Lecture notes on Monte Carlo simulations

Peter Olsson

May 5, 2017

Contents

Chapter 1

Introduction

These notes are intended as an introduction to Monte Carlo methods in physics with an emphasis on Markov chain Monte Carlo and critical phenomena. Some simple stochastic models are also introduced; many of them have been selected because of there interesting collective behavior. The term *Monte Carlo* is used in the broad sense to contain all kinds of calculations that can be performed with the help of random numbers.

1.1 Buffon's needle

The common first example of a stochastic calculation is *Buffon's needle* – the calculation of the value of π by throwing a needle on a plane surface with parallel straight lines separated by a fixed distance. If the length of the needle is equal to the distance between the lines, the probability that the needle will cross a line is equal to $\frac{1}{\pi}\int_0^\pi d\theta \sin\theta = 2/\pi$. By throwing the needle a large number of times it is then possible to estimate π through the fraction of throws that hit the line.

Whereas the above example nicely illustrates the stochastic element of Monte Carlo simulations it doesn't properly convey the strength, beauty, and usefulness of MC simulations. This example differs in at least the two following ways from usual MC simulations:

- The calculation of π may be done in numerous other more efficient ways. In contrast MC methods are normally used for problems that would otherwise be considered very difficult or even intractible.

- The calculation of π would be much more precise if the experiment could be done in a systematic instead of a random way. One could then be lead to believe that a carefully chosen set of numbers would do better than the random numbers in the MC simulations, but that is indeed not the case. In some applications it is even the case that one needs a very high quality of the random numbers (very low degree of correlations) to obtain the correct result.

1.2 Models in Physics

Virtually all real world phenomena are complex and complicated. In spite of this we will here focus on a number of simple models. There are actually several good reasons for doing so:

- When one is trying to arrive at an *understanding* of some phenomenon it is necessary to turn to simplified versions of the system. Consider the opposite case that one actually managed to include everything in the calculation and obtained perfect agreement with experiments. This would be a great triumph for the people involved in the work, but would not provide any new understanding of what is going on. In contrast, the construction and analysis of a simple model that could be found to capture certain key features of the original system could add some new understanding about the mechanisms that are responsible for the obtained behavior.

- Many simple models are general enough to describe a large number of different systems. This means that even though the original motivation for a study usually comes from a single specific problem, the result can often be taken over to an entirely different context. The simple models may therefore function as tools in a conceptual toolbox.

- The development of physics since the beginning of the 70's has stressed the importance of *universality* – the insight that certain properties may be altogether insensitive to many of the details of the system. This means that the most efficient approach is to try to find and analyze the simplest possible model that belongs to the same *universality class* as the problem under consideration.

After stressing the use of the simple models it should be emphasized that most MC methods are general and flexible enough to work with models of arbitrary complexity.

1.3 Different kinds of Monte Carlo simulations

There are at least three different kinds of Monte Carlo simulations:

- **Transport simulations.** The basic problem here is an energetic particle (e.g. a neutron) that reaches a shield. It will then collide with the atoms in the shield and cause different kinds of reactions. The question is how much that will get through. Instead of keeping track of the position of the atoms in the shield and try to calculate the events according to the precise position and momentum of the incident particle, (which would be terribly complicated and according to quantum mechanics would still not be deterministic) the idea is here to choose the outcome of the collision by random. This collision could in turn give rise to a complex chain reaction where all the different events are similarly chosen by random.

- **Markov Chain Monte Carlo.** This is a method that is very useful in statistical physics where we want the configurations to appear with a probability proportional to the Boltzmann factor. This is achieved by constructing a Markov chain with the desired property. Monte Carlo in statistical physics is a big field that has exploded into a number of different methods of which several are very beautiful. Beside the methods for classical statistical mechanics that are covered in this course there are also many different ways to do Quantum Monte Carlo.

- **Simple stochastic models.** These should maybe not be called Monte Carlo, but the term is frequently used for all kinds of simulations that make use of random numbers.

Chapter 2

Random variables and distributions

Random variables arise when repeated attempts under apparently identical conditions fail to give the same results. We will denote the random variable by x. The random variables are usually distributed in a way which can be approximated by a simple mathematical function — the probability density function (pdf), $p(x)$. The meaning of this function is that the probability for a sample to be within the interval x to $x + dx$ is

$$p(x)dx.$$

The probability density function should of course be normalized,

$$\int dx\ p(x) = 1,$$

and for the case of discrete variables that can only take certain values $x^{(i)}$, the corresponding expression is

$$\sum_i p(x^{(i)}) = 1.$$

Content of this chapter

We will below first define two properties—mean and variance—that are used to characterize various distributions. We then turn to the art of *estimating* the same properties from a finite number of random variables. To estimate

the mean is usually trivial; the focus is therefore on estimating the variance which is needed to get an estimate of the uncertainity in the obtained mean. We will then be concerned with various methods to generate random samples from various simple distributions. In the last subsection we introduce Markov chains that are very important e.g. to generate multidimensional samples from complicated distributions.

2.1 Distributions

Two quantities are used for characterizing distributions: the *mean* μ and the *variance* σ^2. The mean is obtained as

$$\mu = \int dx\ x\ p(x).$$

The variance is a characteristic measure of the spread of the variables around the mean. It is given by

$$\sigma^2 = \int dx\ (x-\mu)^2\ p(x).$$

For the case of discrete variables the integrals above should of course be replaced by summations.

2.1.1 Distribution of a combination of variates

Consider the common case where z is the sum of two random *independent* variables, $z = x + y$. In that case

$$\mu_z = \mu_x + \mu_y,$$

and for the variance we obtain

$$\begin{aligned}
\sigma_z^2 &= \int dx\ p_x(x) \int dy\ p_y(y)\ ((x+y) - (\mu_x + \mu_y))^2 \\
&= \int dx\ dy\ p_x(x) p_y(y) \left[(x-\mu_x)^2 + 2(x-\mu_x)(y-\mu_y) + (y-\mu_y)^2\right] \\
&= \int dx\ p_x(x)(x-\mu_x)^2 + \int dy\ p_y(y)(y-\mu_y)^2 = \sigma_x^2 + \sigma_y^2
\end{aligned}$$

A generalization that follows directly is for the average of N independent random variables,

$$m = \frac{1}{N} \sum_{i=1}^{N} x_i, \tag{2.1}$$

for which we find $\mu_m = \mu_x$, but also the important result for the variance of the average of N independent variables,

$$\sigma_m^2 = \frac{1}{N}\sigma_x^2. \tag{2.2}$$

2.1.2 Distributions with two correlated variables

It is also common to have correlations between the variables. The distribution is then a single function of two variables $p(x, y)$ instead of two functions of one variable each. Introduce the covariance,

$$\mathrm{cov}[x, y] = \int dxdy\ (x - \mu_x)(y - \mu_y)p(x, y).$$

From the derivation above we find

$$\sigma_z^2 = \sigma_x^2 + \sigma_y^2 + 2\ \mathrm{cov}[x, y].$$

A quantity that gives a normalized measure of the strength of the correlations is the *correlation coefficient*,

$$\rho = \frac{\mathrm{cov}[x, y]}{\sqrt{\sigma_x^2\ \sigma_y^2}}.$$

2.1.3 Distributions of discrete variables

We now turn to some useful distributions. They naturally divide into two groups: distributions with discrete variables, and distributions with continuous variables.

The binomial distribution

The binomial distribution appears whenever the outcome may be classified into two mutually exclusive classes, $x = 0,\ 1$. The distribution is then

$$p(x) = \begin{cases} 1 - \alpha, & x = 0, \\ \alpha, & x = 1, \end{cases}$$

or

$$p(x) = \alpha^x (1 - \alpha)^{(1-x)},$$

The mean and the variance are readily found to be

$$\begin{aligned} \mu &= \alpha, \\ \sigma^2 &= \alpha(1-\alpha). \end{aligned}$$

The common use of the binomial distribution is to determine n, the number of successes ($x = 1$) out of N attempts. This is given by

$$P_N(n) = \frac{N!}{n!(N-n)!}\alpha^n(1-\alpha)^{(N-n)}, \quad 0 \leq n \leq N.$$

For this case the above expressions for the mean and the variance should just be multiplied by N.

The Poisson distribution

Consider some kind of process that depends on a number of independent events. If the average time between two successive events is τ then the average number of such events during a time t is $\mu = t/\tau$. The Poisson distribution, which describes the probability that exactly n events have taken place, is given by

$$p(n) = \frac{\mu^n}{n!}e^{-\mu}.$$

This is properly normalized, $\sum_n p(n) = 1$. How can you see that? For the variance of this distribution we have $\sigma^2 = \mu$.

2.1.4 Distributions of continuous variables

The uniform distribution

The pdf of the uniform distribution is constant for an interval of x, and zero otherwise,

$$p(x) = \frac{1}{b-a}, \quad a \leq x \leq b.$$

The mean and variance are $\mu = (b+a)/2$, $\sigma^2 = (b-a)^2/12$. For the standardized form, with $a = 0$ and $b = 1$ the variate will be denoted by ξ.

The Gaussian distribution

This is the most important distribution in physics. Its probability density function has the form

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

where the mean and variance appear explicitely in the formula.

One reason for the importance of the Gaussian distribution is that it appears as soon as one considers the mean m of a number of variates, x_i, Eq. (2.1). The remarkable fact is that the variable m will to a good approximation be Gaussian distributed with variance σ_x^2/N regardless of the distribution of x. The only requirement is that σ_x^2 is finite and the N is not too small. $N \geq 10$ is often enough. This is the Central Limit Theorem.

The exponential distribution

The time interval between successive random events (as discussed in the Poisson distribution) follow an exponential distribution,

$$p(x) = \frac{1}{\mu} e^{-x/\mu}, \quad x \geq 0.$$

The variance is $\sigma^2 = \mu^2$.

The power law distribution

This distribution is only defined for $x \geq x_0$, with $x_0 > 0$ and is only possible to normalize for $\gamma > 1$, the mean is only defined for $\gamma > 2$ and the variance only for $\gamma > 3$. The pdf is given by

$$p(x) = C_0 x^{-\gamma},$$

where the normalization constant is $C_0 = (\gamma - 1)x_0^{(\gamma-1)}$.

2.2 Estimation

We now leave the mathematically well-defined and precise distributions and turn to analyses that can be made from a finite number of samples. It is

then necessary to distinguish between the true values of the mean and the standard deviation and the corresponding values that may be estimated on the basis of finite samples. We therefore introduce m for the estimate of the mean, μ, and s^2 for the estimate of the variance, σ^2.

2.2.1 Mean and variance

With a population of N observables x_i the obvious estimate of the mean is

$$m = \frac{1}{N}\sum_i x_i, \tag{2.3}$$

whereas the estimate of the variance of x_i is

$$s^2 = \frac{1}{N}\sum_i (x_i - \mu)^2. \tag{2.4}$$

Note first that the prefactor $1/N$ compensates for the N terms in the sum. This value is therefore not expected to change with N. It also has a very direct interpretation: a plot of x_i vs i would produce a cloud where s is a measure of the with of this cloud of points. To be more precise, about 68% of the points would be between $\mu - s$ and $\mu + s$.

Note also that the true value of the mean ($= \mu$) is used in this expression. In some cases the correct value of μ may be known e.g. from symmetry properties of the problem, but in most cases we have to instead make use of the *estimated* mean m. With the estimated mean m instead of μ Eq. (2.4) instead becomes

$$s^2 = \frac{1}{N-1}\sum_i (x_i - m)^2. \tag{2.5}$$

This follows from the observation that the variance in the estimate of m is $\sigma_m^2 = \sigma^2/N$, cf. Eq. (2.2). The correction factor of $(N-1)/N$ appears below:

$$\begin{aligned}
s^2_{\text{naive}} &= \frac{1}{N}\sum_i (x_i - m)^2 = \frac{1}{N}\sum_i (x_i - \mu - (m-\mu))^2 \\
&= \frac{1}{N}\sum_i \left[(x_i-\mu)^2 - 2(m-\mu)(x_i-\mu) + (m-\mu)^2\right] \\
&= \left[\frac{1}{N}\sum_i (x_i-\mu)^2\right] - (m-\mu)^2 \\
&= s^2 - s_m^2 = \frac{N-1}{N}s^2.
\end{aligned}$$

which shows that $s^2 = s^2_{\text{naive}} N/(N-1)$ gives an unbiased estimate of σ^2.

In practice one often collects $\sum x_i$ and $\sum x_i^2$ in the simulations. Two useful equations are then

$$s^2 = \frac{1}{N-1}\left(\sum x_i^2 - \frac{1}{N}\left(\sum x_i\right)^2\right), \tag{2.6}$$

and

$$s^2 = \frac{N}{N-1}\left(\left\langle x^2\right\rangle - \langle x\rangle^2\right) \approx \left\langle x^2\right\rangle - \langle x\rangle^2. \tag{2.7}$$

2.3 Error estimates of averages—variance of the mean

We now turn to a different question: What is the precision in our estimated mean? This is necessary as soon as we like to compare one estimate with another. It is only if the difference is much bigger than the estimated uncertainty that we can say that two values are truly different.

Under the common assumption that the obtained averages are from a normal distribution we can also estimate the likelyhood that our average is within $\pm\sigma_A$, $\pm 2\sigma_A$, or $\pm 3\sigma_A$ from the true average μ_A. These figures are

$$\begin{aligned} P(|\langle A\rangle - \mu_A| < \sigma_A) &\approx 68\%, \\ P(|\langle A\rangle - \mu_A| < 2\sigma_A) &\approx 95\%, \\ P(|\langle A\rangle - \mu_A| < 3\sigma_A) &\approx 99.7\%. \end{aligned}$$

It turns out that the precision of an average—and thereby the determinations of s_m^2—are strongly affected by correlations between the data points. We therefore begin with the simplest case of independent data.

2.3.1 Independent data – no correlations

In the absence of correlations between the measurements x_0 through x_{N-1} the estimate of the variance of m is

$$s_m^2 = \frac{1}{N-1}\left(\left\langle x^2\right\rangle - \langle x\rangle^2\right). \tag{2.8}$$

This could e.g. apply to cases like percolation where the random configurations are generated independently of one another. Note that this formula is for the variance of the mean and much as expected this is a quantity that decreases when we have more samples.

2.3.2 Correlated data points

It is very common to have time series with correlations between the successive samples, i.e. $\langle \delta x_i \delta x_{i+j} \rangle \neq 0$, where $\delta x_i = x_i - \mu$. These correlation usually decay with a correlation time τ,

$$r_j = \langle x_i x_{i+j} \rangle - \langle x_i \rangle^2 \sim e^{-j/\tau},$$

which we will return to later.

2.3.3 The blocking method

The common solution to the problem of correlations is to group the N measurements together into N_b blocks with N_s samples each, X_0 through X_{N_b-1},

$$X_j = \frac{1}{N_s} \sum_{i=jN_s}^{(j+1)N_s-1} x_i, \quad j = 0, \ldots, N_b - 1.$$

If N_s is chosen larger than τ ($N_s \gg \tau$) one expects the correlations between successive blocks to vanish and since the values, to a good approximation, are uncorrelated we may again use the simple formula

$$s_m^2 = \frac{1}{N_b - 1} \left(\left\langle X^2 \right\rangle - \langle X \rangle^2 \right).$$

2.4 Generating random variables

We will here mainly be concerned with pseudo random numbers that are generated by some kind of deterministic algorithm and therefore in some sense not are very random at all. They should be distinguished from genuine random numbers that usually are generated from measurements on radiactive substances, see `http://www.fourmilab.ch/hotbits`. It is actually possible to buy CD:s with huge numbers of genuine random numbers that at least historically have had their use within cryptography.

2.4.1 Standard random numbers

The discussion below is very short in spite of the fact that the random number generators is a very important and thorougly studied subject. Incorrect

results are now and then produced even in research papers due to bad quality of the random numbers. The most famous such case is the special purpose processor that was built for performing extensive simulations on the 3D Ising model (see Sec. 4) but produced incorrect results because of the bad built-in random number generator. Furthermore, a paper from 1992 revealed problems with several well-known and often used random number generators. The situation may be summarized with this quote from Donald E. Knuth:

> ... random numbers should not be generated by a method chosen at random.

The basic problem in random number generation is to generate a sequence of random real numbers uniformly distributed in the range $0 \leq \xi < 1$. It is then possible to transform them into random numbers from other distributions.

Most algorithms used to produce random numbers work with integers between 0 and $m-1$ where m is a large number. On a computer with 32 bit integers it is common to choose $m = 2^{31} = 2147483648$ which means that $m - 1 =$ `INT_MAX` $= 2147483647$. The random number generator gives a sequence r_0, r_1, r_2,..., r_n and the standard random numbers are then obtained as $\xi_n = r_n/m$.

The simplest choice is to iterate with a simple function

$$r_n = f(r_{n-1}).$$

Regardless of the choice of function this method will give a cycle of integers that starts to repeat itself as soon as $r_k = r_0$ for some k. The cycle length k can not be larger than the available number of integers, i.e. $k \leq m$. To get a good random number generator one should have a long cycle.

It could seem that the cycle length $k = 2^{32} \approx 10^9$ would never pose a problem. However, considering the study of site percolation on a 256×256 lattice. The generation of each such lattice requires 2^{16} random numbers which means that the random number generator would have gone through its cycle after $2^{32}/2^{16} = 65536$ configurations have been produced. If the program was run longer the same results would be obtained over again and the precision in the results would not improve any further.

The linear congruential generator

A common kind of random number generator is the linear congruential generator,

$$r_n = (a\, r_{n-1} + c) \bmod m$$

where "mod" is the modulus operation, the remainder after a division. To get the longest possible cycle a and c have to be chosen with care. There are however some hidden correlations between the random numbers produced in this way that in some cases might have adverse effects on the results.

Reshuffling algorithm

To wipe out the correlations one can use a reshuffling algorithm. A simple one makes use of an array of N integers j_0 through j_{N-1}. The random number generator is first used both to fill this array with integers and to calculate the value y. The random numbers are then produced by repeating the following steps. Here $\lfloor \ldots \rfloor$ denotes the floor function which is what you automatically get in the computer if all the numbers involved in the operations are integers.

1. calculate an index $k = \lfloor yN/m \rfloor$,

2. set $y = j_k$ and return y as the new random number,

3. assign a new number to j_k from the linear congruential generator.

Lagged Fibonacci generator

The idea here is to generate r_n from a set of older random numbers. The choice

$$r_n = (r_{n-r} + r_{n-s}) \bmod m$$

with $r = 24$ and $s = 55$ is common and will give the random number generator a cycle of at least $2^{55} \approx 10^{16}$. If $m = 2^{32}$ is choosen the modulus operation is made automatically through overflow in the addition. To make this method work one first needs to feed the generator with a vector of numbers produced with another method. A listing of the C code for a Lagged Fibonacci generator is found in Appendix A.1.

2.4.2 Uniformly distributed integers

There are many cases where one wants to generate integers $0 \leq i < i_{\max}$. One such situation is in the simulation of a random walk in d dimensions where one wants to choose one out of $2d-1$ directions by random. A simple way to achieve this is to use

$$\lfloor i_{\max}\, \xi \rfloor$$

but since this involves a conversion back and forth between integer and floating point variables it isn't very efficient. A more efficient method makes use of integer division:

$$\frac{r_n}{r_{\max}/i_{\max}+1},$$

where $r_{\max} = m-1$ is the largest possible value from the random number generator. Note first that the denominator usually may be evaluated already at compilation which means that only a single operation has to be performed at run-time. Second, note that the "+1" in the denominator is necessary to get random numbers $< i_{\max}$. Without the "+1" the random number may be $= i_{\max}$ which could well be disastrous. With the large $r_{\max}$ normally used in simulations this will not happen very often and could therefore be an error that is difficult to find.

2.4.3 Integers from arbitrary distributions

The task is here to sample from an arbitrary distribution of discrete values $x^{(i)}$ with probabilities $p(x^{(i)})$. One way to solve it is through the following recipe:

> Generate ξ. Choose the value $x^{(k)}$ where k is the smallest integer such that $P(x^{(k)}) \equiv \sum_{i=1}^{k} p(x^{(i)}) \geq \xi$.

2.4.4 Method of inversion

The recipe described above may also be used for certain continuous distributions:

> Generate ξ, then choose the value x such that $P(x) \equiv \int_{-\infty}^{x} dx' p(x') = \xi$.

This method relies on the possibility to integrate and to find the inverse function. As an example consider sampling from the exponential distribution,

$$p(x) = \frac{1}{\mu} e^{-x/\mu}, \quad x \geq 0.$$

We then have

$$P(x) = \frac{1}{\mu} \int_0^x dx' e^{-x'/\mu} = 1 - e^{-x/\mu},$$

and the requirement that $P(x) = \xi$ leads to

$$e^{-x/\mu} = 1 - \xi \quad \Rightarrow \quad x = -\mu \ln(1 - \xi).$$

Since the distribution of $1 - \xi$ is the same as the distribution of ξ, this may be simplified one step further to

$$x = -\mu \ln \xi. \tag{2.9}$$

2.4.5 Generate Gaussian random numbers

The Gaussion distribution is commonly used and there is a nice method – the Box Müller method – that produces random numbers from the Gaussian distribution. Note first that the inversion method may not be used since the integral $P(x) = \int_{-\infty}^{x} dx' e^{-x'^2/2}$ cannot be expressed in terms of elementary functions.

The Box-Müller method makes use of two random numbers ξ_1 and ξ_2 to produce two independent variables x and y from the Gaussian distribution,

$$\begin{aligned} p(x) &= \frac{1}{\sqrt{2\pi}} e^{-x^2/2}, \\ p(y) &= \frac{1}{\sqrt{2\pi}} e^{-y^2/2}. \end{aligned}$$

Changing variables to r and θ with $x = r\cos\theta$, and $y = r\sin\theta$ gives

$$p(x)dx\ p(y)dy = \frac{1}{2\pi} e^{-(x^2+y^2)/2} dx\ dy = \frac{d\theta}{2\pi} e^{-r^2/2} r dr \equiv p(\theta) d\theta p(r) dr.$$

We may then get an expression for $P(r)$,

$$P(r) = \int_0^r dr'\ r' e^{-r'^2/2} = \left[-e^{-r'^2/2} \right]_0^r = 1 - e^{-r^2/2},$$

and the method of inversion gives

$$\begin{aligned} r &= \sqrt{-2\ln\xi_1}, \\ \theta &= 2\pi\xi_2, \end{aligned}$$

from which x and y may be calculated directly.

2.4.6 The acceptance-rejection method

In many cases none of the above methods are directly applicable and one can then often make use of von Neumann acceptance-rejection method. Suppose that we want random numbers from the distribution $p(x)$ that is rather similar to the distribution $p'(x)$. Write

$$p(x) = CQ(x)p'(x).$$

where the constant C is chosen such that the maximum value of Q is equal to unity. The idea is then to first generate a random variable x, from a random number ξ_1, and then accept that variable with the probability $Q(x)$. That is, accept if $Q(x) < \xi_2$, reject and try over again otherwise. The efficiency factor is the inverse of C.

A different version of the acceptance-rejection method may be used to generate points uniformly within the unit circle: The idea is to first generate points uniformly within the square given by $-1 < x < 1$ and $-1 < y < 1$ and then reject the points that are not within the unit circle, i.e. all points for which $x^2 + y^2 < 1$. The accepted points will then be uniformly distributed within the unit circle.

2.5 Sampling a distribution with Markov chains

Markov chains may be used to sample *multidimensional variates* and are often used when the normalization constant is unknown, i.e.

$$\pi(\mathbf{x}) = Cf(\mathbf{x}).$$

One common example which will be our main use of MCMC is statistical mechanics, where the probability distribution is proportional to the Boltzmann

factor,

$$\pi(\mathbf{x}) = \frac{1}{Z} e^{-\beta E(\mathbf{x})},$$

and the partition function Z is unknown.

We will first describe a "Markov chain" in general terms and will then second consider how we should construct a Markov chain to give $\mathbf{x}$ from a certain probability distribution.

2.5.1 Markov chain

Suppose that there is a finite number of possible configurations or multidimensional variables $\mathbf{x}^{(i)}$. A Markov chain is a random chain of these variables, $\mathbf{x}_1$, $\mathbf{x}_2$,... produced by means of a transition matrix p_{ij} with the following properties:

1. $p_{ij} \geq 0$,
2. $p_{ii} \neq 1$,
3. $\sum_j p_{ij} = 1$ for all i.

Note that (1) is necessary since probabilities have to be non-negative, (2) means that the chain may never come to a halt, and (3) means that the total probability to go to some state must always be equal to unity.

If the system is left to perform a random walk according to these transition probabilities this will lead to a certain probability distibution $\pi_i \equiv \pi(\mathbf{x}^{(i)})$. We have here for simplicity assumed a discrete set of possible configurations but Markov chains may easily be defined for the more general case of a continuous configuration space.

2.5.2 Construct the transition matrix

The task is now to choose the transition matrix for the Markov chain such that the desired probability distribution $\pi(\mathbf{x})$ is obtained. A sufficient (but not necessary) condition is to require *detailed balance*, i.e.

$$\pi_i p_{ij} = \pi_j p_{ji}.$$

That this gives the required probability distribution may be seen by considering the probability to be in state j after a given step which is

$$\sum_i \pi_i p_{ij} = \pi_j \sum_i p_{ji} = \pi_j,$$

where we have made use of the detailed balance condition followed by condition (3) above. The transition probability usually consists of two parts. We write

$$p_{ij} = q_{ij}\alpha_{ij},$$

where

$$\begin{aligned} q_{ij} &= \text{probability for the transition to be suggested} \\ \alpha_{ij} &= \text{probability for the transition to be accepted} \end{aligned}$$

Detailed balance may e.g. be fulfilled with the following choice for α_{ij}:

$$\alpha_{ij} = \min\left(1, \frac{\pi_j q_{ji}}{\pi_i q_{ij}}\right) \tag{2.10}$$

The Metropolis algorithm

One often has a symmetric targeting probability $q_{ij} = q_{ji}$. The acceptance probability then simplifies to

$$\alpha_{ij} = \min(1, \pi_j/\pi_i), \quad i \neq j, \tag{2.11}$$

which is the choice in the *Metropolis* algorithm[1].

2.5.3 Correlations

As already mentioned the samples generated by the Markov chain will be correlated to one another. This may be described by the time correlation function which essentially measures how fluctuations from the average at a certain time affect the same quantity a time t later. In terms of

$$\delta A(t) = A(t) - \langle A \rangle ,$$

the time correlation function for the quantity A is

$$C_A(t) = \langle \delta A(t')\delta A(t'+t) \rangle . \tag{2.12}$$

The correlation time

The correlations usually decay exponentially. To see this, consider a simple model where the dynamics at each time step contains both a memory and a random part, r:

$$\delta A(t'+1) = a\ \delta A(t') + (1-a)r.$$

After t units of time (neglecting the random contribution that will not contribute to the average), this gives

$$\delta A(t'+t) = a^t\ \delta A(t'),$$

and, after multiplying by $\delta A(t')$ and identifying $a = e^{-1/\tau}$, we get

$$\delta A(t'+t)\delta A(t') = e^{-t/\tau}\delta A(t')\delta A(t').$$

Since $\langle(\delta A(t'))^2\rangle$ is a constant we expect

$$C_A(t) \sim e^{-t/\tau}. \tag{2.13}$$

For multidimensional variables $\mathbf{x}^{(i)}$ and functions of these variables $A(\mathbf{x}^{(i)})$ this behavior is often only seen for reasonably large t, say $t > \tau$. The correlation time may also be different for different observables.

Chapter 3

The Lennard-Jones gas

The point with this chapter is to introduce Monte Carlo simulations of a classical gas. To that end we first shortly review the different ensembles in Sec. 3.1. In Sec. 3.2 we then show that the equations in the preceding section may be understood by considering the different free energies as Legendre transformations of the *entropy*. Sec. 3.3 is devoted to a short comparison of molecular dynamics and Monte Carlo simulations. In Sec. 3.4 we consider the steps necessary to go from the quantum formulation of statistical mechanics (with a number of discrete quantum states) to a classical formulation where each microscopic state is defined through the positions and the momenta of all the particles. In that section we also give the simple recipe for a Monte Carlo simulation in a gas of interacting particles. The last two sections are devoted to the additional steps needed to include volume fluctuations (Sec. 3.6), which are needed to do simulations at constant pressure, and the creation and annihilation of particles (Sec. 3.7), needed for simulations in the grand canonical ensemble, where the number particles is not a constant.

3.1 Summary of different ensembles

3.1.1 Microcanonical ensemble

The basic assumption behind equilibrium statistical physics is that all states with the same energy are equally likely to show up. In a system where we can control the entropy S (and thereby the heat transfer), the volume V, and

the number of particles N, the system is described by the internal energy

$$E(S, V, N).$$

Infinitesimal changes of the control variables then change E according to

$$dE = TdS - pdV + \mu dN.$$

In statistical mechanics we often turn it the other way around and consider the entropy as a function of the energy, $S(E, V, N)$. The entropy is related to the number of states with a certain energy,

$$\Omega(E, V, N) = \sum_{\nu} \delta(E - E_\nu),$$

(where $\delta(x) = 1$ if $x = 0$; otherwise $\delta(x) = 0$) through

$$S(E, V, N) = k_B \ln \Omega(E, V, N). \tag{3.1}$$

3.1.2 Canonical ensemble

In the common situation where our system is kept at a given temperature the entropy as well as the energy will fluctuate and it is more convenient to consider a quantity that is a function of the temperature (instead of a function of the entropy). The Helmholtz free energy is given by the Legendre transform,

$$F(T, V, N) = E(S, V, N) - TS,$$

and with $d(TS) = TdS + SdT$ the differential becomes

$$dF = -SdT - pdV + \mu dN.$$

The probability for a configuration to show up is proportional to the Boltzmann factor, $e^{-\beta E_\nu}$, where $\beta = 1/(k_B T)$, and a basic quantity is the partition function

$$Z(T, V, N) = \sum_{\nu} e^{-\beta E_\nu}. \tag{3.2}$$

This quantity is more than just a normalization constant and the relation to the Helmholtz free energy is

$$F(T, V, N) = -k_B T \ln Z(T, V, N). \tag{3.3}$$

The expectation value of an observable A may now be written

$$\langle A \rangle = \sum_\nu P_\nu A_\nu = \frac{1}{Z} \sum_\nu A_\nu e^{-\beta E_\nu}, \tag{3.4}$$

and as a special case the average energy is

$$\langle E \rangle = \frac{1}{Z} \sum_\nu E_\nu e^{-\beta E_\nu} = -\frac{1}{Z} \frac{\partial Z}{\partial \beta} = -\frac{\partial \ln Z}{\partial \beta}. \tag{3.5}$$

The heat capacity may be determined from the fluctuations in the energy. Using $dT = -k_B T^2 d\beta$ we get

$$\begin{aligned} C = \frac{\partial \langle E \rangle}{\partial T} &= -\frac{1}{k_B T^2} \frac{\partial \langle E \rangle}{\partial \beta} = \frac{1}{k_B T^2} \frac{\partial}{\partial \beta} \left(\frac{1}{Z} \frac{\partial Z}{\partial \beta} \right) \\ &= \frac{1}{k_B T^2} \left[\frac{1}{Z} \frac{\partial^2 Z}{\partial \beta^2} - \left(\frac{1}{Z} \frac{\partial Z}{\partial \beta} \right)^2 \right] = \frac{1}{k_B T^2} \left(\left\langle E^2 \right\rangle - \langle E \rangle^2 \right) \end{aligned} \tag{3.6}$$

3.1.3 Gibbs ensemble—constant pressure

The common experimental situation is that the pressure is constant but that the volume changes as e.g. the temperature is changed. [1] We now make a Legendre transformation from the usual free energy $F(T, V, N)$ to the Gibbs free energy,

$$G(T, p, N) = F(T, V, N) + pV,$$

with the differential

$$dG = -SdT + Vdp + \mu dN.$$

The generalization of the partition function is

$$\Phi(T, p, N) = \sum_V e^{-\beta p V} Z(T, V, N),$$

and the relation to the thermodynamic function is $G = -k_B T \ln \Phi$.

[1] In some cases this is preferred in the simulations as well and it is actually rather easy to generalize the MC simulations to allow for a fluctuating volume.

3.1.4 Grand canonical ensemble

The grand canonical ensemble is useful in simulations with a given chemical potential and a fluctuating number of particles. For the corresponding free energy we have

$$F_G(T, V, \mu) = F(T, V, N) + \mu N,$$

the differential is

$$dF_G = -SdT - pdV + Nd\mu,$$

and $F_G = k_B T \ln \Xi$,

$$\Xi(T, V, \mu) = \sum_N e^{\beta\mu N} Z(T, V, N).$$

3.2 Gibbs entropy formula

In Sec. 3.1 we introduced the internal energy and Legendre transformations to a number of different free energies. We also gave equations for the generalizations of the partition function and the free energy in different ensembles. The connection between all these ensembles may be clearer by instead considering the different free energies as Legendre transforms of the entropy. A good starting point is the Gibbs entropy formula which is valid in all ensembles,

$$S = -k_B \sum_\nu P_\nu \ln P_\nu. \tag{3.7}$$

This equation is—except for the presence of k_B—the same as the expression for the Shannon entropy, which plays a central role in information theory.

In the **microcanonical ensemble** we have

$$\Omega(E, V, N) = \sum_\nu \delta_{E-E_\nu} \delta_{V-V_\nu} \delta_{N-N_\nu},$$

and since $P_\nu = 1/\Omega(E, N, V)$, the Gibbs entropy formula Eq. (3.7), gives

$$S(E, N, V) = -k_B \sum_\nu (1/\Omega) \ln(1/\Omega) = k_B \ln \Omega,$$

in accordance with Eq. (3.1). The microcanonical ensemble may be summarized through

$$e^{S(E,N,V)/k_B} = \Omega(E, V, N) = \sum_\nu \delta_{E-E_\nu} \delta_{V-V_\nu} \delta_{N-N_\nu}. \tag{3.8}$$

In the **canonical ensemble** the dimensionless Helmholtz free energy, βF, may be considered a Legendre transformation of S/k_B,

$$-\beta F = S/k_B - \beta E, \tag{3.9}$$

We will now show that this follows from the application of the Gibbs entropy formula Eq. (3.7), to the Boltzmann factor, $P_\nu = \frac{1}{Z} e^{-\beta E_\nu}$ together with Eq. (3.3):

$$\begin{aligned} S/k_B &= -\frac{1}{Z}\sum_\nu e^{-\beta E_\nu}\left[-\beta E_\nu - \ln Z\right] \\ &= \beta\left\langle E\right\rangle + \ln Z = \beta\left\langle E\right\rangle - \beta F. \end{aligned}$$

In the thermodynamic limit we may identify $\langle E\rangle$ with E and this is then in agreement with Eq. (3.9). We write (compare with Eq. (3.8))

$$Z(\beta,V,N) = e^{-\beta F(\beta,V,N)} = \sum_E \Omega(E,V,N) e^{-\beta E} = \sum_\nu e^{-\beta E_\nu}\delta_{V-V_\nu}\delta_{N-N_\nu}. \tag{3.10}$$

Likewise the free energy in the **Gibbs ensemble** is

$$-\beta G = -\beta F - \beta pV = S/k_B - \beta E - \beta pV, \tag{3.11}$$

and we have

$$\Phi(\beta,p,N) = e^{-\beta G(\beta,p,N)} = \sum_V Z(\beta,V,N) e^{-\beta p V_\nu} = \sum_\nu e^{-\beta E_\nu - \beta p V_\nu}\delta_{N-N_\nu}.$$

Note the similarity between the exponents in this expression and the Legendre transform, Eq. (3.11).

3.3 Monte Carlo versus Molecular Dynamics

A straightforward way to simulate a classical gas is Molecular Dynamics (MD) where one simply integrates the equations of motion. The state of the system is then specified by both position $\mathbf{r}_i$ and velocity $\mathbf{v}_i$ (or momentum) for each particle. The equations of motion are

$$\begin{aligned} \dot{\mathbf{r}}_i &= \mathbf{v}_i, \\ m\dot{\mathbf{v}}_i &= \sum_{j\neq i} \mathbf{F}_{ij}. \end{aligned}$$

There are efficient techniques that makes it possible to perform this kind of integrations with high precision, but with the drawback that the simulations are rather time consuming. The reasons are, first that there are lots of calculations to be done for each time step and, second, that a small time step is often needed to get reliable results.

When the goal is only to determine static (time-independent) quantities, Monte Carlo simulations is a simpler and more efficient alternative. The simulation may even in this case be thought of as a kind of dynamics in the many-dimensional configurational space, but since (1) the configuration is specified with only the position coordinates $\mathbf{r}_i$, and (2) we don't need a small time step (since we don't try to mimic a realistic dynamics), the simulation program becomes both considerably simpler and faster.

In contrast to MD where the dynamics is deterministic, the steps in a Monte Carlo simulation are entirely stochastic. One uses the random number generator first to suggest the change to the configuration and second to accept or reject this change with a probability that (in most cases) only depends on the temperature and the change in energy. Such a simulation is therefore an implementation of a Markov chain, discussed in Sec. 2.5.

Monte Carlo simulations are commonly performed in the canonical ensemble where the temperature is a controlling parameter and the configurational energy fluctuates in the simulation. The average energy is then a result from the simulation. Molecular Dynamics on the other hand is done in the microcanonical ensemble where the *total energy* is specified in the starting configuration and the temperature is measured from the average kinetic energy, $m \langle v^2 \rangle /2$. [2]

3.4 Classical statistical mechanics

As discussed above the properties of a system at a fixed temperature T may be determined from the partition function

$$Z = \sum_{\nu} e^{-\beta E_\nu}.$$

[2] These two cases are however not altogether different since the *configurational energy* actually varies in Molecular Dynamics as well, since the kinetic degrees of freedom act as a (finite) heat bath that couples to the configurational degrees of freedom. However, in MD the temperature of the heat bath is not specified but has to be measured during the simulation.

The index ν serves to specify the quantum state. Since the set of states is discrete, the partition function may be written as a sum.

For many purposes one can instead use classical statistical mechanics. Each microscopic state is then a point in phase space,

$$(\mathbf{r}_1, \mathbf{r}_2, \ldots \mathbf{r}_N; \mathbf{p}_1, \mathbf{p}_2, \ldots \mathbf{p}_N) \equiv (r^N, p^N).$$

The energy associated with a point in phase space is a sum of the kinetic and the potential energies,

$$\mathcal{H}(r^N, p^N) = K(p^N) + U(r^N),$$

where the kinetic energy is usually given by $K(p^N) = \frac{1}{2m}\sum_i \mathbf{p}_i^2$. The potential energy may take on many different forms; one common approximation will be discussed shortly.

With phase space given by a continuum, the sum over ν instead becomes a many-dimensional integral over r^N and p^N,

$$Z = \frac{1}{N!h^{3N}} \int dr^N \int dp^N e^{-\beta \mathcal{H}(r^N, p^N)}.$$

The prefactor comes from a comparison with a quantum formulation of an ideal gas. The division by $N!$ makes sense if we recall that we are dealing with identical particles. This means that the same state will be generated $N!$ times by the integral and that is compensated for by the division by $N!$.

For the momentum part of the integral we have

$$\int dp^N e^{-(\beta/2m)\sum_i \mathbf{p}_i^2} = \left(\int dp\, e^{-(\beta/2m)p^2} \right)^{3N} = \left(\sqrt{2\pi m/\beta} \right)^{3N},$$

and with

$$v_Q = \left(h/\sqrt{2\pi m k_B T} \right)^3, \tag{3.12}$$

we are left with

$$Z = \frac{1}{N!} \frac{1}{v_Q^N} \int dr^N e^{-\beta U(r^N)}. \tag{3.13}$$

3.4.1 To calculate expectation values

For a set of discrete states we have Eq. (3.4). The analogue for a continuous set of states is

$$\langle A \rangle = \frac{1}{Z} \frac{1}{N!} \frac{1}{v_Q^N} \int dr^N A(r^N) e^{-\beta U(r^N)}.$$

The calculation of such averages (for simplicity we stick to the quantum notation) may now in principle be done in two ways:

- Generate the configurations r^N randomly, with the $\mathbf{r}_i$ independent of one another. The expectation value is then

$$\langle A \rangle = \frac{\sum_{\nu=1}^{n} A_\nu e^{-\beta E_\nu}}{\sum_{\nu=1}^{n} e^{-\beta E_\nu}}, \tag{3.14}$$

  where "$\sum_\nu$" now denotes a summation over n different random configurations. However, if one attempts to do this, one will find that most configurations have a rather large energy and therefore a very small Boltzmann factor. It will therefore be impossible to calculate the properties at most temperatures of interest. Except for very small systems this is therefore an entirely impracticable method.

- Use some valid dynamics or Markov Chain Monte Carlo to generate the configurations with a probability $\pi_\nu \propto e^{-\beta E_\nu}$. The expectation values should then be calculated as simple averages,

$$\langle A \rangle = \frac{1}{n} \sum_{\nu=1}^{n} A_\nu. \tag{3.15}$$

  This is a useful method that allows for a simple calculation of many quantities that would otherwise be very difficult—if not altogether impossible—to calculate. The only drawback is that the configurations, and thereby the A_ν, are not independent of one another.

3.4.2 Pair interaction

To get further we also need to specify the configurational part of the energy. We then approximate the interaction energy by a sum of pair interactions:

$$U(r^N) = \sum_i \sum_{j>i} u(|\mathbf{r}_i - \mathbf{r}_j|).$$

The pair interaction is attractive, $u(r) \sim -r^{-6}$, at large distances and repulsive at short distances. A common choice is to let the short distance repulsion be $\sim r^{-12}$, which gives the Lennard-Jones potential

$$u_{\mathrm{LJ}}(r) = 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right].$$

The minimum of the potential is

$$u_{\mathrm{LJ}}(r_{\min}) = -\epsilon, \quad r_{\min} = 2^{1/6}\sigma.$$

3.4.3 Expression for the pressure – the virial theorem

In a simulation with molecular dynamics it is possible to calculate the pressure from the change of momentum the particles get when they bounce off the walls. In a Monte Carlo simulation this isn't possible and one instead has to determine the pressure from a kind of correlation function by means of the virial theorem. Note that this equation is only valid in ensembles with a fixed volume. This follows since the starting point for the derivation is the differential for the Helmholtz free energy.

The starting point is the differential of the free energy,

$$dF = -SdT - pdV + \mu dN,$$

which gives

$$p = -\left(\frac{\partial F}{\partial V}\right)_{N,T}.$$

Since $\beta F = -\ln Z$ we get

$$\beta p = \left(\frac{\partial \ln Z}{\partial V}\right)_{N,\beta}.$$

Because the Hamiltonian separates into two parts we may write

$$Z = Z_{\mathrm{kin}} Z_{\mathrm{conf}},$$

where the kinetic part is independent of volume and is equal to $Z_{\mathrm{kin}} = 1/v_Q^N$ and

$$Z_{\mathrm{conf}} = \frac{1}{N!}\int dr^N e^{-\beta U(r^N)},$$

where the integration for each $\mathbf{r}_i$ should be performed over the volume V. Since only the configurational part depends on V, the pressure becomes

$$\beta p = \frac{\partial}{\partial V} \ln\left[\int dr^N e^{-\beta U(r^N)}\right].$$

This differentiation is tricky since the limits of integration depend on V. To get around this, change coordinates to

$$\mathbf{x} = \mathbf{r}/V^{1/d} \quad \Rightarrow \quad dr^N = V^N dx^N.$$

We then get

$$\begin{aligned}
\beta p &= \frac{\partial}{\partial V} \ln \left[V^N \int dx^N e^{-\beta \sum_{i<j} u(V^{1/d} x_{ij})} \right] \\
&= \frac{d}{dV} \ln V^N + \frac{\partial}{\partial V} \ln \left[\int dx^N e^{-\beta U} \right] \\
&= \frac{N}{V} + \left[\int dx^N \left(-\beta \frac{\partial U}{\partial V} \right) e^{-\beta U} \right] \Big/ \left[\int dx^N e^{-\beta U} \right].
\end{aligned}$$

(The integrations are here understood to be over a unit volume.) We also have

$$\frac{\partial U}{\partial V} = \sum_{i<j} \frac{\partial}{\partial V} u(V^{1/d} x_{ij}) = \sum_{i<j} \frac{\partial V^{1/d}}{\partial V} x_{ij} u'(V^{1/d} x_{ij}) = \frac{1}{Vd} \sum_{i<j} r_{ij} u'(r_{ij})$$

which follows since

$$\frac{\partial V^{1/d} x_{ij}}{\partial V} = \frac{1}{d} V^{1/d-1} x_{ij} = \frac{1}{Vd} V^{1/d} x_{ij} = \frac{1}{Vd} r_{ij}$$

Putting things together we get the virial theorem

$$\beta p = \frac{N}{V} - \frac{\beta}{Vd} \left\langle \sum_i \sum_{j>i} r_{ij} u'(r_{ij}) \right\rangle.$$

By doing the above a little bit more carefully with $\frac{d}{dc} u(c\mathbf{x}) = \mathbf{x} \cdot \nabla u(c\mathbf{x})$, $\mathbf{F} = -\nabla u$, and finally multiplying by $k_B T V$, the expression becomes

$$pV = N k_B T + \frac{1}{d} \left\langle \sum_i \sum_{j>i} \mathbf{r}_{ij} \cdot \mathbf{F}_{ij}) \right\rangle. \tag{3.16}$$

Note that this simplifies to the ideal gas law in the absence of interactions, $\mathbf{F}_{ij} = 0$.

3.5 Recipe for a Monte Carlo simulation

The Monte Carlo simulation of a classical gas of particles is our first implementation of Markov Chain Monte Carlo (MCMC), discussed in Sec. 2.5. The standard implementation uses the Metropolis algorithm[1], Eq. (2.11).

0. Initialize by generating N positions $\mathbf{r}_i$ by random.
1. Set $i = 0$.
2. Let ν denote the present configuration. Suggest a change $\mathbf{r}_i \to \mathbf{r}_i + \Delta\mathbf{r}$; denote this trial configuration by ν'. Here $\Delta\mathbf{r}$ is a random vector where each component is usually from a uniform distribution, $-\epsilon \leq \Delta r_\mu \leq \epsilon$.
3. Calculate the energy difference $U_{\nu'} - U_\nu$ and
$$\alpha_{\nu\to\nu'} = \min\left(\frac{\pi_{\nu'}}{\pi_\nu}, 1\right) = \min\left(e^{-\beta(U_{\nu'}-U_\nu)}, 1\right),$$
and accept the new configuration with probability α.
4. $i \to i + 1$; if $i < N$ goto 2.

Points 1–4 are often called a Monte Carlo sweep. Any serious MC simulation should consist of many (say 10^6) such sweeps, first for thermalization and then to collect data.

The parameter ϵ in "2." is not specified and one is actually free to choose it at will. The efficiency of the simulation will depend on this choice. A very big ϵ will lead to a large fraction of big moves that will often be rejected. A small ϵ, on the other hand, will give a high acceptance ratio but since the particles move very short distances the configurations will change very slowly. To get a high efficiency one tries to choose ϵ such that the acceptance ratio is not far from 50%.

3.6 Ensemble with a fluctuating volume

To use a constant pressure and a fluctuating volume, the starting point is the thermodynamic function, $G(T, p, N) = -k_B T \ln \Phi(T, p, N)$ where

$$\Phi = \int_0^\infty dV e^{-\beta pV}\ Z(T, V, N) = \int_0^\infty dV\ e^{-\beta pV} \frac{V^N}{N! v_Q^N} \int dx^N e^{-\beta U(V;x^N)}.$$

The probability density for a certain configuration, $(V, x_1, \ldots, x_N)$, specified by volume (or length L) and the relative position coordinates $\mathbf{x}_i = \mathbf{r}_i/L$, is then

$$\pi(V, \mathbf{x}_1, \ldots, \mathbf{x}_N) = \frac{V^N e^{-\beta p V} e^{-\beta U(V;x^N)}}{\Phi}.$$

The calculation of the acceptance probability when *the position* of an atom changes is the same as before, but for *changing the volume* from V_{old} to V_{new} the acceptance probability should be calculated as $\min(\pi_{\text{new}}/\pi_{\text{old}}, 1)$ where

$$\frac{\pi_{\text{new}}}{\pi_{\text{old}}} = \left(\frac{V_{\text{new}}}{V_{\text{old}}}\right)^N \exp\left(-\beta p[V_{\text{new}} - V_{\text{old}}] - \beta[U(L_{\text{new}}, x^N) - U(L_{\text{old}}, x^N)]\right).$$

We again stress that the virial theorem cannot be used to calculate the pressure in an ensemble with constant pressure, since the starting point in that derivation is the differential of the Helmholtz free energy $F(T, V, N)$.

3.7 The grand canonical ensemble

One more difference between Monte Carlo and molecular dynamics is the possibility to do the simulation with a varying number of particles. One then uses the grand canonical ensemble with the chemical potential μ as a parameter. The grand sum is

$$\Xi = \sum_{N=0}^{\infty} e^{\beta\mu N} Z(T, V, N) = \sum_{N=0}^{\infty} \frac{e^{\beta\mu N}}{N! v_Q^N} \int dr^N e^{-\beta U(N, r^N)}.$$

The acceptance ratio for moving a particle is the same as in the canonical ensemble, but the steps to create or annihilate particles require a non-trivial derivation. Consider *creation* that takes us from N to $N+1$ particles and *annihilation* that reduces the number of particles from $N+1$ to N. The basic requirement of detailed balance is

$$\pi_i q_{ij} \alpha_{ij} = \pi_j q_{ji} \alpha_{ji},$$

where π is the desired probability density, q_{ij} is the trial probability, and α_{ij} is the acceptance probability. Let 'i' stand for a state with N particles and 'j' a state with $N+1$ particles. For the probability densities we then have

$$\pi_i \;=\; \frac{e^{\beta\mu N}}{N!\; v_Q^N} e^{-\beta U(N)} dr^N$$

$$\pi_j = \frac{e^{\beta\mu(N+1)}}{(N+1)!\, v_Q^{N+1}} e^{-\beta U(N+1)} dr^{N+1},$$

whereas the trial probabilities are

$$\begin{aligned} q_{ij} &= \frac{d\mathbf{r}}{V(N+1)}, \\ q_{ji} &= \frac{1}{N+1}, \end{aligned}$$

where we have also considered the probability that the created or annihilated particle has a certain index out of the $N+1$ possible. From Eq. (2.10) the acceptance probabilities should be

$$\begin{aligned} \alpha_{ij} &= \min\left(1, \frac{\pi_j q_{ji}}{\pi_i q_{ij}}\right) = \min\left(1, \frac{V\, e^{\beta\mu}}{(N+1)\, v_Q} e^{-\beta[U(N+1)-U(N)]}\right), \\ \alpha_{ji} &= \min\left(1, \frac{\pi_i q_{ij}}{\pi_j q_{ji}}\right) = \min\left(1, \frac{(N+1)\, v_Q}{V\, e^{\beta\mu}} e^{-\beta[U(N)-U(N+1)]}\right). \end{aligned}$$

Chapter 4

The Ising model

The kind of phenomenon we like to model with the Ising model is the behavior of a ferromagnet. In a ferromagnet the spontaneous magnetization M decreases with increasing temperature and vanishes at the Curie temperature as

$$M \sim (T_c - T)^{\beta}. \tag{4.1}$$

For many different magnets one finds $\beta \approx 0.3$.

Beside being a model for a ferromagnet the Ising model has been extremely important for the developement of our understanding of critical phenomena. The Ising model has been called the Drosophila[1] of statistical mechanics because it has been so thoroughly studied and manipulated in all conceivable ways.

4.1 Lattices

Many of the models we will discuss live on a lattice. Even though there are many different kinds of lattices we will usually do well with the simplest kind that in one dimension (1D) is a set of points along (say) the x axis with unit distance. This concept easily generalizes to higher dimensions: the square lattice (2D) contains all points (i, j) with i and j integers in the x-y plane and the cubic lattice – as well as higher dimensionalities – are obvious generalizations.

[1] A fruit fly that has been thoroughly studied in genetics by innumerable experiments with different kinds of mutations.

In numerical calculations the mathematical objects with an infinitude of points are not so relevant. We instead need the finite lattices of size L in all directions. We will throughout use the C-style convention to let the indices go from 0 up to $L-1$. The number of points in a d-dimensional lattice is $N = L^d$. Another important quantity is the coordination number that is the number of nearest neighbors to each site, $z = 2d$.

Associated with the lattice is the way to treat the boundaries and the simplest and best way is to use periodic boundary conditions. In 1D this means that we connect the beginning and the end of the line of sites to make a closed circle. In higher dimensions the opposite edges (edges in 2D, planes in 3D) are considered connected to one another and there is then nothing special with a site at a boundary; all sites have an identical surrounding with the same number of nearest neighbors.

Since quantites determined from simulations often have a dependence on the system size, the subscript L will often be used to stress this fact without further notice.

4.2 Definition of the model

The Ising model is the simplest possible model of a ferromagnet. (To be more precise it models a magnet with uniaxial symmetry.) The basic building blocks are called "spins" even though they rather correspond to the domains in a ferromagnet with different direction of magnetization. To define the model we have to specify (1) the properties of the individual degrees of freedom (2) their spatial organization, and (3) an expression for the energy.

1. A set of spins s_i with two different states, $s = \uparrow, \downarrow$, or in a convenient mathematical language, $s = +1, -1$.

2. A lattice that specifies the position of the spins. This may e.g. be a 1D lattice, a 2D square lattice, a 2D triangular lattice or various kinds of lattices in 3D, e.g. the simple cubic.

3. The expression for the Hamiltonian is

$$\mathcal{H}(\{s_i\}) = -J \sum_{\langle ij \rangle} s_i s_j - h \sum_i s_i, \tag{4.2}$$

   Here the notation "$\{s_i\}$" means "$s_1, \ldots, s_N$" and

- J is a coupling constant with the dimension of energy,
- $\sum_{\langle ij \rangle}$ denotes a sum over all pairs i, j that are nearest neighbors,[2]
- h is the magnetic field (with the coupling constant between the field and the spins absorbed in h).

Note that we are usually interested in the behavior of the model in the limit of an infinite system, $N \to \infty$. One reason is that the physical systems, that we ultimately want to compare with, are made up of a huge number of particles or degrees of freedom. Another reason is that the phase transition at a critical point is only perfectly sharp and well-defined in the limit $N \to \infty$.

4.2.1 Basic properties

We will in the following usually be interested in the behavior of the model at zero magnetic field, $h = 0$. From the Hamiltonian we then have the following values for the energy at the link between spins s_i and s_j,

s_i	s_j	$\mathcal{H}_{ij}$
$\uparrow$	$\uparrow$	$-J$
$\downarrow$	$\downarrow$	$-J$
$\uparrow$	$\downarrow$	$+J$
$\downarrow$	$\uparrow$	$+J$

It is difficult to determine the behavior at general temperatures. In the present model it is however easy to determine the properties in the limits $T \to 0$ and $T \to \infty$.

The low-temperature limit

At zero temperature, $\beta \to \infty$, the dominating term in the partition function is from the configuration with lowest energy, the ground state. In the Ising model the ground state is doubly degenerate since there are two states with lowest energy; the states with only spin up or only spin down. With the coordination number z the number of nearest neighbors for each spin, the total number of nearest neighbor pairs is $Nz/2$. The ground state energy is then $E_0 = -NzJ/2$. Note that the ground states are perfectly ordered and each have maximum magnetization.

[2]This notation is unfortunately similar to expectation values, but because of the exclusive use of this notation in summations, the meaning is usually clear from the context.

The high-temperature limit

At infinite temperature, $\beta \to 0$, we have $e^{-\beta E_\nu} = 1$ which means that all configurations are equally probable. We then expect $\langle E \rangle = 0$ and a system that is totally disordered.

Three possibilities

With perfect order at $T = 0$ and zero order at $T \to \infty$ there are three possibilities for how the "order" can depend on temperature. These are illustrated in Fig. 4.1. Of these three possibilities it turns out that the last one, $T_c = \infty$ is never realized but, depending on the dimensionality of the system, one either has $T_c = 0$ or a finite value of T_c.

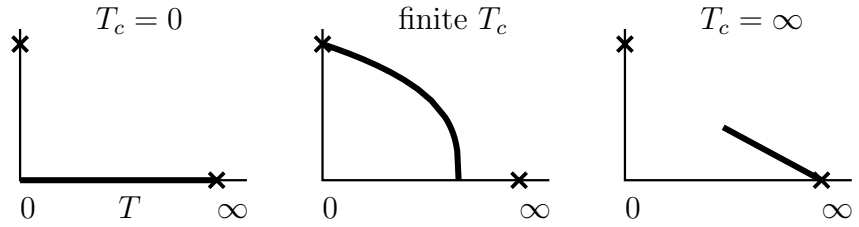


Figure 4.1: The three panels show the three possibilities for how the "order" may vanish as the temperature is increased. The first two possibilities are realized in the Ising model in 1D and higher dimensionality, respectively.

4.3 Monte Carlo simulations for the Ising model

We will now discuss the implementation of Monte Carlo for the 2D Ising model but before doing that we will first consider calculating properties (as e.g. the average energy) of the Ising model. Before jumping into Markov chain Monte Carlo we will discuss some methods that at first could look like possible alternatives but really are dead ends unless the system size is very limited. (A similar discussion was presented in the context of a simulation of classical particles in Sec. 3.4.1.)

4.3.1 Some dead ends

Complete summation The simplest approach is to try and perform the sum over all states on the computer. For all but the smallest sizes we however quickly run into difficulties since the number of different configurations increases very rapidly with the system size. We may well handle $L = 6$ that gives $N = 36$ and $2^{36} \approx 7 \times 10^{10}$ different configurations, but already with $L = 8$ such a straighforward summation becomes completely hopeless because of the more than 10^{19} different configurations.

Randomly generated configurations With the impossibility of performing a complete summation the next approach becomes to try and perform the sums over a small but hopefully representative part of the configurations. With the same configurations included in both the numerator and the denominator in Eq. (3.14) the hope is that the value of the fraction will stay close to the true value. The most natural idea is then to generate a large number of configurations by random and use the found energy, E_ν, together with the observable A_ν to calculate the expectation value.

Quite naturally, the values of E_ν obtained in this way are centered around 0 with a standard deviation of $J\sqrt{2N}$. However, we are usually interested in properties of the system in the vicinity of the ordering temperature, and for the case of the 2D Ising model the average energy close to the transition happens to be close to $-1.4JN$. For large N this is far out in the tail of the distribution. A consequence of this is that the Boltzmann factor becomes very small for most of the configurations generated in this way. This could mean that the sums in Eq. (3.14) often are heavily dominated by only a few terms. An estimate for $L = 8$ at $T_c \approx 2.269$ shows that the fraction of sites with $E_\nu < \langle E \rangle$ is less than 10^{-14} and this is a fraction that rapidly gets even smaller as the system size increases. This clearly indicates that the approach to randomly generate a subset of the configurations isn't any very promising one. It is clear that an entirely new idea is needed to get an efficient and usable method and this is importance sampling by means of Markov chains.

Importance sampling The basic idea here is to generate the configurations in a way that makes the probability for a configuration to show up in our simulation proportional to the Boltzmann factor. As discussed above that may be done by means of a Markov chain. When this is achieved, the average is calculated with Eq. (3.15).

4.3.2 Metropolis Monte Carlo

Requirements of Monte Carlo methods

There are many ways to design a Markov chain and thereby a Monte Carlo method, but in order to give configurations from the correct probability distribution the method has to fulfill the following two requirements:

1. Detailed balance. The transition probabilities $p_{\nu\nu'}$ and $p_{\nu'\nu}$ (for the transitions $\nu \to \nu'$ and back) should obey

$$\pi_\nu p_{\nu\nu'} = \pi_{\nu'} p_{\nu'\nu}. \tag{4.3}$$

2. Ergodicity. It should be possible to access the whole phase space through a finite number of Monte Carlo steps.

With these two requirements and symmetric suggestion probabilities, $q_{\nu\nu'} = q_{\nu'\nu}$, the probability for the transition to be accepted is according to Eq. (2.11) $\alpha_{\nu\nu'} = \min(1, \pi_{\nu'}/\pi_\nu)$.

Implementation

A sweep through the system with the Metropolis[1] Monte Carlo method consists of the following:

- Step sequentially over the system.
- For each given site with spin s_i generate a new trial configuration ν' from ν by the change $s_i = -s_i$.
- Calculate the energy change $\Delta E = E_{\nu'} - E_\nu$.
- Accept the configuration with probability (cf. Eq. (2.11)),

$$\alpha_{\nu\nu'} = \min(e^{-\beta\Delta E}, 1),$$

which follows from $\pi_\nu \propto e^{-\beta E_\nu}$.

Critical slowing down

A Monte Carlo simulation performs a random walk in configuration space through a large number of single-spin flips. As a consequence the successive configurations are very similar to one another, the correlation times become large, and the number of truly independent configurations can become considerably reduced which means a poor precision in the calculated averages.

This effect rapidly becomes very severe close to criticality because of the large correlation length and the very large regions of correlated spins that are present close to T_c. The process to turn over a large region of spins may be exceedingly slow and this phenomena is called *critical slowing down*. The following section describes a method that is designed to eliminate this problem. This is (together with a closely related method) the first Monte Carlo algorithm with non-local updates and represent an important improvement. In recent years one has also managed to device other methods with such non-local Monte Carlo steps.

4.3.3 Cluster update methods

The general idea in the cluster update methods is to construct clusters of spins pointing in the same direction, and then flip the whole cluster.

Swendsen-Wang method

In the Swendsen-Wang method[2] one creates a number of clusters by two simple steps[3]:

1. Create cluster boundaries:
   i) Between all spins with opposite orientation (solid lines in Fig. 4.2).
   ii) With probability $e^{-2\beta J}$ between each nearest neighbor pair with *the same* orientation (dashed lines in Fig. 4.2).
2. Identify clusters and flip each cluster with 50% probability. (Note that the dashed lines that do not form closed paths do not play any role.)

[3]The presentation here is different from what is given in most text books where the focus is commonly on making clusters by creating *links* between spins. There is a direct connection between these two presentations as a cluster boundary implies the absence of a link and vice versa. I do however prefer the discussion in terms of cluster boundaries since I feel that it is easier to give them a direct intuitive interpretation.

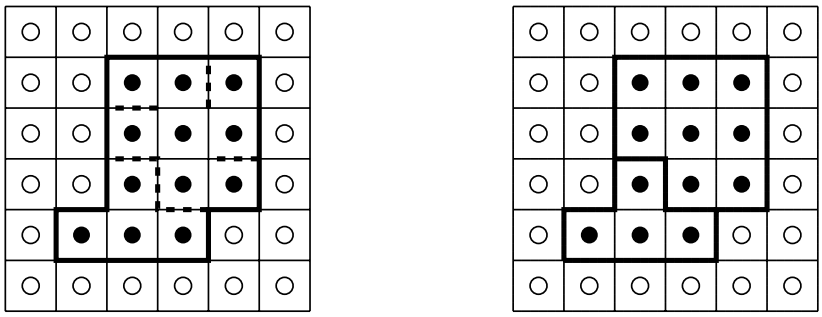


Figure 4.2: Construction of clusters in the Swedensen-Wang method. The left figure illustrates the creation of bondary segments i) between all spins with different orientation and ii) between *some* spins with same orientation. Note that this is just one out of many possible ways of doing the second step. The right figure shows the two clusters that the interior part of the system is split up into, but note that one more cluster boundary (in the random part "ii)") above could easily have made the region split up into three clusters instead.

Note that the probability $e^{-2\beta J}$ corresponds to $\Delta E = 2J$ which is the difference in energy for a pair with different and equal orientations, respectively. In this method the question we repeatedly ask is therefore "is it OK to put a cluster boundary here" whereas in Metropolis MC we ask "is it OK to flip this spin"?

The Wolff cluster update method

In the Wolff cluster update[3] one instead grows a single cluster which is then always flipped. It is actually convenient to flip the cluster as it is grown. One Wolff update step—which in a sense is equivalent to one MC sweep—is as follows:

- **Initialize:**

  a) Chose a starting position, i, by random.

  b) Store the spin direction: $S = s_i$.

  c) Flip the spin: $s_i = -s_i$.

  d) Put i in the queue.

- **Repeat as long as the queue is non-empty:**
  - Get a position, i. from the queue.
  - Loop over j, the nearest neighbors of i: If $s_j = S$ add the spin to the cluster with probability $1 - e^{-2\beta J}$, i.e.
    c) Flip the spin, $s_j = -s_j$.
    d) Put j in the queue.

The implementation of a queue in C is described in Sec. 7.3.

Acceptance probability and cluster simulations at low temperatures

We have above given a simple argument why it is reasonable to accept cluster boundaries with probability $e^{-2\beta J}$. That this choice fulfills detailed balance is furthermore shown in Sec. 4.3.4. It is nevertheless instructive to examine the low-temperature case where it is clear that the Metropolis single spin MC and the Wolff cluster MC are very similar.

In Metropolis MC, starting from a system in the ground state with only spin up, the energy cost for flipping a spin is $\Delta E = 8J$ and the fraction of spin down will be $e^{-8J/T}$. At $T/J = 1$ (which must be considered a low temperature as it is well below $T_c/J \approx 2.269$) this probability is $e^{-8} = 0.03\%$.

Things are quite a bit different when doing Wolff cluster MC since the clusters at low temperatures always span essentially the whole system, changing a (mostly) spin-up system to a spin-down system and vice versa. Not all spins are however added to the cluster. The probability for a given spin not to be added to the cluster is the same as the probability that there are four cluster boundary around that site, and this is something that happens with probability $(e^{-2J/T})^4 = e^{-8J/T}$. Since this is the same probability as found in Metropolis MC this does indeed suggest that $e^{-2J/T}$ is the correct probability for a cluster boundary.

4.3.4 The Wolff cluster update method – traditional presentation

In the Wolff cluster update one grows a cluster that is then flipped as a whole. Since this method in a single step may cause a big change in a configuration the correlations between successive configurations become considerably

smaller and the problem of critical slowing down is practically eliminated. Especially for large systems at temperatures close to T_c this means an enormous increase in efficiency.

With a clever method for growing the cluster it turns out that the change is always accepted, $\alpha_{\nu\nu'} = 1$. The detailed balance condition may therefore be expressed in terms of the targeting probability q and becomes

$$\frac{q_{\nu\nu'}}{q_{\nu'\nu}} = \frac{\pi_{\nu'}}{\pi_\nu}.$$

The method to build the cluster is to start from a random spin and then look at its neighbors to see if any of them points in the same direction and add these spins to the cluster with probability P_{add}. This is done recursively.

A specific growth process

To be specific we describe below the growth of a specific cluster, illustrated in Fig. 4.3. The spins are here numbered in the order that they are accessed. One sequence of steps that gives the cluster shown in the left panel (the sites within the boundary) are the following:

1. Choose the starting spin, spin number 5. The probability for this spin to be chosen is $P_1 = 1/N$.

Then consider all the neighbors to spin 5:

2. Add spin 6 to the cluster with probability P_{add}. In this case the spin was not accepted. The probability for this to happen is $P_2 = 1 - P_{\text{add}}$.

3. Spin 9 is added to the cluster. The probability for this is $P_3 = P_{\text{add}}$.

4. Spin 4 is not added, $P_4 = 1 - P_{\text{add}}$.

5. Spin 1 is of the wrong sign and cannot be added, $P_5 = 1$.

After considering all nearest neighbors to spin 5 we consider the neighbors to spin 9 that do not already belong to the cluster:

6. Spin 10 is added to the cluster. The probability for this is $P_6 = P_{\text{add}}$.

7. Spin 13 is not added, $P_7 = 1 - P_{\text{add}}$.

8. Spin 8 is of the wrong sign, $P_8 = 1$.

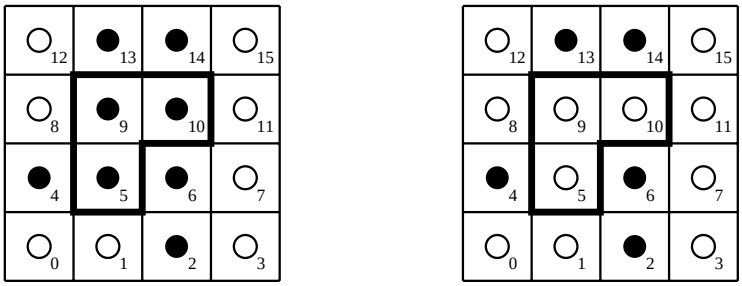


Figure 4.3: Wolff cluster update

Finally consider the neighbors of spin 10:

9. Spin 11 is of the wrong kind, $P_9 = 1$.

10. Spin 14 is not added to the cluster. $P_{10} = 1 - P_{\text{add}}$.

11. Finally, spin 6 is getting another chance, but it is not added to the cluster this time either, $P_{11} = 1 - P_{\text{add}}$.

The probability for getting precisely this sequence of events is $q_{\nu\nu'} = \prod_{i=1}^{11} P_i$. The inverse process, of going from ν' to ν, is illustrated in the right panel in Fig. 4.3. To make a list as the above to describe the inverse process, change all "is of the wrong sign" to "is not added" and vice versa.

Addition probability

We now focus on the domain boundary in the left and the right panels of Fig. 4.3. Because the two panels differ only in the sign of the spins within the cluster it follows that the boundary segments that separate aligned spins in the one panel separate anti-aligned spins in the other panel. We denote the number of segments that separate identical spins in configurations ν and ν' by n and n', respectively. It is then possible to express the energies of these configurations in terms of a common background term plus the contribution from the boundary as,

$$\begin{aligned} E_\nu &= E_{\text{bg}} - nJ + n'J, \\ E_{\nu'} &= E_{\text{bg}} + nJ - n'J, \end{aligned}$$

which means that the energy difference becomes

$$E_{\nu'} - E_{\nu} = 2(n - n')J. \tag{4.4}$$

It turns out that it is also possible to express the transition probabilities in terms of n and n'. Denoting the size of the cluster by s (in our example $s = 3$) the number of spins added to the original spin is $s - 1$ and the transition probabilities become

$$\begin{aligned} q_{\nu\nu'} &= \frac{1}{N}(P_{\text{add}})^{s-1}(1 - P_{\text{add}})^{n}, \\ q_{\nu'\nu} &= \frac{1}{N}(P_{\text{add}})^{s-1}(1 - P_{\text{add}})^{n'}. \end{aligned}$$

We then find

$$\frac{q_{\nu\nu'}}{q_{\nu'\nu}} = (1 - P_{\text{add}})^{(n-n')}$$

and from the detailed balance condition, Eq. (4.3), together with Eq. (4.4) for the energy difference we find

$$(1 - P_{\text{add}})^{(n-n')} = e^{-2\beta(n-n')J} = (e^{-2\beta J})^{(n-n')},$$

and it is clear that this is fulfilled if the addition probability is chosen as

$$P_{\text{add}} = 1 - e^{-2\beta J}.$$

What we have examined above is not really the probability of going from ν to ν', and the inverse process, but rather the probability of growing the cluster that takes us between the two states in a specific way. The full $p_{\nu\nu'}$ should have contributions from each of several possible ways to grow the cluster. However, since we have shown that it is possible to fulfill detailed balance for each specific way of growing the cluster, it follows immediately that detailed balance will hold for the total quantities $p_{\nu\nu'}$ and $p_{\nu'\nu}$.

4.3.5 Improved estimators

The idea behind improved estimators is the Swendsen-Wang cluster update method, where the whole system is split into clusters and each cluster is flipped with 50% probability.

From this follows that one can generate a large number of different configurations, this number is equal to 2^{N_c}, where N_c is the number of clusters, and the size of cluster number c is n_c. Each such configuration is then specified by the signs of all the clusters, t_c, for $c = 1, \ldots N_c$, and the magnetization becomes $M = \sum_c t_c n_c$. For each such configuration we have $M^2 = \sum_c \sum_{c'} t_c t_{c'} n_c n_{c'}$. The average of M^2 for all these configurations becomes

$$\begin{aligned} M^2 &= \frac{1}{2^{N_c}} \sum_{t_1=\pm 1} \cdots \sum_{t_{N_c}=\pm 1} \left[\sum_c \sum_{c' \neq c} t_c t_{c'} n_c n_{c'} + \sum_c t_c^2 n_c^2 \right] \\ &= \sum_c n_c^2. \end{aligned}$$

Note that the first term inside square brackets vanishes due to the sum over the t-variables.

Going instead to the Wolff algorithm the probability to find cluster c depends on its size and we get $p_c = n_c/N$. The average cluster size generated with the Wolff algorithm becomes

$$\langle n_c \rangle_W = \left\langle \sum_c p_c n_c \right\rangle = \frac{1}{N} \langle \sum_c n_c^2 \rangle,$$

and we therefore find

$$m^2 = \frac{1}{N^2} \left\langle \sum_c n_c^2 \right\rangle = \frac{1}{N} \langle n_c \rangle_W.$$

The same idea may also be used to determine the spin correlation function from the properties of the Wolff clusters. This is best done with FFT analyses of the clusters as discussed in Sec. 7.4.

4.4 Exact solutions

The behavior and complexity of the Ising model is highly dependent on the dimensionality. In one dimension the model only orders at $T = 0$ but for $d \geq 2$ there is a transition at a finite T_c where the order parameter vanishes as $M \sim (T_c - T)^\beta$.

One dimension

It is a simple exercise to solve the 1D Ising model exactly, i.e. to calculate the partition function in terms of elementary functions. The result is

$$Z^{\mathrm{1D}} = (2\cosh\beta J)^N.$$

Two dimensions – Onsagers solution

The solution of the two-dimensional Ising model is a very much more demanding problem. The model was solved by the norwegian physicist Lars Onsager in 1944. This was an impressing tour de force which by some is considered as one of the greatest scientific achievements of the 20th century. Because of the combination of a non-trivial behavior and a known exact solution, the 2D Ising model is often used as a testing ground for various kinds of methods for MC simulations. Onsager found among other things that the free energy is non-analytic at

$$T_c = \frac{2}{\ln(1+\sqrt{2})} \tag{4.5}$$

such that the heat capacity (which is $\propto \partial^2 F/\partial\beta^2$) diverges logarithmically,

$$C \sim -\ln|T - T_c|.$$

Three dimensions – an unsolved problem

The three-dimensional Ising model is still an unsolved problem which continues to attract both mathematicians and physicists. Both T_c and the critical exponents are however known with rather high precision from Monte Carlo simulations and analytic calculations.

4.5 Behavior at a critical point

A consequence of the non-analytic free energy is that various quantities diverge or vanish at the critical temperature. The behavior is described by the *critical exponents* α, β, γ, and δ. The main reason for the great interest in these exponents is because they are universal, which means that their values are the same for a number of different models that have certain properties in common. Such a class of models is said to constitute a *universality class*.

4.5.1 The heat capacity

The exponent α governs the behavior of the specific heat. In a typical first order transition there is a jump in the energy at the transition and this jump gives rise to a δ-function like peak in the heat capacity $C/N = d\langle e\rangle/dT$, where $\langle e\rangle = \langle E\rangle/N$. In *continuous* phase transitions on the other hand the specific heat often diverges like

$$C \sim |T - T_c|^{-\alpha}.$$

Since[4]

$$\lim_{\alpha\to 0}\frac{1}{\alpha}(x^{-\alpha}-1) = -\ln x,$$

the logarithmic behavior of the 2D Ising model is taken to imply $\alpha = 0$.

In simulations the heat capacity is usually determined from the fluctuations in the energy, c.f. Eq. (3.6)

4.5.2 The magnetization

The ensemble average of the magnetization is

$$\langle M\rangle = \frac{1}{Z}\sum_\nu P_\nu M_\nu, \quad \langle m\rangle = \frac{\langle M\rangle}{N},$$

where

$$M_\nu = \sum_i s_i.$$

Note that we make use of upper case letters for extensive quantities like the total magnetization whereas the magnetization *density* is denoted by a lower case, $\langle m\rangle$.

The magnetization is however a difficult quantity to handle both in the simulations and in analytical calculations. The reason is that the expectation value is always $M = 0$ independent of temperature because of the up-down symmetry of the model which means that are exactly as many states with $M_\nu > 0$ as with $M_\nu < 0$. It is nevertheless possible to define a quantity that behaves as an order parameter by considering applying a small magnetic field and taking the limit $h \to 0$ after $L \to \infty$,

$$\langle m\rangle = \lim_{h\to 0}\lim_{L\to\infty}\frac{1}{Z}\sum_\nu m_\nu e^{-\beta E_\nu(h)}. \tag{4.6}$$

[4]This follows from $x^{-\alpha} = e^{-\alpha\ln x} \approx 1 - \alpha\ln x$.

Both in simulations and in real ferromagnets one does however find a non-vanishing magnetization below T_c. The reason for this is that the system has become stuck in one part of the phase space and actually is non-ergodic. The fact that nature (as well as simulations) finds a solution to a problem with a symmetry that is different from the symmetry present in the problem itself, is called *spontaneous symmetry breaking*.

4.5.3 Critical exponents related to the magnetization

With the critical point defined by $T = T_c$ and $h = 0$ it is clear that it is possible to approach the critical point in two directions, either by changing the temperature or by changing the applied magnetic field. The behavior along these two directions in parameter space define two different critical exponents.

- The exponent β is defined from the vanishing of the magnetization as T_c is approached from below,

$$\langle m \rangle \sim (T_c - T)^{\beta}.$$

- The response of the magnetization to the applied field is governed by the exponent δ,

$$\langle m \rangle \sim h^{1/\delta}, \quad T = T_c.$$

  In this case the up-down symmetry is broken and there is no difficulty in the definition of the magnetization.

From the exponents β and δ it is possible to determine all the other exponents. Table 4.1 expresses these exponents in terms of y_t and y_h which describe the rescaling of the free energy density in the thermal and field directions, respectively. More on this in Sec. 4.8.1.

4.5.4 Susceptibility

The susceptibility is the response of the system to a weak applied magnetic field and is defined as

$$\chi = \frac{1}{N} \left. \frac{\partial \langle M \rangle}{\partial h} \right|_{h=0}.$$

The susceptibility may actually be written in terms of the fluctuations in the magnetization. From Eq. (4.2) we have

$$\langle M\rangle = \frac{1}{Z}\sum_{\nu} M_\nu e^{-\beta E_\nu} = \frac{1}{Z}\frac{\partial Z}{\partial(\beta h)} = \frac{\partial \ln Z}{\partial(\beta h)},$$

and we obtain

$$\begin{aligned}\chi &= \frac{\beta}{N}\frac{\partial^2 \ln Z}{\partial(\beta h)^2} \\ &= \frac{\beta}{N}\left(\left\langle M^2\right\rangle - \langle M\rangle^2\right).\end{aligned}$$

Below T_c we again get problems with determining $\langle M\rangle$ but in the high-temperature phase we may put $\langle M\rangle = 0$ and obtain

$$\chi = \beta N\left\langle m^2\right\rangle.$$

The exponent γ is defined from the divergence of the susceptibility as T_c is approached from either side,

$$\chi \sim |T - T_c|^{-\gamma}.$$

	definition	$d=2$	$d=3$	With d, y_t, and y_h
α	$C \sim \lvert T-T_c\rvert^{-\alpha}$	0	0.10940	$2-d/y_t$
β	$\langle m\rangle \sim (T_c-T)^{\beta}$	1/8	0.3267	$(d-y_h)/y_t$
γ	$\chi \sim \lvert T-T_c\rvert^{-\gamma}$	7/4	1.2372	$(2y_h-d)/y_t$
δ	$\langle m\rangle \sim h^{1/\delta}$, $T=T_c$	15	4.787	$y_h/(d-y_h)$
η	$g(k) \sim 1/k^{2-\eta}$, $T=T_c$	1/4	0.0368	$d+2-2y_h$
ν	$\xi \sim \lvert T-T_c\rvert^{-\nu}$	1	0.630199	$1/y_t$
k_BT_c/J		2.26919	4.51153	

Table 4.1: Critical properties of the Ising model in two and three dimensions. The exponents are universal quantities whereas the critical temperatures are for the usual square and cubic lattices, respectively.

4.5.5 The correlation function

The Ising model may also be described in terms of the correlation function $g(\mathbf{r})$. One may then extract two exponents: one that describes the behavior at T_c and another that describes the approach to T_c. Because of the relations between the critical exponents these two together completely determine the critical behavior of the model.

We define the correlation function

$$g(r) = \langle s_{\mathbf{r}} s_{\mathbf{0}} \rangle \,,$$

which has the following behaviour below, at, and above T_c,

$$g(r) \sim \begin{cases} g_\infty + Ce^{-r/\xi}, & T < T_c, \\ r^{-(d-2+\eta)}, & T = T_c, \\ Ce^{-r/\xi}, & T > Tc. \end{cases}$$

This defines the critical exponent η from the decay of the correlations at T_c. The Fourier transform of the correlation function is also of interest. We have

$$g(\mathbf{k}) = \sum_{\mathbf{r}} g(\mathbf{r}) e^{-i\mathbf{k}\cdot\mathbf{r}},$$

and the behavior at T_c is given by

$$g(k) \sim \frac{1}{k^{2-\eta}}.$$

The other exponent that may be extracted from the correlation function is the correlation length exponent ν from the divergence of ξ,

$$\xi \sim |T - T_c|^{-\nu}.$$

4.6 Mean field theory

One of the simplest methods to examine a model in statistical physics is to use mean field theory. The basic approximation is that the effect on a certain particle of the interactions with the other particles may be treated by introducing a field, and that this field may be calculated self-consistently.

In mean field theory of the Ising model one focuses on the effect on spin s_i of the surrounding spins s_j where j are the z nearest neighbors. The part of the Hamiltonian which is related to s_i is

$$\mathcal{H}(s_i) = -J\sum_j s_i s_j = -h_i s_i,$$

where $h_i = J\sum_j s_j$. The expectation value of s_i is then

$$\langle s_i\rangle = \frac{e^{\beta h_i} - e^{-\beta h_i}}{e^{\beta h_i} + e^{-\beta h_i}} = \tanh(\beta h_i).$$

Up to this point everything is exact, but we now assume that we may consider the average field $\langle h_i\rangle$ and that this may be calculated from $\langle s_j\rangle = \langle s_i\rangle = m$. This gives

$$m = \tanh(\beta J z m), \tag{4.7}$$

which is shown graphically for two different values of βJz in Fig. 4.4. From

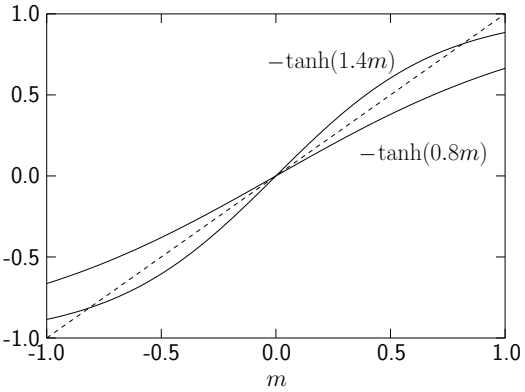


Figure 4.4: Graphic solutions to Eq. (4.7).

the figure we may conclude that Eq. (4.7) only has the trivial solution for $\beta Jz = 0.8$ but also a non-trivial solution for $\beta Jz = 1.4$. It is clear that the non-trivial solution will exist only if

$$\left.\frac{d\tanh(\beta Jzm)}{dm}\right|_{m=0} > 1.$$

From the Taylor expansion $\tanh x \approx x - x^3/3$ we conclude that this condition gives $\beta_c Jz = 1$, and the critical temperature

$$k_B T_c = Jz.$$

4.6.1 The exponent β

To avoid confusion we will in the present subsection use $\tilde{\beta}$ for the inverse temperature. For $\tilde{\beta}$ close to $\tilde{\beta}_c$, the definition of the exponent β is

$$M \sim (\tilde{\beta} - \tilde{\beta}_c)^{\beta}.$$

From Eq. (4.7) together with the Taylor expansion and $\tilde{\beta}_c Jz = 1$ we get

$$\tilde{\beta}_c Jzm = \tilde{\beta} Jzm - \frac{1}{3}(\tilde{\beta} Jzm)^3.$$

By rearranging and dividing by m we get

$$(\tilde{\beta} - \tilde{\beta}_c) \sim m^2,$$

which gives

$$m \sim (\tilde{\beta} - \tilde{\beta}_c)^{1/2},$$

and we find the exponent $\beta = 1/2$ in the mean field approximation.

4.6.2 Dimensionality and the MF approximation

With a knowledge of the true behavior it is appropriate to return to the approximative methods considered before, especially the mean field approximation.

Since the dimensionality of the system never plays any role in the mean field approximation[5] the character of the transition is independent of dimensionality. Table 4.2 shows a comparison between the predictions from the MF approximation and the true behavior. It is clear that this approximation is terribly bad at low dimensions but works reasonable well in 3D.

4.7 Energy-entropy argument

As mentioned above the behavior of the Ising model depends strongly on the dimensionality but that is not seen in the mean field approximation. A

[5]The dimensionality does enter through the coordination number, but that is equivalent with a rescaling of the coupling constant or (equivalently) a change in temperature scale only. It doesn't give any qualitative difference.

d	k_BT_c/J	$k_BT_c^{\rm MF}/J$	$T_c/T_c^{\rm MF}$	β
1	0	2	0	–
2	2.269	4	0.57	0.125
3	4.511	6	0.75	0.3267

Table 4.2: Comparison between the predictions from mean field theory and the true behavior.

simple way to judge about the existence of a phase transition, which takes the dimensionality into account, is through an energy-entropy argument.

The state with lowest energy is an ordered state and that will dominate the behavior at zero temperature. There is however a huge number of disordered states and the idea behind the energy-entropy argument is to compare the ordered ground state with the different disordered states to decide which is more probable.

We now consider Ω_0 states with energy E_0 and Ω_1 states with energy E_1 and we want to compare the two probabilities

$$\begin{aligned} P_0 &\propto \Omega_0 e^{-\beta E_0}, \\ P_1 &\propto \Omega_1 e^{-\beta E_1}. \end{aligned}$$

To phrase this in terms of entropy and energy we recall

$$S = k_B \ln \Omega \quad \Rightarrow \quad \beta k_B T S = k_B \ln \Omega \quad \Rightarrow \quad \Omega = e^{\beta TS},$$

which means that we may write

$$P \propto \Omega e^{-\beta E} = e^{-\beta(E-TS)} = e^{-\beta F},$$

where $F = E - TS$ is the free energy. This implies that the condition $P_1 > P_0$ corresponds to $F_1 < F_0$, and with $\Delta E = E_1 - E_0$ and $\Delta S = S_1 - S_0$ we may say that disorder wins if $\Delta F < 0$ which is the same as $T\Delta S > \Delta E$.

4.7.1 1D Ising model

We now apply the energy-entropy argument to a 1D Ising chain with $L+1$ spins in the limit $L \to \infty$. The model has two different ground states with either $s_i = 1$ for all i or $s_i = -1$, i.e. $\Omega_0 = 2$. The ground state energy

is $E_0 = -JL$. There is then also $\Omega_1 = 2L$ different disordered states with energy $E_1 = E_0 + 2J$, and we find that the difference in free energy is

$$\Delta F = \Delta E - T\Delta S = 2J - k_B T \ln L,$$

which means that $\Delta F < 0$—the disorder wins—for all $T > 0$. The conclusion is that the system disorders for all $T > 0$ and that there is only a trivial phase transition at $T_c = 0$.

4.7.2 2D Ising model

When considering the 2D model it is more appropriate to reformulate the order-disorder question slightly. The question is instead whether the ground state is stable against the formation of large domains. If there are no large domains the system will be in the ordered phase whereas the presence of such domains means a disordered system.

We then need to determine both energy and entropy for a domain boundary with perimeter p. For the energy we immediately get

$$E(p) = 2Jp.$$

It is more tricky to estimate the number of different domain boundaries with perimeter p. Note first that this is a self-avoiding walk (see Sec. 5.3.2) but of a somewhat special kind since it has to close onto itself. As an upper bound we take the number of non-backtracking walks which means $\Omega(p) < 3^p$, and as a estimated lower bound we take $\Omega(p) > 2^p$. Taken together this becomes

$$\Omega(p) = c^p, \quad 2 < c < 3,$$

and we may write an expression for the free energy of a domain with perimenter p as

$$\Delta F = 2Jp - k_B T p \ln c.$$

This means that the change in free energy turns negative for $T > T_c$ where

$$k_B T_c = \frac{2J}{\ln c} \quad \Rightarrow \quad 1.8J < k_B T_c < 2.9J.$$

4.8 Universality, RG theory, and scaling

A remarkable property of the phenomena we have discussed here is the fact that certain properties are insensitive to the details of the system. This is the case for the exponents which are kinds of fingerprints of the "universality class" that characterizes the transition. To e.g. change the kind of lattice from square to hexagonal or triangular doesn't affect these exponents. Similarly, adding couplings along the diagonals or even to sites several lattice spacings away, does not change the universality class. It is only quantities like the critical temperature that will be different for these different cases.

The reason for the existence of universality classes is that the details of the system start to become increasingly less important when it is the big clusters that start to dominate the system. What governs the approach to T_c is how the clusters merge, and on the larger length scales this looks the same independent of several kinds of details of the system. With this way of thinking it may also be concluded that the change of dimensionality of the system also should lead to a different universality class.

The concept of universality classes is actually not only important for how we think about various kinds of transitions but also for the kind of simulations we choose to do. The computationally most efficient method is to consider the simplest possible model that belongs to the same universality class as the problem under consideration. Even in cases where we are interested in the non-universal quantities like T_c it could be wise to first examine the simplest model expected to be in the same universality class, and only as a second step attack the more realistic model.

4.8.1 Renormalization Group theory

As discussed above the crucial property of systems at the critical point is the absence of a characteristic length scale. This is e.g. seen in the correlation function which in the Ising model is $g(r) \sim r^{-(d-2+\eta)}$ at T_c, see Sec. 4.5.5. The absence of a characteristic scale implies that fluctuations of all sizes are important and this is the main reason why the critical phenomena for a long time defied solution by analytical techniques.

Renormalization Group theory which is usually attributed to K. G. Wilson was the result of a process in the end of the 60's where several people (including e.g. Fisher and Kadanoff) made significant contributions. One very impressive part of the RG theory is the possibility to calculate critical

exponents. This is however a technical and highly specialized art which will not concern us here. From our perspective the importance of RG theory is rather that it gives a way of thinking about critical phenomena and gives a theoretical motivation for the scaling assumptions.

The idea behind Renormalization Group theory is to consider the effect of a *change of scale* to the system. To make this more concrete one may consider looking at a system through a microscope with the possibility to change the magnification. The absence of a characteristic length at T_c means that the configurations right at T_c would look the same independent of the magnification. In contrast, at temperatures above or below T_c we will find that an increase of the length scale will have the effect to make the configurations look as if they were farther away from T_c. Figure 4.5 is a simple sketch

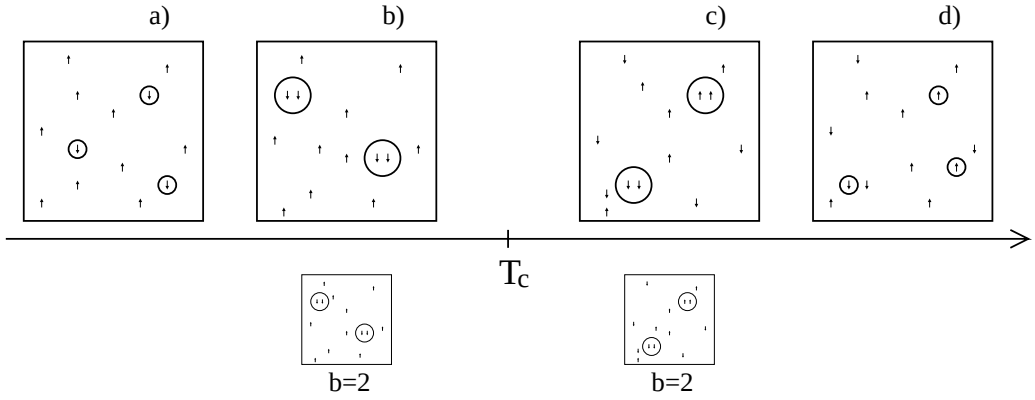


Figure 4.5: Sketch of typical configurations below and above T_c. The configuration below T_c have an ordered background and the fluctuations are regions with opposite spins. The fluctuations are of course of different size and shape but have here for simplicity been illustrated by circular regions with a size proportional to the correlation length. The configurations above T_c have a disordered background and the fluctuations are here ordered regions where (most of) the spins either point up or down. The figures below the temperature axis show the effect of a change of scale with the scale factor $b = 2$. These configurations look much the same as the configurations farther away from T_c.

of typical configurations at four temperatures around T_c. Panel a) and b) are for $T < T_c$. In that case the system has a non-vanishing magnetization and the main part of the system has spins pointing up. There is then some

regions with spins of the opposite direction and the figure shows the typical size of these islands which is directly related to the correlation length. For $T > T_c$, panels c) and d), the background of the system is disordered and the fluctuations are now ordered regions with spins pointing either upwards or downwards. Far above T_c these regions are small but as T_c is approached the typical size of these regions grows larger.

Consider a system at the temperature T. Introduce $t = T/T_c - 1$. The effect of a change of scale with a factor b may be considered as a change of temperature with $t_b = s(b)t$. The unknown function $s(b)$ may be determined from the fact that a single change of scale with the factor $b_1 b_2$ is the same as two consecutive scale changes with the factors b_1 and b_2. We therefore have

$$s(b_1 b_2) = s(b_1)s(b_2),$$

which holds with $s(b) = b^y$ for arbitrary y since $(b_1 b_2)^y = b_1^y b_2^y$.

The above discussion may illustrate the effect of a rescaling but is very different from the usual formulation of the Renormalization Group approach where one doesn't consider the changes to individual configurations, but rather how the parameters in the Hamiltonian (e.g. the "coupling" $K = J/T$) change with the scale factor b. The effect of the scale factor is then that b^d spins are collected into a single "block spin" where the coupling between these block spins is given by K'. The critical point is associated with $K' = K = K^*$ and for K away from K^* the effect of the renormalization is that K' moves away from K^*. The "fixed point" K^* associated with a critical point is therefore said to be an "unstable" fixed point.

4.8.2 The scaling behavior of the free energy

Consider a system of size $N = L^d$ and a change of scale to $L' = L/b$ and $N' = N/b^d$. The total free energy should be the same regardless of the scale factor[6], $F(t, L) = F(t', L')$. For the free energy density we may write

$$f(t)L^d = f(t')L'^d,$$

which means

$$f(t) = \frac{L'^d}{L^d} f(t'),$$

[6]This is true for the part of the free energy that is relevant for the phase transition, the singular part of the free energy.

or

$$f(t) = b^{-d} f(tb^{y_t}).$$

We may make this slightly more general by also including the magnetic field h,

$$f(t, h) = b^{-d} f(tb^{y_t}, hb^{y_h}).$$

4.8.3 Relation to critical exponents

The scaling relation for the free energy density may now be used to derive the expressions in Table 4.1 for the the critical exponents in terms of d, y_t, and y_h. Consider the magnetization,

$$m(t, h) \sim \frac{\partial f}{\partial h} = b^{-d+y_h} f_h(tb^{y_t}, hb^{y_h}),$$

where f_h is another function, the derivative of f with respect to its second argument.

So far we have considered b to be a fixed constant but the relation above may also be used in a somewhat different way. One may e.g. choose $tb^{y_t} = -1$, which gives $b = (-t)^{-1/y_t}$. For $h = 0$ we then find

$$m(t, 0) = (-t)^{(d-y_h)/y_t} f_h(-1, 0),$$

and since $f_h(-1, 0)$ is a constant and $m \sim (-t)^\beta$, we may identify $\beta = (d - y_h)/y_t$. The other relations in Table 4.1 may be derived with the same techniques, and from these expressions follow several relations between the exponents.

4.8.4 Finite size scaling

The finite size scaling relations are derived in a similar way. The critial behavior, i.e. the divergence of ξ, is only seen at $t = 0$ (i.e. $T = T_c$), $h = 0$, and $1/L = 0$, and we therefore include $1/L$ as an additional argument to the free energy. According to the same pattern as the other quantities the argument on the right hand side would be b^{y_L}/L, but since a change of scale with a factor b gives the linear size L/b it follows that $y_L = 1$ and the scaling assumption becomes

$$f(t, h, 1/L) = b^{-d} f(tb^{y_t}, hb^{y_h}, b/L).$$

The finite size scaling relation for the magnetization now becomes

$$m(t, h, 1/L) = b^{-d+y_h} f_h(tb^{y_t}, hb^{y_h}, b/L),$$

and for $h = 0$ we may choose $b = L$ which gives

$$m(t, 0, 1/L) = L^{-d+y_h} f_h(tL^{y_t}, 0, 1).$$

With the above derived relation for β together with $y_t = 1/\nu$ we finally find (where we change the last argument on the left side from $1/L$ to L),

$$m(t, L) = L^{-\beta/\nu} \tilde{f}_h(tL^{1/\nu}).$$

This suggests that plotting $mL^{\beta/\nu}$ versus $(T/T_c - 1)L^{1/\nu}$ should make the points collapse on a single curve, described by the (unknown) function $\tilde{f}_h(x)$.

As discussed above, the up-down symmetry of the model means that $m = 0$ at all temperatures and the way out is to define the magnetization in the limit $h \to 0$, which is to be taken after $L \to \infty$. This procedure is of course not relevant for the finite system sizes to be used in finite size scaling, and the way out is to either take the absolute value,

$$m = \frac{1}{N} \langle |M| \rangle ,$$

or to examine the magnetization squared,

$$m^2 = \frac{1}{N^2} \left\langle M^2 \right\rangle .$$

The latter quantity has the scaling behavior

$$m^2(t, L) = L^{-2\beta/\nu} f_2(tL^{1/\nu}). \tag{4.8}$$

4.8.5 Binder's cumulant

When the critical behavior is unknown there are three unknown quantities in Eq. (4.8), T_c, β, and ν. It is often difficult to simultaneously adjust all these three parameters to get the best possible fit. A way out is to use a different quantity that has a known scaling dimension, and thereby a known behavior at T_c. One such quantity is Binder's cumulant which was originally

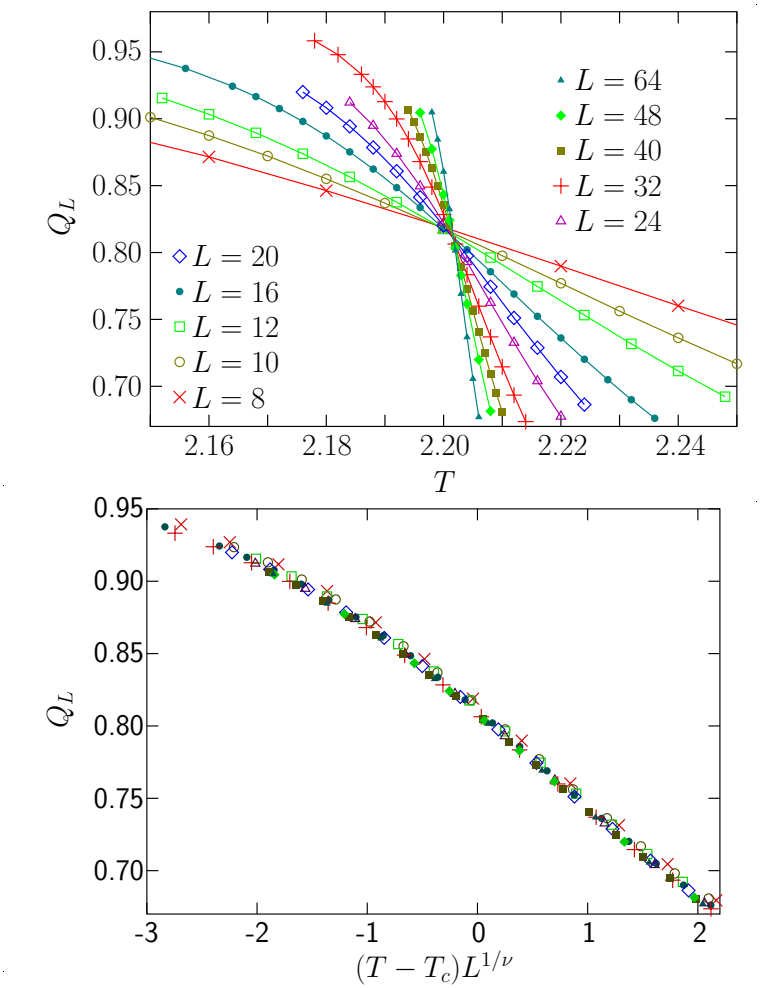


Figure 4.6: Binder's cumulant obtained from the 3D XY model. The first panel illustrates the crossing of the cumulant Q for different L at a single temperature which is identified as T_c. The second panel shows the collapse obtained with $T_c = 2.2018$ and $\nu = 0.672$. A closer look would reveal that the crossing points for data with L and $2L$ shift to higher temperatures for larger L, this is the usual behavior and is called "corrections to scaling". This adds some complexity to high-precision determinations of the exponents.

introduced as $U_L = 1 - \langle m^4 \rangle / (3 \langle m^2 \rangle^2)$. A simpler, closely related, formula is

$$Q = \frac{\langle m^2 \rangle^2}{\langle m^4 \rangle}. \tag{4.9}$$

The scaling behavior of this quantity is simply,

$$Q(t, L) = f_Q(tL^{1/\nu}), \tag{4.10}$$

where $f_Q(x)$ is a scaling function. To determine T_c one may plot $Q_L(T)$ for a number of different sizes L. The common intersection point is the critical temperature. With this value of T_c it is then possible to adjust ν such that the data for the different sizes collapse on top of one another. This kind of analysis of the 3D XY model is shown in Fig. 4.6.

4.9 More on analytical techniques

4.9.1 High temperature expansion

It is possible to examine the Ising model by two different kinds of expansions. In the following sections we will make use of the dimensionless coupling $K = J/k_B T$. In the high-temperature expansion this (or rather $\tanh K$) will be used as the small parameter.

The starting point is the partition function

$$Z = \sum_\nu e^{-\beta E_\nu} = \sum_\nu e^{K \sum_{\langle ij \rangle} s_i s_j} = \sum_\nu \prod_{\langle ij \rangle} e^{K s_i s_j}.$$

From

$$e^{\pm K} = \frac{e^K + e^{-K}}{2} \pm \frac{e^K - e^{-K}}{2},$$

we get

$$e^{K s_i s_j} = \cosh K + s_i s_j \sinh K,$$

and the partition function becomes

$$\begin{aligned} Z_H(K) &= \sum_\nu \prod_{\langle ij \rangle} (\cosh K + s_i s_j \sinh K) \\ &= (\cosh K)^{2N} \sum_\nu \prod_{\langle ij \rangle} (1 + s_i s_j \tanh K). \end{aligned}$$

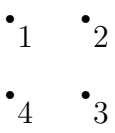
The product is now a large number of different terms with various powers of $\tanh K$, e.g.

$$1 + s_1 s_2 \tanh K + s_3 s_4 \tanh K + \ldots + s_1 s_2 s_3 s_4 \tanh^2 K + \ldots$$

It is however easy to show that most of these terms do not contribute to the partition function since the sum over configurations means summing over $s_i = \pm 1$. We e.g. have

$$\sum_{s_1=\pm 1} \sum_{s_2=\pm 1} s_1 s_2 \tanh K = 0.$$

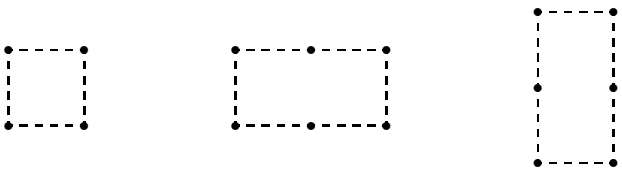
The only terms that contribute are those where each s_i appears an even number (0, 2, or 4) of times. If we assume that sites 1 through 4 are located as shown here,



then the term below will give a non-vanishing contribution to the partition function,

$$(s_1 s_2)(s_2 s_3)(s_3 s_4)(s_4 s_1) \tanh^4 K.$$

It turns out that all the non-vanishing terms may be represented in terms of a closed path (or a set of closed paths) on the lattice. The terms up to sixth order in $\tanh K$ may be represented by the following closed paths



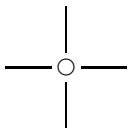
and since each path may be put on $N = L \times L$ different places on the lattice the contributions to the partition functions become

$$Z_H = (2\cosh^2 K)^N \left[1 + N \tanh^4 K + 2N \tanh^6 K + \ldots\right]. \tag{4.11}$$

It is of course possible to consider further terms in this expansion but for our purposes the first few will be enough.

4.9.2 Low temperature expansion

The ground state corresponds to all the spins pointing in the same direction, $+1$ or -1 with energy $-2NJ$. At zero temperature these configurations are the only relevant one, but at finite temperatures the configurations with one or a few spins in the opposite direction will contribute as well. Note that a single flip will give $N-4$ links with $s_i s_j = 1$ and 4 links with $s_i s_j = -1$ which gives the energy $-(2N-8)J$. A pictorial representation of such a configuration with the mis-aligned spin and the links with energy $= +J$ is shown here:

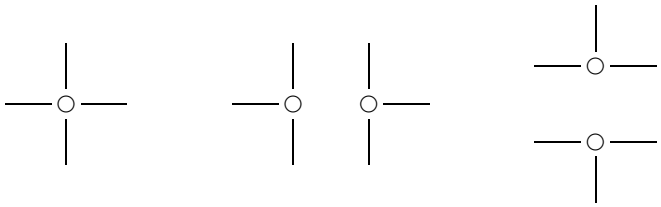


The single misaligned spin may be any of the N spins in the system. The contribution to the partition function therefore becomes

$$Z_L = 2\left[e^{2KN} + Ne^{2K(N-4)} + \ldots\right] = 2e^{2KN}\left[1 + N(e^{-2K})^4 + \ldots\right],$$

where the prefactor of 2 comes from the up-down symmetry in the problem.

The next terms in Z_L come from configurations with two misaligned spins that are nearest neighbors; the energy is $-(2N-12)J$.



Including these terms we find

$$Z_L(K^*) = 2e^{2K^*N}\left[1 + N(e^{-2K^*})^4 + 2N(e^{-2K^*})^6 + \ldots\right], \tag{4.12}$$

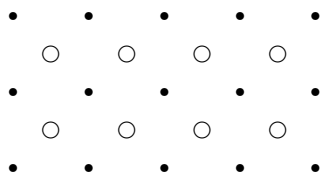
where K^* is introduced instead of K for the coupling in the low-temperature region.

4.9.3 Duality relation

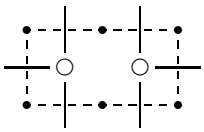
Note that the first few terms of the high- and the low-temperature expansions are of the same form. The only difference is that the expansion parameter

is e^{-2K^*} in the low temperature expansion and $\tanh K$ in the high temperature expansion. This is not just accidental but rather holds for all terms to arbitrary order. To see this we will need the concept of a *dual lattice*.

The figure below illustrates the *dual lattice* for the simple 2D square case. The filled circles are the ordinary lattice where the spins are located. The open circles are the points of the dual lattice. Note that the dual of a square lattice is itself a square lattice. (What is the dual lattice of a triangular lattice?)



The correspondence between the geometrical objects of the high- and low-temperature expansions may now be made more precise. Putting two such objects on top of one another gives,



and this demonstrates that each closed path that may be formed on the original lattice (that describes a non-vanishing term in the high-temperature expansion) may be transformed into a set of solid lines (links) on the dual lattice that characterizes a term in the low-temperature expansion of the Ising model on the dual lattice.

The implication is that the Ising model with coupling K on the original lattice is closely related to an Ising model with coupling K^* where the two coupling constants are related to one another through

$$\tanh K = e^{-2K^*}, \tag{4.13}$$

and we find $K^* = -(1/2)\ln\tanh K$. This relation gives a K^* of the low-temperature region from a K that is from the high-temperature region and as K increases K^* will decrease. There is then a special point where $K^* = K$ and it seems reasonable to associate this with the phase transition. To determine that coupling we rewrite the above equation in a more symmetric

form. From Eq. (4.13) we get

$$e^{2K^*} - e^{-2K^*} = \frac{e^K + e^{-K}}{e^K - e^{-K}} - \frac{e^K - e^{-K}}{e^K + e^{-K}} = \frac{4}{e^{2K} - e^{-2K}},$$

which may be rewritten as

$$\sinh 2K \sinh 2K^* = 1.$$

To solve for $K_c = K = K^*$ we see that $\sinh 2K_c = 1$ gives

$$e^{2K_c} - e^{-2K_c} = 2 \Rightarrow e^{4K_c} - 2e^{2K_c} = 1,$$

which becomes $e^{2K_c} = 1 + \sqrt{2}$ and we find

$$K_c = \frac{\ln(1+\sqrt{2})}{2},$$

which is equivalent to the expression for T_c in Eq. (4.5).

4.10 The lattice gas

We have so far discussed the Ising model in terms appropriate to a magnet with terms as magnetisation, susceptibility, and so forth. It is however worth pointing out that the model may be of relevance to other seemingly unrelated systems. By a simple change of variables the Ising model may be changed into a lattice gas model where the variables are $n_i = 0$, 1, corresponding to empty or occupied sites, and the attraction of molecules at short distances is captured by assigning the energy $-\epsilon$ to each nearest neigbor pair,

$$\mathcal{H} = -\epsilon \sum_{\langle ij \rangle} n_i n_j.$$

This model is defined in the grand canonical ensemble where the number of particles fluctuates and is controlled by the chemical potential μ. The grand sum is then

$$\Xi = \sum_\nu e^{-\beta E_\nu + \beta \mu N_\nu},$$

where $N_\nu = \sum_i n_i$.

4.10.1 Relation between the Ising model and the lattice gas

It is not difficult to derive the relation between the lattice gas and the Ising model. (Since N in the grand canonical ensemble by convention is the number of particles, we use $N_c = L^d$ for the number of cells.) The starting point is the relation between the occupancy n_i and the spin variable s_i:

$$n_i = \left(\frac{s_i + 1}{2}\right).$$

Substituting this into the grand sum for the lattice gas we get

$$\Xi = \sum_{\nu} \exp\left[\beta\epsilon \sum_{\langle ij \rangle}^{zN_c/2} \left(\frac{s_i + 1}{2}\right)\left(\frac{s_j + 1}{2}\right) + \beta\mu \sum_i \left(\frac{s_i + 1}{2}\right)\right],$$

and for the expression within brackets we get

$$\begin{aligned} &\frac{\beta\epsilon}{4} \sum_{\langle ij \rangle} s_i s_j + \frac{\beta\epsilon z}{4} \sum_i s_i + \frac{\beta\epsilon z N_c}{8} + \frac{\beta\mu}{2} \sum_i s_i + \frac{\beta\mu N_c}{2} \\ &= \frac{\beta\epsilon}{4} \sum_{\langle ij \rangle} s_i s_j + \beta\left(\frac{\epsilon z}{4} + \frac{\mu}{2}\right) \sum_i s_i + \beta\left(\frac{\epsilon z}{8} + \frac{\mu}{2}\right) N_c. \end{aligned}$$

A comparison with the Ising model gives

$$\begin{aligned} J &\Leftrightarrow \epsilon/4 \\ h &\Leftrightarrow \frac{\epsilon z}{4} + \frac{\mu}{2} \end{aligned}$$

4.11 A few words about all the other models

Since the 2D Ising model is the simplest model in statistical physics with a critical behavior it is a common starting point for studies of critical phenomena. The purpose with this section is to stress the fact that this is not the complete story of critical phenomena. It is rather only the first page . There is a vast number of different models that differ in many aspects. Starting from the Ising model (a spin model with a one-component spin) we can imagine a few different changes that all give rise to models with (usually) qualitatively different behavior. (Other models will give additional possibilities.)

1. Change the number of spin components. Two-component spins give the XY model which has been used to describe superconductivity. The phase transition in the 2D XY model is a key phenomenon behind the Nobel prize for Physics 2016 and has also been studied a lot here at the Physics department.

2. Three spin components gives the Heisenberg model.

3. Introduce frustration, i.e. put a negative coupling constant at some links such that it becomes impossible to satisfy all couplings around an elementary square simultaneously.

4. Put disorder into the system. That can be done through couplings with random strength, e.g. uniformly distributed between $-J$ and J, or by introducing a random field.

The changes listed above are all sufficient for changing the universality class. It is also possible to give examples of more trivial changes that do not affect the universality class (i.e. the critical exponents) though they do give different values of T_c. Such examples are

- Changing from a square lattice to a triangular lattice.

- Changing the interaction from only nearest neighbors to also include interactions to next nearest neighbors.

There are many other kinds of models, beside the spin models, in statistical physics. One example is the solid-on-solid model where each point i has an associated height h_i, and the Hamiltonian is

$$H = J \sum_{\langle ij \rangle} (h_i - h_j)^2,$$

yet another one is a model with interacting loops in three dimensions. One of many fascinating thing about statistical physics is that there are relations between many of these seemingly unrelated models.

The "spins" considered above are classical spins but the behavior (of course) becomes radically different with quantum spins. Such models may be studied with quantum Monte Carlo—a big field with a large number of interesting, and often rather complicated, algoritms.

Chapter 5

Simple stochastic models

5.1 Scale free behavior

We have seen that there is no characteristic length in the 2D Ising model at T_c. We say that the Ising model is *scale free* at criticality. This property is in marked contrast to most physical phenomena where an understanding may be obtained by focusing on the behavior at a certain length and/or energy scale and neglecting possible processes at larger and/or smaller lengths. As an example, the behavior of an atom or a molecule may be examined without considering on the one hand the details of the nucleus and on the other hand the macroscopic environment of the molecule. This separation of length scales is important since it means that we can consider one part of the awfully complicated system at a time, and this is crucial for the scientific development of the last centuries.

Problems that include many widely different length scales are often difficult to handle—both analytically and numerically—and may well require a new way of thinking. One example of a successful attack on such problems is the Renormalization Group theory by Wilson who was awarded with the Nobel price 1982 "for his theory for critical phenomena in connection with phase transitions". There are also several examples of problems that are as yet not understood that are difficult to handle at least to some degree because of the very different length (or time) scales involved:

- Turbulence is the main unsolved problem in classical physics. The onset of turbulence depends on the behavior of the system on all scales from the atomic scales to macroscopic dimensions.

- The Gutenberg-Richter law is extracted on the basis of the statistics of the magnitudes of earth quakes. Formulated in terms of the released energy the histogram obeys a power law, $H(E) \sim E^{-C}$.
- A well known scale free phenomenon is the $1/f$-noise that has been observed in many diverse systems like the flow of the river Nile, light from quasars, and high-way traffic.

The possibility of scale free behavior is a common property of many of the models in the present chapter and has for quite some time been an area of very active research.

5.2 Site percolation

Percolation is our first example of a simple stochastic models with a surprisingly complex behavior.

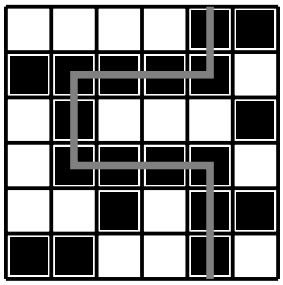


Figure 5.1: A configuration is said to *percolate* if it is possible to get from one side of the system (say the bottom) to the other (the top) by only stepping on occupied bonds/sites. They grey line shows the percolating path across the system. All sites that are connected to this path constitute the percolating (or spanning) cluster.

In percolation one considers the possibility to find a path across a medium with randomness. There are two kinds of percolation on a lattice, bond percolation and site percolation. In both cases there is a occupation probability p for each bond/site to be occupied. A configuration is said to *percolate* if it is possible to get from one side of the system (say the bottom) to the other

(the top) by only stepping on occupied bonds/sites. One may then define percolation probabilities P^{bond} and P^{site} that depend on both the system size and p. In the following we will only consider site percolation.

It turns out that the natural definition of a percolating cluster becomes slightly different with open and periodic boundary conditions. With OBC the bottom and top rows are special and the question to consider is the possibility to get from the bottom to the top. This criterion for percolation has to be modified when periodic boundary conditions are used. One reason for this is that with PBC the top row may be reached by going a single step *downwards* from the bottom row. Another reason is that we need a criterion that is independent of the position of the boundary. With PBC, the move of a row from the bottom to the top should never change a percolating system to a non-percolating one, or vice versa. With this requirement, an acceptable criterion for percolation is whether there exists of a path where the sum of the steps is equal to $(0, L)$. Whereas the two definitions often agree about the percolation, Fig. 5.2 shows two configurations where the two definitions actually do give different answers.

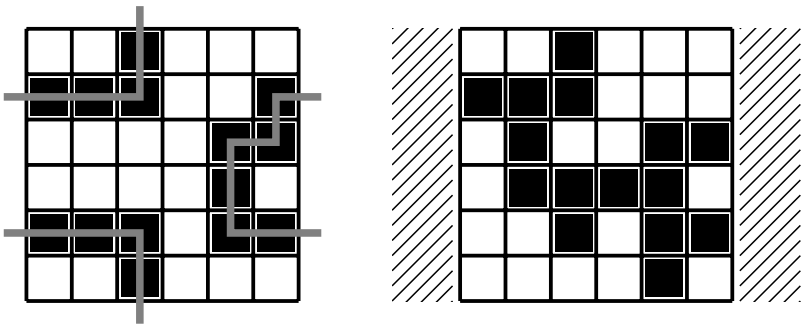


Figure 5.2: Percolating paths for periodic and open bondary conditions. The left figure percolates only with PBC because of the possibility to cross the boundary and then return back again. The right figure on the other hand percolates with OBC since it is possible to get from the bottom to the top. Since it is not possible to return back to the original site after crossing the system there is no percolation with PBC.

We now turn to some results for site percolation on a 2D square lattice. Figure 5.3 shows the percolation probability, $P_L(p)$, for $L = 8$, 16, and 32. Two things are immediately obvious:

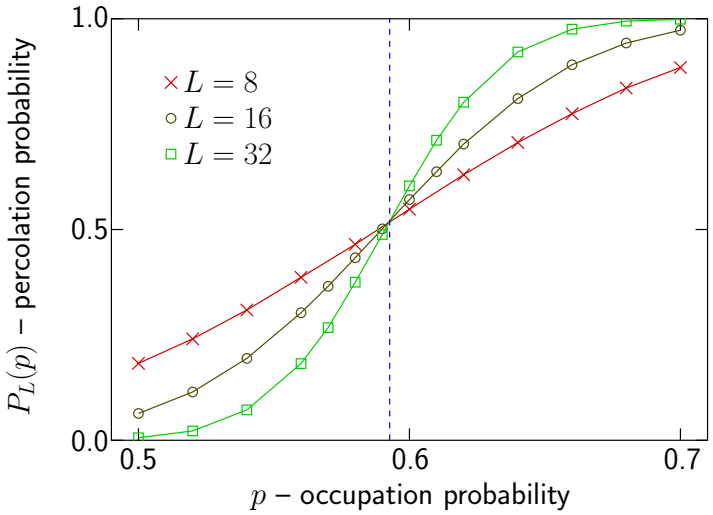


Figure 5.3: Percolation probability for three different system sizes together with the step function that holds in the limit of infinite size.

1. The lines for the different sizes cross to a good approximation at a single point.

2. The slope of the curves becomes larger for larger L

Since the slope becomes larger for larger system sizes one may infer that it is infinite in the limit $L \to \infty$ and that the function, in this limit, becomes equal to the step function $P_\infty(p) = \theta(p - p_c)$, shown by the dashed line. p_c is called the critical occupation probability.

For bond percolation on a square lattice the critical occupation probability is $p_c^{\text{bond}} = 1/2$ for symmetry reasons. For site percolation there is no corresponding symmetry and the analytic determination of p_c is still an unsolved problem. An approximate determination of p_c with simulations is however relatively simple and as we will see the simulations may also give lots of information about the behavior of the model as p_c is approached from above or from below.

5.2.1 Distribution of cluster sizes

The possibility to examine and measure all kinds of properties of the model is one of the great advantages of simulations in comparison to experiments.

The reason why this is possible is of course that all the underlying variables are accessible for investigation which is in sharp contrast to experiments that usually allow for only a small set of measurements.

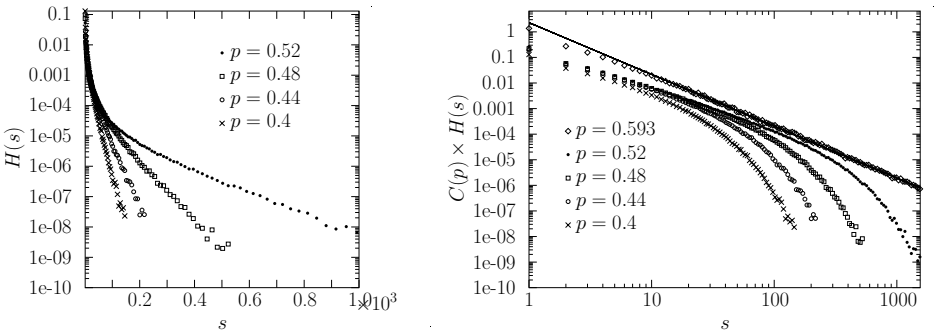


Figure 5.4: The distribution of cluster sizes. The left panel which shows the data for $p < p_c$ on a log-lin scale gives clear evidence for an exponential decay, $\sim e^{-s/\mu}$, whereas the right panel illustrates the algebraic decay of $H(s)$ at $p = 0.593 \approx p_c$.

In site percolation one such quantity that gives additional information is the distribution of cluster sizes, s. A "cluster" is here understood to contain all the interconnected sites. For small p one expects to find only rather small clusters, but as p increases towards p_c, larger clusters start to appear. To illustrate this behavior, Fig. 5.4 shows histograms of s for some different values of $p \leq p_c$. When plotted on a log-lin scale the data sets give evidence of a straight line behaviors, $\ln H = \text{const} - s/\mu$. This is the same as

$$H(s) \sim e^{-s/\mu},$$

where μ is a measure of a characteristic cluster size. From the decreasing magnitude of the slope we conclude that the characteristic size μ increases as $p \to p_c$ in accordance with the expectations. Figure 5.4a only shows data for $p < p_c$. In Fig. 5.4b the same data plotted on a log-log scale now also includes data for $p = p_c$. In that figure the characteristic cluster size is seen by the curves starting to bend downwards at different values of s.

For $p = p_c$ the data fits nicely to a straight line, $\ln H = \text{const} - a \ln s$, with $a \approx 2$. This may also be written

$$H(s) \sim s^{-a}.$$

Note that there is a crucial difference between this form and the exponential decay that the latter has no characteristic scale. This means that there is no "typical" cluster size but that there are clusters of all sizes. This shows nicely that the percolation model is scale free at p_c.

5.2.2 Fractal dimension

The usual relation between the mass m and the linear extension ℓ of an object is

$$m \sim \ell^d,$$

where d is the dimensionality; $d = 2$ in our present problem or $d = 3$ for bulky materials. Objects that obey the above relation are said to be *compact*.

In recent years Mandelbrot and others have introduced the concepts fractal and fractal dimension, d_f, to describe properties of ramified and airy objects. Objects are considered to be *fractal* if they obey a similar relation, but with $d_f < d$,

$$m \sim \ell^{d_f}.$$

This means that fractals are less dense on larger scales, which is possible only if the objects have holes of all sizes.

The percolation cluster at p_c is an example of a fractal object. Since $d_f < d$ this has the surprising consequence that the density of the percolating cluster vanishes in the large-L limit.

5.2.3 The correlation function

The distribution of cluster sizes discussed above, is the conceptually simplest quantity that provides evidence for scale invariance. Because of the fractal property of the clusters that quantity is however not useful for examining the length dependence. To get a measure of a characteristic length we instead turn to the correlation function. From the usual definition of the correlation function e.g. in the Ising model a natural definition of $g(\mathbf{r})$ could be

> $g^{\text{usual}}(\mathbf{r})$ = the probability that the position $\mathbf{r}' + \mathbf{r}$ is occupied granted that $\mathbf{r}'$ is an occupied site.

However, in the present model the occupance probability for any site is independent of all the others, and we have trivially, $g^{\text{usual}}(\mathbf{r}) = p$, which means that this function not is useful for characterizing the system.

It is nevertheless possible to define a "correlation function" by keeping in mind that the issue at focus is connectivity. We therefore define

> $g(\mathbf{r})$ = the probability to find a connected path from an arbitrary occupied position $\mathbf{r}'$ to $\mathbf{r}' + \mathbf{r}$.

The behavior of this function is qualitatively similar to the distribution of cluster sizes. For $p < p_c$ we have

$$g(r) \sim e^{-r/\xi}, \quad p < p_c,$$

and right at the percolation threshold the behavior is again scale free:

$$g(r) \sim r^{-\eta}, \quad p = p_c.$$

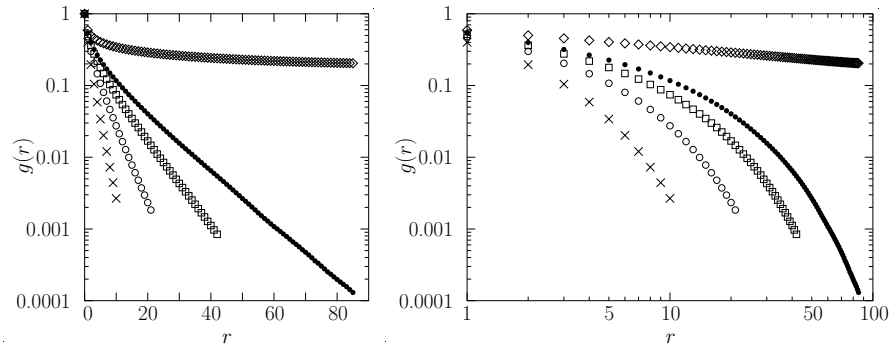


Figure 5.5: Correlation function in 2D percolation.

5.2.4 The correlation length

Analytical results for $p \to 0$

Before turning to the simulation results we will discuss the possibility of an analytic approach. It turns out that this can be done exactly in the simple 1D case and that the same results holds in arbitrary dimensions in the limit when $p \to 0$. The possibility to compare the simulation data with analytically obtained results is very valuable as a check that the simulations are really doing the right thing.

Consider first the one-dimensional case. Since $g^{1D}(r=1)$ is equal to the occupance probability we have $g^{1D}(1) = p$. This may also be generalized to arbitrary r,

$$g^{1D}(r) = p^r = e^{r \ln p}.$$

Written in this way this is seen to be an exponential decay and we may identify $\xi = -1/\ln p$.

In two dimensions the paths connecting two sites may be of many shapes but in the limit of small p the shortest path will contribute the most to the correlation function. Since these shortest paths are identical to the 1D configurations the same result for the correlation length holds in 2D (and higher dimensions) as well,

$$\xi = -1/\ln p, \quad p \to 0.$$

For larger p and 2D the correlation function gets contributions from a large number of paths and therefore decreases more slowly; we therefore expect $\xi^{2D} \geq \xi^{1D}$.

Numerical results

Figure 5.6a shows the dependency of the correlation length ξ on the occupancy probability, p in 2D. The correlation length has been obtained with the methods discussed in Sec. 7.5. As p approaches p_c (shown as a dashed line) from below, ξ is seen to increase rapidly. To analyze this behavior further we plot ξ versus $p_c - p$ with a log-log scale in Fig. 5.6b. The data is found to obey

$$\xi \sim (p_c - p)^{-\nu}, \quad \nu = 4/3,$$

where the value of the exponent ν is a known exact result.

It is similarly possible to define a correlation function that has a non-trivial behavior at $p > p_c$ by first removing the spanning cluster from the system. The above equation for the divergence of ξ may then be written in the more general form,

$$\xi \sim |p - p_c|^{-\nu}. \tag{5.1}$$

Radius of gyration and the correlation length

The radius of gyration is often used to characterize various geometric objects and may actually also be used to determine the correlation length. With a

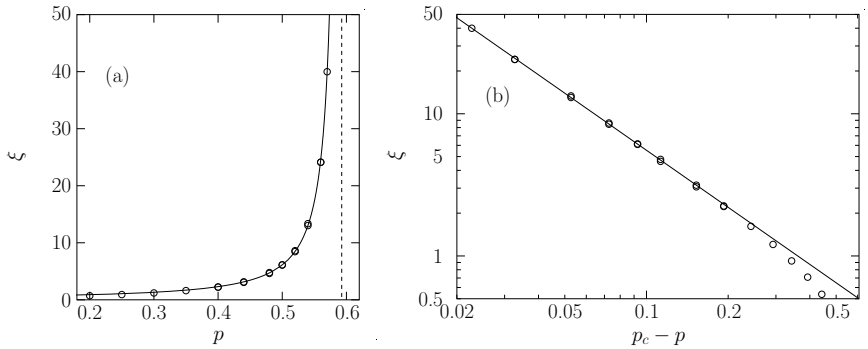


Figure 5.6: Correlation length in site percolation in 2D.

cluster of size s consisting of the occupied points $\mathbf{r}_i$, $i = 1 \ldots s$, the center of mass is $\bar{\mathbf{r}} = (1/s)\sum_i \mathbf{r}_i$, and the radius of gyration is defined as

$$R^2 = \frac{1}{s}\sum_{i=1}^{s}(\mathbf{r}_i - \bar{\mathbf{r}})^2. \tag{5.2}$$

Note that the center of mass, and thereby the radius of gyration, cannot be defined on the percolating cluster. For the below expressions to work properly we therefore need to remove the percolating cluster if it happens to appear. Below p_c, where a percolating cluster is a finite size effect, it is however much better to try and do the analysis on systems that are large enough that they practially never appear.

Now assume that the distribution of distances between pairs of points in the same cluster falls off exponentially, $P(r) \sim e^{-r/\xi}$. We then have

$$\left\langle r^2 \right\rangle = \frac{\int dr\ r^2 e^{-r/\xi}}{\int dr\ e^{-r/\xi}}.$$

By partial integration (with the limits at $r = 0$ and infinity) we have

$$\begin{aligned}\int dr\ r^2 e^{-r/\xi} &= \left[(-\xi)r^2 e^{-r/\xi}\right] + \xi \int dr\ 2re^{-r/\xi} \\ &= 2\xi^2 \int dr\ e^{-r/\xi},\end{aligned}$$

which gives $\langle r^2 \rangle = 2\xi^2$.

To apply this to site percolation with a number of non-spanning clusters we write

$$\xi_g^2 = \frac{1}{2}\left\langle (\mathbf{r}_i^c - \mathbf{r}_j^c)^2 \right\rangle = \frac{\sum_c \sum_{ij} (\mathbf{r}_i^c - \mathbf{r}_j^c)^2}{2\sum_c s_c^2},$$

where the index c is a cluster identifier, s_c is the cluster size, and the indices i and j only include sites that belong to a given cluster. To relate this expression to Eq. (5.2), define $\mathbf{u}_i = \mathbf{r}_i - \bar{\mathbf{r}}$ and make use of $\sum_i \mathbf{u}_i = 0$:

$$\sum_{ij}(\mathbf{r}_i - \mathbf{r}_j)^2 = \sum_{ij}(\mathbf{u}_i - \mathbf{u}_j)^2 = \sum_{ij}\mathbf{u}_i^2 + \sum_{ij}\mathbf{u}_j^2 + 2\sum_i \mathbf{u}_i \sum_j \mathbf{u}_j = 2s\sum_i(\mathbf{r}_i - \bar{\mathbf{r}})^2. \tag{5.3}$$

Putting all this together gives the correlation length in terms of the radius of gyration for each cluster,

$$\xi_g^2 = \frac{\sum_c s_c^2 R_c^2}{\sum_c s_c^2},$$

or in terms of R_s – the average radius of gyration of a cluster of size s,

$$\xi_g^2 = \frac{\sum_s n_s s^2 R_s^2}{\sum_s n_s s^2},$$

where n_s is the number of clusters of size s. Note that the above equations give considerably more weight to larger clusters than what would be obtained by just taking a straight average of the radii of gyration of all clusters.

5.2.5 The order parameter

The order parameter in the Ising model is the magnetization. The corresponding quantity in percolation theory is the relative size of the spanning cluster,

$$q^{\text{span}} = s^{\text{span}}/L^d.$$

The exponent β is defined from the vanishing of the order parameter,

$$q^{\text{span}} \sim (p - p_c)^\beta. \tag{5.4}$$

For percolation in 2D the value is known exactly, $\beta = 5/36$.

The size dependence of the order parameter q^{span} is shown in Fig. 5.7, together with a curve that shows the approach to zero $\sim (p - p_c)^\beta$ for the $L \to \infty$ limit. Note that larger sizes are needed when we like to probe the behavior closer to p_c.

The above equation implies that the spanning cluster right at the percolation threshold in the limit $L \to \infty$ constitutes a vanishing fraction of the total system. The same conclusion was also reached in Sec. 5.2.2 and was there found to be a consequence of the fractal property of the percolating cluster.

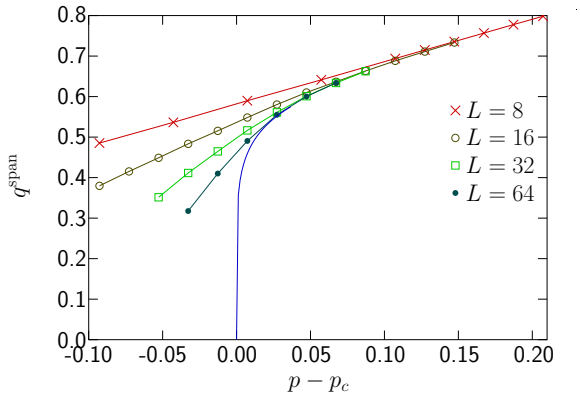


Figure 5.7: The relative size of the spanning cluster at p_c together with the behavior in the large-L limit.

5.2.6 Average size of non-spanning clusters

Another quantity that characterizes the phase transition is the average size of the non-spanning clusters,

$$S(p) = \frac{1}{N_c} \sum_{c}^{N_c} s_c.$$

In analogy with the susceptibility this quantity behaves as

$$S(p) \sim |p - p_c|^{-\gamma}, \tag{5.5}$$

with $\gamma = 43/18$ in 2D.

5.2.7 Finite size scaling in percolation

As discussed above our percolation model has a diverging correlation length, $\xi \sim |p - p_c|^{-\nu}$. When one is doing simulations on finite systems one expects the behavior close to p_c to be a function of L/ξ, or, equivalently a function of $(L/\xi)^{1/\nu}$ which may also be written $(p - p_c)L^{1/\nu}$. For a quantity A with scaling dimension y_A the standard finite size scaling assumption is then

$$A_L(p) \sim L^{y_A/\nu} f_A((p - p_c)L^{1/\nu}),$$

where f_A is a scaling function. The scaling dimension may sometimes be deduced through scaling arguments but is otherwise obtained from finite size scaling of the data.

Percolation probability

The simplest quantity for finite size scaling analysis is the percolation probability $P_L(p)$ shown in Fig. 5.3. A scaling collapse of the same data is shown in Fig. 5.8.

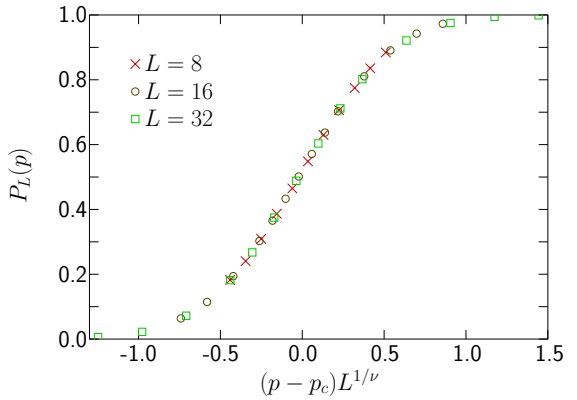


Figure 5.8: Scaling collapse of the percolation probability for three different system sizes.

The order parameter The size dependence of the order parameter q^{span} was shown in Fig. 5.7. To collapse the data one plots $L^{\beta/\nu}q^{\text{span}}$ versus $(p - p_c)L^{1/\nu}$ as shown in Fig. 5.9.

5.3 Random walk

Random walk is another simple stochastic problem with numerous applications as e.g. diffusing atoms or the shape of polymers. Whereas random walks often live in a continuum, we will here put them on a lattice. The reason is, as before, that the interesting behavior not should depend on such details and that this will simplify and speed up the simulations.

5.3.1 Simple random walk

A simple random walk may be described as a number of vectors $\mathbf{d}_i$ with unit length but random directions in a d-dimensional space. How far will such a

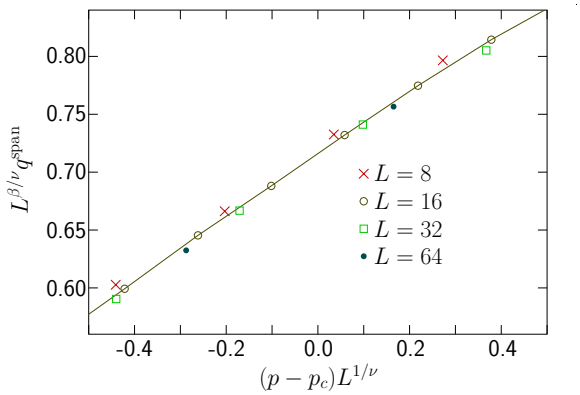


Figure 5.9: Scaling collapse of $L^{\beta/\nu}q^{\mathrm{span}}$ in a narrow region around p_c. Note that the data for the smallest size deviates slightly. Such effects are called corrections to finite size scaling.

walk typically reach after N steps? Define

$$\mathbf{S} = \sum_{i=1}^{N} \mathbf{d}_i.$$

Since the steps are independent of one another we have

$$\langle \mathbf{d}_i \cdot \mathbf{d}_j \rangle = \begin{cases} 1, & i = j, \\ 0, & i \neq j, \end{cases}$$

independent of the dimensionality, and the expectation value of $\mathbf{S}^2$ is,

$$\left\langle \mathbf{S}^2 \right\rangle = \left\langle \sum_i \mathbf{d}_i \cdot \sum_j \mathbf{d}_j \right\rangle = \left\langle \sum_{i=1}^{N} \mathbf{d}_i^2 \right\rangle + \left\langle \sum_{i=1}^{N} \sum_{j \neq i} \mathbf{d}_i \cdot \mathbf{d}_j \right\rangle = N, \tag{5.6}$$

which means that the root-mean-square distance is

$$\sqrt{\langle \mathbf{S}^2 \rangle} = \sqrt{N}.$$

Note that this result holds independent of dimension, d. Since this result describes the dependence of a typical length, we write $\nu = 1/2$.

Another quantity that characterizes the random walk is the radius of gyration of the $N+1$ points $\mathbf{r}_0, \ldots, \mathbf{r}_N$ of the random walk with N steps,

$$R^2 = \frac{1}{N+1} \sum_{i=0}^{N} (\mathbf{r}_i - \bar{\mathbf{r}})^2.$$

With the use of Eq. (5.3) this may be written

$$R^2 = \frac{1}{2(N+1)^2} \sum_{ij} (\mathbf{r}_i - \mathbf{r}_j)^2,$$

and together with $\langle (\mathbf{r}_i - \mathbf{r}_j)^2 \rangle = |i-j|$ and $N+1 \approx N$, which holds for large N, this becomes

$$\left\langle R^2 \right\rangle = \frac{1}{2N^2} \int_0^N dx \int_0^N dy \, |x-y| = \frac{1}{N^2} \int dx \int_0^x dz \, z = \frac{1}{N^2} \int_0^N dx \frac{x^2}{2} = \frac{N}{6},$$

and together with Eq. (5.6) we find

$$\frac{\langle S_N^2 \rangle}{\langle R_N^2 \rangle} = 6.$$

5.3.2 Self-avoiding walk

After this short introduction to the analytically solvable simple random walk we now turn to the more interesting and considerably more difficult case of self-avoiding random walk. We will now sketch three different methods with increasing sophistication. Even though a "better" method of course normally is much more efficient than a simple one it is always wise to start out with a program that implements the simplest possible method. The results from that program may then be used to verify that the more efficient and complicated program actually is correct.

We would really like to calculate the averages over the whole set of distinct self-avoiding paths. The methods below that select samples out of this huge set, therefore need be constructed with care to chose samples that are representative of the whole set.

Random generation

The simplest method is to just generate a number of random walks and then discard the ones that happen to be self-intersecting. In practice this is done by aborting the run when a site is visited a second time. In this method we choose by random between $z-1$ different directions and therefore at the outset discards the possibility to go back along the same path.

This method works nicely for medium long random walks, but because the run has to be aborted at self-intersection it becomes difficult to obtain good statistics for runs with say $N = 100$ in two dimensions.

Survival biasing

In the next level of sophistication we only choose among the steps that are acceptable, i.e. non-self-intersecting. It is then necessary to compensate for this biased choise by introducing a weight factor for all such steps. With z_i acceptable directions for step i to choose among out of the $z-1$ non-backtracking directions, the walk should be weighted by a factor $z_i/(z-1)$. Each walk then gets a total weight factor

$$w_\mu = \prod_{i=1}^{N} \frac{z_i}{z-1}, \tag{5.7}$$

which is used to calculate the averages of various quantities,

$$\langle Q \rangle = \frac{\sum_\mu w_\mu Q_\mu}{\sum_\mu w_\mu}. \tag{5.8}$$

This is our first example of *survival biasing* that is a common method in Monte Carlo simulations of particle transport, cf. Sec. 9.2.4.

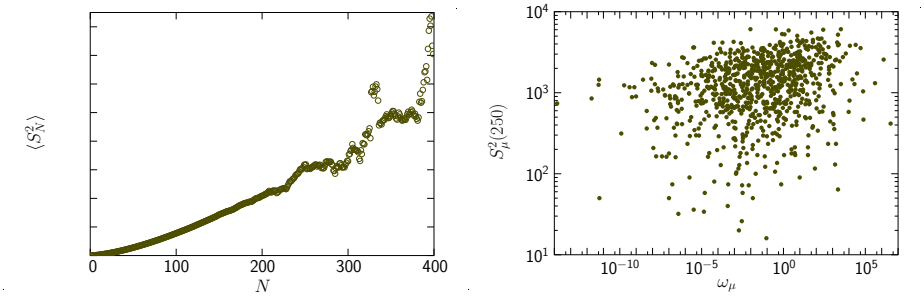


Figure 5.10: End-to-end distance for walks of length N from survival biasing. The left figure which is $\langle S_N^2 \rangle$ vs. N obtained from 10^8 random walks, shows that the precision in S^2 for longer walks—say with $N > 200$—are bad. The explanation is that there is a huge spread in the weight factors ω_μ. This is shown in the right figure (from a shorter run with 10^5 random walks) with the main message that ω_μ (here determined for $N = 250$) spreads a lot. Shown on the y axis are the corresponding values of $S_\mu^2(N = 250)$. From this figure it becomes clear that the numerator of Eq. (5.8) will be dominated by a small number of terms. (For this calculation $z-1$ (=3) in the denominator of Eq. (5.7) was replaced with 2.6, to avoid getting very small values of ω_μ.)

There are limitations that affect the use of this method as well. As shown in Fig. 5.10 the spread in w_μ becomes large for very long walks and this has

adverse effects on the calculations since the average may be dominated by a small number of walks. For very long walks it is then better to consider the next level of sophistication.

Chain of configurations

An even more efficient method is the pivot algorithm. Instead of producing new random walks from scratch over and over again, the idea is now to modify an existing walk. The following steps are then performed many times:

1. Choose a point along the chain at random.
2. Select a symmetry operation (see below) by random and perform that operation on one part of the chain.
3. Accept this modified chain if it is self-avoiding. Otherwise, restore the old configuration.
4. Measure and collect various properties of the chain.

The symmetry operations can be rotations, reflections or combinations of the two. The most time-consuming part is to check for self-avoidance. With the pivot point denoted by $\mathbf{r}_p$ we need to check for each $0 \leq i < p$ and $p < j \leq N$ whether $\mathbf{r}_i = \mathbf{r}_j$. This is a test where the average number of operations goes as N^2.

The pivot algorithm is yet an example of Markov chain Monte Carlo and that also implies correlations between generated configurations and the need for "burn in" in the beginning of the simulation:

- It is perfectly acceptable to start from a straight walk and apply the pivot algorithm over and over again. It is however clear that the first configurations would be biased towards untypically large end-to-end distances. Before one starts to collect data it is therefore necessary to apply the pivot algorithm quite a few times. In thermal simulations this corresponds to the *thermalization* of the system.
- The random walks produced in this way are not independent but strongly correlated to one another. This has to be considered when estimating the statistical uncertainities.

5.4 Self-organized criticality

We have now the existence of scale free behavior both in the Ising model and in site percolation. In both cases we could define a correlation length that diverges at criticality and it was also found that the correlation function in both cases decay algebraically. In percolation the distribution of cluster sizes again gave evidence for a scale free behavior. Similar behaviors are found in many different models. One thing these models have in common is that one has to adjust some parameter (e.g. temperature or occupation probability) in order to see the scale free behavior.

As mentioned above scale free behaviors are common in nature, one example being the Gutenberg-Richter law for the occurence of earthquakes. In this case it is not obvious that there is any parameter that happens to be precisely at the critical value. Instead, the processes behind the earthquakes are postulated to have the property to automatically adjust their properties such that they become scale-free. This is what lies behind the name *self-organized* criticality.

Self-organization is maybe best explained with the sand pile model. Assume that we drop grains of sand one at a time on a sand pile. The effect would be that the slope of the sand pile would gradually increase and when it becomes big enough we would have an avalanche. Whereas most avalanches are small and only involve a few grains of sand they would occasionally be very big and the distribution of avalanche sizes could (hopefully) be shown to be algebraic. Whereas the experiments on real sand piles do not quite show the expected behavior (rice piles do better) the sand pile is the paradigmatic example of self-organized criticality.

5.4.1 The sand pile model in one dimension

We will now describe three different 1D sand pile models to see what is needed to get a non-trivial behavior. The sand pile is described by a height variable, h_x for $x = 0$ through $L - 1$.

- We first try with the following rule: (1) at each time step add one grain of sand at $x = 0$. (2) For each site with $h_x - h_{x+1} > 1$, move a grain of sand to the left.

$$\begin{aligned} h_x &\rightarrow h_x - 1, \\ h_{x+1} &\rightarrow h_{x+1} + 1 \end{aligned}$$

This gives a simple behavior with $H(s) = \text{const.}$

- Next, let the toppling event mean that two grains of sand are moved to the right and land at the neighbor to the right and the next nearest neighbor to the right. If $h_x - h_{x+1} > 2$,

$$\begin{aligned} h_x &\rightarrow h_x - 2, \\ h_{x+1} &\rightarrow h_{x+1} + 1, \\ h_{x+2} &\rightarrow h_{x+2} + 1. \end{aligned}$$

- To get a non-trivial behavior we need to introduce some randomness. That may be done by adding the grain of sand to a site chosen by random. Considering the distribution of sizes of avalanches, s, we then find an algebraic distribution as shown in Fig. 5.11.

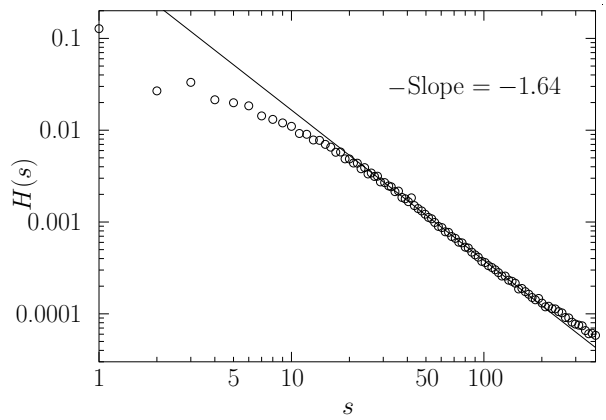


Figure 5.11: Distribution of sizes of avalanches in a 1D sand pile.

5.4.2 The sand pile model in two dimensions

A popular sand pile model lives on a 2D lattice with coordinates $\mathbf{r} = (x, y)$, $0 \leq x < L$ and $0 \leq y < L$. It is most common to model the local slope and the rules for the model are:

1. Choose a site $\mathbf{r}$ by random and increase the slope by one,

$$Z_{\mathbf{r}} \rightarrow Z_{\mathbf{r}} + 1.$$

2. If Z reaches the critical value $Z_{\rm cr} \geq 4$, spread out the slope at the four neighboring sites, $\mathbf{r}'$,

$$\begin{aligned} Z_{\mathbf{r}} &\rightarrow Z_{\mathbf{r}} - 4, \\ Z_{\mathbf{r}'} &\rightarrow Z_{\mathbf{r}'} + 1. \end{aligned}$$

   Iterate this until $Z_{\mathbf{r}} < Z_{\rm cr}$ for all $\mathbf{r}$. For this iteration note that the update should be synchronous. This is obtained by *first* checking for the sites with $Z_{\mathbf{r}} = Z_{\rm cr}$ and *after* that updating the Z.

The model we have described has been compared to a set of lazy bureaucrates in an office with their tables arranged in a square grid. Even now and then a random bureaucrat gets a piece of paper from the outside. He doesn't deal with it until he finds too many papers on his desk. He then sends one piece of paper to each of his four neighbors. All bureaucrates handle the pieces of paper in the same way except for those placed at the wall who simply throws one of the papers through the window.

In this model one has found power laws in distributions of both the size and the time of the avalanche, which are defined as follows:

- The size S of an avalanche is given by the number of times the critical value has been reached.

- The time τ of an avalanche is the number of times that synchronous updates have been performed. Since several sites may be affected at a single time step one usually finds that $\tau < S$ at all but the smallest sizes.

5.5 Complex networks

The different models considered above all have the property in common that they live in some simple space in three or two dimensions. It is then computationally convenient to put the variables on some kind of lattice.

It is however not difficult to find phenomena that cannot be put on any kind of regular lattice. Some examples are relations between people, links in html-documents, metabolic pathways, and the neural network in the brain. The field of Complex networks attempts to answer questions related to properties of such networks.

5.5.1 What is a network?

A network consists of some *nodes* connected by *links*. (Or, *vertices* connected by *edges*.) It should be noted that links are of two different kinds, directed/undirected. In some cases there is also a capacity related to each link and there are many other possible generalizations.

5.5.2 Examples of networks:

- Social networks: a set of people with certain interactions between them.
- Information networks: citation networks, links in html-documents.
- Technological networks: the electric power grid, internet (direct contact between computers).
- Biological networks:
  - neural network in the brain
  - the metabolic pathways (node: chemical reaction, link: molecule)
  - the genetic regulatory network (node: presence or absence of a particular protein (A), link: the effect of protein A on the expression of a gene and thereby the production of protein B.
  - the food web, node: a species, links specify predator-prey interactions.
  - blood vessels.

5.5.3 Small world networks – Watts & Strogatz

An often mentioned experiment was performed by Stanley Milgram in 1960. In his experiment a number of letters were able to reach a destinated target individual in on the average six steps. The restriction was that each person handling the letter was only allowed to hand it over to one of his aquintancies. The usual interpretation is that a connection may be made between two arbitrary persons through only six steps – the small world phenomena.

In the pioneering work by Watts & Strogatz[4] they pointed out that many naturally occuring networks in some sense are between random networks and ordered networks. A characteristic property of a random network is precisely the small world phenomena, that the mean path length increases very slowly (logarithmically) with system size. With d_{ij} denoting the minimum path length (number of links) between node i and j the mean path length is defined as

$$\ell = \frac{1}{\frac{1}{2}n(n-1)} \sum_{i \geq j} d_{ij}$$

In ordered networks on the other hand the mean path length grows linearly with system size but a property that characterizes ordered networks is clustering. Clustering means that the probability for a link between node i and j is large if they are both linked to another node k. The clustering coefficient is defined as

$$C = \frac{3 \times \text{number of triangles in the network}}{\text{number of connected triples of vertices}}$$

The interesting thing is now that many naturally occuring networks seem to have properties in common both with random and ordered networks; both a large clustering coefficient and a small mean path length. Some examples of networks with both these properties are

- Film actors
- Power grid
- Neuron connections in *C. elegans*

Watts and Strogatz also suggested a simple model that may be gradually taken from an ordered network to a random one. The model is illustrated in Fig. 5.12

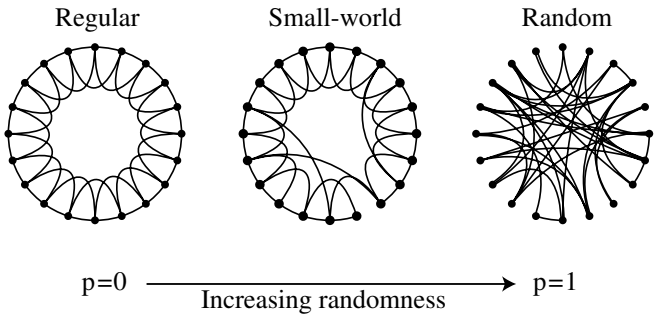


Figure 5.12: The Watts & Strogatz model.

The starting point is a perfectly ordered network with n nodes and links both between nearest and next nearest neighbors. The idea to introduce randomness is to step through all links and *rewire* the link with probability p. The dependence of the mean path length and the clustering coefficient is shown in Fig. 5.13. Note that there is a large region where the system has both a small mean path length ℓ (typical of random systems) and a large clustering coefficient (typical of ordered networks).

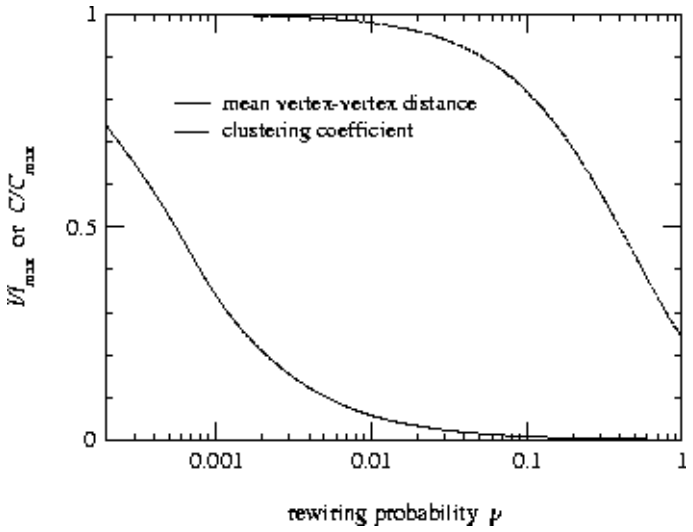


Figure 5.13: The mean path length ℓ and the clustering coefficient C as a function of the rewiring probability p for the Watts & Strogatz model.

An important concept to describe describe the resulting network is the degree, k which is the number of links per node. For the perfectly ordered system $k = 4$ for each link but when randomness is introduced the different nodes will have different k the system may then be described with a degree distribution p_k.

5.5.4 Degree distribution – Barabási & Albert

The degree distribution of a random network is either a binomial or a Poisson distribution but it turns out that many real world networks has a rather different behavior where p_k has a tail up to values that are far above the mean. Some examples of networks with that property are:

- Actor collaborations – a link exists between two actors if they have played in the same movie.
- The world wide web – the links are the references from one web page to another.
- The connectivity of the power grid for the USA.

For several of these networks the degree distribution obeys $p_k \sim k^\alpha$; they are called *scale-free networks*.

Barabási & Albert have suggested a way to construct networks with a scale-free degree distribution[5]. The basic idea is to add new nodes to the network one after another. The links that belong to the new node are then more likely to attach to a node with a higher degree. This is called preferential attachment. The probability for a certain link to connect to node i is given by

$$\Pi(i) = \frac{k_i}{\sum_j k_j}.$$

5.5.5 Present research

The short description above of the papers by Watts & Strogatz and Barabási & Albert[5] is just a brief introduction to the research on networks. There is by now several hundreds of papers that try to attack networks in different ways. So far the research is however mainly descriptive and many different quantities have been defined to try to capture the essential behavior of different networks.

Many people working in this area have a background in statistical physics and are used to models that belong to various universality classes characterized by certain values of the critical exponents. Are there any (more or less similar) rules for the complex networks that remain to be unveiled. As things stand today nobody knows, and the alternative – that there are no really unifying principles that govern the properties of networks – still seems to be a possibility.

The present research of networks focuses on three different questions:

1. Describe the structure in different networks.
2. How are complex networks formed?
3. Try to understand processes on networks
    - Stability of networks against failuring nodes. (Where the power grid is an interesting and important application.)
    - Epidemological processes
    - Search on networks
    - Phase transitions on networks

Chapter 6

Quantum Monte Carlo with the SSE method

The stochastic series expansion method (SSE) is a recent addition to the methods for doing quantum Monte Carlo and because of certain advantages before other methods it is rapidly becoming popular. The simplest model is perhaps the spin 1/2 model with nearest-neighbor interactions and before describing the method we briefly review some basic facts from quantum mechanics.

6.1 Basic relations for quantum spins

6.1.1 A single spin

A spin has three components, s_x, s_y, and s_z but since their corresponding operators do not commute we can only known the value of one of these at the same time. The standard choice is to use spin states that are eigenvalues of the S^z operator,

$$S^z\phi = s^z\phi.$$

The eigenstates are $|\uparrow\rangle$ and $|\downarrow\rangle$, with eigenvalues

$$\begin{aligned} S^z\,|\uparrow\rangle &= \frac{\hbar}{2}|\uparrow\rangle, \\ S^z\,|\downarrow\rangle &= -\frac{\hbar}{2}|\downarrow\rangle. \end{aligned}$$

The commutation relations for the operators in the three different directions are

$$[S^x, S^y] = i\hbar S^z, \quad [S^y, S^z] = i\hbar S^x, \quad [S^z, S^x] = i\hbar S^y.$$

It is sometimes convenient to use a vector notation; the eigenstates are then

$$|\uparrow\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |\downarrow\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

and the operators are given by the following matrices

$$S^x = \frac{\hbar}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad S^y = \frac{\hbar}{2}\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad S^z = \frac{\hbar}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

In the following we will need the step operators with the properties

$$\begin{aligned} S^+|\uparrow\rangle &= |0\rangle, \\ S^+|\downarrow\rangle &= \hbar|\uparrow\rangle, \\ S^-|\uparrow\rangle &= \hbar|\downarrow\rangle, \\ S^-|\downarrow\rangle &= |0\rangle, \end{aligned}$$

and it is easy to see that they will be given by the matrices

$$S^+ = \hbar\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad S^- = \hbar\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}.$$

Comparing with the matrices for S^x and S^y it is clear that they are given by

$$\begin{aligned} S^+ &= S^x + iS^y, \\ S^- &= S^x - iS^y, \end{aligned}$$

and we also have

$$S^x = \frac{1}{2}(S^+ + S^-), \quad iS^y = \frac{1}{2}(S^+ - S^-).$$

6.1.2 A model of interacting spins

The starting point is the Hamiltonian for the Heisenberg antiferromagnet,

$$\begin{aligned} \mathcal{H}_J &= J\sum_{\langle ij\rangle} \mathbf{S}_i \cdot \mathbf{S}_j - h\sum_i S_i^z, \\ &= J\sum_{\langle ij\rangle} \left[S_i^x S_j^x + S_i^y S_j^y + S_i^z S_j^z - \frac{h}{2J}(S_i^z + S_j^z) \right] \end{aligned}$$

The first two terms may then be written

$$\begin{aligned} S_i^x S_j^x + S_i^y S_j^y &= \frac{1}{4}(S_i^+ + S_i^-)(S_j^+ + S_j^-) - \frac{1}{4}(S_i^+ - S_i^-)(S_j^+ - S_j^-) \\ &= \frac{1}{4}(S_i^+ S_j^- + S_i^- S_j^+ + S_i^+ S_j^- + S_i^- S_j^+) \\ &= \frac{1}{2}(S_i^+ S_j^- + S_i^- S_j^+), \end{aligned}$$

and the Hamiltonian becomes

$$\mathcal{H}_J = J \sum_{\langle ij \rangle} \left[S_i^z S_j^z + \frac{1}{2} \left(S_i^+ S_j^- + S_i^- S_j^+ \right) \right], \tag{6.1}$$

6.1.3 Natural basis vectors

The natural basis vectors for a 1D spin model look like $|\uparrow\uparrow\downarrow \cdots \downarrow\uparrow\rangle$. For a spin-1/2 system with N spins there are 2^N such states which we will denote by $|\alpha_k\rangle$. We assume normalization,

$$\langle \alpha_k | \alpha_\ell \rangle = \delta_{k\ell}.$$

These states are however not eigenvectors to the Hamiltonian since the step operators have the effect to flip the spins,

$$S_1^+ S_2^- \, |\downarrow\uparrow\rangle = \hbar^2 \, |\uparrow\downarrow\rangle.$$

Generally speaking the application of the Hamiltonian gives

$$\mathcal{H}_J |\alpha_k\rangle = \sum_\ell C_{k\ell} |\alpha_\ell\rangle,$$

where $C_{k\ell}$ is a matrix. One way to approach a quantum mechanical problem is to determine the eigenvectors $|\phi_k\rangle$ and eigenvalues E_k of the matrix C that obey

$$\mathcal{H}_J |\phi_k\rangle = E_k |\phi_k\rangle.$$

This is OK for reasonably small systems. A spin chain with $N = 8$ spins gives $2^N = 256$ states which requires the solution of a 256×256 matrix. However, $N = 64$ would generate a matrix of dimension $10^{19} \times 10^{19}$. This makes it perfectly clear that the study of all but the smallest sizes will need other methods, as e.g. Monte Carlo.

6.1.4 Expectation values

The partition function is

$$Z = \sum_\alpha \langle\alpha|e^{-\mathcal{H}_J/k_BT}|\alpha\rangle = \sum_\alpha \langle\alpha|e^{-\beta\mathcal{H}}|\alpha\rangle,$$

where the dimensionless Hamiltonian is $\mathcal{H} = \mathcal{H}_J/J$, $\beta = J/k_BT$. The expectation values may be written

$$\langle A\rangle = \frac{1}{Z}\sum_\alpha \langle\alpha|A(\alpha)e^{-\beta\mathcal{H}}|\alpha\rangle.$$

From our previous experience of Monte Carlo it is clear that we want to generate the configurations $|\alpha\rangle$ with probability $\propto \langle\alpha|e^{-\beta H}|\alpha\rangle$ but in order to do that we need another set of manipulations.

6.1.5 Expand the Boltzmann factor

With $e^x = \sum_{n=0}^\infty x^n/n!$ the partition function may be written as a power series expansion,

$$Z = \sum_\alpha \sum_{n=0}^\infty \frac{\beta^n}{n!}\langle\alpha|(-\mathcal{H})^n|\alpha\rangle.$$

Already $H|\alpha\rangle$ is difficult to handle and $H^n|\alpha\rangle$ is even worse, but since the Hamiltonian consists of a sum of several terms,

$$-\mathcal{H} = \sum_i H_i,$$

we have

$$(-\mathcal{H})^n = \sum_{i_n} H_{i_n} \cdots \sum_{i_1} H_{i_1}$$

For each fixed n we then have

$$\langle\alpha|(-\mathcal{H})^n|\alpha\rangle = \sum_{\{i_n\}} \langle\alpha|H_{i_n}\cdots H_{i_1}|\alpha\rangle,$$

and the advantage with this step is that a single term $H_{i_n}\cdots H_{i_1}|\alpha\rangle$ is a simple thing that is possible to handle.

6.1.6 The splitting of the Hamiltonian

As seen in Eq. (6.1) the term for each ij pair may be split into a term that leaves the configuration unchanged (a diagonal part) and a part with step operators which changes the configuration (an off-diagonal part). We then write the Hamiltonian as a sum over the N_b bonds,

$$\mathcal{H} = \sum_{\langle ij \rangle} \cdots \quad \Rightarrow \quad -\mathcal{H} = \sum_{b=1}^{N_b} \left(\mathcal{H}_b^{\rm diag} + \mathcal{H}_b^{\rm off} \right),$$

where

$$\mathcal{H}_b^{\rm diag} = C - S_{i(b)}^z S_{j(b)}^z + \frac{h}{2J}(S_{i(b)}^z + S_{j(b)}^z),$$

and

$$\mathcal{H}_b^{\rm off} = -\frac{1}{2}(S_{i(b)}^+ S_{j(b)}^- + S_{i(b)}^- S_{j(b)}^+).$$

We here also introduce a constant C which is necessary to avoid negative values. Our dimensionless Hamiltonian is now

$$\mathcal{H} = \sum_{\langle ij \rangle} \left[S_i^z S_j^z + \frac{1}{2} \left(S_i^+ S_j^- + S_i^- S_j^+ \right) - C - \frac{h}{2J}(S_i^z + S_j^z) \right], \tag{6.2}$$

and we get

$$\begin{aligned} (-\mathcal{H})^n &= \left(\sum_{b_n} \mathcal{H}_{b_n}^{\rm diag} + \sum_{b_n} \mathcal{H}_{b_n}^{\rm off} \right) \cdots \left(\sum_{b_1} \mathcal{H}_{b_1}^{\rm diag} + \sum_{b_1} \mathcal{H}_{b_1}^{\rm off} \right) \\ &= \sum_{\{\mu_n\}} \sum_{\{b_n\}} \mathcal{H}_{b_n}^{\mu_n} \cdots \mathcal{H}_{b_1}^{\mu_1}. \end{aligned}$$

In the last step we introduce $\mu_i =$ "diag", "off" and $\{\mu_n\}$ as a sequence of n such variables.

6.1.7 The partition function

The partition function may now be written

$$Z = \sum_{n=0}^{\infty} \frac{\beta^n}{n!} \sum_{\{\mu_n\}} \sum_{\{b_n\}} \sum_{\alpha} \langle \alpha | \mathcal{H}_{b_n}^{\mu_n} \cdots \mathcal{H}_{b_1}^{\mu_1} | \alpha \rangle,$$

and the expectation value becomes

$$\langle A\rangle = \frac{1}{Z}\sum_{n=0}^{\infty}\frac{\beta^n}{n!}\sum_{\{\mu_n\}}\sum_{\{b_n\}}\sum_{\alpha} A(\alpha)\langle\alpha|\mathcal{H}_{b_n}^{\mu_n}\cdots\mathcal{H}_{b_1}^{\mu_1}|\alpha\rangle.$$

For a small system, say $N = 4$, it is possible to peform these summations but for bigger systems that is not possible. It is however possible to construct a Markov chain in the configuration space. In this context a "configuration" is a specification of n, $\{b_n\}$, $\{\mu_n\}$, and α. In the following we will use a graphic representation of the configurations. The table below illustrates the configuration with $n = 4$, $\{b_n\} = 1, 2, 3, 2$, $\{\mu_n\} =$ diag, off, diag, off, and $|\alpha\rangle = |\uparrow\uparrow\downarrow\downarrow\rangle$,

$$\mathcal{H}_2^{\text{off}}\mathcal{H}_3^{\text{diag}}\mathcal{H}_2^{\text{off}}\mathcal{H}_1^{\text{diag}}\,|\uparrow\uparrow\downarrow\downarrow\rangle.$$

The table uses the the notation $|\alpha(p)\rangle$ for the normalized basis vector after the operation of p operators on $|\alpha\rangle$. The diagonal operators are marked with an $=$ sign whereas the off-diagonal ones are given by $\Leftrightarrow$. Note that the off-diagonal operators interchange the spins whereas the diagonal ones leave them unchanged.

p	$\mathcal{H}_{b_p}^{\mu_p}$	1		2		3		4	$\mathcal{H}_{b_p}^{\mu_p}\|\alpha(p-1)\rangle$
0		$\uparrow$		$\uparrow$		$\downarrow$		$\downarrow$	$\|\alpha(0)\rangle$
	$\mathcal{H}_1^{\text{diag}}$		$=$						
1		$\uparrow$		$\uparrow$		$\downarrow$		$\downarrow$	$(C-\frac{1}{4})\,\|\alpha(1)\rangle$
	$\mathcal{H}_2^{\text{off}}$				$\Leftrightarrow$				
2		$\uparrow$		$\downarrow$		$\uparrow$		$\downarrow$	$\frac{1}{2}\,\|\alpha(2)\rangle$
	$\mathcal{H}_3^{\text{diag}}$						$=$		
3		$\uparrow$		$\downarrow$		$\uparrow$		$\downarrow$	$(C-\frac{1}{4})\,\|\alpha(3)\rangle$
	$\mathcal{H}_2^{\text{off}}$				$\Leftrightarrow$				
4		$\uparrow$		$\uparrow$		$\downarrow$		$\downarrow$	$\frac{1}{2}\,\|\alpha(4)\rangle$

For this spin vector and sequence of operators the final spin configuration is the same as the initial one. Since the states are orthogonal, $\langle\alpha'|\alpha\rangle = \delta_{\alpha',\alpha}$, it is only the configurations with this property that contribute to the sum. The matrix element is

$$\langle\uparrow\uparrow\downarrow\downarrow|\mathcal{H}_2^{\text{off}}\mathcal{H}_3^{\text{diag}}\mathcal{H}_2^{\text{off}}\mathcal{H}_1^{\text{diag}}\,|\uparrow\uparrow\downarrow\downarrow\rangle = \left(\frac{1}{2}\right)^2\left(C-\frac{1}{4}\right)^2$$

Note also that the off-diagonal operators always connect sites with different spins; the application of the operator would otherwise give the null state, $|0\rangle$ and a vanishing matrix element.

6.2 Monte Carlo update steps

The task is now to generate configurations specified by $|\alpha\rangle$, n, and the sets $\{\mu_n\}$ and $\{b_n\}$. We usually want them to be proportional to the Boltzmann factor but in the present case the probability for the states should instead be proportional to

$$\frac{\beta^n}{n!}\langle\alpha|\mathcal{H}_{b_n}^{\mu_n}\cdots\mathcal{H}_{b_1}^{\mu_1}|\alpha\rangle.$$

The update steps are of two different kinds:

1. The operator loop—steps that change operators from diagonal to off-diagonal and vice versa. This is done by flipping the spins along a certain loop. This operation changes $\{\mu_n\}$ but leaves n and $\{b_n\}$ unchanged. Such a loop cluster often changes the spin states $|\alpha\rangle$.

2. Diagonal updates—steps that create or destroy diagonal operators. This leads to a change of n and one of the b_i but $|\alpha\rangle$ and all the other μ_i are left unchanged.

These two kinds of update steps are described in more detail below.

To calculate the correct probabilities we will need to know the contribution to the Hamiltonian from the diagonal and off-diagonal operators:

$$\begin{aligned}
W(\begin{smallmatrix}+&+\\+&+\end{smallmatrix}) &= \langle\uparrow\uparrow|\mathcal{H}_b|\uparrow\uparrow\rangle = C - 1/4 - h/(2J),\\
W(\begin{smallmatrix}-&-\\-&-\end{smallmatrix}) &= \langle\downarrow\downarrow|\mathcal{H}_b|\downarrow\downarrow\rangle = C - 1/4 + h/(2J),\\
W(\begin{smallmatrix}+&-\\+&-\end{smallmatrix}) &= \langle\uparrow\downarrow|\mathcal{H}_b|\uparrow\downarrow\rangle = C - 1/4,\\
W(\begin{smallmatrix}-&+\\-&+\end{smallmatrix}) &= \langle\downarrow\uparrow|\mathcal{H}_b|\downarrow\uparrow\rangle = C - 1/4,\\
W(\begin{smallmatrix}-&+\\+&-\end{smallmatrix}) &= \langle\uparrow\downarrow|\mathcal{H}_b|\downarrow\uparrow\rangle = 1/2,\\
W(\begin{smallmatrix}+&-\\-&+\end{smallmatrix}) &= \langle\downarrow\uparrow|\mathcal{H}_b|\uparrow\downarrow\rangle = 1/2.
\end{aligned}$$

6.2.1 The operator loop

In the first kind of updates we think of the operators as having four legs (numbered 1 through 4) with a spin associated to each leg. Since the position

of each such operator is fixed, each leg is connected to a fixed leg on a given other operator. The idea is now to construct a loop and flip the spin at the legs that the loop passes through. The following steps are used:

1. Choose one of the operators and one of its legs by random and flip the corresponding spin.

2. Choose an outgoing leg according to probabilities to be discussed below and flip the corresponding spin.

3. From this outgoing leg one can always reach another leg on another (or the same) operator. Exit, if this means that the loop closes onto itself. Otherwise continue with step 2.

   In a later version there should be some figures here.

The choice in step 2 should be done according to the *detailed balance* prescription. With configurations s and s' with their corresponding weights $W(s)$ and $W(s')$ the probabilities of change from s to s' and vice versa should obey

$$P(s \to s')W(s) = P(s' \to s)W(s').$$

For an example, consider $\left(\begin{smallmatrix} + & - \\ - & + \end{smallmatrix}\right)$ where we enter in the lower left corner. Flipping that spin we get $\left(\begin{smallmatrix} + & - \\ + & + \end{smallmatrix}\right)$ and depending on the output leg we may then get four different spin configurations with weights:

$$W\left(\begin{smallmatrix} + & - \\ - & + \end{smallmatrix}\right) = \frac{1}{2}, \quad W\left(\begin{smallmatrix} + & - \\ + & - \end{smallmatrix}\right) = C - \frac{1}{4}, \quad W\left(\begin{smallmatrix} - & - \\ + & + \end{smallmatrix}\right) = 0, \quad W\left(\begin{smallmatrix} + & + \\ + & + \end{smallmatrix}\right) = C - \frac{1}{4}.$$

Let us for concreteness, assume that leg number 2 is the chosen exit leg which means that the new set of spins is $s' = \left(\begin{smallmatrix} + & - \\ + & - \end{smallmatrix}\right)$. To examine detailed balance we then need to consider the *reverse* step, $s' \to s$ where the entry leg is leg number 2. After entering on leg number 2 and flipping the corresponding spin, the spin set is $\left(\begin{smallmatrix} + & - \\ + & + \end{smallmatrix}\right)$ which is the same intermediate set as in the $s \to s'$ process considered above. This means that the possibilities and weight factors are the same for both processes.

The simplest choice for $P(s \to s')$ is

$$P(s \to s') = \frac{W(s')}{\sum_s W(s)},$$

and it is straightforward to show by direct substitution that this fulfills the detailed balance condition. However, this expression for the probabilities gives a rather large probability for backtracing—exiting through the input leg. This is a drawback since backtracing often leads to shorter operator loops. It is however possible to choose these probabilities in different ways which excludes backtracing and still fulfills detailed balance.

6.2.2 Diagonal update

To discuss the diagonal update we introduce an unit operator $\mathcal{H}^{\text{unit}}$ (which does nothing) and change from operator sequences with n operators to sequences with M operators. If such a sequence includes $M-n$ unit operators this may be identical to a sequence with n (non-unit) operators.

The sum of $2N_b$ operators $\sum_\mu \sum_b \mathcal{H}_b^\mu = \left(\sum_b \mathcal{H}_b^{\text{diag}} + \sum_b \mathcal{H}_b^{\text{off}}\right)$ is then replaced by $2N_b + 1$ terms. We introduce the notation $\nu = (\mu, b)$, where $\nu = (\text{“unit”}, 0)$ for the unit operator.

$$\mathcal{H}^{\text{unit}} + \sum_b \mathcal{H}_b^{\text{diag}} + \sum_b \mathcal{H}_b^{\text{off}} \equiv \sum_\nu \mathcal{H}_\nu.$$

The expression for the partition function is somewhat modified since it is possible to choose n out of M in $M!/[(M-n)!\, n!]$ different ways. After compensating for this overcounting the new expression for the partition function becomes

$$Z = \sum_\alpha \sum_{\{\nu_M\}} \frac{\beta^n (M-n)!}{M!} \langle \alpha | \mathcal{H}_{\nu_M} \cdots \mathcal{H}_{\nu_1} | \alpha \rangle.$$

For each set $\{\nu_M\}$, n is determined as the number of non-unit (i.e. diagonal or off-diagonal) operators in the operator chain. With the notation $|\alpha(p)\rangle$ for the *normalized* state after the propagation of a fraction of the operator string,

$$|\alpha(p)\rangle \propto \mathcal{H}_{\nu_p} |\alpha(p-1)\rangle,$$

the (dimensionless) energy associated with a single operator $\mathcal{H}_{\nu_p}$ is

$$\langle \alpha(p) | \mathcal{H}_{\nu_p} | \alpha(p-1) \rangle,$$

and the weight for each set $\nu \equiv \{\nu_M\}$ becomes

$$\pi_\nu = \frac{\beta^n (M-n)!}{M!} \prod_{p=1}^{M} \langle \alpha(p) | \mathcal{H}_{\nu_p} | \alpha(p-1) \rangle. \tag{6.3}$$

We now loop over the M operator positions and try to change unit operators to diagonal ones and diagonal ones to unit operators. The off-diagonal operators are left unchanged. To give the correct probability distribution π_ν the acceptance probability for these changes should be chosen to fulfill detailed balance. The expression from Eq. (2.10) is[1]

$$a_{\nu\nu'} = \min\left(1, \frac{\pi_{\nu'} q_{\nu'\nu}}{\pi_\nu q_{\nu\nu'}}\right).$$

We now consider a transition from a state $\nu \equiv \{\nu_M\}$ with n non-unit operators to a state with $n+1$ which differs from ν only in that ν_p is changed from "unit" to ("diag", b'). The probability to suggest b' out of the N_b possible bonds is $q_{\nu\nu'} = 1/N_b$ whereas the transition back always takes us to ("unit",0). The probability for this to be suggested is therefore $q_{\nu'\nu} = 1$. Equation (6.3) gives

$$\pi_{\nu'}/\pi_\nu = \left(\frac{\beta^{n+1}(M-n-1)!}{M!}\right) \Big/ \left(\frac{\beta^n (M-n)!}{M!}\right) \times \langle \alpha(p)|\mathcal{H}_{b'}^{\mathrm{diag}}|\alpha(p-1)\rangle,$$

and we get

$$a_{\nu\nu'} = \min\left(1, \frac{\beta N_b \, \langle \alpha(p)|\mathcal{H}_{b'}^{\mathrm{diag}}|\alpha(p-1)\rangle}{M-n}\right) \tag{6.4}$$

The probability for deleting a diagonal operator at a given p, i.e. to go from ν' with $n+1$ non-unit operators to ν with n is precisely the inverse of Eq. (6.4). If we however want n to denote the number of non-unit operators in the *initial* state (whereas the value at the final state is $n-1$) the probability for the transition to be accepted should be written

$$a_{\nu'\nu} = \min\left(1, \frac{M-n+1}{\beta N_b \, \langle \alpha(p)|\mathcal{H}_{b'}^{\mathrm{diag}}|\alpha(p-1)\rangle}\right).$$

[1]To avoid confusion we now use $a_{\nu\nu'}$ for the acceptance probability instead of $\alpha_{\nu\nu'}$ since α in the present context denotes the spin state.

Chapter 7

Technical considerations

We now turn to a number of different technical considerations that are relevant for various kinds of MC programs. We will start by considering different ways to handle 2- (or more) dimensional arrays in C. We then turn to methods for handling periodic boundary conditions. We also discuss a simple implementation of two different kinds of queues.

7.1 2D arrays

The problems we are examining are usually defined on a d-dimensional space with $d \geq 2$. Since the arrays in C are one-dimensional constructs we need some kind of method that allows us to address the arrays with two, or more, coordinates. There are at least three different methods to choose between.

7.1.1 Multidimensional arrays

The simplest choice is to make multidimensional arrays in the declaration,

```
double arr[N][M];
```

where `N` and `M` usually are constant values. With this approach the address of `arr[x][y]` is calculated as `arr + x * M + y`, but the code needed to do that is put in by the compiler and need not bother us. To allow for this the dimension of the last (well, all but the first) index has to be known.

The drawback with this solution is that it doesn't allow for use of dynamic allocation, with `malloc` and similar functions. With the following construct it is however possible to create multidimensional arrays with adjustable size,

```
void first_func(int L) {
  int arr[L][L];
```

The array is allocated on the stack which means that it is lost when the program is returning from `first_func`. The array may only be used in calls to other functions that e.g. may be declared like this:

```
void second_func(int L, int arr[][L])
```

7.1.2 Indexing yourself

A possibility that allows for more flexibility is to only work with one-dimensional arrays and do the index calculations yourself. With this method `arr[x][y]` would instead be written

```
arr[x * L + y]
```

To make the program easier to read it could be convenient to define a macro

```
#define XY(x, y) ((x) * L + (y))
```

or a `static inline` function

```
static inline int index(int x, int y, int L) {
  return x * L + y;
}
```

The `static inline` declaration means that the code is included directly in the function which saves the overhead associated with a function call. This is in many respects like a macro but makes it easy to define more complicated functions.

7.1.3 Matrices with arrays of pointers

A third alternative is to make use of an array of pointers. In that case the $L \times L$ array may be declared and allocated as shown below

```
int **arr;

arr = malloc(L * sizeof(int *));
for (i = 0; i < L; i++)
  arr[i] = malloc(L * sizeof(int));
```

Note the use of `sizeof(int *)` in the first malloc and `sizeof(int)` in the second one. This method requires a little space beside the $L \times L$ variables that hold the actual data.

7.2 Periodic boundary conditions

One commonly needs to find the nearest neighbors of a certain point, say (x, y). In 2D the nearest neighbors would normally be $(x-1, y)$, $(x+1, y)$, $(x, y-1)$ $(x, y+1)$ but because of the periodic boundary conditions this has to be modified for (x, y) at a boundary to give coordinates in the interval $0 \leq x, y < L$. Again, it is possible to choose between a few different methods:

1. A simple choice is to define macros for plus and minus with the periodicity of the system,

   ```
   #define PLUS(x, L) ((x) == (L) - 1 ? 0 : x + 1)
   #define MINUS(x, L) ((x) == 0 ? (L) - 1 : x - 1)
   ```

2. If one restricts the simulations to system sizes $L = 4$, 8, 16, 32,... the bit-and operation with $L - 1$ takes us back to the allowed interval:

   ```
   xp = (x + 1) & (L - 1);
   xm = (x - 1) & (L - 1);
   ```

3. Yet another possibility is to construct a set of arrays that hold the index of the nearest neighbors. The advantage is that we may then do with a single loop over the particles regardless of the number of dimensions. The value of `arr` at the positition $(x+1, y)$ may then be specified as

   ```
   arr[xplus[ixy]]
   ```

7.3 The implementation of queues

7.3.1 A simple stack

To handle a stack (a last-in-first-out queue) one only needs a single pointer to put to and get from the queue. The "put" and "get" operations are performed with `*ptr++` and `*--ptr`, respectively. A code snippet could look like the following

```
int *arr, *ptr;
ptr = arr = malloc(SIZE * sizeof(int));

*ptr++ = something;
while (ptr - arr) {
  var = *--ptr;
  for (depends on the application)
    if (some condition)
      *ptr++ = othervariable;
}
```

7.3.2 Fifo queue

For a fifo (first in first out) queue one needs two pointers. One for input and the other for output. Both the put and the get operations are followed by incrementing the pointers. A skeleton could look like the following:

```
int *arr, *in, *out;
in = out = arr = malloc(SIZE * sizeof(int));

*in++ = something;
while (in - out) {
  var = *out++;
  for (depends on the application)
    if (some condition)
      *in++ = othervariable;
}
```

With this construct the number of items processed through the queue is obtained from the difference `in - arr`.

7.4 Correlation function through FFT

The correlation function is defined as

$$g(\mathbf{r}) = \left\langle \frac{1}{N} \sum_{\mathbf{r}'} s_{\mathbf{r}'} s_{\mathbf{r}'+\mathbf{r}} \right\rangle .$$

The time needed for a direct calculation of this quantity goes as N^2. With a Fast Fourier transform this instead becomes $N \ln N$ and this is therefore the standard way to calculate the correlation function. Define the Fourier transforms

$$\begin{aligned} s_{\mathbf{r}'} &= \frac{1}{N} \sum_{\mathbf{k}'} e^{i\mathbf{k}'\cdot\mathbf{r}'} s_{\mathbf{k}'}, \\ s_{\mathbf{r}'+\mathbf{r}} &= \frac{1}{N} \sum_{\mathbf{k}} e^{i\mathbf{k}\cdot(\mathbf{r}'+\mathbf{r})} s_{\mathbf{k}}. \end{aligned}$$

For the sum above we then get

$$\begin{aligned} \sum_{\mathbf{r}'} s_{\mathbf{r}'} s_{\mathbf{r}'+\mathbf{r}} &= \frac{1}{N^2} \sum_{\mathbf{k},\mathbf{k}'} \sum_{\mathbf{r}'} e^{i(\mathbf{k}+\mathbf{k}')\cdot\mathbf{r}'} s_{\mathbf{k}} s_{\mathbf{k}'} e^{i\mathbf{k}\cdot\mathbf{r}} \\ &= \frac{1}{N} \sum_{\mathbf{k}} s_{\mathbf{k}} s_{-\mathbf{k}} e^{i\mathbf{k}\cdot\mathbf{r}} = \frac{1}{N} \sum_{\mathbf{k}} |s_{\mathbf{k}}|^2 e^{i\mathbf{k}\cdot\mathbf{r}}, \end{aligned}$$

and since $g(\mathbf{r}) = \frac{1}{N} \sum_{\mathbf{k}} e^{i\mathbf{k}\cdot\mathbf{r}} g(\mathbf{k})$ we identify

$$g(\mathbf{k}) = \frac{1}{N} \left\langle |s_{\mathbf{k}}|^2 \right\rangle.$$

Note that the Fourier transform presupposes a periodicity in the system and therefore requires the use of periodic boundary conditions. Another consequence of this periodicity is that the correlation function also becomes periodic with e.g. $g_L(L - x, y) = g_L(x, y)$.

7.5 Determination of the correlation length

The most obvious way to get the correlation length is to fit the function $g(r) = Ae^{-r/\xi}$. In most cases of interest this behavior is correct only for $r \geq \xi$ and with the choice of the fitting interval comes a certain arbitrariness in the determination of ξ. An alternative approach is to start from the Fourier representation of an exponentially decaying function,

$$g(k) \sim \frac{1}{\tilde{k}^2 + \xi^{-2}},$$

where[1] $\tilde{k} = 2\sin(k/2)$. Using only the two smallest k-values, $k = 0$ and $k_{\min} = 2\pi/L$ gives the relation

$$\xi = \frac{1}{2\sin(\pi/L)}\sqrt{\frac{g(0)}{g(k_{\min})} - 1}. \tag{7.1}$$

This is often the most efficient method for determining the correlation length and becomes very handy if the correlation function is determined with FFT. On the other hand, if the purpose is only to determine the correlation length (and not the full correlation function $g(\mathbf{r})$) there is no point in calculating the full function $g(\mathbf{k})$. A routine that determines only $g(0)$ and $g(k_{\min})$ is then far more efficient: just calculate $s_x = \sum_y s_{(x,y)}$ for each x and determine

$$\begin{aligned} g(0) &= \frac{1}{N}\left\langle \left| \sum_x s_x \right|^2 \right\rangle, \\ g(k_{\min}) &= \frac{1}{N}\left\langle \left| \sum_x s_x e^{-ik_{\min}x} \right|^2 \right\rangle. \end{aligned}$$

[1]This is related to the discrete version of $\nabla e^{ikx} = ke^{ikx}$ which is $De^{ikx} \equiv e^{ik(x+1/2)} - e^{ik(x-1/2)} = 2\sin(k/2)e^{ikx}$.

Chapter 8

To organize large scale simulations

8.1 The two-step approach

8.1.1 Background

After writing a Monte Carlo program and doing some runs one usually collects the results into a file which is then used as input to the plotting program. It will often soon be obvious that this simple approach has some short-comings, as problems may arise in several ways:

- After attempting to analyze the data it often becomes clear that it is necessary to run more for some of the parameters to get higher precision. It will then be necessary to calculate the averages from all runs together. However, performing these calculations and putting the new data into the plotting file quickly becomes somewhat tedious.
- It may become clear that some of the manipulations of the measured data in the simulation program were incorrect. This can e.g. be an incorrect normalization in the determination of Binder's cumulant. In some cases the errors may be compensated for, but in other cases they make the data useless.
- With access to a batch system with a number of fast computers the large amount of data produced makes the refreshing of the plotting file a difficult, tedious and error-prone task.

A solution

This problem calls for a flexible, error-tolerant method for organizing the Monte Carlo simulations and for collecting the data of interest. The method described below is a two-step approach to this task where the first step is the actual MC simulation together with the accumulation of the measured data whereas the second step is the processing of this raw data to get out the desired quantities. Splitting things up like that has several advantages:

- The calculation of averages in step II may automatically merge the results from several simulations with the same parameters.
- The MC program may be kept simple and is therefore more likely to be free from errors.
- It is easy to add more analyses when they are needed. One example is error analyses that usually only are needed at the end of a project to estimate the uncertainity in the results.

Whereas errors in the MC program often will make it necessary to discard the data and re-run the simulations, errors in the analysis program are easily corrected. As a general rule it is therefore wise to move as many manipulations as possible from the simulation step to the analysis program.

8.1.2 Names for config- and data files

It is convenient to name both the configuration files and the data files on the basis of the parameters. To achieve this one may use the `sprintf` function:

```
sprintf(fname, "%3.3d_%5.3f", n, t);
```

which e.g. could give `016_2.200` or `256_2.300`. The use of `%3.3d` instead of just `%3d` prepends the number with zeros instead of spaces. One could of course also do with `%d` that produces variable-length output. The fixed-width output could however be advantageous in some cases, e.g. because the sorting done by the `ls`-command (see below) automatically orders the data by increasing size.

8.1.3 Different types of files

To be specific we consider a MC simulation of the 2D Ising model. One can then make use of three different kinds of files. For each kind of file we create a directory:

`data/` A data file that is easily read by the analysis program. In the very beginning this file should contain all the relevant input parameters (system size, temperature,...) and simulation parameters (e.g. number of samples per average) needed to analyze the data. The rest of the file is just the collected data.

`conf/` A file that stores the configuration at the end of each run. The idea with this file is to be able to continue the simulation in another job without the need to thermalize the system again. Each time a simulation is started the program should check for the existence of a configuration file with the right name. If it exists it should be used as the starting configuration.

`log/` For batch jobs we need log files that give human-readable summaries of the runs. This should list the parameters used, tell if the configuration was read from a `conf` file, the name of the `data` file, and also print out some results.

8.1.4 The analysis program

The input to the analysis program should be names of data files and the output is usually written to a file. On the command line in a Linux/Unix system this may be written

```
ls data/* | analyze > anal.res
```

The "`|`" sign is a *pipe* that instructs the shell (the program that interprets what is written on the command line) to connect the output from one program (here `ls`) to the input of another one (here `analyze`). The "`>`" redirects the output that normally should have appeared on the screen to the file `anal.res`. If possible, a convenient alternative would be to run the analysis program directly from within the plotting program without the need for storage in a file.

8.1.5 Accessing binary data files

Both the configuration files and the data files could be either formatted or unformatted (binary). The advantage with formatted files is that they may be examined with an ordinary editor. For binary files, on the other hand, the advantage is that it is easy to read or write big chunks of data directly between the memory and the file. Formatted data files are opened/closed with `fopen` and `fclose` and written to and read from with `fprintf` and `fscanf`. In this section we give some information about functions for access of unformatted files.

As mentioned above it is convenient to have all the parameters readily accessible at the very beginning of the data file. That may be done with a struct:

```
typedef struct Par {
  double t;
  int n, nstep, nsamp;
} Par;
```

Create and initialize the data file

The following small function first attempts to open the file for read only. If that doesn't succeed it creates the file and writes the parameter struct to the file.

```
int check_datafile(char *fname, Par *par) {
  int fdesc;
  char filename[64] = "data/";
  strcat(filename, fname);
  fdesc = open(filename, O_RDONLY, 0644);  // Open for read-only
  if (fdesc == -1) {            // Not successful, try to create
    fdesc = open(filename, O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (fdesc == -1) return 0;
    write(fdesc, par, sizeof(Par));
  }
  close(fdesc);
  return 1;
}
```

We usually want to append new data to the end of the data file. To that end one needs to specify `O_APPEND` in the second argument to the `open` call. To also specify write access one gives `O_WRONLY | O_APPEND`. (The "`|`" character specifies a bit-or operation.)

Read from the data file

In the `analyze` program one first reads the parameter struct and then all the data till end-of-file. Since `read` returns the number of bytes read the construct `while(read(...))` will only continue reading as long as there is any data left in the file.

```
int read_datafile(char *filename, Par *par, double *vec) {
  int fdesc;
  fdesc = open(filename, O_RDONLY, 0644);
  if (fdesc == -1) return 0;
  read(fdesc, par, sizeof(Par));
  while(read(fdesc, v, SIZE)) {
    // accumulate data from v into vec.
  }
  close(fdesc);
}
```

The functions `open`, `write`, `read`, and `close` used above are the basic low-level input/output functions in the C library and are very convenient for accessing binary files. The documentation is accessed in the `info` system. Note the use of `sizeof(Par)` to determine the number of bytes in the struct. In this case `sizeof(par)` will not do since that is only the size of a pointer.

8.2 Organizing the source code

At least for somewhat larger projects one often needs a few slightly different versions of the program for examining the same problem in different ways. It is however not convenient to have several separate sets of the source code since one often wants new features to be available in all the different versions without too much work. We now sketch a method to make that work where the compilation is done in various subdirectories to the directory with the sources. In these subdirectories the compilations are controlled in different ways with the use of preprocessor variables.

Binaries in different subdirectories

We need several different directories with different ".o"-files produced from a single set or source and header files. It is possible to instruct `make` to get the source from other directories (if they don't exist in the current directory) with the `VPATH` variable. With the following in `Makefile`

```
VPATH = ..
```

`make` may construct the following command for making `ising.o`:

```
gcc -g -O4 -c -o ising.o ../ising.c
```

With this construction we may have several subdirectories below the directory with the source code.

Control compilation with preprocessor directives

To get different versions of the program in the different subdirectories we need a way to select alternative blocks of the code on the basis of certain variables. The mechanism for doing that is *preprocessor directives*. To select the code that performs cluster update or else select the code that does single spin update we may use the construct

```
#ifdef CLU
 // Code for cluster update.
#else
 // Code for single spin update.
#endif
```

To make this work we need to define the variable (or macro) `CLU` in one of the directories. That may be done through the variable `CPPFLAGS` (for C Pre-Processor flags) in two different ways. The most direct one is throught the `-D` option followed by the variable name:

```
CPPFLAGS = -DCLU
```

The second method is to instead put the definitions in a file, e.g. `define.h` and instruct the preprocessor to include that file before everything else. In the `Makefile` this becomes

```
CPPFLAGS = -include define.h
```

Separate simulations from the source code

Finally, it is good to separate the simulation results from the source code. A good way to achieve this is to have two subtrees, `~/src` and `~/mc` for the source code and the simulations, respectively. The necessary connection is then provided with soft links:

```
ln -s ~/src/ising/ising-clu/sim Sim
```

The good things with a soft link here are: (1) after a re-compilation the latest version of the program is immediately used, and (2) it is always easy to check the origin of the program and how it was compiled. This is in contrast to the situation if the executable file had instead been copied to the simulation directory.

Chapter 9

Particle transport – a brief orientation

Particle transport is an important field for Monte Carlo methods with many different applications:

- radiation shields for nuclear reactors,
- the critical state for neutrons in a reactor,
- charge transport in semiconductor devices,
- estimates of the dose in radiation theraphy.

Generally speaking, particle transport may be considered to be a great branching random walk where random numbers determine

- when there is a collision,
- what kind of process that occurs (e.g. pair production, Compton scattering, photoelectric effect),
- the properties of the resulting particles (direction and energy).

The programs used in such simulations are normally rather big and complicated and need information both about scattering cross sections and angle distributions for different energies of the particle of interest. Some well known programs for such simulations are ETRAN and EGS4.

9.1 Basic methods

Consider a photon with a certain energy. There is then three possible processes:

- photoelectric effect (p),
- Compton scattering (C),
- pair production (pp).

The probability of a particle interacting along a path $d\ell$ is $\mu_t d\ell$ where the total interaction coefficient μ_t depends on both the parameters of the particle and the nature of the medium. When there are three kinds of relevant processes as listed above one has

$$\mu_t = \mu_p + \mu_C + \mu_{\rm pp}.$$

9.1.1 Differential sampling

The simples method is differerential sampling. The idea is here to split the medium into parts of length $\Delta\ell$ and repeat the following steps:

- generate a random number ξ,
- if $\mu_t \Delta\ell > \xi$ then there is some kind of interaction:
  - photoelectric effect if $\xi < \mu_P \Delta\ell$,
  - Compton scattering if $\mu_P \Delta\ell < \xi < (\mu_P + \mu_C)\Delta\ell$,
  - pair production if $(\mu_P + \mu_C)\Delta\ell < \xi$,
- else, move the particle the distance $\Delta\ell$.

This is a very slow and tedious procedure which is not to be used in practice.

9.1.2 Integrated sampling

Integrated sampling is a much more efficient method. The idea is to instead choose the distance ℓ to the interaction. Define the interaction length,

$$\lambda = 1/\mu,$$

which in the more general case may depend on the position. For the simpler case of constant λ we have

$$P(\ell) = 1 - \prod_i e^{-\Delta\ell/\lambda} = 1 - e^{-\ell/\lambda},$$

and with the method of inversion, Eq. (2.9), we get

$$\ell = -\lambda \ln \xi. \tag{9.1}$$

For the more complicated case where λ depends on ℓ we have

$$1 - e^{-\int d\ell'/\lambda(\ell')} = 1 - \xi \quad \Rightarrow \quad \int_0^\ell d\ell'/\lambda(\ell') = -\ln \xi.$$

It is often impossible the solve the above integral equation analytically. A useful trick is then to introduce a null process "self-scattering" that does nothing. Define a constant $\mu^* \geq \mu_t(x)$ and define self-scattering with the probability coefficient

$$\mu_0(x) = \mu^* - \mu_t(x).$$

The selection of the interaction point may then be done with the simpler Eq. (9.1). To select the kind of process we calculate

$$\int_0^\ell d\ell' \mu_P, \quad \int_0^\ell d\ell' (\mu_P + \mu_C), \quad \int_0^\ell d\ell' (\mu_P + \mu_C + \mu_{\rm pp}),$$

and the choice between one of these and the null process is done by means of a random number ξ.

9.2 Variance reduction

Assume that we are to examine a radiation shield and expect a transmission probability of $T = 10^{-6}$. A calculation with $N = 10^8$ particles would then on the average let 100 particles through which gives a variance

$$\sigma^2 = NT(1 - T) = 100, \quad \Rightarrow \sigma = 10.$$

The statistical error is then 10% which is not very impressive and calls for better methods.

The kind of calculation discussed above is an *analogue MC simulation* where there is a direct relation between the process in the computer and a possible experiment. In more advanced techniques different particles are assigned different statistical weights that varies during the propagation.

9.2.1 Bias in the distribution of particles

Assume that the particles are from a distribution $n(E)$ where only the particles at the high end of the distribution are likely to contribute to the transmission. It is then possible to make use of a distribution $n^*(E)$ and assign a weight factor $w = n(E)/n^*(E)$ to each particle in the beginning of the simulation.

9.2.2 Potential scoring

Consider the measurement of the energy absorbed in a region V. The simplest method would be to let

$$E_{\text{abs}} \to E_{\text{abs}} + E,$$

each time a particle with energy E is absorbed in the region under consideration. If the probability for absorption is small compared to the total number of interactions in the volume, $\mu_{\text{abs}} \ll \mu^*$, not very many particles would be absorbed during the simulation and the uncertainity in the result would be large. To increase the precision one may instead assume that each interaction deposits an energy that is scaled by the probability of absorption,

$$E_{\text{abs}} \to E_{\text{abs}} + \frac{\mu_{\text{abs}}}{\mu^*} E.$$

Because of the larger number of events the uncertainity would be smaller than in the analogue simulation.

9.2.3 Splitting

In this method each particle that reaches far enough (for example in the radioactive shield) is replaced by ν particles with identical properties but with weight

$$w_{\text{out}} = \frac{1}{\nu} w_{\text{in}}.$$

The larger number of particles again help improve the statistics.

9.2.4 Survival biasing

In this method the possibility of absorption is entirely suppressed and this is compensated for by modifying the weight factor,

$$w_{\text{out}} = \left(1 - \frac{\mu_{\text{abs}}}{\mu_t}\right) w_{\text{in}}.$$

In this way all the produced particles make a contribution, though often very small, to the final score.

9.2.5 Russian roulette

Here the focus is to reduce the amount of work on particles with low probability to contribute to the final result. This means that we skip particles with a probability $\alpha(E)$ and that the ones that continue do that with a weight factor

$$w = \frac{1}{1 - \alpha(E)}.$$

One possible choice of $\alpha(E)$ is

$$\alpha(E) = \begin{cases} 0, & E > E_{\text{ref}} \\ 1 - E/E_{\text{ref}}, & E < E_{\text{ref}}. \end{cases}$$

Computer lab I

Two Monte Carlo methods for the Ising model

I.1 Monte Carlo programs for the 2D Ising model

The task is to write two programs to do Monte Carlo on the 2D Ising model. You should implement both the simpler single spin Metropolis algorithm and the more complicated Wolff cluster algorithm.

I.1.1 Get source files

Use the command below to extract some files for the Ising lab in three different directories.

```
tar xzf /home/peol0002/MonteCarlo/ising.tgz
```

If you are at not at the Linux system at the physics department you will first have to download `ising.tgz` from the link at the web page `http://www.tp.umu.se/mc`. Then extract its content in much the same way:

```
tar xzf ising.tgz
```

If everything is correct you will now have three new directories: `src`, `Metro`, and `Cluster` which each contains some files. Some source files are in `src` and the `Makefile`s in `Cluster` and `Metro` are set up to compile the cluster version and the Metropolis version, respectively.

I.1.2 Metropolis Monte Carlo

Several essential parts are missing in `src/ising.c`. You will therefore have to go through the following steps to get a working program:

1. Start by looking at `ising.h` and the `main` function in `ising.c`.

   The program is meant to be run with several temperatures in a sequence, e.g. with `./ising L=16 nblock=16 T=2.2 run T=2.22 run T=2.24 run`. Try to understand how how `main` and `read_args` work. Also examine the functions `initialize_mc` and `mc`. Check the effect of `ntherm`, `nblock`, and `nsamp` in the code.

   The comments like "Fix this (2)" should be taken care of in the suggested order. The numbers 2–5 refer to the steps in this list.

2. Write a routine `update` that performs a Monte Carlo sweep over the system and returns the number of accepted changes.

   Note that Chapter 7 in the lecture notes, "Technical considerations", have suggestions regarding the implementation of both 2D arrays and periodic boundary conditions.

   - For 2D arrays I prefer "Indexing yourself" and that is also consistent with the functions in `config.c`. You are however free to implement the 2D arrays in a different way.
   - Boundary conditions: If you choose the second alternative in Sec. 7.2, you also have to include a test in the program that the given system size is a valid one.

   When the `update` routine is completed it should be possible to run the program, even though it doesn't produce any measured results. Make a test run with $L = 8$ and $T = 2.2$:

   ```
   ./ising L=8 T=2.2 run
   ```

   If everything is correct you should get acceptance ratio $\approx 14\%$.

3. Write a function `measure` to measure magnetization and energy. Also put in some code in `mc` to accumulate $|M|$, E, and E^2 (which is needed for the heat capacity.) Then complete the function `result` which

should print out $|m|$, e, and c (i.e. magnetization, energy and heat capacity per spin). It is good to print out both averages for each block and the total values. The output could e.g. look like

```
  energy     c          magn
-1.563313  1.056386  0.819306
-1.560219  1.103098  0.817187
-1.574837  1.035064  0.826266
-1.565069  1.046963  0.819550
 --------  --------  --------
-1.565859  1.060773  0.820577
```

If everything is correct the energy per spin for $L = 8$ and $T = 2.2$ should be ≈ -1.568. For a longer run and higher precision, you can try

```
./ising L=8 T=2.2 nblock=64 run
```

4. In the simplest implementation the exponential function is called many times. This is a vaste of time since there will only be a few different values of ΔE. Change the program by first filling an array with values for $\exp(-\Delta E/T)$ in `init_tables` and then using this array in the update routine. Use the Unix command `time` (see below) with the old and new versions to check that the new version really is considerably faster.

```
mv ising slow-ising
make
time ./slow-ising L=64 T=2.2 seed=1 run
time ./ising L=64 T=2.2 seed=1 run
```

Note that the commands above set the seed to the random number generator by hand `seed=1` and this is convenient to check that two versions of the program do the same thing. If everything is correct the two versions should therefore produce identical results. (With the default value seed=0 in the call to `init_ran` a new seed is instead taken from the clock and the sequence of random numbers is therefore different each time, see `ran.c`.)

5. We also need to be able to store configurations in files and read configurations from file into memory. Note the `sprintf(fname,...)` statement in `initialize_mc`. Uncomment the `read_config` and `write_config` calls in function `mc`. Also create a directory `conf` where the configuration files will be stored (see `config.c`).

Plotting with gnuplot

To get the data into the plotting program in a convenient way it might be good to write parameter (temperature) and results (energy, heat capacity, and magnetization) as four columns to one or several files directly from the simulation program. To get different symbols for different system sizes (which is what we want) in gnuplot, one would better have one file per system size. In gnuplot one can then use the lines shown below to (1) set a bigger symbol size, (2) to plot data from a single file, (3) to plot data from several files (where we use the short forms of some keywords and show how to continue the command on a second line), (4) put text on the x axis, (5) write the current figure to 'file.ps' (which only works at the physics Linux system where `/home/local/print.gp` has the proper content).

```
set pointsize 2
plot 'L016.gp' using 1:3 with linespoints
plot 'L008.gp' u 1:3 w lp, 'L016.gp' u 1:3 w lp,\
  'L032.gp' u 1:3 w lp, 'L064.gp' u 1:3 w lp
set xlabel 'T'
call '/home/local/print.gp' file
```

I.1.3 Check data for the heat capacity

We will here check that the two different ways to calculate the heat capacity give similar results. Running the program with

```
./ising  L=16 nblock=64 T=2.0 run T=2.1 run T=2.2
```

will give you values for both E and C (from the flucutation formula) at the specified temperatures. Use

$$C\left(\frac{T_1+T_2}{2}\right) = \frac{E(T_2)-E(T_1)}{T_2-T_1},$$

report to calculate $C(2.05)$ and $C(2.15)$. Plot the two determinations of the heat capacity per spin in the same diagram with different symbols.

I.1.4 Phase transition and size dependency

The phase transition in the Ising model is seen clearly in both the magnetization and the specific heat. The transition is perfectly sharp only in an infinite system and the quantities that ideally have a sharp and abrupt behavior become smoothened in smaller systems. You should do the simulations with `nblock=64` and four different system sizes, $L = 8$, 16, 32, and 64. At $L = 8$ it is sufficient to take temperatures $T = 2.0, 2.1, \ldots 2.8$ but for the other sizes you should take a larger number of temperatures, $T = 2.00$, 2.10, 2.16, 2.18, 2.20, 2.22, 2.24, 2.25, 2.26, 2.27, 2.28, 2.29, 2.30, 2.32, 2.34, 2.36, 2.38, 2.40, 2.44, 2.48, 2.52, 2.56, 2.60, 2.70.

Note that the simulations for each size may conveniently be taken care of by a single run of the program since the program then continues from a configuration from a nearby temperature, which means that we can do with rather short times for thermalization (which are chosen automatically). Note also that the configurations are stored automatically in the `conf` directory and that it is possible to read in a configuration with e.g. `read=064_2.250` specified on the command line (as seen in `read_args`).

Plot $\langle|m|\rangle$ versus T for the four different sizes in a single figure. Connect the symbols with lines and remember to label the axes properly. Make another figure with C versus T. **report**

I.1.5 Cluster update

Write a routine that performs a Wolff cluster update step as described in Sec. 4.3.4. Note that a call to the routine only should give a single cluster. Again test your program by running for $L = 8$ and $T = 2.2$.

1. Do simulations with $L = 256$ and several temperatures close to T_c: $T = 2.20$, 2.22, 2.24, 2.25, 2.255, 2.26, 2.262, 2.264, 2.266, 2.268 and 2.269. (To save time you can here use `nblock=1`.)

2. Also run with $L = 1024$ and temperatures $T = 2.265$, 2.266, 2.267, 2.268, and 2.269. (To get good precision you now have to run longer; each run should take at least a few minutes. Good data will make the analysis in the next step easier.)

3. Plot your data as m versus $T_c - T$ close to T_c on a log-log scale (`set logscale` in gnuplot) for $L = 64$, 256, and 1024. If everything is correct **report**

it should be possible to identify a straight line through the data, but there should also be clear deviations from this behavior for smaller L. Make use of the seemingly reliable data points close to T_c to determine the slope, and thereby the exponent β in $m \sim (T_c - T)^\beta$. Draw a line in the same figure to show the fitted line.

I.2 For extra points

I.2.1 The correlation time (2p)

The drawback of Markov chains is that the produced configurations are correlated to one another. To study this effect we will examine the time correlation function for the energy

$$C_E(t) = \langle \delta E(t') \delta E(t' + t) \rangle.$$

One expects the correlations to decay exponentially with time,

$$C_E(t) \sim e^{-t/\tau},$$

which is usually true for all but the smallest t. $t > \tau$ is usually safe.

- Add some code to measure $C_E(t)$. Determine the energy correlations for the Metropolis algorithm with $L = 32$ for times up to $t = 200$ at temperature $T = 1.8$, 2.0, 2.2, 2.3, 2.4, 2.6, and 2.8. Around T_c where the correlation time is longer you will need longer runs to get good precision in the data. Plot $C_E(t)$ versus t in a single figure with a log scale on the y axis. Skip data which is only noise.

report

- Determine the correlation time and plot τ versus T.
- Also make a separate figure with $C_E(t)$ obtained with the Wolff cluster algorithm at $T = 2.3$.

I.2.2 Visualization of the simulation process (2p)

Use the `g2` library to be able to see the update moves on the screen. Some information about the `g2` library is available at a link from the `mc` web page. If the `g2` library isn't installed (check if `/usr/lib/libg2.so.0` exists on the computer you are using) contact Peter.

Computer lab II

Scaling analyses of critical phenomena

II.1 Data handling

In this lab you should improve the data handling in your simulations by storing data in some data files:

- Before the simulation is started (e.g. in `initialize_mc`) the simulation program should check if the data file is already there. If it is not the file should be created and the information needed for the calculations in the `summary` program should be put at the beginning of the file. (This is essentially the information in the `par` struct.)

  More information may be found in Chapter 8, "To organize large scale simulations".

- After each block is completed the simulation program should write the data associated with that block to the data file.

- You also have to write a `summary` program that reads the data files and writes the output in a way which is convenient for the plotting program. This program should take the file names from `stdin` and write results to `stdout`. This means that it should be possible to use it by writing

  ```
  ls data/256* | summary > L256.gp
  ```

An partial program is found at the physics computer system, `/home/peol0002/MonteCarlo/summary.c` and is also available from a link on the course web page, `http://www.tp.umu.se/mc`.

II.2 Finite size scaling analysis

II.2.1 The magnetization

Do runs with the Wolff cluster update method for system sizes $L = 16$, 32, 64, 128, and 256. Set `nsamp=1000` (put it in the code for `CLU`) and run with at least `nblock=10`. Define $M = \sum_i s_i$, $m = \langle |M| \rangle / L^2$, and $t = T - T_c$. Choose the temperatures differently for different sizes such that you always have at least seven values with $-1 < tL^{1/\nu} < 1$. Plot four figures with different symbols for the different sizes:

1. m versus T (connect the points with lines, `linespoints`)
2. $mL^{\beta/\nu}$ versus T (again connect the points with lines)
3. $mL^{\beta/\nu}$ versus $tL^{1/\nu}$ (no lines)
4. Zoom in figure 3 for $-0.25 < tL^{1/\nu} < 0.25$ and include error bars in the plot. If possible (in gnuplot) draw a line from a second order polynomial through the data.

II.2.2 Binder's cumulant

Plot two figures with different symbols for the different system sizes:

1. Q_L versus T (with lines),
2. Q_L versus $tL^{1/\nu}$ (no lines).

II.3 For extra points

II.3.1 Spin-spin correlation (2p)

The most efficient way to determine the spin-spin correlation function,

$$g(x) = \langle s(x', y)s(x' + x, y)\rangle \,,$$

is with a Fast Fourier Transform. Here we will instead do it with a simple raw calculation. To save time you should only examine a single randomly chosen row of spins in each measurement. Your program should do the following:

- Choose a row y by random.
- For each $x = 0, \ldots, L - 1$ accumulate

$$\sum_{x'} s(x', y)s(x' + x, y).$$

- Write the results from each block to a file.

Note that you now want different symbols for different temperatures.

Run simulations with $L = 256$ and $T = 2.16$, 2.20, 2.22, 2.24, 2.26, 2.2692, 2.28, 2.30, 2.32, and 2.36. At $T_c \approx 2.2692$ you should also run at $L = 1024$ and maybe also some bigger size (2048 and/or 4096).

First plot the raw data $g(x)$ vs. x for $L = 256$ and all the different temperatures. We will then have to plot data in the different regions, $T > T_c$, $T = T_c$, and $T < T_c$ in different ways:

For $T > T_c$

We could here determine ξ from $g(x)$ for the different temperatures and try to examine how ξ depends on $T - T_c$. We will instead do a quick test that just demonstrates that

$$\xi \sim \frac{1}{|T - T_c|},$$

by plotting $g(x)$ for different temperatures (on a log scale) versus $x/\xi \equiv x|T - T_c|$. The data should fall on straight lines with the same slope. (Cut off the data for large x; avoid plotting data which is only noise.)

For $T = T_c$

At T_c we expect $g(x)$ to decay algebraically. Plot $g(x)$ for $T = 2.2692$ versus x with log scale on both axes for two (or more) different sizes. Also draw a straight line with the expected slope through the data that is not affected by finite size effects.

For $T < T_c$

Below T_c $g(x)$ approaches a finite constant at large x. To extract the exponential decay, plot $g(x) - g(L/2)$ (on a log scale) versus x. Then make a second figure where the same data is plotted versus $x \times (T_c - T) \sim x/\xi$. Again, the data should fall on straight lines with the same slope.

The correlation length

Finally determine the slopes in the figures for $T < T_c$ and $T > T_c$, respectively. Use these slopes to determine the correlation length $\xi(T)$ (for $T \neq T_c$) and plot ξ versus T.

II.3.2 The 2D Ising model on a triangular lattice (2p)

The task is here to determine the critical temperature for the 2D Ising model on a triangular lattice. According to universality the critical exponents should be the same whereas quantities like T_c (of course) are different for different two-dimensional lattices.

Start from your program for the square lattice Ising model and change to a triangular lattice by adding links along one of the diagonals. Do simulations for a few different sizes and determine T_c with an uncertainity less than 0.001. To do this you need to get data with very good precision close to T_c. Avoid using too small sizes since finite size scaling is not quite OK for the smallest sizes. Very big systems, for which you don't have time to get good precision, are not good either.

Appendix A

Algorithms

A.1 Lagged Fibonacci random number generator

A.1.1 The source file ran.c

```
#include <stdlib.h>
#include <time.h>
#include "ran.h"

// Adapted from Newman & Barkema, Monte Carlo Methods in Statistical Physics.

#define SIZE 1279
#define OFFSET 216
#define A 2416
#define C 374441
#define M 1771875
#define CONV 2423.9674
#define UCONV (1.0/(UINT_MAX + 1.0))
#define ICONV (1.0/(INT_MAX + 1.0))

static unsigned int vec[SIZE] = {0};
static int p, pp;

/*** Initialize the vector. ***/
```

```
void init_ran(int seed) {
  int i;

  if (!seed)
    seed = time(NULL);
  for (i = 0; i < SIZE; i++) {
    seed = (A * seed + C) % M;
    vec[i] = CONV * seed;
  }
  p = 0;
  pp = OFFSET;
}

/*** This does the real job ***/
inline unsigned int uran() {      // Integer 0...2^32 - 1
  if (--pp < 0) pp = SIZE - 1;
  if (--p  < 0) p  = SIZE - 1;

  vec[p] += vec[pp];
  return vec[p];
}

double dran() {                   // Double [0, 1)
  return UCONV * uran();
}

double dran_sign() {              // Double [-1, 1)
  return ICONV * (int) uran();
}

int iran() {
  return uran() & INT_MAX;        // Integer 0...2^31-1
}

int iran_sign() {
  return (int) uran();            // Integer -2^31...2^31-1
}
```

A.1.2 The header file `ran.h`

```
#include <limits.h>

#define URAN_MAX UINT_MAX
#define IRAN_MAX INT_MAX

void init_ran(int iseed);       // Initialize

unsigned int uran();            // Integer 0...URAN_MAX
int iran();                     // Integer 0...IRAN_MAX
int iran_sign();                // Integer INT_MIN...IRAN_MAX
double dran();                  // Double [0, 1)
double dran_sign();             // Double [-1, 1)
```

Bibliography

[1] N. Metropolis *et al.*, The Journal of Chemical Physics **21**, 1087 (1953).

[2] R. H. Swendsen and J.-S. Wang, Phys. Rev. Lett. **58**, 86 (1987).

[3] U. Wolff, Phys. Rev. Lett. **62**, 361 (1989).

[4] D. J. Watts and S. H. Strogatz, Nature **393**, 409 (1998).

[5] A.-L. Barabási and R. Albert, Science **286**, 509 (1999).