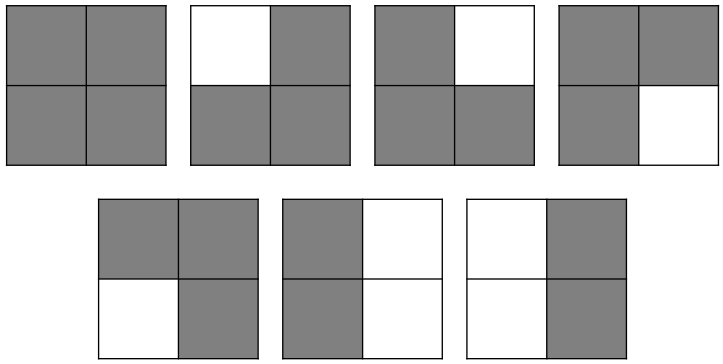


Figure 5.5: The seven vertically spanning configurations of a 2×2 cell

For simplicity we will consider here vertical spanning criterion. In this case there are seven spanning 2×2 configurations, which are shown in Figure 5.5. In order to preserve the main features during a renormalisation step, we will replace all spanning 2×2 cells by an occupied site and the non-spanning cells by an empty site. In Figure 5.6 four successive steps in the renormalisation process are shown for a 16×16 lattice with p=0.7

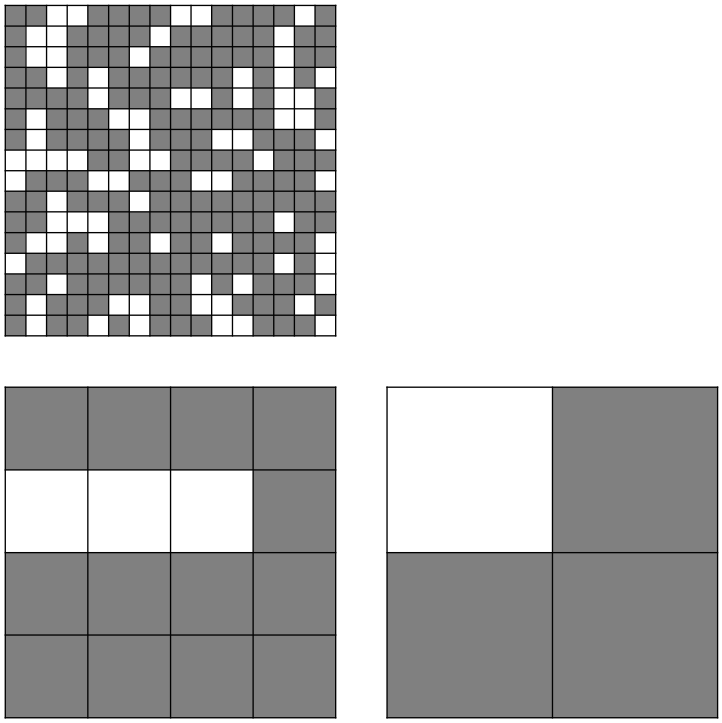


Figure 5.6: A percolating configuration generated at p=0.7 and three successive steps of the renormalisation procedure.

Note that using this procedure allows one to estimate $p_c$but is not capable yet of providing critical exponents. The renormalisation group method we need to estimate the critical exponents consists of two parts:

1) an average over the underlying variables together with a specification of the variables that determine the state of the renormalised configuration.

2) a parametrisation of the renormalised configuration in terms of the original parameters and possibly others.

Using the same average as before and replace the b^2 sites on the original lattice by a single cell with linear dimension b, which represents whether the original lattice sites were spanning or not. We assume that the new cells are independent and are occupied with a probability p', which is some function $R(p)$ of the original probability p. In the case of b=2 we find (see Figure 5.5) that the *renormalisation transformation* $R(p)$ is given by

$$p' = R(p) = p^4 + 4\,p^3(1-p) + 2\,p^2(1-p)^2. \tag{5.8}$$

The fixed points of this transformation are found by solving

$$p^* = R(p^*), \tag{5.9}$$

which leads to the trivial fixed points p^*= 0 and p^*= 1, and a non-trivial fixed point p^*= 0.61804, which can be associated with p_c. This calculated value of p^* for b=2 should be compared with the estimate p_c=0.5927.

The connectedness length ξ transforms as

$$\xi' = \xi / b, \tag{5.10}$$

which can be transformed by using (5.6) p$\approx p_c$ to

$$\left| p' - p^* \right|^{-\nu} = \frac{| p - p^* |^{-\nu}}{b}, \tag{5.11}$$

where p_c is identified with p^*. Expanding the renormalisation transformation (5.9) near p^* in terms of (p-p^*) gives

$$p' - p^* = R(p) - R(p^*) \approx \lambda(p - p^*),$$
$$\lambda = \frac{d\,R(p = p^*)}{d\,p.} \tag{5.12}$$

Combining these results give

$$| p' - p^* |^{\nu} = \lambda^{\nu}(p - p^*)^{\nu} = b\,| p - p^* |^{\nu}, \tag{5.13}$$

hence the critical exponent ν can be obtained

$$\nu = \frac{\log b}{\log \lambda} = \frac{\log 2}{\log 1.5279} = 1.635. \tag{5.14}$$

The result of this simple calculation compares not bad with the exact result ν=4/3 (see Table 5.1), the accuracy of the result, however, is not known. Better estimates for both p_c and ν can be obtained by

considering larger cells. However, the number of configurations 2^{b^2} for a $b \times b$ cell increases rapidly and for $b>7$ this exact enumeration is not practical anymore, but could be combined with for instance a Monte Carlo approach.

The main assumption in this renormalisation approach is that the occupancy of each cell is independent of the other cells. Although this is correct for the initial lattice, it is not true anymore after a renormalisation step, because some information will be lost in the process. Some connections can be lost, while new connections can be formed. An example of the later is shown in Figure 5.7, where an interface separates the left and right side and disappears after renormalisation.

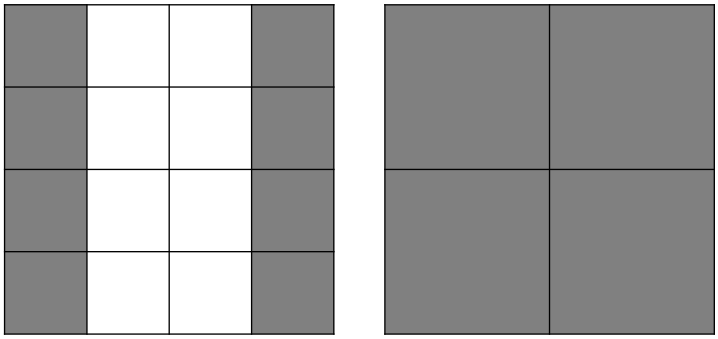


Figure 5.7: Example of an interface, which after renormalisation is connected.

6 Modelling Accretion Processes

Main reference:

Computer Simulations with Mathematica: Explorations in Complex Physical and Biological Systems. Richard J. Gaylord and Paul R. Wellin. Copyright 1994 TELOS/Springer-Verlag

General references:

Leonard Sander. Fractal Growth Phenomena. *Scientific American*, **256**(1), 94-100, 1987.

Brian Hayes. Nature's Algorithm. *American Scientist*, **82**(May-June), 206-210, 1994.

6.1 Introduction

Accretion is a process in which particles undergo motion until they encounter another particle to "stick to". As particles coalesce, large, irregularly shaped clusters are formed. If the clusters are free-floating, the process is known as aggregation, and if the clusters are attached to a surface, the process is known as deposition. We will look at one example of each of these processes:

In diffusion-limited aggregation, a particle undergoes Brownian motion until it makes contact with another particle. The free-floating cluster that is formed is known as a diffusion-limited aggregate or *DLA*. This mechanism underlies a wide variety of natural phenomena, including crystallization, colloidal and polymeric condensation, soot formation, and dielectric breakdown.

In ballistic deposition, a particle starts above a solid substrate or surface and follows a straight-line downward trajectory until it reaches the surface or makes contact with another particle. Deposition has many applications in the area of materials fabrication, such as: thin-film formation, vapor deposition, sputtering, and molecular-beam epitaxy.

6.2 The Diffusion-Limited Aggregation Model

The DLA model is most easily described in physical terms. At any time during DLA growth, the system consists of a **cluster** of particles and a particle executing a random walk. Initially, the cluster contains just a single **seed** particle. The cluster grows via a simple process: A particle starts at a randomly chosen location along the perimeter of a circle centered on the seed, and executes a random walk until the particle is either a certain distance outside the circle, in which case it vanishes, or until the particle is adjacent to the cluster, in which case it joins the cluster. This process is repeated, one particle at a time, until the cluster reaches a given size.

In this chapter we give examples of the principal algorithms and possible implementations in *Mathematica*.

■ 6.2.1 The Algorithm

The model employs a two-dimensional square lattice.

(1) Create a list, called **occupiedSites**, containing the lattice site {0, 0}.

The following sequence of steps 2 through 3 will be executed a number of times (this is described in step 4), first using the value of the initial cluster **occupiedSites**, given in step 1, and then using the value of **occupiedSites** resulting from the previous run-through of the sequence.

(2a) Determine the lattice site nearest to a randomly chosen location along the circumference of a circle whose radius, rad, equals a specified value, s, plus the maximum absolute coordinate value in occupiedSites.

(2b) Starting at the selected lattice site, execute a lattice walk until the step location is either at a distance greater than (rad + s), or on a site that is contiguous (adjacent) to a site in **occupiedSites**. Call the final step location of the walk, **loc**.

(3) Check if **loc** is adjacent to a site in the occupiedSites list and if it is, add loc to occupiedSites.

(4) Execute the sequence of steps 2 through 3 until the length of **occupiedSites** reaches a value *n*.

■ 6.2.2 Implementation (only for those interested in Mathematica implementations).

Step 1. The list containing the seed site is written as

```
occupiedSites = {{0, 0}}
```

Step 2. The radius of the circle which a random walker starts from is given by

```
rad = Max[Abs[occupiedSites]] + s
```

The walk starts from a randomly chosen location on the circle, which is given by

```
(Round[rad {Cos[#1], Sin[#1]}] &)[Random[Real, {0, N[2 π]}]]
```

Each new step of the walk is generated using

```
#1 + {{1, 0}, {0, 1}, {-1, 0}, {0, -1}}[[Random[Integer, {1, 4}]]] &
```

where # represents the current step location of the walk.

Note: The one-dimensional analog to this anonymous function was discussed in Chapter 1.

The random walk is terminated when **True** is returned using

```
Plus @@ #1^2 > (rad + s)^2 || occupiedSites ⋂
    (Function[y, y + #1] /@ {{1, 0}, {0, 1}, {-1, 0}, {0, -1}}) ≠ {} &
```

Note: The first test determines if the step location is at a square distance greater than $(\text{rad} + s)^2$ from the seed site and the second test determines if the step location is adjacent to a site in **occupiedSites**.

The final location of the random walk, which is called **loc**, is determined using the **FixedPoint** function with the functions given above.

```
loc = FixedPoint[
  #1 + {{1, 0}, {0, 1}, {-1, 0}, {0, -1}}[[Random[Integer, {1, 4}]]] &,
  (Round[rad {Cos[#1], Sin[#1]}] &)[Random[Real, {0, N[2 π]}]],
  SameTest → (Plus @@ #2^2 > (rad + s)^2 || occupiedSites ⋂
        (Function[y, y + #2] /@ {{1, 0}, {0, 1}, {-1, 0}, {0, -1}}) ≠ {} &)]
```

Note: The notation **#2** (rather than #) is used with the **SameTest** option in **loc** because in **SameTest**, **#2** refers to the last element in a list.

Step 3. **loc** is checked to see if the walk ended because it was too far from the cluster and if not, it is appended to **occupiedSites**.

```
If[Plus @@ loc^2 ≤ (rad + s)^2, AppendTo[occupiedSites, loc]]
```

Step 4. The repeated application of the sequence of steps 2 through 3 until the length of **occupiedSites** equals *n* is performed with a conditional function.

```
While[Length[occupiedSites] < n, rad = Max[Abs[occupiedSites]] + s; loc =
  FixedPoint[#1 + {{1, 0}, {0, 1}, {-1, 0}, {0, -1}}[[Random[Integer, {1, 4}]]] &,
   (Round[rad {Cos[#1], Sin[#1]}] &)[Random[Real, {0, N[2 π]}]],
   SameTest → (Plus @@ #2^2 > (rad + s)^2 || occupiedSites ⋂
         (Function[y, y + #2] /@ {{1, 0}, {0, 1}, {-1, 0}, {0, -1}}) ≠ {} &)];
 If[Plus @@ loc^2 ≤ (rad + s)^2, AppendTo[occupiedSites, loc]]]
```

We can now assemble these code fragments to create a program.

■ 6.2.3 The Program

```
DLA[s_Integer, n_Integer] := Module[{loc, rad,
   particleCount = 0, stepChoices = {{1, 0}, {0, 1}, {-1, 0}, {0, -1}}},
  occupiedSites = {{0, 0}}; While[Length[occupiedSites] < n,
   ++particleCount; rad = Max[Abs[occupiedSites]] + s;
   loc = FixedPoint[#1 + stepChoices[[Random[Integer, {1, 4}]]] &,
     (Round[rad {Cos[#1], Sin[#1]}] &)[Random[Real, {0, N[2 π]}]],
     SameTest → (Plus @@ #2^2 > (rad + s)^2 ||
         occupiedSites ⋂ (Function[y, y + #2] /@ stepChoices) ≠ {} &)];
   If[Plus @@ loc^2 < rad^2, occupiedSites = Join[occupiedSites, {loc}]]];
  Print["The number of particles released was ", particleCount];
  occupiedSites]
```

Note: We have included a counter, **particleCount**, to keep track of how many particles are released during the growth process and the final tally is printed out.

■ 6.2.4 Visualizing Diffusion-Limited Aggregation

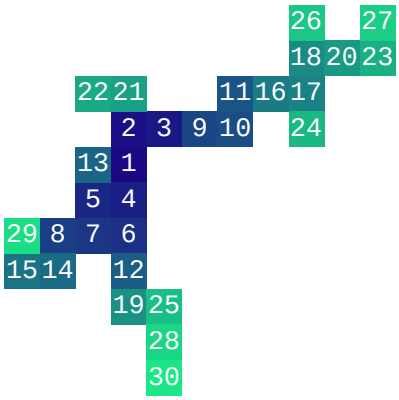
The graphics of the DLA structure can be created using the **Graphics** primitive, **Rectangle**, to represent each particle in the cluster. The history of the accretion process can be seen by numbering and shading the rectangles in the order in which they were added to the cluster, with earlier rectangles being darker in color and lower in number.

```
ShowDLA[sites_, ptsize_: 0.1, opts___Rule] :=
 Module[{len = Length[sites], points, colors},
  points = Point /@ sites; colors = Hue /@ Range[len]/len;
  Show[Graphics[{PointSize[ptsize], Transpose[{colors, points}]},
    Axes → None, AspectRatio → Automatic, PlotRange → All, opts]]]
ShowDLA[sites_, opts___Rule] := ShowDLA[sites, 0.1, opts]
```

Some typical output from running the **dLA** program are shown below.

```
agg = DLA[10, 30];
The number of particles released was  109
```

```
ShowDLA[agg];
```



For large DLAs, it is quicker to render each particle as a single point, as opposed to the above rectangles. Here is a program that accomplishes this and colors each point according to its history.

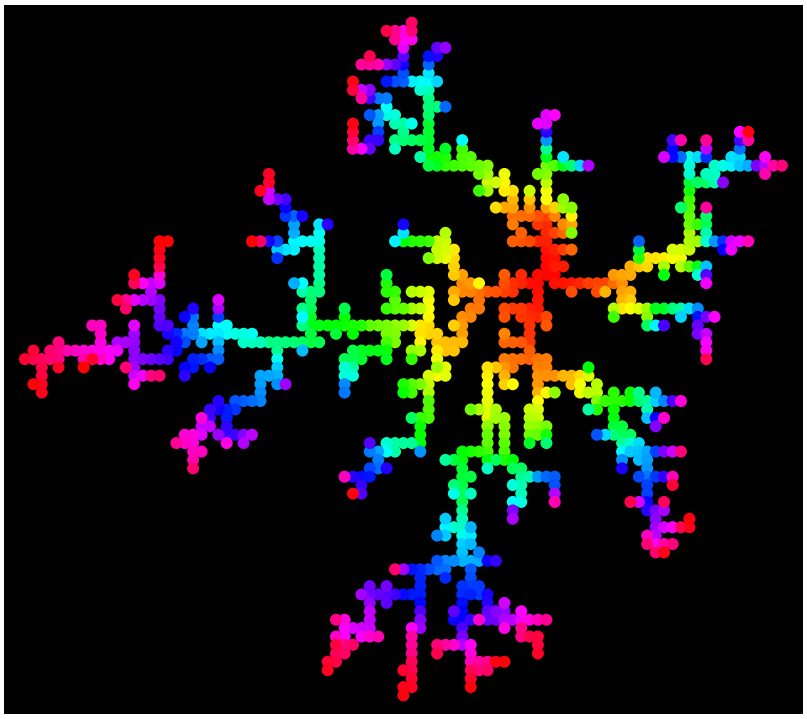
```
ShowDLA2[sites_, opts___Rule] := Module[{len = Length[sites], points, colors},
  points = Point /@ sites; colors = Hue /@ Range[len]/len;
  Show[Graphics[{PointSize[1/Sqrt[len]], Transpose[{colors, points}]},
    opts, Axes → None, AspectRatio → Automatic, PlotRange → All]]]
```

This function allows you to specify any number of additional graphics options to the plot. In addition, it computes a pointsize that scales with the size of the DLA. So, for example, you could display a DLA agg on a black background and connect the points with lines by using ShowDLA2[agg, Epilog->Line[agg], Background->GrayLevel[0]].

Here then is a much larger DLA.

```
agg = DLA[2, 1000];
```

```
ShowDLA2[agg, Background → GrayLevel[0]];
```



6.3 The Fractal Dimension of a DLA

As DLA growth proceeds, the shape of the aggregate or cluster becomes increasingly irregular and tenuous. This occurs as the result of a "screening" effect which increases the likelihood that a meandering particle will contact an exposed exterior portion of the DLA before it penetrates into a more shielded interior portion (this effect can be seen by looking at the locations of sites in the cluster as a function of when they joined the cluster, in the DLA graphics above).

We can get a feel for the compactness of a DLA (i.e., how it fills space) by measuring its fractal dimension. There are a number of fractal dimensions that can be measured (e.g., the dependence of the radius of gyration of the cluster on the size of the cluster) and while the various fractal measures give different quantitative results, they display a universal trend, namely that the fractal dimension of a DLA decreases with increasing size to a limiting value at large sizes. Here we will follow the density of space occupied by the cluster as a function of size.

Note: The overall DLA process is stochastic, and it is therefore necessary take an average of the fractal dimension (or any other quantity) over a number of randomly generated DLA's. However, for illustrative purposes, we will only consider a single DLA .

6.3.1 Computing The Fractal Dimension

The following steps are used to compute the fractal dimension of the DLA:

(1) A fractalDataList of ordered pairs is constructed.

```
fractalDataList = Table[{2 r, occSiteDensity[r]}, {r, Max[occupiedSites]}]
```

where occupiedSites is a list of the sites in the cluster and occSiteDensity is the number of sites in occupiedSites that lie within a square running between -r and r in each direction, divided by the total number of sites in the square.

The value of occSiteDensity is calculated for a given square size, using

```
occSiteDensity[t_Integer] :=
  1/(2 t + 1)^2 (N[Count[occupiedSites, {x_?(Abs[#1] ≤ t &), y_?(Abs[#1] ≤ t &)}]])
```

(2) The fractal dimension of the DLA structure is determined using

```
fractaldim = Fit[N[Log /@ fractalDataList], {1, x}, x]
```

These computations can be put together in a program.

■ 6.3.2 The Fractal Dimension Program

```
FractalDimension[occupiedSites_List] :=
  Module[{occSiteDensity, fractalDataList, fractaldim}, occSiteDensity[t_Integer] :=
    1/(2 t+1)^2 (N[Count[occupiedSites, {x_?(Abs[#1] ≤ t &), y_?(Abs[#1] ≤ t &)}]]);
   fractalDataList = Table[{2 s, occSiteDensity[s]}, {s, Max[occupiedSites]}];
   fractaldim = Fit[N[Log /@ fractalDataList], {1, x}, x];
   Print[The fractal dimension of the DLA is , Coefficient[fractaldim, x]];]
```

The fractal dimension of a small DLA is calculated below.

```
FractalDimension[DLA[3, 12]]
The number of particles released was 49
The fractal dimension of the DLA is -0.0588937
```

6.4 The Ballistic Deposition Model

The ballistic deposition model is easily described in physical terms. The surface is initially smooth, consisting of a single row of particles. A particle is released at a randomly chosen location a certain distance above the surface. It falls straight downwards towards the surface until it is reaches the surface or is adjacent to another particle where it remains.

■ 6.4.1 The Algorithm

The model employs a two-dimensional rectangular lattice.

(1) Create a 2 by n matrix in which the top row consists entirely of 0's and the bottom row consists entirely of 1s.

The following sequence of steps 2 through 3 will be executed a number of times (this is described in step 4)

(2a) Randomly select a site, **depositColumn**, in the first row of the matrix (this value indicates which lattice column the particle will travel down).

(2b) Using the value of **depositColumn**, create the list, **nnColumns**, of the selected matrix column and its nearest-neighbor columns to the right and left .

Note: When the selected column is the rightmost (leftmost) column, the nn column on the right (left) is taken to be the leftmost (rightmost) column (this is known as a periodic boundary condition).

(2c) Find the positions of the first occupied site in each of the three columns in **nnColumns** and determine which one occurs first. Calculate the position, **depositRow**, in the selected column which is adjacent to the position which occurs earliest (this value is the lattice row where the particle stops).

(2d) Place the particle in the lattice in the position given by {**depositRow**, **depositColumn**}.

(3) Determine if the first row of the matrix is all 0s (it will be unless the particle was placed there in the previous step) and if it is not, then add a row of zero's to the top of the matrix.

(4) Execute the sequence of steps 2 through 3 until the number of occupied sites is t.

■ 6.4.2 Implementation

Step 1. The initial lattice is written as

```
init = Transpose[Table[{0, 1}, {n}]]
```

In the following steps 2a-d, the # symbol represents the lattice configuration at a given time step.

Step 2a. The location of the column in the lattice in which the particle falls is given by

```
depositColumn = Random[Integer, {1, n}]
```

Step 2b. The selected column and its two nearest-neighbour columns are determined using

```
nnColumns = Transpose[#1][[
   {depositColumn - 1 /. 0 → n, depositColumn, depositColumn + 1 /. n + 1 → 1}]] &
```

Step 2c. The row position in the lattice where the falling particle stops is determined.

```
depositRow =
 Min[Flatten[Function[y, First[Position[y, 1]]] /@ nnColumns[#1]] - {0, 1, 0}] &
```

Step 2d. The particle is placed into its final resting place in the lattice using

```
ReplacePart[#1, 1, {depositRow[#1], depositColumn}] &
```

We can combine steps 2a-d in an anonymous function, which we'll call newLat

```
newLat = (depositColumn = Random[Integer, {1, n}];
   nnColumns = Transpose[#1][[{depositColumn - 1 /. 0 → n,
       depositColumn, depositColumn + 1 /. n + 1 → 1}]] &; depositRow =
    Min[Flatten[Function[y, First[Position[y, 1]]] /@ nnColumns[#1]] -
       {0, 1, 0}] &; ReplacePart[#1, 1, {depositRow[#1], depositColumn}]) &
```

where # represents the lattice.

Step 3. A row of 0s is added to the top of the lattice resulting from applying newLat to the lattice, unless the topmost row is already empty, using

```
emitLayer = If[#1[[1]] ≠ Table[0, {n}], Prepend[#1, Table[0, {n}]], #1] &
```

where # represents the result of applying newLat to the lattice.

Step 4. The sequence of steps 2-3 is repeated t times, using

```
deposition = Nest[emitLayer[newLat[#1]] &, init, t]
```

The program for ballistic deposition is given by combining the code fragments.

■ 6.4.3 The Program

```
MolecularDeposition[n_, t_] := Module[{}, init = Transpose[Table[{0, 1}, {n}]];
  newLat = (depositColumn = Random[Integer, {1, n}];
     nnColumns = Transpose[#1][[{depositColumn - 1 / .0 → n, depositColumn,
         depositColumn + 1 /. n + 1 → 1}]] &; depositRow = Min[Flatten[
          Function[y, First[Position[y, 1]]] /@ nnColumns[#1]] - {0, 1, 0}] &;
     ReplacePart[#1, 1, {depositRow[#1], depositColumn}]) &;
  emitLayer = If[#1[[1]] ≠ Table[0, {n}], Prepend[#1, Table[0, {n}]], #1] &;
  Nest[emitLayer[newLat[#1]] &, init, t]]
```

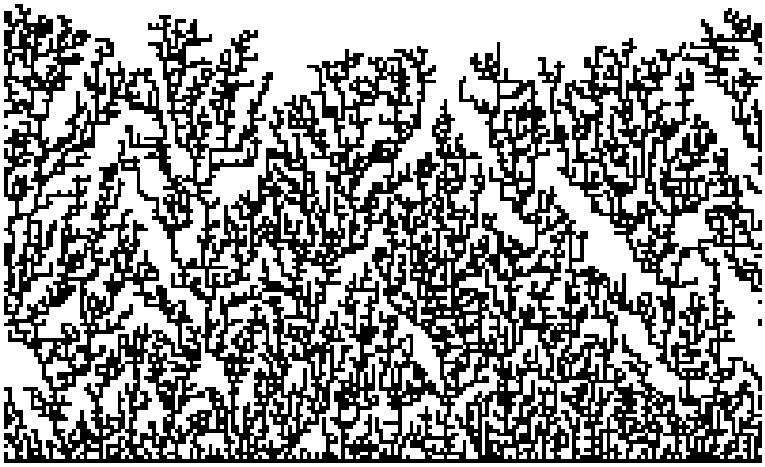
■ 6.4.4 Visualizing Ballistic Deposition

A simple graphics display of the final state of a ballistic deposition process can be created using

```
ShowDeposition[sites_, opts___] :=
 ListDensityPlot[Reverse[sites /. {0 → 1, 1 → 0}], opts,
  AspectRatio → Automatic, FrameTicks → None, Frame → None, Mesh → False]
```

A typical graphical output of running the ballistic deposition program is shown below.

```
ShowDeposition[MolecularDeposition[200, 10 000]];
```



```
ShowDeposition[MolecularDeposition[500, 20 000]];
```

Appendix A: Elements of Probability and Random Variables

In this appendix we refresh your memory on the following concepts: Discrete and continuous probabilistic distributions, expectation, variance, covariance, conditional probabilities, central limit theorem and the strong law of large numbers. Furthermore we will give an introduction in the generation of random variables that are distributed according to discrete and continuous probability distributions.

A.1 Elements of Probability

If we do an experiment of which the outcome is not determined in advance, for example throwing a die or flipping of a coin, we may try to use stochastics to describe the system in question. If we do a simulation experiment where we try to mimic a specific stochastic system we need random variables in order to be able to do so.

■ A.1.1 Distributions: Discrete and Continuous

We speak of discrete random variables if the outcome X of an experiment can take a finite or at most countable number of possible values. For example you may think of flipping one coin. The outcome X of this experiment is restricted to heads or tails. In this case we say that such a system can be described by a *probability mass function* $p(x)$:

$$p(x) = P[X = x], \tag{1.1}$$

which denotes the probability that the outcome of the experiment X is equal to x.

For example in case of the coin flipping experiment:

$$p(\text{heads}) = P[X = \text{heads}] = \frac{1}{2}. \tag{1.2}$$

In general X is a discrete random variable if it can take on only a finite or countable set of possible values $x_1, x_2, \ldots, x_n, \ldots$. Since X must take on one of these values we have:

$$\sum_{i=1}^{\infty} p(x_i) = 1. \tag{1.3}$$

This means that the probability that the outcome of the experiment lies in the set of outcomes $\{x_1, x_2, \ldots, x_n\}$ is equal to unity.

On the other hand if the random variable X can take on a continuous range of values we will have to describe it by means of a *cumulative distribution function*. From this function one can derive the probability that X will lie in a specific continuous range of values. The function

$$F(x) = P[X \leq x],$$

then describes the probability that X takes on a value that is less than or equal to x. Analogous to the probability mass function in the discrete case we define in the continuous case the *probability density function* $f(x)$:

$$f(x) = \frac{d}{dx} F(x). \tag{1.4}$$

This density function has to be normalised to unity. This means that

$$\int_{-\infty}^{\infty} f(x)\, dx = 1, \tag{1.5}$$

which says that the outcome of the experiment X lies in the total range of possible values of x with probability 1.

■ A.1.2 Conditional probabilities

Say we perform an experiment where we flip a coin twice. The sample space of possible outcomes is given by: $S = \{(H, H), (H, T), (T, H), (T, T)\}$ where H denotes heads and T denotes tails. Each of the possible outcomes is equally likely to occur and thus has probability 1/4. We denote the probability that the second coin flip gives heads given the fact that the first flip gives tails by the conditional probability $P(H|T)$. We can easily see that this probability is equal to 1/2. If the first flip has given T only two possible outcomes are left (T, H) and (T, T) which have equal probability.

In general we denote the probability that an outcome (e.g. first tails, then heads) of an experiment is in some set of possible outcomes A under the condition that the outcome is also in another set of outcomes B (e.g. first tails) by the following:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}. \tag{1.6}$$

For example in the case of the coin flipping experiment:

$$P(H|T) = \frac{P(\{T, H\} \cap \{T, *\})}{P(\{T, *\})} = \frac{1/4}{1/2} = \frac{1}{2}, \tag{1.7}$$

where * denotes the wild-card. If $P(A|B) = P(A)$ we say that $P(A)$ and $P(B)$ are independent.

■ A.1.3 Expectation: Discrete and Continuous

The expected value or *expectation* of X, also called the mean of X is denoted by $E[X]$ or $\langle X \rangle$. For a discrete random variable it is defined by:

$$\langle X \rangle = \sum_i x_i \, p(x_i). \tag{1.8}$$

As an example you may think of the coin flipping experiment. Say that we denote heads by 0 and tails by 1. Both events occur with probability 1/2. Therefore the expectation value of this experiment:

$$\langle X \rangle = 0 \underbrace{P[X = 0]}_{1/2} + 1 \underbrace{P[X = 1]}_{1/2} = \frac{1}{2}. \tag{1.9}$$

Analogously for a continuous random variable we can calculate the expectation using the probability density function for X.

$$\langle X \rangle = \int_{-\infty}^{\infty} x \, f(x) \, d\, x \, . \tag{1.10}$$

As an example we take the expectation of a random variable that behaves according to the probability density function

$$f(x) = \begin{cases} e^{-x} & x > 0 \\ 0 & x < 0 \end{cases}, \tag{1.11}$$

then the expectation value of X is given by:

$$\langle X \rangle = \int_0^{\infty} x \, e^{-x} \, d\, x = \underbrace{\left(\left[-x \, e^{-x} \right] \right)_0^{\infty}}_{0} - \int_0^{\infty} e^{-x} \, d\, x = - \left(\left[e^{-x} \right] \right)_0^{\infty} = 1. \tag{1.12}$$

If we are interested in the random variable $g(X)$, where g is some given function we simply write

$$\langle g(X) \rangle = \sum_i g(x_i) \, p(x_i) \tag{1.13}$$

for the discrete case and

$$\langle g(X) \rangle = \int_{-\infty}^{\infty} g(x) \, f(x) \, d\, x \tag{1.14}$$

for the continuous case.

The expectation is a linear operation in the sense that for any two random variables X_1 and X_2:

$$\langle X_1 + X_2 \rangle = \langle X_1 \rangle + \langle X_2 \rangle, \tag{1.15}$$

which generalises to:

$$\left\langle \sum_i X_i \right\rangle = \sum_i \langle X_i \rangle. \tag{1.16}$$

Equation (A.16) holds for the continuous as well as the discrete case.

A.1.4 Variance and Covariance: Discrete and Continuous

The expectation value $\langle X \rangle$ of the random variable X, is a weighted average of the possible values of X. However it doesn't yield any information about the variation of these values. One way of measuring this variation is to consider the average value of the square of the difference between X and $\langle X \rangle$.

If X is a random variable with mean $\langle X \rangle$, then the variance of X, denoted by Var(X) is defined by:

$$\mathrm{Var}(X) = \left\langle (X - \langle X \rangle)^2 \right\rangle. \tag{1.17}$$

It can simply be derived that :

$$\mathrm{Var}(X) = \left\langle X^2 \right\rangle - \left\langle X \right\rangle^2. \tag{1.18}$$

The covariance of two random variables X and Y, denoted Cov(X, Y) is defined by:

$$\mathrm{Cov}(X, Y) = \left\langle (X - \langle X \rangle)(Y - \langle Y \rangle) \right\rangle. \tag{1.19}$$

It can be derived that

$$\mathrm{Cov}(X, Y) = \left\langle X\, Y \right\rangle - \left\langle X \right\rangle \left\langle Y \right\rangle \tag{1.20}$$

The covariance gives an idea of the correlation between two variables X and Y. What can be understood by "correlation" between two variables can be see as follows. For example in the coin flipping experiment. Assume you flip n times the same coin. Set X to be the stochastic variable that describes the number of heads that will come out. Simultaneously let Y be the variable that describes the number of times that tails will come out. It is clear that the outcomes of the simultaneous experiments X and Y depend on each other. Let n = 1, heads = 0 and tails = 1. Then $\langle X \rangle = \langle Y \rangle = 1/2$ and $\langle XY \rangle = 0$, thus Cov(X,Y)=-1/4. This is an example of anti-correlation. In other words: if X has a high value consequently Y must have a low value. If X and Y are independent Cov(X,Y)=0.

Given formulas (A.18) and (A.20) we can derive furthermore that

$$\mathrm{Var}(X + Y) = \mathrm{Var}(X) + \mathrm{Var}(Y) + 2\,\mathrm{Cov}(X, Y). \tag{1.21}$$

If now X and Y are independent random variables then Cov(X,Y)=0, and thus

$$\mathrm{Var}(X + Y) = \mathrm{Var}(X) + \mathrm{Var}(Y). \tag{1.22}$$

The formulas given above hold for the discrete as well as the continuous case. For a more detailed discussion of this section the reader is referred to the book by Ross [18].

■ A.1.5 Some Important Theorems

Strong law of large numbers

The so called strong law of large numbers :

$$\lim_{n \to \infty} \frac{X_1 + X_2 + \ldots + X_n}{n} = \left\langle X \right\rangle. \tag{1.23}$$

This law says that with certainty, the long run average of a sequence of independent and identically distributed random variables will converge to its mean.

Central Limit Theorem

A random variable X is said to be normally distributed with mean μ and variance σ^2 if its probability density function is given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{1.24}$$

If μ=0 and σ^2= 1 then X is said to have a unit normal distribution. We denote the cumulative distribution function of a unit normal random variable by the function ϕ, that is:

$$\phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{y^2}{2}} d\,y \tag{1.25}$$

Now the **Central Limit Theorem** states:

Let X_1, X_2, … be a sequence of independently and identically distributed random variables having mean μ and finite variance σ^2 then

$$\lim_{n\to\infty} P\left[\frac{X_1 + \ldots + X_n - n\,\mu}{\sigma\sqrt{n}} < x\right] = \phi(x). \tag{1.26}$$

So in the limit of an infinite sequence of independently and identically distributed random variables the "random variable"

$$Y = \frac{X_1 + \ldots + X_n - n\,\mu}{\sigma\sqrt{n}}, \tag{1.27}$$

behaves as a variable that is distributed according to the unit normal distribution. The wide applicability of normal random variables results from this theorem. We can summarise this important theorem as follows.

The sum of a large number of independent random variables approximately behaves as a normal distribution.

■ A.1.6 Some frequently encountered discrete probability laws

Probability law and parameter values	probability mass function $p(k)$	Mean $\langle X \rangle$	Variance $\mathrm{Var}(X)$
Binomial $n = 1, 2, \ldots$ $0 \le p \le 1$	$\binom{n}{k} p^k (1-p)^{n-k}$ for $k = 0, 1, \ldots, n$ 0 otherwise	$n\,p$	$n\,p\,(1-p)$
Poisson $\lambda > 0$	$e^{-\lambda}\,\frac{\lambda^k}{k!}$ for $k = 0, 1, \ldots$ 0 otherwise	λ	λ
Geometric $0 \le p \le 1$	$p(1-p)^{k-1}$ for $k = 1, 2, \ldots$ 0 otherwise	$\frac{1}{p}$	$\frac{q}{p^2}$

Table A.1: Discrete probability laws.

■ A.1.7 Some frequently encountered continuous probability laws

Probabilty law and parameter values	probability mass function $f(x)$	Mean $\langle X \rangle$	Variance $\mathrm{Var}(X)$
Uniform over $[a, b]$	$\frac{1}{b-a}$ for $a < x < b$ 0 otherwise	$\frac{a+b}{2}$	$\frac{(b-a)^2}{12}$
Normal or $N(\mu, \sigma^2)$	$\frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	μ	σ^2
Exponential $\lambda > 0$	$\lambda\, e^{-\lambda x}$	$\frac{1}{\lambda}$	$\frac{1}{\lambda^2}$

Table A.2: Continuous probability laws.

A.2 Generation of Random Numbers and Random Variables

The building block of a simulation study is the ability to generate random numbers, where a random number represents the value of a random variable uniformly distributed on the interval [0,1]. Nowadays the approach to generate random numbers is to use a computer to generate pseudo-random numbers. These pseudo-random numbers constitute a sequence of values, which although they are deterministically generated, have all the appearances of being independent uniform [0,1] random variables. There are several methods to generate pseudo-random numbers.

Very often the random number generator included with many programming languages is based on the ongruential method. In this method each term in the sequence can be found from the preceding one by the relation

$$x_{n+1} = (a\, x_n + c) \bmod m,$$

where x_0 is the "seed" and a, c and m are non negative integers. The random numbers on the unit interval [0,1] are given by $r_n = x_n/m$.

We will assume that we have a "good" random number generator to our disposal. For more information on random number generation, see for example [16] and [18]. Here we mention some bugs/features of generic random number generators:

- Random number generators require a "seed" to start with. For two different seeds two different sequences of random numbers are generated, otherwise two identical sequences are generated.
- Random number generators will have a periodic behaviour. At some point in the sequence the initial value will be reached again. After this the whole sequence is repeated identically as the first sequence. A good random number generator has a "very" long period.

■ A.2.1 The Inverse Transform methods

Suppose that we want to simulate a discrete random variable X, which has probability mass function

$$P[X = x_j] = p_j \text{ for } j = 0,\ 1,\ \ldots$$
$$\sum_j p_j = 1. \tag{1.28}$$

To accomplish this, we generate a random number U that is uniform on the interval [0,1] and set

$$X = \begin{cases} x_0 & \text{if} \quad p_0 < U < p_1 \\ x_1 & \text{if} \quad p_0 < U < p_1 \\ \vdots & \\ x_j & \text{if} \quad \sum_{i=1}^{j-1} p_i < U < \sum_{i=1}^{j} p_i \\ \vdots & \end{cases} \tag{1.29}$$

For example if we wish to simulate the coin flipping:

```
1] Generate U
2] if U < 0.5 set X = heads
   if U ≥ 0.5 set X = tails
```

For the continuous case: Let U be a uniform [0,1] random variable. For any continuous distribution function F the random variable X defined by

$$X = F^{-1}(U) \tag{1.30}$$

has distribution F.

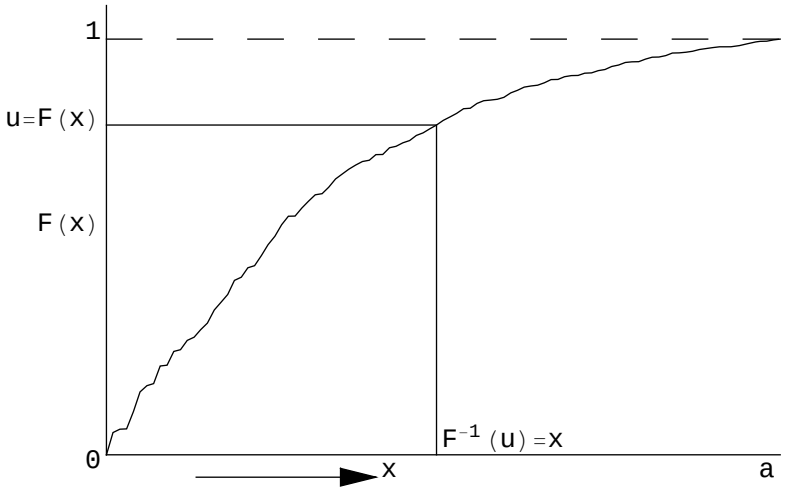


Figure A.1: Example of a cumulative distribution function $F(x)$, where x is in $[0,a]$

For example if X is an exponential random variable with rate 1, then its distribution function is given by:

$$F(x) = 1 - e^{-x} \tag{1.31}$$

if we let $F^{-1}(u) = x$ then

$$u = 1 - e^{-x} \iff 1 - u = e^{-x} \iff x = \ln(1 - u). \tag{1.32}$$

Hence we can generate an exponential random variable with rate 1 by generating a random number U and setting X = -ln(1-U). A small savings in computer time can be obtained by noting that 1-U is also uniform on [0,1]. Thus -ln(1-U) has the same distribution as -ln(U). That is the negative logarithm of a random number is exponentially distributed with rate 1.

■ A.2.2 The Rejection method

Suppose we have an efficient method for simulating a random variable having probability mass function $\{q_j, j \geq 0\}$. We can use this as the basis for simulating another distribution which has mass function $\{p_j, j \geq 0\}$.

First we simulate a random variable Y having mass function $\{q_j, j \geq 0\}$. Then we accept this simulated value with probability proportional to p_Y / q_Y. Specifically, let C be a constant such that $p_j / q_j \leq C$ for all j and $p_j \neq 0$. The following technique, called the rejection method generates random variables X having mass function $p_j = P[X = x_j]$.

Rejection Method for discrete random variables

1] Simulate the value of Y, having probability mass function
2] Generate a random number U

```
3] If U<p_Y / C q_Y set X=Y
   Else          return to 1
```

The rejection method for continuous random variables can be formulated analogously:

Suppose we have a method for generating a random variable having density function $g(x)$. We can use this as the basis for generating random variables from the continuous distribution having density function $f(x)$ by generating Y from g and then accepting this generated value with a probability proportional to

$$f(Y)/g(Y). \tag{1.33}$$

Specifically, let C be a constant such that

$$f(Y)/g(Y) \le C \tag{1.34}$$

for all y, then we have the following technique for generating a random variable X having density f.:

Rejection Method for continuous random variables

```
1] Generate Y having density g
2] Generate a random number U
3] If U<p_Y / C g_Y set X = Y
   Else          return to 1
```

This method is very useful for example when it is not possible to calculate F^{-1} analytically.

Lets use the rejection method to generate a random variable having density function

$$f(x) = 20\,x\,(1-x)^3 \qquad 0 \le x \le 1 \tag{1.35}$$

Since this random variable is concentrated in the interval [0,1] let us consider the rejection method with

$$g(x) = 1 \qquad 0 \le x \le 1 \tag{1.36}$$

Then

$$\frac{f(x)}{g(x)} = 20\,x\,(1-x)^3 \tag{1.37}$$

This function has a maximum value at $x = 1/4$. So

$$\frac{f\left(\frac{1}{4}\right)}{g\left(\frac{1}{4}\right)} = 20 \times \frac{1}{4}\left(1-\frac{1}{4}\right)^3 = \frac{135}{64} = C \tag{1.38}$$

And thus

$$\frac{f(x)}{C\,g(x)} = \frac{256}{27}\,x\,(1-x)^3 \tag{1.39}$$

The rejection procedure

1] Generate random numbers $U_1$and U_2
2] If $U_2 \leq \frac{256}{27} U_1 (1 - U_1)^3$ set X = U_1
   Else return to 1

Appendix B: Generating Normally distributed Random Variables

B.1 Box-Müller Method

Let X and Y be independent unit normal random variables and let R and θ denote the polar coordinates of the vector (X, Y). So

$$R^2 = X^2 + Y^2,$$
$$\tan\theta = \frac{Y}{X}. \tag{2.1}$$

Since X and Y are independent, their joint density is the product of their individual densities and thus is given by:

$$f(x, y) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} = \frac{1}{2\pi} e^{-\frac{(x^2+y^2)}{2}}. \tag{2.2}$$

To determine the joint density of R^2 and θ we make the change of variables

$$d = x^2 + y^2,$$
$$\theta = \tan^{-1}\left(\frac{y}{x}\right). \tag{2.3}$$

The Jacobian of this transformation-that is, the determinant of partial derivatives of d and θ with respect to x and y-is easily shown to equal 2. Therefore the joint density function of R^2 and θ is given by

$$f_{R^2,\theta}(d, \theta) = \frac{1}{2}\frac{1}{2\pi} e^{-d/2} \qquad 0 < d < \infty, \quad 0 < \theta < 2\pi. \tag{2.4}$$

However, as this is equal to the product of an exponential density having mean 2 and the uniform density on $[0,2\pi]$, it follows that R^2 and θ are independent with R^2 being exponential with mean 2 and θ being uniformly distributed over $[0, 2\pi]$. We can now generate a pair of independent normal random variables X and Y by using (B.4) to first generate their polar coordinates and then transforming back to rectangular coordinates. This is accomplished as follows:

1. Generate random numbers U_1 and U_2 .

2. R^2=-2 ln U_1 and thus R^2 is exponential with mean 2.

3. Set θ =2 π U_2 and thus θ is uniform between 0 and 2π.

4. Now let

$$X = R\cos\theta = \sqrt{-2\ln U_1}\cos(2\pi U_2)$$

$$Y = R\cos\theta = \sqrt{-2\ln U_1}\sin(2\pi U_2)$$

This method is generally know as the Box-Müller method.

B.2 Random Walk using Metropolis

The next piece of Mathematica ™ code generates a random walk based on the Metropolis algorithm, with a probability distribution $Exp[-0.5(x^2 + y^2)]$ (a 2-dimensional normal distribution with mean $\{x,y\}=\{0,0\}$, $\sigma_{xy} = \sigma_{yx} = 0$, and $\sigma_{xx} = \sigma_{yy} = 1$) :

(For an introduction to Mathematica ™ see[23]).

```
Mathematica ™ code for generating a Random Walk

ProbDist[point_] := Exp[-0.5 * (point[[1]] * point[[1]]
                     + point[[2]] * point[[2]])]

MetropolisStep[oldpoint_] :=
    Module[{newpoint,Prob},
        newpoint=oldpoint+With[{dir = Random[Real, range]},
                {Cos[dir], Sin[dir]}];
        Prob = ProbDist[newpoint] / ProbDist[oldpoint];
        If[Prob>=1,Return[newpoint],
            If[Prob>Random[Real,{0,1}],Return[newpoint],
            Return[oldpoint]]
        ]
    ]

MetropolisWalk[n_Integer] :=
    Module[{points},
        points = NestList[ MetropolisStep[#]&, {0.0, 0.0}, n];
        Show[ Graphics[{Point[{0,0}], Line[points]}],
            AspectRatio->1.0, Frame->True,
```

```
            FrameTicks->None]
    ]
```

The result of the execution of this code for 10000 Metropolis steps with $p(x,y) = 1$ and $p(x,y) = Exp[-0.5(x^2 + y^2)]$ respectively, is shown in the next figures:

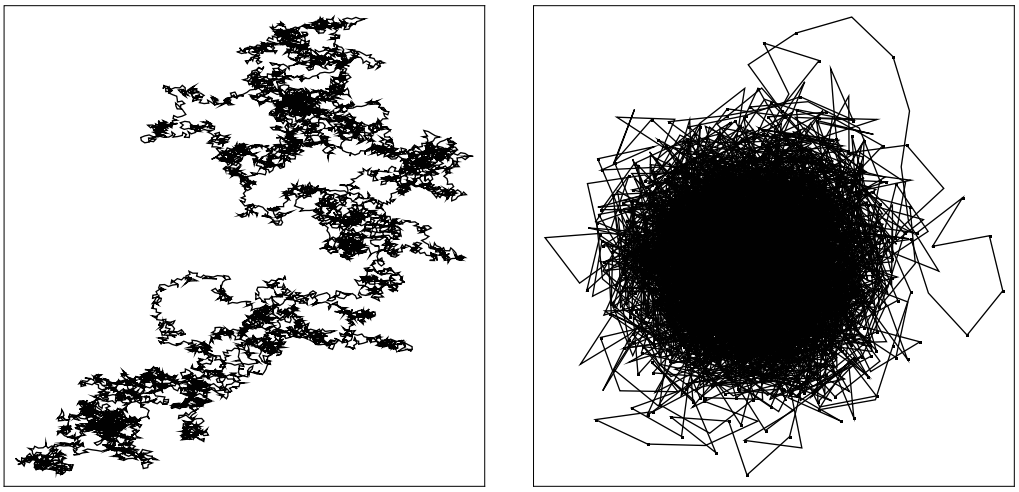


Figure B.1: MetropolisWalk[10000], with $p(x,y) = 1$ and with $p(x,y) = Exp[-0.5(x^2 + y^2)]$.

From these figures we observe that for $p(x,y) = 1$ we obtain a more or less random distribution, whereas the *Z*-normal distribution shows a normal distributed walk around the mean value (0,0). Closer inspection for different numbers of Metropolis steps nicely shows the dependency of the density of the walk relative to the values of the co-variance matrix used (data not shown). The underlying simulation mechanism used here is the so-called *Markov Chain* simulation, see Chapter 2.1.

Appendix C: A Bayesian approach to simulation of a stochastic model

C.1 Introduction to the Method

Let $P(x)$ be the probability that event "x" has occurred, and $P(y \mid x)$ the (conditional) probability that "y" has occurred given the fact that event "x" has taken place. Then with the joint probability (the probability that both occur) we have:

$$P(x \mid y) = \frac{P(x \cap y)}{P(y)}. \tag{3.1}$$

If the occurrence of event "x" is independent of the occurrence of event "y" (and the other way around) then $P(x \mid y) = P(x)$. Therefore the previous equation transforms into:

$$P(x \mid y) = \frac{P(y \mid x)\, P(x)}{P(y)}. \tag{3.2}$$

This is the well known Bayes theorem (reference [1] is worthwhile reading!). A simple formula that has had a enormous impact on a large variety of sciences. Interpreting this theorem in terms of simulation we see that if the user is able to express his beliefs about the values of the parameters under investigation in the form of a prior distribution function $P(x)$, then experimental information based on the likelihood function $P(y \mid x)$ (outcome of the simulation) can be used to convert these initial beliefs into a posterior distribution $P(x \mid y)$. In other words given some ideas about the model we want to study, we can generate via simulation information that can be used to update our initial guesses. This can for instance be applied to complicated markov chain simulations, fuzzy logic models and image recovery [5, 12].

To get a feeling of the use(fullness) of this theorem consider the following head-or-tail experiment.

N-times a coin is tossed with as result d times head. What is the probability $P(x)$, that the result of a toss gives a head? Without a priori knowledge, $P(x) \approx d/N$. If however preceding head-or-tail experiments were carried out, we could give a probability distribution for the random variable X:

$$p(x) = P\{X = x\}. \tag{3.3}$$

With Bayes theorem the best estimate for $P(x)$ is such that:

$$P(x \mid d) = \frac{P(d \mid x)\, P(x)}{P(d)} = \max. \tag{3.4}$$

Where the formula maximises the possibility that hypothesis X is the correct information given

experimental data *d*. This is clearly an experimentalist's interpretation of this theorem. In the experiment we use:

$$P(x|d) \sim P(d \mid x)\, P(x) \tag{3.5}$$

since $P(d)$ can be considered constant. Consequently a simulation experiment consists of the following components:

1. $P(x)$: a priori knowledge (hypothesis before experiment);
2. $P(d \mid x)$: experimental model (how data appears given X; the likelihood function);
3. $P(x \mid d)$: a posteriori knowledge (= a priori knowledge + knowledge from experiment).

The $P(x)$ and $P(d|x)$ are established by "common sense" or by means of "educated guesses."

Let's investigate these concepts by the following example. Suppose that:

- $P(x)$: normal distribution with $\mu = 0.2$;
- $P(d \mid x)$: N trials with $d/N = 0.8$.

We are interested in the influence of N and σ on the a posteriori distribution. The a priori knowledge $P(x)$ is normally distributed; notation $N(\mu,\sigma^2)$. In formula:

$$\begin{aligned} P(x) &= \frac{1}{Norm}\,\frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \\ Norm &= \int_0^1 P(x)\, d\,x. \end{aligned} \tag{3.6}$$

The experimental model $P(d \mid x)$ is the probability distribution that the hypothesis about X is correct, given the data "d" from the N trials. $P(d \mid x)$ has a binomial probability distribution, notation bin(N,x) . In formula:

$$P(d \mid x) = \binom{N}{d} x^d (1-x)^{N-d}. \tag{3.7}$$

Remark: The binomial is not calculated as function of the number of successes, *d*, but rather as function of success per trial $P(X = x)$. The next figures show the results of some of these experiments for different values of the parameters that model the system:

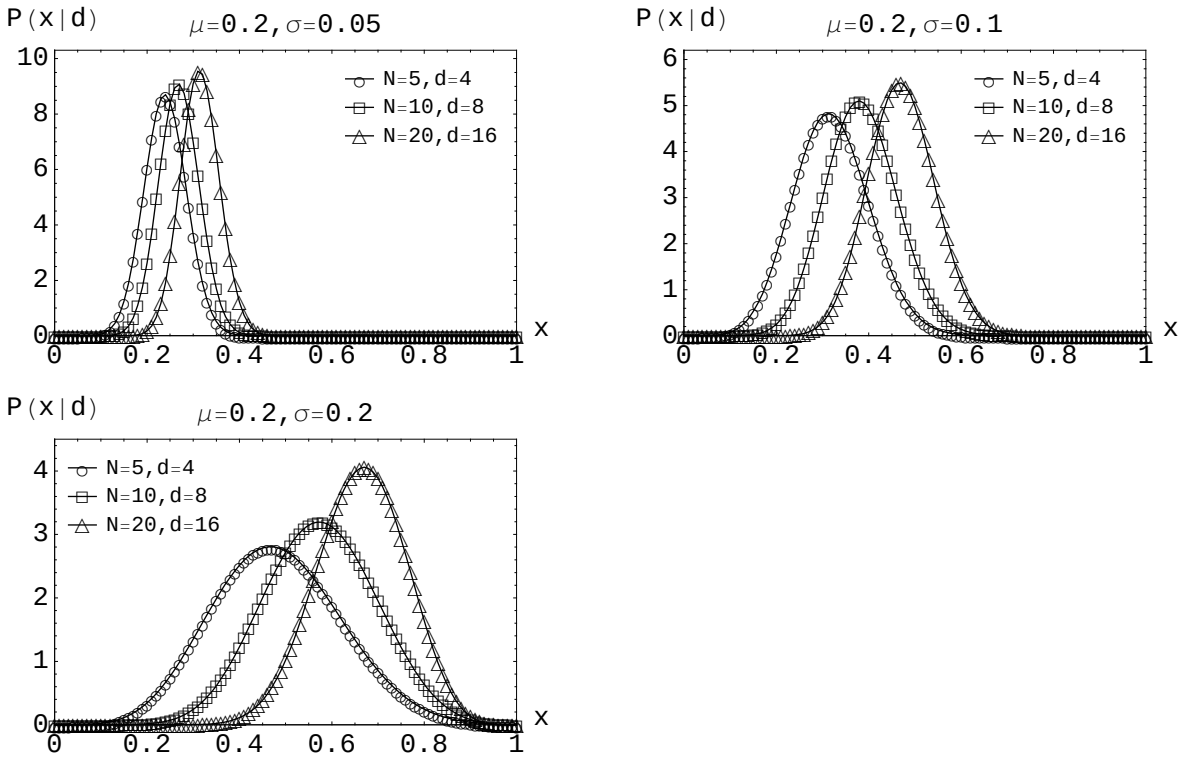


Figure C.1: The results of three experiments with μ=0.2 and various σ.

From these experiments we observe that if the prior information is accurate (σ is small), the estimate for x is dominated by the value of μ (prior knowledge). We see that when the data set is large ($N \longrightarrow \infty$) the prior information gets "washed out." Furthermore we observe that if no prior information is present, the estimate for x is data-driven. This should give us some ideas as how to design and evaluate simulation experiments. The problem of course is that Bayes theorem does not prescribe how to find $P(x)$. This information must be abstracted from the physical model and the system knowledge we have. This inspires us to use the predictive capacity of Bayes theorem wherever possible. Especially in computationally intractable problems where tedious simulation is the only way out.

C.2 Examples of Bayes method in simulation

■ C.2.1 The wildcatter's Problem

In this section we will describe the problem of a wildcatter who must decide whether to drill or not at a particular site [11]. The decision will be based on the profitability of the drilling. This particular problem can be solved using Bayes' theorem. There is some information that the wildcatter can use in making his judgement, about past experience in the drilling area. This past information can be summed up as follows:

If we would drill 8 wells:

- 70% of the time: all 8 wells will break or be profitable.
- 20% of the time: 1 well will be dry, hence not profitable.
- 10% of the time: 2 wells will be dry.

Assume that in order for the wildcatter to break even in the long run, at least 6 out of 8 wells will have to be profitable. For the wildcatter to make money, 7 out of 8 wells will have to be profitable. The current situation is that the wildcatter has drilled 3 wells of which one has been unprofitable, i.e. dry. Obviously the wildcatter wants to make money, so the question to be asked is: "What is the probability that the second well will turn out to be dry?". The historic information, combined with the current situation should deliver the answer to the wildcatter's problem.

Let's try to formalise this question. If the frequency F of dry wells is known, and d_i is the event of drilling a well, then the probability of any particular well being dry is $P(d_i \mid F)$. If F is known we can use the binomial distribution to calculate the occurrence of 1 dry well out of three. These probabilities for all three possible frequencies and the probabilities for the frequencies themselves are summarised in Table C.1.

Frequency(F)	Prob. 1 dry well	$P(F)$
0	0	0.7
0.125	0.287	0.2
0.25	0.422	0.1

Table C.1: Probability of 1 dry well and Probability of occurrence.

Using this information how should we calculate the probability that the F = 0.25 given the current drilling pattern, i.e $P(F{=}0.25 \mid d)$, where d are all patterns with 1 dry well out of three. Using Bayes, we arrive at the following equation:

$$P(F = 0.25 \mid d) = \frac{P(d \mid F = 0.25)\, P(F = 0.25)}{P(d)}. \tag{3.8}$$

We only need to calculate the probability $P(d)$ in order to find the solution to (C.8). $P(d)$ is the probability that the wildcatter drills 1 dry well out of 3. Using the numbers in Table C.1, we can calculate this quantity by:

$$P(d) \approx 0.7 \times 0.0 + 0.2 \times 0.287 + 0.1 \times 0.422 \approx 0.0996. \tag{3.9}$$

Substituting this calculated number in (C.8) we obtain the probability of drilling a second dry well:

$$P(F = 0.25 \mid d) \approx \frac{0.422 \times 0.1}{0.0996} \approx 0.423. \tag{3.10}$$

Therefor we arrive at the final conclusion that the wildcatter has a better than 57% change that the drilling will be profitable.

■ C.2.2 Inductive Physics and the law of Diffusion

In this section we will describe the application of Bayes' theorem to the seemingly phenomenological 1D Fick's law of diffusion [15]. The flow of particles in a dilute solution from high to lower concentrations is described by the following conservation law:

$$\frac{\partial c(x,\, t)}{\partial t} + \nabla^2 (J) = 0, \tag{3.11}$$

where $c(x,t)$ is the concentration of particles at position x at a time t. Fick's law relates the flow of low concentrations to the density gradient:

$$J = -D \nabla (J). \tag{3.12}$$

How do we find the correct diffusion coefficient D? Logically, given where a particle is now, we can sum the velocities of all particles in a small region, from which we can derive the local flux $J(x,t)$. We calculate the average velocity of a particle over a time interval 2τ:

$$\overline{v} = \frac{x(t + \tau) - x(t - \tau)}{2\,\tau}. \tag{3.13}$$

The probability that a particle will move from $x(t)$ to $y(t) = x(t + \tau)$, is given by a distribution $P(y \mid x, \tau)$. The motion of the particle is caused by a very large number of small increments, which is the result of encounters with individual water molecules. According to the Central Limit Theorem (A.26), this large number of fluctuating increments will result in a Gaussian form for the $P(y \mid x, \tau)$ distribution. This distribution will have a zero mean, because of the inherent symmetry:

$$P(y \mid x, I) = A\, e^{-\frac{1}{2}\left(\frac{y-x}{\sigma(\tau)}\right)^2}, \tag{3.14}$$

where I stands for the stated prior information. The spreading function equals the expected square of the displacement, i.e. $\sigma^2 = (\delta x)^2$. The particle is equally likely found on the left site as on the right site of the distribution, i.e. the expectation of y is $\langle\, y \,\rangle = x(t)$. We know that the equations of motion are

time reversal invariant, hence the past motion $z = x(t\text{-}\tau)$ will also have distribution (C.14). This implies that $\langle z \rangle = x(t)$, and therefore, according to Equation (C.13), the estimated velocity is zero.

According to the reasoning stated above, we do not get the correct density gradient. We could suggest that it is not possible to calculate the diffusion coefficient directly from first principles. In other words how do we know that it is the density *gradient* that matters, and not some other function of density. The answer to this question is not to solve the problem of prediction, given the dynamics, but rather state it is an inference problem. We do not want to ask the question: How do the equations of motion require the particles to move about on the average? But rather state the inference question:

What is the best estimate we can make about how the particles are in fact moving in the present instance, based on all the information we have?

It is crucial to note that the equations of motion are symmetric in past and future, but that our information about the particles is not. Equation (C.14) expresses an average over the class of all possible motions compatible with the dynamics; from which symmetry in movements naturally follows. For a real situation, it is only one of all these possible motions that is in fact executed, and we ought to use as prior information a small class of possibilities out of the large class, in which the particle is most likely to be: We need to calculate the inverse probability $P(z \mid x, I)$, which requires the use of Bayes' theorem:

$$P(z \mid x, I) = A\, P(z \mid I)\, P(x \mid z, I). \tag{3.15}$$

The prior probability $P(z \mid I)$ is proportional to $c(z)$, i.e. the concentration at the past position z. Using Equation (C.14) for $P(x \mid z, I)$, we get:

$$\ln P(z \mid x, I) = \ln(c(z)) - \frac{(x - z)^2}{2\,\sigma^2(\tau)} + \text{Constant}. \tag{3.16}$$

We derive the most probable value (i.e. the maximum) of z by differentiating this equation with respect to x and setting result equal to zero:

$$z = x + \sigma^2\, \nabla \ln(c(z)) \equiv x + (\delta x)^2\, \nabla \ln(c(z)). \tag{3.17}$$

If we substitute (C.17) in Equation (C.13) for $x(t\text{-}\tau)$, using x as the expectation for y, we estimate the velocity to be:

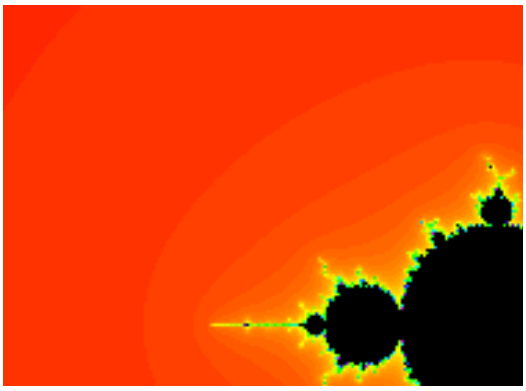
$$\overline{v} = \frac{x - \left(x + (\delta \mathrm{x})^2\, \nabla (\ln c)\right)}{2\,\tau} = -\frac{(\delta x)^2}{2\,\tau}\, \nabla (\ln c), \tag{3.18}$$

from which we can calculate the average diffusion flux over the time interval 2τ:

$$J(x, t) = c\, \overline{v} = -\frac{(\delta x)^2}{2\,\tau}\, \nabla c. \tag{3.19}$$

By comparing Equation (C.19) with (C.12) we have obtained the Einstein diffusion coefficient $D = \frac{(\delta x)^2}{2\,\tau}$ using Bayes' theorem. In addition the phenomenological ∇c is derived automatically.

Appendix D: Fractals and Fractal Dimension



D.1 Introduction

Many objects in nature are so complicated and irregular that they cannot be modelled well using conic sections, polygons, spheres and the other familiar objects of classical geometry. For example, circulatory systems, clouds, trees, mountains, and coastlines cannot be reduced to combinations of simple shapes from classical geometry. Where classical geometry ends as a tool for analyzing the complexity of natural objects, fractal geometry begins. Today, fractals are used to model a wide range of biological and topographical entities and to produce ultra-realistic special effects for movies and video games.

The question "How long is the coastline of Britain?" posed by Benoit Mandelbrot, the father of modern fractal theory, in his book *The Fractal Geometry of Nature* is not as simple as it appears. The problem is that one's answer to this question depends on the length of the ruler one uses. Unlike circles and the other shapes from classical geometry, coastlines are very irregular. They're full of inlets, bays, and rocky shores. A shorter measuring stick will fit more snugly in these nooks and crannies and increase the estimated length of the coastline. Hence, if we measure the length of Britain's coastline using a mile-long ruler, we will get one value. If we use a shorter ruler, say a yardstick, we will get a larger value because a yardstick can more closely approximate Britain's convoluted boundary. In fact, as the scale of measurement decreases, the estimated length increases without limit. Thus, *as the length of the ruler approaches zero, the estimated length of the coastline approaches infinity*. This difficulty in measuring due to the irregularity of the object being measured is characteristic of fractal curves and surfaces.

Fractal theory is grounded in geometry and dimension theory. Geometrically, fractals are independent of scale and appear equally detailed at any level of magnification. This property, called self-similarity, means that any portion of a self-similar fractal curve, if blown up in scale, would appear

identical to the whole curve. In other words, if we shrink or enlarge a fractal pattern, its appearence remains unchanged. This repetition of a pattern at all scales, no matter how small, is exhibited by many natural objects. For example, imagine that you are in space looking at the coastline of Britain. As you approach the Earth, the coastline still looks like a coastline. No matter how close you get to Britain's shore, the coastline appears equally complex. Even after you land your spacecraft and get down on your hands and knees with a microscope at the water's edge, the coastline still looks jagged and irregular.

The term *fractal*, introduced in 1975 by Benoit Mandelbrot, is an abbreviation for "fractional dimension". We all learned in high school that in classical geometry a line is an one dimensional object and a plane is two dimensional. Strangely, if we put enough kinks in a line, the resulting fractal curve will have a dimension somewhere between one and two, so that it is neither a line nor a plane but something in between. Similarly, an extremely convoluted surface will have a dimension between two and three. Such a figure is called a fractal. The concept of fractal dimension provides a way to measure how rough fractal curves are. The more jagged and irregular a curve is, the higher its fractal dimension. Fractional dimension is related to self-similarity in that the easiest way to create a figure that has fractional dimension is through self-similarity.

D.2 Regular non-random fractals

Fractals fall into two categories, random and non-random (regular). It is instructive to discuss a much studied regular fractal, the Koch snowflake. The construction of this curve starts with a triangle. In each construction step an edge is replaced by a set of 8 equal-sized new edges.The boundary of the Koch Snowflake constructed by Helge Von Koch in 1904 is the union of three congruent self-similar fractals. Each third of the snowflake is constructed by starting with one side of an equilateral triangle and performing an iterative process. The following cell defines a function called Snowflake that illustrates the stages in this iterative process (just execute the code).

■ D.2.1 Koch Snowflake Construction

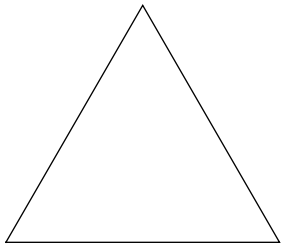
The Koch code

```
Clear[n, Snowflake, start, finish, doline]
Snowflake[n_Integer?NonNegative] :=
 Show[Graphics[
  Nest[ (#1 /. Line[{start_, finish_}] :> doline[start, finish]) &,
     {Line[{{0, 0}, {1 / 2, Sqrt[3] / 2}}],
      Line[{{1 / 2, Sqrt[3] / 2}, {1, 0}}],
      Line[{{1, 0}, {0, 0}}]},
     n]],
  AspectRatio -> Automatic, PlotRange -> All]

doline[start_, finish_] :=
 Module[{vec, normal},
 vec = finish - start;
 normal = Reverse[vec] {-1, 1} Sqrt[3] / 6;
 {Line[{start, start + vec / 3}],
 Line[{start + vec / 3, start + vec / 2 + normal}],
 Line[{start + vec / 2 + normal, start + 2 vec / 3}],
 Line[{start + 2 vec / 3, finish}]
 }
 ];
```

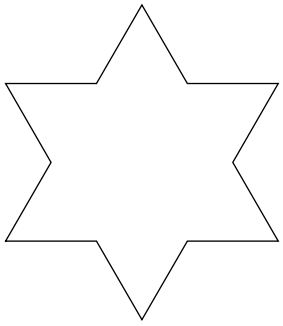
The following cells show examples of different iterations of the Koch code

```
Snowflake[0]
```



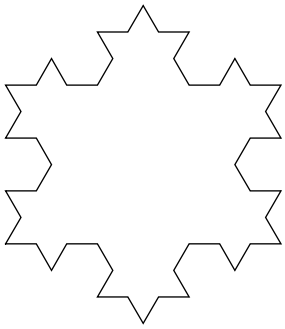
At the next stage of the snowflake's construction, we remove the middle one-third of each side and add two new segments having the same length as the part that was removed. (See the picture generated by the cell below.)

```
Snowflake[1]
```

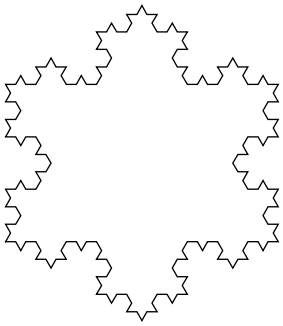


At each stage, we replace the middle-third of every segment in the previous stage by two new segments, creating a "bump" on the original segment. Evaluate the following cells to see the next four stages in the construction of the Koch Snowflake.

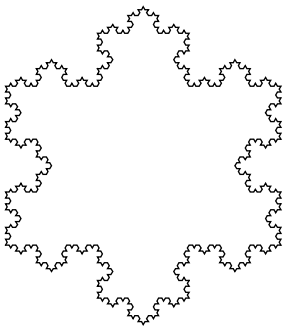
```
Snowflake[2]
```



```
Snowflake[3]
```



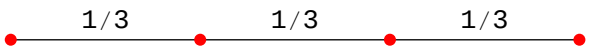
```
Snowflake[4]
```



D.3 The Fractal Dimension

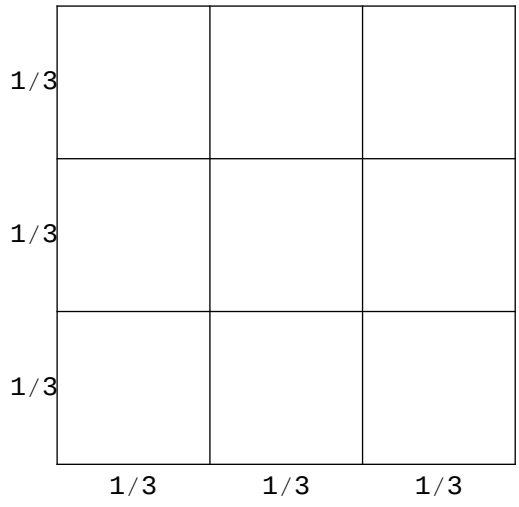
Consider a line segment divided into N equal pieces. Each of these N pieces can be thought of as a scaled version of the whole segment, with scaling ratio $r = \frac{1}{N}$. The relation between N and r is clearly $N\,r = 1$. For example, if $N = 3$, then $r = \frac{1}{3}$ and $N\,r = 3 \times \frac{1}{3} = 1$, see below.

N=3, r=1/3, Nr=1

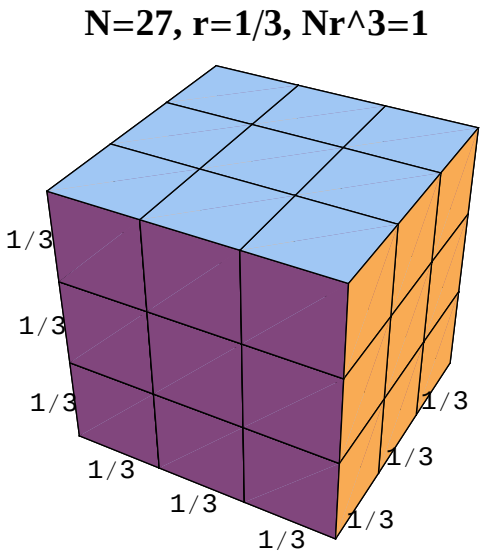


Now suppose the sides of a square are scaled by a factor r to produce N identical subsquares, each of which is a scaled version of the whole square. The relation between N and r in this case is $N\,r^2 = 1$. For example, if $r = \frac{1}{3}$, then $N = 9$ and $N\,r^2 = 9 \times \frac{1}{9} = 1$. See the figure below.

N=9, r=1/3, Nr^2=1



Finally, if a cube is scaled in the x, y, and z directions by a factor r to produce N equal subcubes, then the relation is $N\,r^3 = 1$. For example, if $r = \frac{1}{3}$, then $N = 27$ and $N\,r^3 = 27 \times \frac{1}{27} = 1$. (See the figure generated by the cell below.)



The line, square, and cube above have integer dimensions of one, two, and three respectively. Notice that the dimensions of these objects show up as the exponent d in the relation $N\,r^d = 1$, where N is the number of equal subunits and r is the scaling ratio. In general, if a given set is the union of N essentially disjoint copies of the original that are scaled by a constant factor r, then the value of d that satisfies the equation $N\,r^d = 1$ is called the fractal dimension or similarity dimension of the set. *It turns out that there are configurations for which the value of d in* $N\,r^d = 1$ *is not an integer. Such configurations are called self-similar fractals!*

The explicit formula for d in terms or N and r is given by the cell below.

```
Solve[n r^d==1,d]
```

$$\left\{\left\{d \to \frac{\text{Log}\left[\frac{1}{n}\right]}{\text{Log}[r]}\right\}\right\}$$

The logarithm in the formula above can be taken with respect to any positive base different from 1. The following cell defines a function called *dimension* that takes the values of N and r as input and returns the value of d. This function will be used later in this notebook, so execute it now.

```
Clear[dimension, d]
dimension[n_Integer?Positive, r_?Positive] :=
    Module[{d},
        d = Log[1 / n] / Log[r] // N]
```

■ D.3.1 The Fractal dimension of the Koch Curve

Each third of the Koch Snowflake converges to a limiting curve K that is a self-similar fractal. If K is scaled by a factor of $r = \frac{1}{3}$, then there are $N = 4$ copies of the scaled version making up the entire set K. Hence, the fractal dimension of K is given by the following cell.

```
dimension[4, 1 / 3]
```

Since the dimension of the snowflake (1.26186) is greater than the dimension of the lines making up the curve (1), the Koch Snowflake is a fractal.

■ D.3.2 Boundary length of the Koch Curve

An important property of the Koch Snowflake is that its boundary has infinite length. This is especially surprising in light of the fact that the snowflake encloses only a finite area (after all, it can be completely covered with a square of paper). To show that the boundary of the snowflake has infinite length, it suffices to show that each of the three congruent fractals making up the snowflake has infinite length. Suppose that the initial segment (call it K_0) has length 1. Then K_1, the curve produced by removing the middle one-third of K_0 and adding two new segments having the same length, has length $4/3$. The curve K_2 at the end of the second stage has length $4^2 \big/ 3^2$. Repeating this process, the curve K_n produced after n stages has length $4^n / 3^n$. Hence, the length of the limit curve K is given by the following cell.

```
Limit[(4 / 3)^n, n -> Infinity]
```

■ D.3.3 Fractal dimension of non-regular fractals

For the fractals discussed above, the regular fractals, we can *calculate* the fractal dimension. For irregular fractals such as fractal structures from nature or from random processes (see Simuleren en Modelleren notebook 5) we need an alternative approach.

One approach is to overlay the object with a grid with gridspacing ε and count the number of intersected cells. By decreasing ε we can infer the fractal dimension. Explain in pseudo-code how this process can be used to determine the fractal dimension. (In the lectures this method was referred to as the Feder Box-counting method.)

References

[1] Bayes, T., "An essay toward solving a problem in the doctrine of changes," Philosophical Transactions of the Royal Society, 370-418 (1763).

[2] Callen, E. and Shapero, D., "A theory of social imitation," Physics Today **27**, 23-28 (1974).

[3] Gaylord, R.J., "The Ising Model," Mathematica in Education **3** (1994).

[4] Gaylord, R.J. and Wellin, P.R., *Computer Simulations with Mathematica* (Springer-Verlag, 1994).

[5] Geman, S. and Geman, D., "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," IEEE Trans. Patt. Anal. Mach. Intell. **6**, 721-741 (1984).

[6] Gould, H. and Tobochnik, J., *Computer Simulation Methods part I and II* (Addison Wesley, 1987).

[7] Grinstein, G., "Scale invariance, interfaces and nonequilibrium dynamics," J. Appl. Phys. **69**, 5441 (1991).

[8] Grinstein, G. and Jayaprakash, C., "Simple models of self-organized criticality," Computers in Physics **9**, 164-169 (1995).

[9] Hoshen, J. and Kopelman, R., "Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm," Phys. Rev. **B14**, 3438 (1976).

[10] Koonin, S.E. and Meredith, D.C., *Computational Physics* (Addison Wesley, 1989).

[11] Korsan, R.J., "An example of Bayesian Inference," The Mathematica Journal **3**, 57-59 (1993).

[12] Laarhoven, P.J.M., Boender, C.G.E., Aarts, E.H.L., and Rinnooy Kan, A.H.G., *A Bayesian Approach to Simulated Annealing*, Tech. Rept. 8839/A Philips, 1988.

[13] McCoy, B.and Wu, T.T., *The two-dimensional Ising Model* (Harvard University Press, 1973).

[14] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., and Teller, E., "Equation of State Calculations by Fast Computing Machines," J. of. chem. physics **21**, 1087-1092 (1953).

[15] Nog, D., "Ik wou dat ik wist waar," Zoek op **166**, 149-161 (1988).

[16] Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T., *Numerical recipes in C* (Cambridge University Press, Cambridge, 1988).

[17] Reif, F., *Fundamentals of Statistical and Thermal Physics* (McGraw-Hill, 1965).

[18] Ross, S.M., *A course in Simulation* (Maxwell MacmillianNew York, 1991).

[19] Sloot, P.M.A., *Parallel Scientific Computing and Simulation*, Tech. Rept. CS-95-xx , January, 1995.

[20] Stauffer, D., *Introduction to percolation theory* (Taylor & Francis London and Philadelphia, 1985).

[21] Wall, F.T., "Macromolecular dimensions obtained by an efficient Monte Carlo method without sample attrition," J. Chem. Phys. **63**, 4592 (1975).

[22] Weiss, G.H., "Random numbers and their Applications," American Scientist **71**, 65-71 (1983).

[23] Wolfram, S., *Mathematica: A system for doing mathematics by computer* (Addison Wesley, 1991).

Exercises

Introduction

In the programs you will have to make several subroutines that can be used more than once. So, before you start programming look through the assignments and try to make your subroutines as general as possible.

We strongly advise you to write the algorithms in pseudo-code before implementing. In principle, coding can be done in the language of your choice. However, implementation of the recursive algorithm in the percolation problem needs recursive constructs. Therefore Fortran is not suited for that specific algorithm. If you're eager to use Fortran then you'll have to rewrite the algorithm yourself.

The following subjects are covered in the exercises of Stochastic Simulations:

1. Monte Carlo Integration
2. Diffusion
3. Ising Model
4. Lattice Gas
5. Percolation

Each problem section starts with a reference to the theory in your syllabus:

Theory: section X

■ Monte Carlo (MC) Integration and Variance Reduction
Theory: Section 1 ; Appendix A1, A2

Question A

Use MC integration to estimate

$$\int_0^1 e^{-x}\,dx \qquad (1)$$

1. Estimate the integral for 3600 trials.
2. Compute the standard deviation σ at n = 3600.
3. Does σ change significantly if n > 3600 ?
4. Determine the error in your estimate.
5. How does the error compare to the error estimate $\sigma/\sqrt{n}$

Divide your measurements into s = 20 subsets of 180 trials each.

1. Compute the mean and standard deviation of the collection of s measurements.
2. Is $\sigma_s \,/\, \sqrt{s}$ consistent with your previous error estimates?
3. What are the implications of this observation?

Question B

Choose the importance sampling functions p(x) = C_1 and p(x) = $C_2 e^{-x}$. The constants C_i must be chosen such that the importance sampling functions are unity when integrated over the interval in question.

1. Evaluate by simulation

$$F = \int_0^{10} \frac{1}{1+x^3}\,dx \qquad (2)$$

2. List F_n (approximate value after n trials).
3. List σ and $\sigma/\sqrt{n}$.
4. Do you have benefit from applying importance sampling?

Question C

Choose the importance sampling function $p(x) = c\,\lambda e^{-\lambda x}$ and estimate

$$F = \int_0^{\pi} \frac{1}{1+x^2+\cos^2 x}\, dx \qquad (3)$$

1. Determine the value of λ that minimizes the standard deviation σ.

2. What is remarkable about the optimal importance sampling function when compared to the function that is integrated?

■ Diffusion
Part I: Random Walk

Theory: Section 4.2

A random walk consists of N steps (see section 4.2) of equal length in random directions, starting from an arbitrary position. You will study a random walk on a two-dimensional lattice. On a square and triangular lattice there are 4 and 6 possible moves respectively.

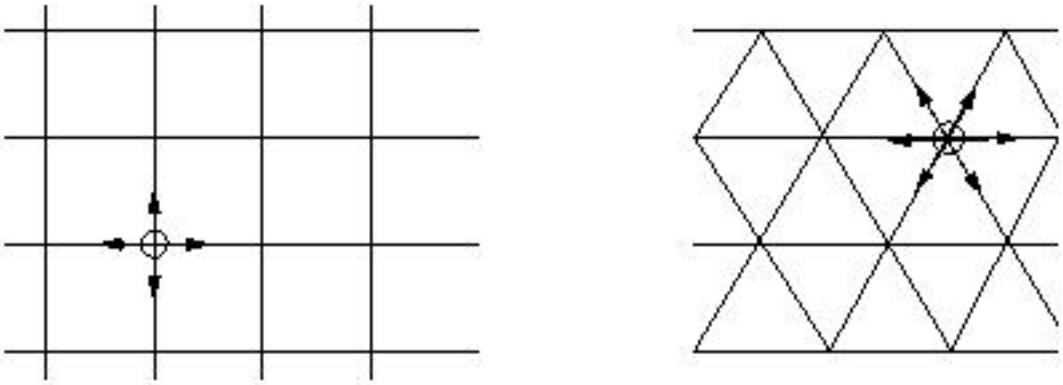


Figure 1: A square and a triangular lattice with possible directions

The quantity that is measured during the simulation is the net mean square displacement $< \triangle R_N^2 >$ given by (Eq. 4). Hence,$< \triangle R_N^2 >$ is determined by averaging over a reasonable number of trials for the same N. From this the root mean square displacement (Eq. 5) is calculated, which can be used to determine the critical exponent ν defined in (Eq. 6) for large N.

$$< \triangle R_N^2 > = < x_N^2 > + < y_N^2 > - < x_N >< x_N > - < y_N >< y_N > \quad (4)$$

Furthermore we define:

$$R_N = \sqrt{< \triangle R_N^2 >} \quad (5)$$

The critical exponent of a random walk is a constant that indicates the following relationship between R_N and the number of steps N in the random walk (see section 4.2 and references therein):

$$R_N = a\, N^{\nu} \quad (6)$$

Question I.A

1. Write a program (see psuedo code below) that simulates a random walk, i.e. a particle is somewhere on the lattice and at each time step it is moved at random to one of the neighboring sites. You'll have to use periodic boundary conditions, i.e. if a particle leaves the lattices at the top, it will reappear at the bottom.

```
REPEAT a number of times
     Clear displacements of walker
    DO N time steps
         Move walker in random direction
          Monitor displacement : dx = dx + x and dy = dy + y
         Accumulate displacements : x = x + dx, y = y + dy,
         x2 = x2 + dx * dx and, y2 = y2 + dy * dy
Calculate < △R_N^2 > (see Eq .4)
```

2. What will happen to the distance that a particle has moved away from its starting position if that particle croses a border? Solve this problem!

Question I.B

1. Enumerate (on paper) all random walks on a square lattice for N = 4 random steps and determine $< x_N >$, $< y_N >$ and $< \triangle R_N^2 >$.

2. Verify that your program gives the same results as the enumeration above. Note that you have to average over multiple trials in order to determine the statistical quantities for a specific value of N.

Question I.C

1. Do a Monte Carlo simulation to determine R_N for N = 8, 16, 32 and 64.

2. For each N you have to do a reasonable number of independent experiments. What do you call

reasonable?

3. What is the influence of putting $< x_N >$ and $< y_N >$ to zero in your calculation of formula (4) ?

4. Estimate the exponent ν from a log-log plot of R_N versus N. That is, the slope of the line equals ν.

5. Calculate the magnitude of the self-diffusion constant D as defined by [4.3] with the "time" t replaced by N. Peform this experiment for different values of N.

The measure of "time" in this context is arbitrary. The usual definition which we will often encounter is that *one unit of time corresponds to one Monte Carlo step per particle*. During one Monte Carlo step per particle, each particle attempts *one jump on the average*. The diffusion constant D is obtained as the limit t of D(t), where D(t) is given by [4.3].

Question I.D

1. Do a Monte Carlo simulation of R_N on the triangular lattice and estimate ν.

2. Can you conclude that ν is independent of the symmetry of the lattice ?

3. Does D depend on the symmetry of the lattice?

■ Part II: Lattice Gas

Theory: Section 4.3.1

Next consider a nonzero concentration c of random walkers (particles) on a square lattice (n × n sites). Each particle moves at random to *empty* nearest-neighbor sites but double occupancy of sites is excluded; otherwise the particles are noninteracting. Such a model is an example of a lattice gas. Note that the motion of an individual particle is correlated with the motion of the other particles. The physical motivation of this model arises from metal physics where diffusion is caused by thermal vacancies whose concentration depends on the temperature. The main physical quantity of interest is the self-diffusion constant D of k tracer particles. The algorithm for a Monte Carlo simulation of < D > for a given value of N, can be stated as follows:

```
REPEAT a number of times
   Init Lattice
   REPEAT a number of trials /* to calculate  △R_N^2 > .* /
      Clear displacements of particles
      DO N time steps
         Move particles
      Accumulate x, y, x2 and y2 for k tracers
      Calculate < △R_N^2 > for all k tracers
   Calculate D from individual tracer contributions
   Accumulate D
   Calculate < D >
```

Question II.A

1. Write a program that simulates a lattice gas with a concentration of c random walkers on a square lattice. You'll have to use periodic boundary conditions, i.e. if a particle leaves the lattices at the top, it will reappear at the bottom.

2. If the initial positions are recorded in an array, what will happen to the distance that a particle has moved away from its starting position if that particle crosses a border? Solve this problem!

Question II.B

1. Simulate a number of random walkers (take a fixed concentration c, take one tracer particle among the random walkers. Calculate the self-diffusion constant and its variation using only this single tracer particle. Note that the calculation resembles that of a simple random walk.

2. Observe that the variation will not decrease if longer simulation times are used (larger N). Explain this. (You can prove this theoretically using the Gaussian distribution and a calculation analogous to the calculation of the Maxwell velocity distribution for gasses).

Question II.C

Now use more than one tracer particle to calculate the self-diffusion constant. Explain the dependence of the deviation of the self-diffusion constant on the number of tracer particles.

Question II.D

1. Do a series of simulations to determine D for c = 0.1, 0.2, 0.3, 0.5 and 0.7.

2. Why is D a monotonically decreasing function of the concentration ?

3. Calculate theoretically the chance that a particle will jump back at time step t+1 to the position it

had at time step t.

4. Calculate theoretically the dependence D = D(c)

Question II.E

1. Consider a one-dimensional lattice model for which the particles move at random but double occupancy of sites is excluded. The latter restriction implies that particles cannot pass each other.

2. Calculate $< \Delta x^2 >$ as a function of t.

3. Do the particles diffuse, i.e. is $< \Delta x^2 >$ proportional to t? If so, explain why,and, if not, what is the *t*-dependence of $< \Delta x^2 >$?

■ Ising Model

Theory: section 3.2

In this assignment the 2D Ising model is examined. For details see chapter 3 in the reader. Use k_B=1 and J =1. Take a lattice of size L x L spins.

Question A

1. Write a program that can simulate a 2D Ising model.

2. Calculate, with the aid of section 3.2.2 in the reader the theoretical temperature dependence of E, M, C and χ on a 2x2 lattice.

3. Check your computer results with the exact results.

Question B

If the computer simulation is started, the configuration is in general not yet in an equilibrium situation. Look at the equilibration times for interesting values of the lattice size L and temperature T.

Question C

1. Compute C and χ for T = 1.5 to 3.0 in small steps for different values of L. Estimate the critical

temperature $T_c(L)$ from the maximum of C and χ.

2. Does T_c determined from C(T) has a stronger L-dependence than $T_c(L)$ determined from $\chi(T)$?

3. Estimate the critical temperature for an infinite lattice.

Question D

1. Calculate the correlation times of the magnetization at various temperatures. The correlation functions are defined by

$$C(k) = \frac{< f_i f_{i+k} > - < f >^2}{< f^2 > - < f >^2} \qquad (7)$$

The average $< f_i f_{i+k} >$ is an average over many different situations where two states are separated by k steps. The correlation time is $\tau = \Sigma_k C(k)$.

2. What happens around the critical temperature ?

Question E

1. Change the spin-flip dynamics and examine the correlation times around the critical temperature. An example is flipping two randomly chosen spins at the same time, try some others!

2. What spin-flip dynamics gives the lattice gas simulation model?

Question F

Since τ diverges near T_c for an infinite lattice, it is possible to define a "dynamic critical exponent" Δ by the relation $\tau \sim (T - T_c)^{-\Delta}$. On a finite lattice we have the relation $\tau \sim L^z$ at $T = T_c$. Use finite-size scaling arguments to obtain the relation of z to Δ. Compute τ for different values of L on a square lattice at $T = T_c$ and estimate z.

Question G

Repeat Question C but now use a triangular lattice and compare the found value for the critical temperature with $T_c = 3.641$.

■ Percolation

Theory: sections 5.1 - 5.4

In the reader the algorithm of Hoshen-Kopelman is shown. This is an efficient method to use for large lattices. Since we're not trying to write an article about percolation but just examining some of its

properties we will not use very large lattices. Therefore you can use a less efficient but simpler algorithm instead. The algorithm works as follows :

put a cluster counter to 1.
walk through the lattice, if you find a cite that is a hole that has not yet been examined then enter a recursive subroutine. If the program returns from this recursive subroutine the cluster counter is incremented.
The recursive subroutine first labels the unexamined hole as examined and puts the value of the corresponding place in the array "labels" to the value of the cluster counter. Then it looks at the neighbors of the (previously) unexamined hole. If one of these neighbors is itself an unexamined hole the subroutine is called again.

In pseudo code it looks like this

```
cluster_number = 1
 For all lattice sites :
  If the current site is an unexamined hole
  call walk_through (cluster _number, current site)
  cluster_number = cluster_number + 1
```

The subroutine walk_through looks like this :

```
 mark current site as examined
label (current site) = cluster \_ number
if (left neighbor = unexamined)
    walk_through (cluster_number, left neighbor)
if (right neighbor = unexamined)
    walk_through (cluster_number, right neighbor)
if (upper neighbor = unexamined)
    walk_through (cluster_number, upper neighbor)
if (lower neighbor = unexamined)
    walk_through (cluster_number, lower neighbor)
```

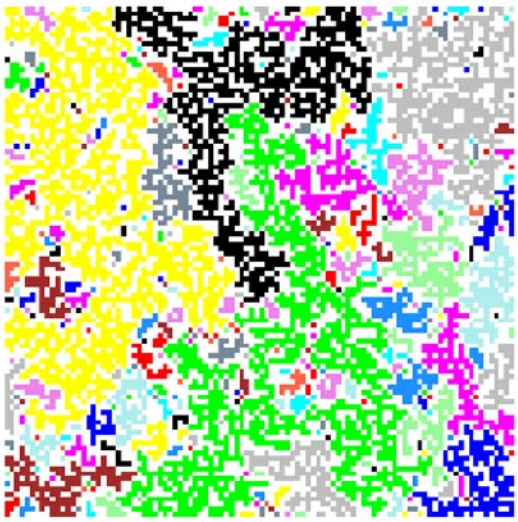


Figure2: An example of a cluster labeling (represented by grey values), P_c = 0.57.

Question A

Calculate the radius of gyration [6.1] for the largest non-spanning cluster as a function of the occupation probability p. The critical value for p is p_c = 0.59 (approximately). If p < p_c and a random lattice is generated with a spanning cluster, that lattice is discarded. If p > p_c and a lattice is generated without a spanning cluster, that lattice is also discarded. Below p_c, only lattices without a spanning cluster are used and above P_c only lattices with a spanning cluster are used.

Question B

p_c is the critical value for p. At this value the largest non-spanning cluster has a fractal shape. The fractal dimension can be calculated with the aid of the radius of gyration. This can be done by making a log-log plot of

the radius of gyration (Use R_s^2 and not R_s) versus the mass of the cluster for many different clusters. The mass is the number of sites belonging to the cluster. The slope of this line is the inverse of the fractal dimension:

slope = $1 / D_f$, where D_f is the fractal dimension. Determine D_f.

Question C

Since we are dealing with finite lattices, we can also use another scaling relation to determine the fractal dimension.

1. For $p > p_c$, the mass of the largest cluster scales with L^2. Check this. Why? In order to determine this scaling relation, you don't discard the spanning cluster, because you are interested in the largest cluster.

2. For $p = p_c$ the mass of the largest cluster scales with L^{D_f}, determine D_f. Is it comparable with the answer found in question B?