Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Proseminar Modern Topics in Statistical Mechanics

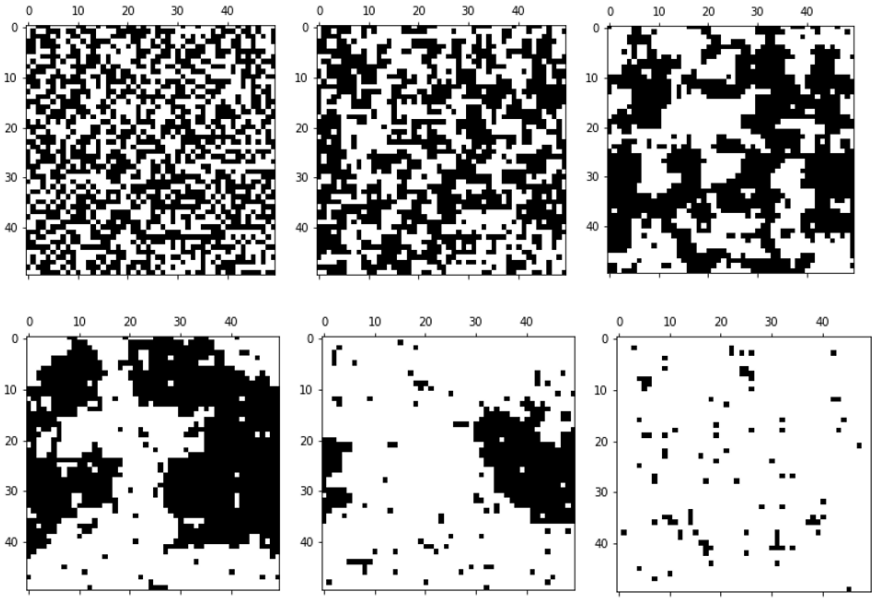
Basic Monte Carlo Introduction including Metropolis and Wolff Algorithms

April 3, 2020

Philipp Binkert

Abstract

This report offers an introduction to Monte Carlo simulations aimed at readers with no prior knowledge on the subject. Two of the most important algorithms, the Metropolis algorithm and the Wolff algorithm, are discussed in detail, with an emphasis being laid implementation, correlation times and critical slowing down. A code for python is referenced as a guideline and to demonstrate these effects. The two-dimensional Ising model serves as a basis on which the algorithms are discussed, although the concepts are presented in such a way that they can be applied to other statistical mechanical systems as well.



Contents

1 Introduction

1.1 Objective and overview

This report aims to give an introduction to a very powerful and widely used class of numerical methods, the Monte Carlo simulations, as well as demonstrate their applications in the field of statistical mechanics. The reader is not required to have any prior knowledge on the subject apart from a basic understanding of numerical methods in general and some statistical mechanics. The report is based on the excellent "Monte Carlo Methods in Statistical Physics" by M.E.J. Newman and G.T. Barkema [1] which can be highly recommended for readers interested in a deeper analysis of the subject.
In this introductory chapter I will give a quick overview over the different areas of application of MC simulations, as well as introduce the Ising Model which will lend itself as a good example to test various concepts on. In the next chapter the Monte Carlo method will then be explained along with some simpler examples to get a feel for how it works and why its applications are so diverse. Following that I will discuss the main principles of equilibrium thermal MC simulations, which lay the foundation for the following algorithms, the Metropolis and the Wolff algorithm. These two algorithms will then be treated in detail, including implementations for the aforementioned Ising model.

1.2 Applications and connection to statistical mechanics

Monte Carlo methods are used in a great number of different fields, such as engineering, physics, biology, mathematics, finance or medicine just to name a few. The reason for their extremely diverse applications should become clear as soon as the basic concept presented in the next chapter is understood.
While MC simulations are performed in almost all branches of physics, they are especially important in statistical mechanics. In statistical mechanics, we are usually dealing with systems that consist of the order of 10^{23} particles and we are interested in some macroscopic properties of the system. The theory tells us that these properties are often given by very large sums over all possible states of the system. Take for instance the internal energy U which is given by

$$U = \frac{1}{Z} \sum_{\mu} E_\mu \mathrm{e}^{-\beta E_\mu} \tag{1}$$

where E_μ denotes the internal energy of state μ and Z is the partition function, which is itself again a sum over the Boltzmann probabilities of the states,

$$Z = \sum_{\mu} \mathrm{e}^{-\beta E_\mu}. \tag{2}$$

While there exist certain textbook examples of systems with energy distributions that allow for analytical solutions of the above sums to be found, it is clear that for most real-world problems in statistical mechanics there is no such possibility and thus a method for

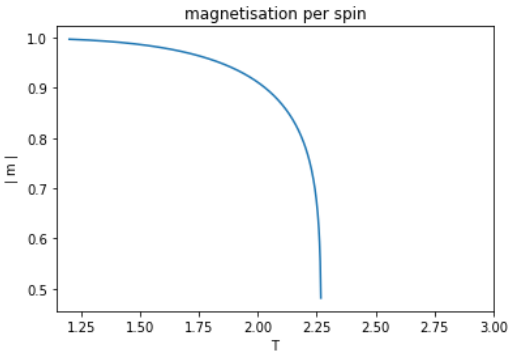


Figure 1: Magnetisation per spin for the Ising model. At $T = 2.27$, there is a phase transition from a ferromagnetic to a paramagnetic phase.

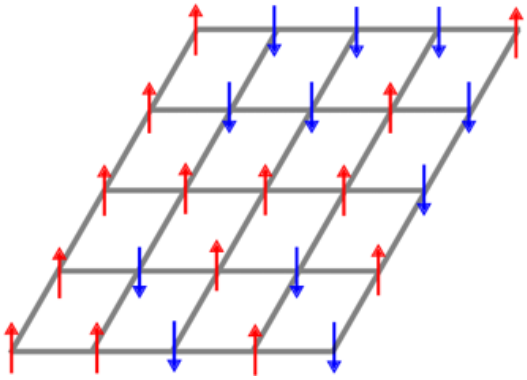


Figure 2: Illustration of the Ising model. Each arrow sits on a fixed lattice point and depicts a spin with value either $+1$ (red) or -1 (blue). [3]

numerically approximating these large sums is required. This is where the MC method comes into play; in fact, it is one of only few known methods, as is also the case for most other sets of high-dimensional problems, that can produce good results within reasonable computation time. This is why they are so important in modern science and therefore worth pursuing.

1.3 The Ising model

The Ising model will be the model upon which we test our simulations later on. It has several advantages: Firstly, for the two-dimensional case there is an analytical solution which we can compare the numerical results to in order to make sure that they are producing reasonable data. The simulation can then be expanded to three dimensions relatively easily, for which there is no more analytical solution. Additionally, the model is relatively simple to implement and shows a phase transition of second order at around $T = 2.27$ (Figure 1), which is important for the numerical approach and will be investigated thoroughly.

The Ising model is a simplified model of a magnet. It assumes that the magnet is built up of many small, discrete magnetic dipole moments, termed spins, which contribute to the overall magnetisation, internal energy and so on. These spins s_i are assumed to be fixed in space on a square lattice and take on one of only two values, ± 1. This is illustrated in Figure 2. A given state of the model is simply defined by the values of all the spins s_i and the total magnetisation of the state is naturally given by summing over all the microscopic spins, $M = \sum_i s_i$. The spins interact with each other through exchange or RKKY interactions. In this simple model we assume interactions only between nearest neighbors and we assume that all are of the same strength J. The Hamiltonian of the

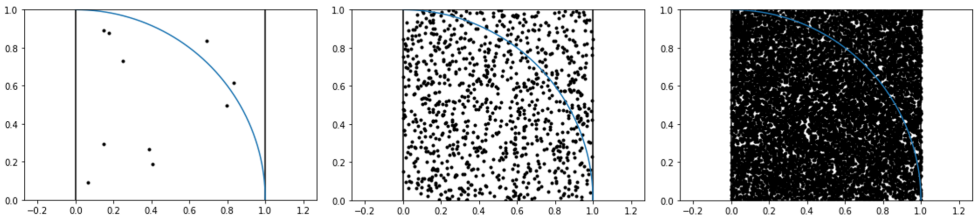


Figure 3: Numerical approximation of π with $N = 10$, 1000 and 10000 sample points.

Ising model is therefore given by

$$H = -J \sum_{\langle ij \rangle} s_i s_j - B \sum_i s_i \tag{3}$$

where $\langle ij \rangle$ indicates the sum over all pairs of nearest neighbors in the lattice and B denotes an external magnetic field which would give rise to a further term but which we will choose to set zero for the remainder of this report.

2 Basic concept and first examples

2.1 Probability distribution and repeated random sampling

It is difficult to give a precise definition of Monte Carlo simulations. As already hinted at in the last section, they are generally useful when the sample size of some system is infinite or too great for all the possibilities to be taken into account. The MC method relies on repeated random sampling to get a good estimate of the final result, while only effectively calculating with a tiny fraction of the entire sample space. In order for this to have a chance to be successful, it is vital that the random sampling process takes the into account the probability distribution of the possible events and selects the random events in accordance; the greater the probability of an event, the more likely it should be selected for the repeated random sampling. The two following examples should help to clarify and grasp this concept.

2.2 Approximation of π using the MC method

Let us begin with an easy, illustrative example, the numerical approximation of π using Monte Carlo. The idea is as follows: Take a unit square and inscribe a quarter circle into it. By geometrical definition, the area of this quarter circle is given by $\frac{\pi}{4}$, whereas the area of the whole square is given by 1. The idea is now to randomly generate points inside the square and determine whether they lie inside or quarter circle or not. If we do this for sufficiently many points, the ratio of points inside the circle to total points

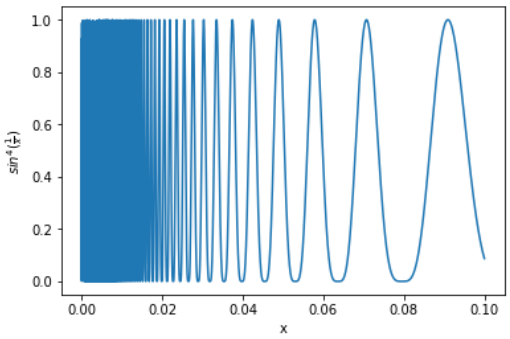


Figure 4: Example of a poorly behaved function on the interval $[0, 0.1]$

will approach $\frac{\pi}{4}$ and thus by rearranging we get a numerical approximation of π by

$$\pi \approx 4 \cdot \frac{\text{points inside}}{\text{points total}}. \tag{4}$$

Figure 3 illustrates the process for 10, 1000 and 10000 points respectively. After one million points, which can be done in a matter of a few seconds on any laptop, one finds an approximation for π of $\pi \approx 3.14174$. Notice the two key concepts from the subsection above that are at work. Repeated random sampling is what gives a good approximation of the result, even though we only took into account one million of the infinitely many points inside the square. The uniform probability distribution for all the points guaranteed that the result was not biased by preferring certain points in certain areas over others. This may seem trivial at the moment, but will be crucial when treating statistical mechanical systems.

2.3 Numerical integration using Monte Carlo

One widely used field for Monte Carlo methods is for numerical integration of badly behaved functions. This technique is closely related to the example above. Let us take as an example the function $f(x) = \sin^4(\frac{1}{x})$ which we would like to integrate between 0 and 0.1, where it is very poorly behaved as evident in Figure 4. Standard quadrature methods will not work because it is not possible to choose the intervals along the input axis small enough near the origin. Instead, the idea is again to randomly select points in the area $[0, 0.1] \times [0, 1]$ and determine whether they belong to the area beneath the curve or not, which can be done simply by evaluating $\sin^4(\frac{1}{x})$ for the x-coordinate of the point and comparing it with its y-coordinate. The integral can then be approximated by repeating this process for many randomly chosen points and observing the ratio of points above and below the curve.

3 Principles of equilibrium thermal Monte Carlo simulations

3.1 Overview

In this chapter I will discuss the key considerations behind equilibrium thermal Monte Carlo simulations in detail, which will conclude in several conditions that must be fulfilled for any MC algorithm. In the next two chapters, we will analyse the two most important algorithms using the insights gained in this chapter.

3.2 Importance sampling and Markov processes

As discussed in the introduction, we are usually interested in calculating macroscopic properties of large systems in statistical mechanics. This is done by calculating the weighted average of some observable quantity Q over all possible states, with the weights in this case being the Boltzmann probabilities:

$$\langle Q \rangle = \frac{\sum_\mu Q_\mu \mathrm{e}^{-\beta E_\mu}}{\sum_\mu \mathrm{e}^{-\beta E_\mu}} \tag{5}$$

In the case of the 2D Ising model, it can immediately be seen that even for a very small system, say 100×100, there are 2^{10000} different possible states and it is beyond hope to ever evaluate the entire sums in equation (5). The idea of the MC simulation is instead again to evaluate just a few of the states and hope to get a good estimate of the actual result. The question is how to choose appropriate states to evaluate. We assign each state μ a probability p_μ to be chosen. If we then choose M states, we will get an estimator Q_M for $\langle Q \rangle$ as follows:

$$Q_M = \frac{\sum_{i=1}^{M} Q_{\mu_i} p_{\mu_i}^{-1} \mathrm{e}^{-\beta E_{\mu_i}}}{\sum_{j=1}^{M} p_{\mu_j}^{-1} \mathrm{e}^{-\beta E_{\mu_j}}} \tag{6}$$

A first, naive attempt would be to just set all of the probabilities $p_{\mu i}$ equal, i.e. give each state the same probability of being selected. Upon closer examination, it quickly becomes clear that this is not a good choice to make. The reason is that the sums in equation (5) are oftentimes dominated by just a few states and the likelihood of catching these states is vanishingly small. Consider as an example a system at low temperature. Because of the lack of thermal energy, the system remains near the ground state and never reaches the higher energy states. We must find a way to ensure that our simulation takes this into account and also selects states near the ground state. This general principle of choosing the appropriate states from the almost infinitely large pool of total states is known as importance sampling. The usual approach to ensure this is to choose each state with a probability proportional to its Boltzmann weight,

$$p_\mu = \frac{1}{Z} e^{-\beta E_\mu}. \tag{7}$$

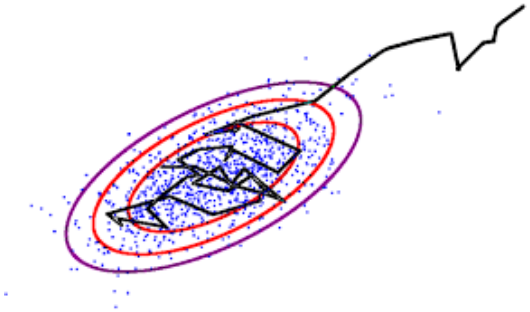


Figure 5: Illustration of a Markov chain. A random state is chosen as a starting point and the Markov process ensures that the following states gravitate towards the equilibrium states. [4]

The estimator from equation (6) then simplifies to $Q_M = \frac{1}{M} \sum_{i=1}^{M} Q_{\mu i}$.

The next question is how to generate the random states according to the probability distribution given by equation (7). Again, a naive approach would be to just randomly select states and then accept them with a probability p_μ. This would be very inefficient however, as most of the states would have a very low probability of being accepted and we would thus be wasting a lot of computational time. Instead, the better approach is to start in any one random state μ and then assign transition probabilities $P(\mu \to \nu)$ for the transition from state μ to each state ν. These transition probabilities should satisfy the following constraints: They should be universal, i.e. not be dependent of time or anything other than the states μ and ν and they must fulfill Kolmogorov's law

$$\sum_{\nu} P(\mu \to \nu) = 1, \tag{8}$$

as we wish to transition to some state in any case. Note that $P(\mu \to \mu)$ can be greater than zero, i.e. the system can stay in the same state with some probability. The Monte Carlo simulation will use these transition probabilities to transition from one state μ to the next state ν and so on, thereby creating a so-called Markov chain (see Figure 5). The goal is for the Markov chain to be created in such a way that it eventually generates states near the equilibrium as defined by the Boltzmann distribution. To ensure this, two further conditions must be applied which will be discussed in the next section.

3.3 Ergodicity and detailed balance

The first condition that the transition probabilities must fulfill is ergodicity. Ergodicity states that any two states μ and ν must be connected by a Markov chain of finite length with non-vanishing probability, i.e. it must be possible to reach any state ν starting in

state μ. The reason for this requirement can quickly be understood if one goes back to the example of the low-temperature system that is predominantly in the lowest few energy states; the randomly chosen initial state is very likely going to be far away from these key states and if it is not possible to reach them through the Markov process, then equilibrium will never be reached.
The second condition is the one that ensures that we generate states in equilibrium according to the Boltzmann distribution. The defining condition of a system in equilibrium is that the rate of transitions into and out of any given state μ must be equal. This can be expressed as follows:

$$\sum_{\nu} p_\mu P(\mu \to \nu) = \sum_{\nu} p_\nu P(\nu \to \mu) \tag{9}$$

In fact, it can be shown that this condition can be tightened. The problem with equation (9) is that it allows in theory for a dynamic equilibrium to be established, which can be thought of as a cycle of states between which the Markov chain rotates. This is unphysical and should be avoided in context with statistical mechanics. It can be shown (the interested reader is referred to [1] for a derivation) that the elimination of this effect along with the help of equation (8) require for equation (9) to be modified to

$$p_\mu P(\mu \to \nu) = p_\nu P(\nu \to \mu) \iff \frac{P(\mu \to \nu)}{P(\nu \to \mu)} = \frac{p_\nu}{p_\mu} = \mathrm{e}^{-\beta(E_\nu - E_\mu)}. \tag{10}$$

This is called the condition of detailed balance and forms the second important condition that guarantees the correct equilibrium distribution along with ergodicity.

3.4 Acceptance ratios

Ergodicity and detailed balance still allow for a great amount of freedom in how to choose the transition probabilities. A very crude but theoretically sound approach would be to just define

$$P(\mu \to \nu) \propto \mathrm{e}^{-\frac{1}{2}\beta(E_\mu - E_\nu)} \tag{11}$$

for all μ, ν. We will later see that this would be a very inefficient algorithm. Instead, we would like to have more control over how the states are selected. To that end we divide the transition process into two parts and introduce selection probabilities $g(\mu \to \nu)$ and acceptance ratios $A(\mu \to \nu)$, $P(\mu \to \nu) = g(\mu \to \nu)A(\mu \to \nu)$. For each step in the Markov process a new state is selected according to probability $g(\mu \to \nu)$ and then accepted with a probability $A(\mu \to \nu)$ or otherwise rejected, in which case a new state is selected. This process is still random, but it has the advantage that we can now control the probabilities $g(\mu \to \nu)$ and adjust the acceptance ratios to ensure that detailed balance,

$$\frac{g(\mu \to \nu)A(\mu \to \nu)}{g(\nu \to \mu)A(\nu \to \mu)} = \frac{P(\mu \to \nu)}{P(\nu \to \mu)} = \frac{p_\nu}{p_\mu} = \mathrm{e}^{-\beta(E_\nu - E_\mu)}, \tag{12}$$

is fulfilled. Note that the acceptance ratios should be always be chosen to be as large as possible in order to avoid wasting computation time selecting states that just get

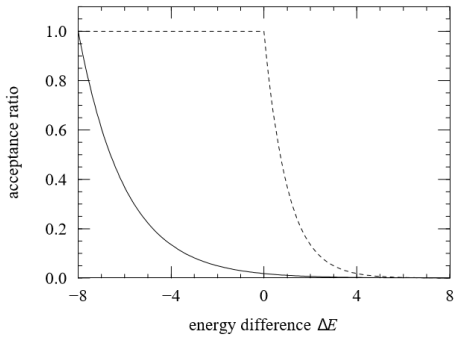


Figure 6: Acceptance ratios of (15) (solid line) and (16)(dashed line). [1]

rejected. It is the choices of the selection probabilities and acceptance ratios that define the different Monte Carlo algorithms, and the next chapters will discuss two prime examples.

4 The Metropolis algorithm

4.1 Definition and basic properties

The Metropolis algorithm was the first Monte Carlo algorithm developed and is still the most widely used today. We will examine it using the Ising model described in the introduction. Consider a magnet modelled by an $l \times l$ Ising Model in a state near equilibrium. Typically, the system will stay in a very narrow energy range around the equilibrium energy for most of the time. For our simulation this implies that we shouldn't be considering transitions into states with completely different energy levels. A simple way to achieve this is to only flip one spin at a time. Algorithms which follow this scheme are said to have single-spin-flip dynamics. In the language of selection probabilities, there are $l \times l = N$ possible states ν for any given state μ, each of them differing from μ by a single spin. All of them should have an equal selection probability, so we set

$$g(\mu \to \nu) = \begin{cases} \frac{1}{N} & \text{if } \mu, \nu \text{ differ by 1 spin} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Note that any algorithm using single-spin flip dynamics automatically fulfills ergodicity, as each spin can theoretically be flipped one after the other to achieve any desired configuration of the lattice. Because $g(\mu \to \nu) = g(\nu \to \mu)$ for any two states μ, ν, the condition of detailed balance (equation (12)), simplifies to

$$\frac{A(\mu \to \nu)}{A(\nu \to \mu)} = \mathrm{e}^{-\beta(E_\nu - E_\mu)}. \tag{14}$$

As already touched upon in the last section in equation (11), we could simply choose our acceptance ratios proportional to $e^{-\frac{1}{2}\beta(E_\nu - E_\mu)}$ and the constant of proportionality

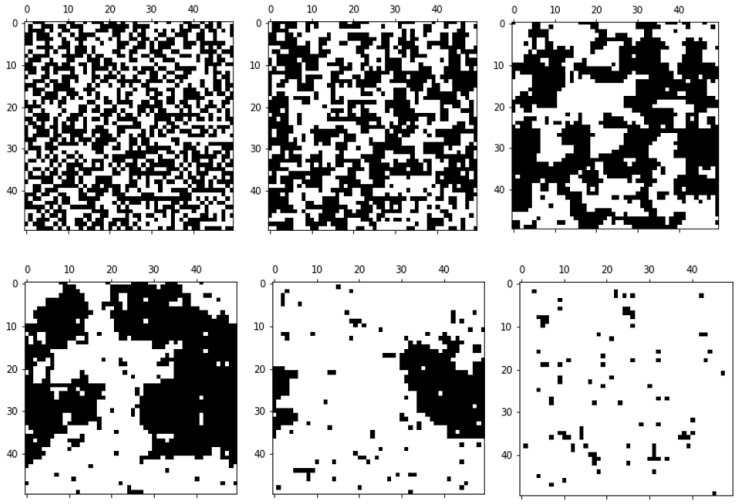


Figure 7: Simulation of a 50×50 Ising model at $T = 2$ starting in initial state $T_{init} = \infty$. The pictures show the boards after $N = 0, 1, 5, 50, 500, 1000$ sweeps. After 1000 sweeps it has reached equilibrium and has a high magnetisation.

would cancel out in (14). We could therefore choose it however we like as long as we make sure that $A(\mu \to \nu)$ is never > 1. Since we are only flipping one spin at a time, a maximum of four bonds between spins can be broken in the 2D Ising model, so it holds that $E_\nu - E_\mu \leqslant 8J$. Recalling that our acceptance ratios should always be as large as possible, we set the constant of proportionality to the maximum $e^{-4\beta J}$ and arrive at

$$A(\mu \to \nu) = \mathrm{e}^{-\frac{1}{2}\beta(E_\nu - E_\mu + 8J)}. \tag{15}$$

Due to the exponential decay, this is very inefficient however; the best approach is usually to set the larger of the two acceptance ratios to 1 and adjust the other one accordingly to satisfy (14). The optimal algorithm for this situation, and this is what defines the famous Metropolis algorithm, is thus defined by

$$A(\mu \to \nu) = \begin{cases} \mathrm{e}^{-\beta(E_\nu - E_\mu)} & \text{if } E_\nu - E_\mu > 0 \\ 1 & \text{otherwise} \end{cases} \tag{16}$$

Figure 6 shows the difference in acceptance ratios and underlines how much more efficient the Metropolis algorithm (16) is compared to the naive approach (15).

4.2 Implementation and equilibration

A possible implementation of the Metropolis algorithm in Python can be found in [5]. In this section I will just touch upon some key considerations:

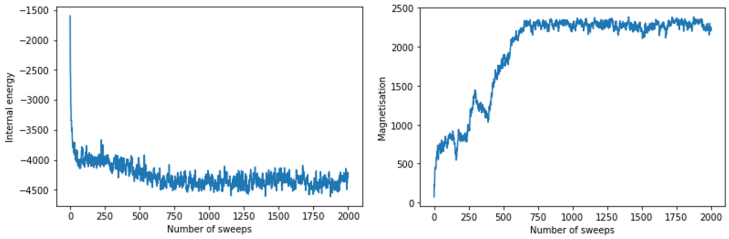


Figure 8: Plots of internal energy and magnetisation as a function of sweeps for the simulation from Figure 7. It can be seen that equilibrium was reached after about 600 sweeps.

- As initial states for the Markov chain, it is advisable to choose either the state where all spins are aligned (this corresponds to the $T = 0$ state, set $J \equiv 1$) or where all the spins have completely random orientation (this implies that the thermal energy completely dominates, the $T = \infty$ state).
- For low temperatures ($T \leqslant 3$) it is advantageous to use $T_{init} = 0$, for higher temperatures $T_{init} = \infty$ in order to reach equilibrium faster.
- When calculating various quantities such as energy or magnetisation, a lot of computation time can be saved by storing them in variables and adjusting them after each step. This avoids having to perform sums over the whole grid to recalculate them every step.

Figure 7 shows the Ising grid after different numbers of sweeps for a system at $T = 2$. In order to subsequently take measurements, it is important to know at what point the simulation reaches equilibrium, i.e. to determine the equilibration time τ_{eq}. The best way to do this is to plot some observable quantities like the energy or magnetisation as a function of the number of sweeps (1 sweep consists of N steps). It should become clear from these plots when equilibrium was reached, as can be seen in Figure 8. After this point the system should stay in equilibrium, as that is how it was designed, and one can start taking measurements.

4.3 Measurements and error calculation

As soon as the system is in equilibrium, one can measure physical quantities directly from the simulation. Of interest may be several properties of a magnet such as internal energy, magnetisation, specific heat, entropy, magnetic susceptibility and so on, at any temperature desired. Figure 9 shows the plots of the first three as a function of different temperatures including the graphs of the exact solution (for a derivation of the exact solutions, the reader is referred to [2]).

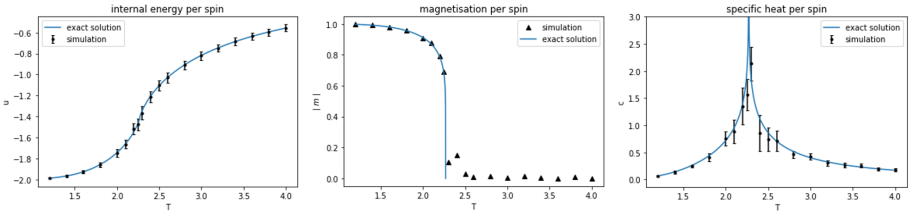


Figure 9: Plots of internal energy, magnetisation and specific heat per spin as a function of temperature. Measurements were made in intervals of $T = 0.2$ and more densely in the vicinity of $T = 2.27$.

Some extra care must be given in determining the statistical errors of certain observables. Most quantities such as the internal energy or magnetisation can be evaluated for every step in the Monte Carlo simulation and it is straightforward to calculate the standard deviation from the mean value using the Gaussian method. However certain other quantities like the specific heat c cannot be defined for any single state; it is rather an inherently average property of a material, depending on averages of u and u^2:

$$c = \frac{\beta^2}{N} \left(\langle u^2 \rangle - \langle u \rangle^2 \right) \tag{17}$$

It is therefore not trivial to calculate the standard deviation of these quantities. The most simple, although not always the best approach to solving this problem is given by the blocking method. The trick is to divide all of the measurements into separate blocks and calculate the required mean properties of these blocks. The final mean and standard deviation of the desired quantity can then be determined using the standard Gaussian method with the means of the blocks as data points. As an example, say the simulation delivers 1000 measurements of u. These can be divided into 10 blocks of 100 measurements. For each block, $\langle u \rangle$ and $\langle u^2 \rangle$ can be calculated and from these values one can obtain the mean and standard deviation of c using equation (17). More involved techniques for this purpose are the bootstrap or the jackknife method which can be found in [1].

4.4 Phase transition

When observing the plots in Figure 9, it evident that something happens around $T_C = 2.27$. The magnetisation changes from ferromagnetic to paramagnetic and the specific heat has a singularity. It is beyond the scope of this report to discuss the nature or the properties of this phase transition, but it has important consequences concerning the choice of the Metropolis algorithm to which I will now turn. Aside from the special behaviour of said quantities around T_C, the Ising grid also shows special configurations when equilibrating around the critical temperature as can be seen in Figure 10. For low temperatures, most of the spins are aligned in the same direction and thus give rise to the net magnetisation. For high temperatures on the other hand, the spins have

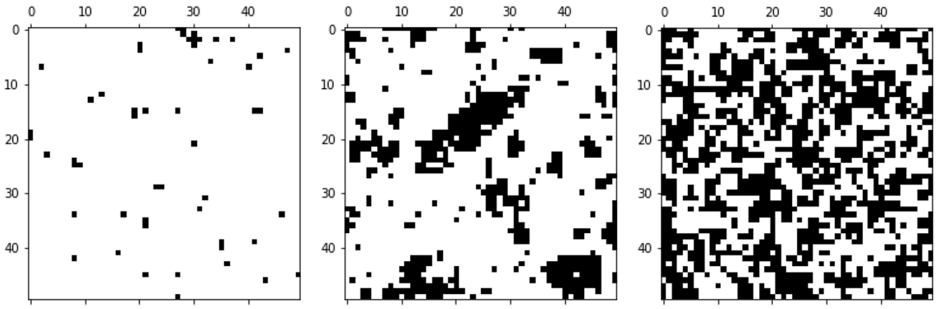


Figure 10: Ising model boards in equilibrium at $T = 1.8$, $T = 2.27$ and $T = 4$.

mostly random orientation. Around T_C though, the spins appear to form clusters of the same orientation. This observation will be important for the upcoming discussion on correlation time and ultimately for a new algorithm.

4.5 Correlation time and critical slowing down

When taking our measurements so far we have taken data after one sweep, i.e. when on average every spin has had the chance to flip once. By doing this, we have tacitly made the assumption that the two states before and after the sweep were uncorrelated. It is now time to be more rigorous in this regard and determine the correlation time of the simulation at the different temperatures. The correlation time can intuitively be described as a measure for how many computational steps it takes for the system to get from one state μ to another state ν such that μ and ν are independent. In order to calculate it, the time-displaced autocorrelation function $\chi(t)$ is introduced. This function is defined as

$$\chi(t) = \int \mathrm{d}t' \left[q\left(t'\right) - \langle q \rangle\right] \left[q\left(t'+t\right) - \langle q \rangle\right] \tag{18}$$

for some observable q and drops off exponentially, $\chi(t) \sim e^{-t/\tau}$, and the correlation time τ is defined as the typical time scale on which it drops off. It is best determined by plotting the autocorrelation function logarithmically and calculating the slope using the least squares method as shown in Figure 11. For more details on why for instance the autocorrelation function shows exponential decay or other possibilities of calculating τ, [1] can again be recommended.
Figure 12 shows the correlation time for the magnetisation τ_m of my Metropolis algorithm as a function of temperature. While for most temperatures one can unhesitatingly take one measurement after each sweep, this is not the case near the critical temperature. This can be understood: Because the spins form clusters, only those at the edges of the clusters have a realistic shot at getting flipped during a sweep and as a result the Ising board will not change much after one sweep. This means that the two states before

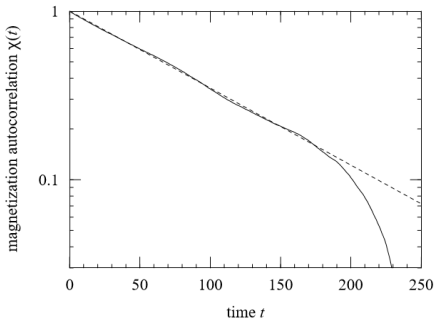


Figure 11: Logarithmic plot of the autocorrelation function. The slope of the best fit line (dashed) determines the correlation time. [1]

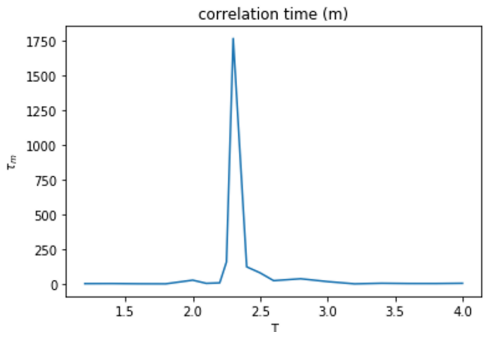


Figure 12: Correlation time of magnetisation as a function of temperature. The correlation time diverges near T_C.

and after the sweep are strongly correlated and we have to wait for a very long time to get two uncorrelated states using the Metropolis algorithm. This effect is called critical slowing down around T_C and it is this that raises the idea for a new class of algorithms that shall be introduced in the next chapter.

5 The Wolff algorithm

5.1 Cluster algorithms

We saw in the last chapter that the Metropolis algorithm produces strongly correlated data near the critical temperature. We will now attempt to improve this by choosing a fundamentally different approach. Instead of flipping one spin at a time, the idea is now to flip entire clusters of similarly-oriented spins at once. Algorithms which follow this technique are called cluster algorithms and form the second important group next to those using single-spin flip dynamics. The Wolff algorithm is the most widely used cluster algorithm.

The first step in implementing any cluster algorithm consists of defining rules for how the clusters to be flipped should be determined. The simplest strategy works as follows: First a spin is chosen at random to be the seed spin. Each of this seed spin's neighbors are then examined and if they are oriented in the same direction as the seed spin the are added to the cluster at a certain probability P_{add}. This process is then repeated for any spins thereby added to the cluster and so on, until no more spins are added and the cluster is complete. The question that remains is how to define P_{add} and the acceptance ratios in such a way that detailed balance is fulfilled (equation (12)). Note that ergodicity is again trivially fulfilled as theoretically a cluster can just consist of just one spin and then it is the same situation as for the single-spin dynamics algorithms.

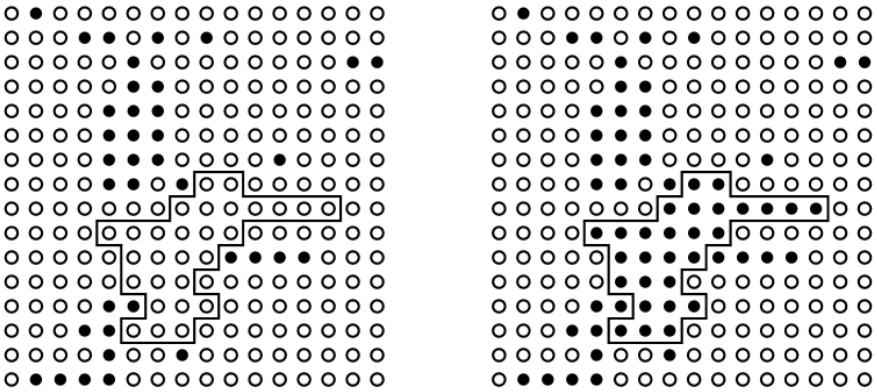


Figure 13: An example of a selected cluster of similarly oriented spins. μ (left) defines the state before and ν (right) defines the state after the cluster being flipped. [1]

5.2 Acceptance ratio and basic properties

To be able to define the acceptance ratios, we must first understand the selection probability of a given cluster. Consider the cluster depicted in Figure 13. Let states μ and ν be the states before and after the cluster is flipped. The crucial difference between μ and ν is the way the spins are oriented at the edge of the cluster. The probability of *not* adding a spin of same orientation to the cluster is $1 - P_{add}$ and thus the selection probability $g(\mu \to \nu)$ of the cluster is $(1 - P_{add})^m$ with m the number of same oriented spins along the edge of the cluster. By the same argument, the selection probability for the reverse move $g(\nu \to \mu)$ is given by $(1 - P_{add})^n$ with n the number of same-oriented spins in state ν along the edge of the cluster, which corresponds to the number of opposite-oriented spins in state μ. Equation (12) can thus be simplified to

$$(1 - P_{\text{add}})^{m-n} \frac{A(\mu \to \nu)}{A(\nu \to \mu)} = \mathrm{e}^{-\beta(E_\nu - E_\mu)}. \tag{19}$$

The change in energy $E_\nu - E_\mu$ can also be determined in terms of m and n: For each same-oriented spin along the cluster edge, a bond is broken when going from μ to ν resulting in a change in energy of $+2J$ according to the Hamiltonian of the system (equation (3)). Conversely, for each of the the n bonds that are created by moving from μ to ν, the internal energy drops by $-2J$. All of the other bonds inside or outside of the cluster do not change and therefore the net change in energy can be described by $E_\nu - E_\mu = 2J(m - n)$. (19) can thus be rearranged and simplified to

$$\frac{A(\mu \to \nu)}{A(\nu \to \mu)} = \left[\mathrm{e}^{2\beta J} (1 - P_{\text{add}})\right]^{n-m}. \tag{20}$$

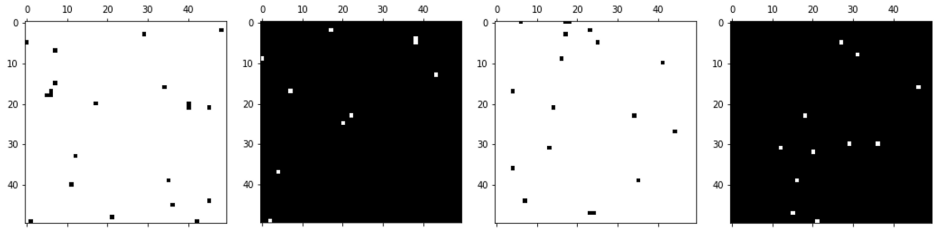


Figure 14: Four consecutive steps of a Wolff algorithm at $T = 1.5$. Most of the time the cluster will consist of almost the entire Ising board.

Recall that the acceptance ratios should always be chosen as great as possible for highest efficiency. Upon examining (20) we realise that if we choose

$$P_{add} = 1 - e^{-2\beta J}, \tag{21}$$

then the right side simplifies to 1 and we can automatically set all acceptance ratios to 1! It is this choice of P_{add} together with the described procedure that defines the Wolff algorithm.

The Wolff algorithm is, like the Metropolis algorithm, a proven algorithm that will generate good data according to the correct Boltzmann probabilities in all temperature ranges. Notice that for high temperatures, there will not be any large clusters due to the fact that the spins are mostly randomly oriented and thus the Wolff algorithm functions similarly to the Metropolis algorithm in these ranges. For low temperatures on the other hand, the Wolff algorithm will flip almost the whole Ising board most of the time as P_{add} is large due to $\beta \gg 1$ (see Figure 14). As a consequence it is important to take the absolute values of certain quantities such as magnetisation as otherwise the average will be close to zero. In both situations, the Wolff algorithm is also pretty efficient, however not quite as good as the Metropolis algorithm. The Wolff algorithm should hopefully be superior at temperatures close to T_C, which is what it was designed for, and this shall be analysed in the next section.

5.3 Critical and dynamic exponents

As we saw in the last chapter, the spins of the Ising model tend to form larger and larger clusters of same orientation as the temperature approaches T_C. This can be formalized by introducing the correlation length ξ, which is a measure for the typical size of these clusters at a given temperature. Let us also introduce the reduced temperature t, which is a dimensionless quantity measuring how far away we are from the critical temperature:

$$t = \frac{T - T_c}{T_c} \tag{22}$$

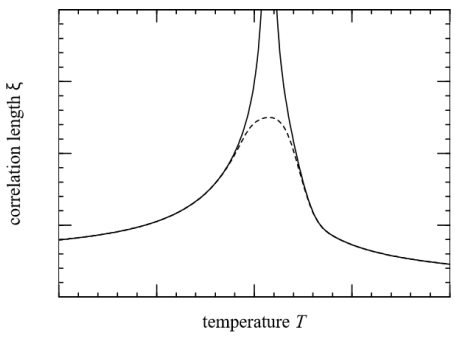


Figure 15: Depiction of the correlation length in the vicinity of T_C. The solid line shows the divergence of an infinite system as opposed to the dashed line which shows the cut-off at around $\xi = L$ for a system with finite dimension. [1]

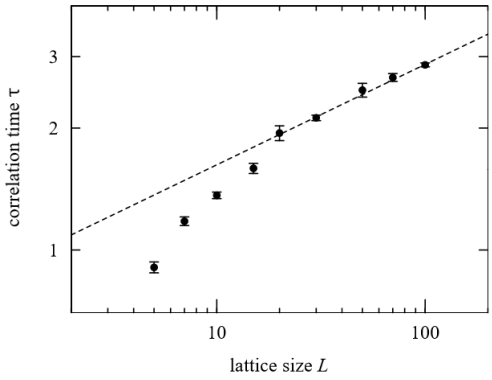


Figure 16: Logarithmic plot of the measured correlation time against the lattice size L. The dynamic exponent can be determined by evaluating the slope of the best fit line (dashed). [1]

It can be shown that the divergence of the correlation length in $t = 0$ behaves as

$$\xi \sim |t|^{-\nu}, \tag{23}$$

where $\nu > 0$ is the critical exponent. An important result of renormalization group theory is that the value of the critical exponent is an intrinsic property of the Ising model itself and thus cannot be influenced by our choice of algorithm. However, the algorithm does have an influence on the correlation *time*, which can be described by

$$\tau \sim |t|^{-z\nu}. \tag{24}$$

z is called the dynamic exponent and allows us to quantify the effect of critical slowing down near T_C.
In order to compare the Metropolis and the Wolff algorithm near the critical temperature, one must therefore approximate their dynamic exponents. This can be done as follows: First equations (23) and (24) are combined to obtain $\tau \sim \xi^z$. For finite sized systems like we are always dealing with in our simulations, it is clear that the correlation length can never truly diverge, as there can never be a larger cluster than L^2 (see Figure 15). For temperatures sufficiently close to T_C, we can therefore write

$$\tau \sim L^z. \tag{25}$$

To determine z, the task is then to measure the correlation times τ at T_C for Ising models of different sizes and then plot them logarithmically against L in order to extract z by calculating the slope of the best fit line as shown in Figure 16.

What is important to keep in mind during this process is that the Wolff algorithm takes a longer time to complete one step than does the Metropolis algorithm as it flips more

spins at once. To have a fair comparison between the two, one should therefore compare $\tau_{Metropolis}$ to

$$\tau_{Wolff} = \tau_{\text{steps}} \frac{\langle n \rangle}{L^d} \tag{26}$$

where $\langle n \rangle$ denotes the average cluster size. Applying this, it turns out that the Metropolis algorithm has a dynamical exponent of $z = 2.1665 \pm 0.0012$, while the Wolff algorithm has a much lower one of $z = 0.25 \pm 0.02$ ([1]). Clearly, the cluster algorithm has fulfilled the expectation in this regard and is much more efficient near the critical temperature.

6 Conclusion

To conclude this report I would like to reiterate that both the Metropolis and Wolff algorithms are good choices to simulate statistical mechanical systems like the Ising model. Far away from the critical temperature (≥ 0.2), the Metropolis algorithm is the ideal choice due to its high efficiency, whereas closer to T_C the Wolff algorithm should be applied. In many cases more complicated than the Ising model it is often not as obvious when a simulation approaches a critical point. In such cases a hybrid solution is an option, meaning that the simulation starts with the Metropolis algorithm, keeps track of the correlation length and if a certain length is reached the program automatically switches to the Wolff algorithm. However, it must be stated that cluster algorithms are not always possible for every system and developing them for complex systems is still an active field of modern research. The Metropolis algorithm on the other hand can be implemented for basically any system and thus can be considered as a blackbox algorithm for Monte Carlo simulations. It will always provide results, however oftentimes a lot of computational time can be saved by putting in some extra effort to develop an algorithm customized for the problem at hand.

References

[1] M.E.J. Newman & G.T. Barkema, "Monte Carlo Methods in Statistical Physics" (1999)

[2] L. Onsager, "Crystal statistics. I. A two-dimensional model with an order-disorder transition" (1944)

[3] Sascha Wald, "Thermalisation and Relaxation of Quantum Systems" (2017)

[4] Iain Murray, lecture on advanced MCMC methods (2006)
http://mlg.eng.cam.ac.uk/zoubin/tut06/mcmc.pdf

[5] .....